UNIVERSITAT ROVIRA i VIRGILI

Fairness and Robustness in Machine Learning

Ashneet Khandpur Singh



DOCTORAL THESIS

2023

Ashneet Khandpur Singh

Fairness and Robustness in Machine Learning

DOCTORAL THESIS

Supervised by Prof. Dr. Josep Domingo-Ferrer

Co-Supervised by Dr. Alberto Blanco-Justicia

Department of Computer Engineering and Mathematics



UNIVERSITAT ROVIRA i VIRGILI

January 2023

I STATE that the present study, entitled "Fairness and Robustness in Machine Learning", presented by Ashneet Khandpur Singh for the award of the degree of Doctor, has been carried out under my supervision at the Department of Computer Engineering and Mathematics of this university, and that it fulfills all the requirements to be eligible for the International Doctorate Award.

Tarragona, January 24, 2023

Doctoral Thesis Supervisor	Doctoral Thesis Supervisor
Prof. Dr. Josep Domingo-Ferrer	Dr. Alberto Blanco-Justicia

The following funding sources are gratefully acknowledged:
European Commission (projects H2020-871042 “SoBigData++” and H2020-101006879 “MobiDataLab”), the Government of Catalonia (ICREA Acadèmia Prize to J. Domingo-Ferrer), the Spanish MCIN/AEI/10.13039/501100011033 /FEDER, UE under project PID2021-123637NB-I00 “CURLING”, and the Martí i Franquès grant from the Universitat Rovira i Virgili (URV).





Acknowledgements

This Ph.D dissertation would not have been possible without the help and support of many people. Herein, I would like to express my greatest acknowledgements to everyone that contributed to the achievement of this new accomplishment.

First and foremost, I would like to express my deepest gratitude to my supervisors Prof. Dr. Josep Domingo-Ferrer and Dr. Alberto Blanco Justicia for the continuous support of my Ph.D study and research with their immense knowledge. Without their guidance, this work would not have been feasible.

I sincerely acknowledge also all the CRISES group members, especially Jesús Manjón, Prof. Dr. David Sánchez, Dr. Sergio Martínez, and Rami Haffar for always being there for helping and advising me. A special mention goes to the past members, Dr. Michael Bamiloshin and Romina Russo.

Besides them, I would like to extend my sincere thanks to Dr. Jordi Castellà and Dr. Blas Herrera, for always guiding me for my future and with the next steps to make after the Ph.D.

I shall not want to forget, of course, the late Dr. David Rebollo Monedero: a heartfelt gratitude to him for his helpful comments and contributions.

I am truly thankful to Prof. Dr. Pino Caballero Gil and Dr. Candelaria Hernández Goya, for introducing me this new world different from my usual Mathematics.

As in every step forward in my life, I cannot forget that this achievement could not have been possible without the support of my family. I am deeply grateful to my parents for their unwavering support in all my endeavors, for providing me with the best education they could and preparing me for my future. Thanks should also go to my sister for always listening to my problems, supporting and encouraging me during all this time.

My PhD time has also been inevitably influenced by the Covid19 pandemic that drastically reduced the opportunities to travel and interact with other people. But I thank my friends and housemates, for not letting me stop and encouraging me during all this time.

I would like to finish these acknowledgements by extending my gratitude to everyone else who, directly or not, consciously or not, has contributed to this thesis.

"What we know is a drop,
what we don't know is an ocean."

- Isaac Newton.

Contents

Acknowledgement **iii**

List of Figures **xv**

List of Tables **xvii**

List of Abbreviations **xxi**

Abstract **xxiii**

Resum **xxv**

Resumen **xxvii**

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 4

1.3 Structure of this thesis 4

2 Background **7**

2.1 Machine learning 7

2.2 Federated learning 9

2.2.1 Attacks to federated learning 11

2.3 Notions of fairness 11
2.4 Distribution of data 16
2.5 Types of data 17
2.6 Microaggregation 18
2.7 Gaussian mixture models 19
2.8 DBSCAN 24
2.9 Generative adversarial networks 26
2.10 Adversarial examples 27

3 Fair Detection of Poisoning Attacks in FL **29**
3.1 Introduction 29
3.2 Contributions 30
3.3 Fair detection of poisoning attacks in FL 32
3.4 Fair attack detection methods 35
3.4.1 **Fair attack detection based on microaggregation** 35
3.4.2 **Fair attack detection based on Gaussian mixtures and expectation-maximization** 39
3.4.3 **Fair attack detection based on DBSCAN** 40
3.5 Experimental results 41
3.5.1 Data sets, preprocessing, and baseline scenarios 41
3.5.2 Detection of malicious updates 48
3.5.3 Performance measures and discussion 50
3.6 Summary 57

4 Measuring Fairness in ML Models via CE **59**
4.1 Introduction 59

4.2 Measuring fairness via counterfactual examples . . . 61
4.3 Measuring fairness via counterfactual examples . . . 62
4.3.1 Measuring fairness in tabular data 62
4.3.2 Measuring fairness in image data 65
4.4 Empirical results 65
4.4.1 Experimental setup 66
4.4.2 Results 68
4.5 Summary 72

5 Conclusions **73**
5.1 Contributions 73
5.2 Future work 74
5.3 Publications 75

Bibliography **77**

List of Figures

2.1 Overview of federated learning 10
2.2 DBSCAN clustering 25
2.3 Architecture of a Generative Adversarial Network . . . 27

4.1 Training the generator of the counterfactual examples 66
4.2 ROC curves of the baseline and biased models on both data sets 69
4.3 Five examples of the counterfactual examples created by the proposed method. 72

List of Tables

3.1 Characteristics of data sets. 42

3.2 Performance measures for the Adult baseline scenario 44

3.3 Performance measures for the Athletes baseline scenario 46

3.4 Performance measures for the Marketing baseline scenario 48

3.5 Number of genuine minority updates misclassified as malicious. 50

3.6 Performance measures for Adult Income data set with different microaggregation parameters. 'B' stands for black (minority clients), 'NB' for non-black (majority clients), 'Att' for attacker clients. 52

3.7 Performance measures for Athletes data set with different microaggregation parameters. 'SA' stands for South Asian (minority clients), 'NSA' for non-South-Asian (majority clients), 'Att' for attacker clients. . . . 53

3.8 Performance measures for Bank Marketing data set with different microaggregation parameters. 'D' stands for divorced (minority clients), 'ND' stands for non-divorced (majority clients), 'Att' for attacker clients. . 53

3.9 Performance measures for Adult Income data set with different GMM parameters 54

3.10 Performance measures for Athletes data set with different GMM parameters 54
3.11 Performance measures for Bank Marketing data set with different GMM parameters 55
3.12 Performance measures for the Adult Income data set with different DBSCAN parameters 55
3.13 Performance measures for the Athletes data set with different DBSCAN parameters 56
3.14 Performance measures for the Bank Marketing data set with different DBSCAN parameters 56
3.15 Performance measures for Adult Income data set with [1] approach. 57

4.1 Architectures of the models used in the experiments for the Adult and the CelebA classification data sets. $C(3, 32, 3, 0, 1)$ denotes a convolutional layer with 3 input channels, 32 output channels, a kernel of size 3×3, a stride of 0, and a padding of 1; $MP(2, 2)$ denotes a max-pooling layer with a kernel of size 2×2 and a stride of 2; and $FC(18432, 2048)$ indicates a fully connected layer with 18,432 inputs and 2,048 output neurons. We used $ReLU$ as an activation function in the hidden layers; lr stands for learning rate. 67
4.2 Accuracy of the baseline and biased models evaluated on the full evaluation data set, on both data sets . . . 69
4.3 Accuracy of the baseline and biased model evaluated on the targeted versions of Adult and CelebA. 70

4.4 Two examples of records from the Adult data set. Symbol '"' indicates that the value of the attribute did not change during the creation of the counterfactual example. 71

List of Abbreviations

IoT	***I**nternet **o**f **T**hings*
AI	***A**rtificial **I**ntelligence*
ML	***M**achine **L**earning*
DL	***D**eep **L**earning*
FL	***F**ederated **L**earning*
iid	***i**dentically and **i**ndependently **d**istributed*
GDPR	***G**eneral **D**ata **P**rotection **R**egulation*
GMM	***G**aussian **M**ixture **M**odel*
DBSCAN	***D**ensity **B**ased **S**patial **C**lustering of **A**pplications with **N**oise*
NN	***N**eural **N**network*
CNN	***C**onvolutional **N**eural **N**network*
ANN	***A**rtificial **N**eural **N**network*
DNN	***D**eep **N**eural **N**network*
CE	***C**ounterfactual **E**xamples*
GAN	***G**enerative **A**dversarial **N**etwork*
EM	***E**xpectation **M**aximization*
MDAV	***M**aximum **D**istance to **A**verage **V**ector*
AUC	***A**rea **U**nder the **C**urve*
ROC curve	***R**eceiver **O**perating **C**haracteristic curve*
TPR	***T**rue **P**ositive **R**ate*
FPR	***F**alse **P**ositive **R**ate*

Abstract

The rise of the IoT and other distributed environments are causing an increase in the number of devices that constantly collect and exchange data. Machine learning models learn from these data to model concrete environments and problems and predict future events but, if the data are biased, they may reach biased conclusions. Such models can be used to make essential and life-changing decisions in a variety of sensitive contexts. Therefore, it is critical to make sure their predictions are fair and not based on discrimination against specific groups or communities, like those of a particular race, gender, or sexual orientation. Federated learning, a type of distributed machine learning, has become one of the foundations of the next-generation AI in distributed settings and needs to be equipped with techniques to tackle this grand and interdisciplinary challenge. Even if FL provides stronger privacy guarantees to the participating clients than centralized learning, in which the clients' raw data are collected in a central server, it is vulnerable to some attacks whereby malicious clients submit bad updates in order to prevent the model from converging or, more subtly, to introduce artificial biases in the models' predictions or decisions (poisoning). Poisoning detection techniques compute statistics on the updates sent by participants to identify malicious clients. A downside of anti-poisoning techniques is that they might lead to discriminating against minority groups whose data are significantly and legitimately different from those of the majority of clients. This would

not only be unfair but would yield poorer models that would fail to capture the knowledge in the training data, especially when data are not independent and identically distributed. In this work, we strive to strike a balance between fighting poisoning and accommodating diversity to help learn fairer and less discriminatory federated learning models. In this way, we forestall the exclusion of diverse clients while still ensuring the detection of poisoning attacks. Additionally, we explore the impact of our proposal on the performance of models on non-i.i.d local training data.

On the other hand, in order to develop fair models and verify the fairness of these models in the area of machine learning, we propose a method, based on counterfactual examples, that detects any bias in the ML model, regardless of the data type used in the model.

Resum

El desplegament massiu de l'IoT provoca que un nombre creixent de dispositius recullin i intercanviïn dades constantment. Els models d'aprenentatge automàtic aprenen d'aquestes dades per modelar entorns i problemes concrets, i predir esdeveniments futurs, però si les dades presenten biaixos, donaran lloc a prediccions i conclusions esbiaixades. Aquests models es poden utilitzar per prendre decisions essencials i que canvien la vida en diversos contextos sensibles. Per tant, és fonamental assegurar-se que llurs prediccions són justes i no es basen en la discriminació contra grups o comunitats específics, com els d'una raça, gènere o orientació sexual en particular. L'aprenentatge federat, una forma d'aprenentatge automàtic distribuït, ha esdevingut una de les bases de l'IA de nova generació en entorns distribuïts i li cal equipar-se amb tècniques per afrontar aquest gran repte interdisciplinari. L'aprenentatge federat proporciona millors garanties de privadesa als clients participants que no pas l'aprenentatge centralitzat, on les dades dels clients en clar són recollides en un servidor central. Tot i així, l'aprenentatge federat és vulnerable a atacs en els quals clients maliciosos presenten actualitzacions incorrectes per tal d'evitar que el model convergeixi o, més subtilment, per introduir biaixos arbitraris en les prediccions o decisions dels models (enverinament o *poisoning*). Les tècniques de detecció de l'enverinament calculen estadístiques sobre les actualitzacions per identificar clients maliciosos. Un desavantatge d'aquestes tècniques és que podrien conduir a la dis-

criminació de grups minoritaris, les dades dels quals són significativament i legítimament diferents de les de la majoria dels clients. Això no tan sols seria injust, sinó que també produiria models amb rendiment més baix que no podrien capturar tot el coneixement de les dades d'entrenament, especialment quan les dades no són independentment i idènticament distribuïdes. En aquest treball, ens esforcem per trobar un equilibri entre combatre els atacs d'enverinament i acomodar la diversitat, tot per a ajudar a aprendre models d'aprenentatge federats més justos i menys discriminatoris. D'aquesta manera, evitem l'exclusió de clients de minories legítimes i alhora garantim la detecció d'atacs d'enverinament. A més, explorem l'impacte de la nostra proposta en el rendiment de models sobre dades d'entrenament locals no idènticament i independentment distribuïdes.

D'altra banda, per tal de desenvolupar models justos i verificarne la imparcialitat en l'àrea d'aprenentatge automàtic, proposem un mètode basat en exemples contrafactuals que detecta qualsevol biaix en el model de ML, independentment del tipus de dades utilitzat en el model.

Resumen

El auge del IoT está provocando que un número cada vez mayor de dispositivos recojan e intercambien datos constantemente. Los modelos de aprendizaje automático aprenden de estos datos para modelar entornos y problemas concretos y predecir eventos futuros, pero si los datos están sesgados, darán lugar a predicciones sesgadas. Dichos modelos se pueden usar para tomar decisiones esenciales y que cambian la vida en una variedad de contextos sensibles. Por lo tanto, es fundamental asegurarse de que sus predicciones sean justas y no se basen en la discriminación contra grupos o comunidades específicos, como los de una raza, género u orientación sexual en particular. El aprendizaje federado, una forma de aprendizaje automático distribuido, se ha convertido en una de las bases de la IA de próxima generación en entornos distribuidos y debe estar equipado con técnicas para abordar este gran desafío interdisciplinario. Aunque el aprendizaje federado ofrece mayores garantías de privacidad a los clientes participantes que el aprendizaje centralizado, en el que los datos brutos de los clientes son recogidos en un servidor central, este es vulnerable a algunos ataques en los que clientes maliciosos envían malas actualizaciones para evitar que el modelo converja o, más sutilmente, para introducir sesgos artificiales en sus predicciones o decisiones (envenenamiento o *poisoning*). Las técnicas de detección de envenenamiento calculan las estadísticas de las actualizaciones para identificar a los clientes maliciosos. Una desventaja de las técnicas

contra el envenenamiento es que pueden llevar a discriminar a grupos minoritarios cuyos datos son significativamente y legítimamente diferentes de los de la mayoría de los clientes. Esto no sólo sería injusto, sino que produciría modelos más pobres que no capturarían el conocimiento de los datos de entrenamiento, especialmente cuando los datos no están igual e independientemente distribuidos (no i.i.d.). En este trabajo, nos dedicamos a lograr un equilibrio entre la lucha contra el envenenamiento y dar espacio a la diversidad para contribuir a un aprendizaje más justo y menos discriminatorio de modelos de aprendizaje federado. De este modo, evitamos la exclusión de diversos clientes y garantizamos la detección de los ataques de envenenamiento. Además, exploramos el impacto de nuestra propuesta en el rendimiento de los modelos con datos de entrenamiento locales no i.i.d.

Por otro lado, para desarrollar modelos justos y verificar la equidad de estos modelos en el área de ML, proponemos un método, basado en ejemplos contrafactuales, que detecta cualquier sesgo en el modelo de aprendizaje automático, independientemente del tipo de datos utilizado en el modelo.

Chapter 1

Introduction

1.1 Motivation

In this digital age, data are key social and economic assets. Sources of data often include edge devices, such as smartphones, IoT sensors attached to home and industrial equipment, or activities conducted at organizations or other entities, such as hospitals. However, collecting, sharing, or releasing these data lead to important privacy concerns. As companies and institutions collect growing amounts of data on their clients, they need to ensure that the privacy of the clients is not unjustifiably breached and that data protection policies and regulations are enforced throughout the data lifetime. The data collected from everyday objects like smartphones, smartwatches or fitness trackers almost invariably end up in centralized servers where they are aggregated, packaged and then, more often than not, shared with or sold to third parties. This may create privacy issues since these data sets can include a person's confidential data, such as her browsing history,

sexuality, political affiliation, and even medical conditions. These issues have led to the enactment of strict data protection laws, such as the European Union's General Data Protection Regulation (GDPR), which is binding for any organization operating in the EU. Organizations that process the personal data of EU citizens must comply with a number of principles laid out in the GDPR [2]. These principles include the following: (1) lawfulness, transparency and fairness, (2) purpose limitation, (3) data minimization, (4) storage limitation, (5) integrity and confidentiality, (6) accuracy, and (7) accountability.

Privacy concerns have become more prominent during the COVID-19 pandemic because, on the one hand, life has become more digital than before and, on the other hand, data collection aimed at controlling the spread of the pandemic might be perceived as a double-edged sword. While contact and mobility tracing are powerful instruments to preserve public health, their potential for misuse is high. More generally, the privacy expectations of individuals are confronted with the data-hungry artificial intelligence (AI) methods increasingly adopted by organizations. Specifically, for deep learning to be effective, vast amounts of data are required to train the models. Service providers collect data at massive scales for such training purposes. Traditionally, these large amounts of data have been stored in centralized databases and processed in central servers owned or hired by the service providers. Such central facilities need tight protection to prevent data leaks. Even if no leaks arise, central data collection and processing generate an asymmetry between the service provider and the customer, because the former accumulates a wealth of personal data on the latter.

Federated learning (FL) [3, 4] attempts to address some of these issues. FL is a machine learning technique that operates in a decentralized manner and allows the training of ML models with the help of a set of clients, each of whom privately owns a local data set. In FL, clients receive an initial global model from a service provider, of-

ten called the model manager. Then each client updates the received model based on their private local data and then uploads the model update to the model manager. The model manager aggregates the client updates to produce a new version of the global model. In this way, the global model can be iteratively improved and shared without the model manager ever accessing the private data of clients. The process iterates until the model converges.

The most usual situation in FL is that there is a crisp divide between the model manager, who orchestrates the different steps of the process, and the clients, who update the global model based on their private data. Yet, it is also conceivable to use federated learning in a peer-to-peer scenario, where each peer may be both a model manager of their own model and a client who updates the models of other peers. In any case, clients transmit to the model manager the bare minimum data to improve the model. This is inherently more privacy-preserving than centralized approaches in which client data are collected by a central server to build a machine learning model. Another advantage of FL is that the learning effort is distributed among the clients, instead of being centralized in a single entity.

For all its many advantages, FL is not free of issues. In particular, it is vulnerable to security attacks whereby malicious clients sabotage the learning process by sending bad model updates. These attacks may seek to prevent convergence to a model (Byzantine attacks) or to cause convergence to a flawed model whose output is determined by the attacker, at least for designated inputs (poisoning attacks). Poisoning attacks are described in [5], along with several solutions to thwart them. A well-known poisoning attack is label flipping, where the attacker is assumed to be able to flip the labels of a fraction of training points. In [6], more effective attacks are investigated, that achieve 100% accuracy on the attacker's task within just a single learning round.

Techniques to prevent security attacks compute statistics on the client updates to detect outlying values. Since abnormal and malicious behaviors are usually associated with outlying updates, these are filtered out as an attack prevention strategy when updating the global model. Even though this type of approaches are effective to prevent attacks, systematically rejecting outlying updates might also lead to unfair global models [7] if the outlying data correspond to a legitimately different minority. Apart from facing the unfairness issue, attack prevention countermeasures for FL often struggle to correctly treat non-i.i.d. (non-independently and identically distributed) data. Most research proposals assume that the clients' private data are i.i.d.

1.2 Contributions

In this thesis, we contribute mechanisms to reconcile security with fairness in FL on non-i.i.d data, and, further, we present a method based on counterfactual examples to detect any biases in trained ML models. Specifically:

1. We propose three methods to distinguish members of minority groups from attackers. Our first method is based on microaggregation, the second one uses Gaussian Mixture Models, and the third one is based on DBSCAN.

2. We propose a method, based on CE, that detects biases in trained ML models regardless of the data type, and in particular for image and tabular data.

1.3 Structure of this thesis

The rest of the thesis is organized as follows. Chapter 2 introduces background concepts of interest throughout the whole work. Chapter 3 presents our three methods for fair detection of attacks based on

microaggregation, Gaussian mixture models and DBSCAN, respectively. Chapter 4 presents the proposed methodologies to measure fairness in the ML models. Finally, conclusions and future research lines are discussed in Chapter 5.

Chapter 2

Background

This section presents the theoretical background of this work. Our aim is to recall the knowledge which the reader needs to understand the subsequent sections.

2.1 Machine learning

Machine learning is a type of artificial intelligence that allows software systems to learn and improve their performance without being explicitly programmed. It is based on the idea that systems can learn from data, identify patterns, and make decisions with minimal human intervention. There are three main types of machine learning, namely supervised learning, unsupervised learning, and reinforcement learning:

- In supervised learning, the system is trained on a labeled dataset, where the correct output is provided for each example in the training set. The goal is for the system to make predictions on new, unseen examples that are drawn from the same distribution

as the training set. Common applications of supervised learning include image classification, speech recognition, and natural language processing.

- In unsupervised learning, the system is not given any labeled training examples. Instead, it must discover the underlying structure of the data through techniques such as clustering or dimensionality reduction. Common applications of unsupervised learning include anomaly detection, data compression, and density estimation.

- In reinforcement learning, the system learns by interacting with its environment and receiving feedback in the form of rewards or penalties. The goal is for the system to learn a policy that will maximize the cumulative reward over time. Reinforcement learning has been used to develop game-playing agents, robotic control systems, and recommendation engines.

There are many algorithms and techniques used in machine learning, including linear models, decision trees, random forests, support vector machines, neural networks, and deep learning. These methods can be used alone or in combination to solve a wide range of problems.

One of the key challenges in machine learning is choosing the right algorithm and set of hyperparameters for a given task. It is also important to have a sufficient amount of high-quality data, as the performance of most machine learning algorithms depends heavily on the data they are trained on. Access to these data and their quality are important factors that can impact the privacy of data respondents and the fairness of the resulting models.

Another crucial component of ML is the problem of how a computer program determines which of its outcomes were appropriate and which contained mistakes. The most important factors for evaluat-

ing the performance of an ML model are accuracy, precision, recall, F-measure, confusion matrix, and the misclassification rate.

Our contributions are aimed at measuring fairness in ML models and attaining fair and robust classification models via supervised learning. Although we focus on deep neural networks trained in distributed environments, our contributions are also applicable to other optimization-based algorithms, such as linear regression models. Techniques to detect outliers and/or attackers in federated learning are usually based on unsupervised learning algorithms (*e.g.*, clustering algorithms).

2.2 Federated learning

In an FL scenario, a model manager initializes a learning model, such as a neural network, with weights θ^0, loss function L and learning rate ρ. Other hyper-parameters may apply, such as dropout rate, decay, or momentum, but we restrict to a general model using stochastic gradient descent (SGD). The model manager may or may not pre-train the model with available public or private data already in her possession. Each of the m clients, whose devices are called client or edge devices, has access to a data set $D_u = \{x_i^u, y_i^u\}_{i=1}^{n_u}$ of size n_u. The total size of the available data is $n = \sum_{u=1}^{m} n_u$. At epoch t —where epoch means learning iteration—, the model manager sends the current global model θ^{t-1} to all clients; these use their devices to train local models from the global model using their respective private data sets D_u; then, clients send their respective updates δ_u^t to the model manager, who updates the global model θ^{t-1} into θ^t by averaging the updates, possibly subject to a parameter η which regulates the model substitution rate. Additionally, a vector $\vec{\alpha}$ can be used to adjust the weight of each client's contribution in the federated aggregate. The intuition of FL is depicted in Figure 2.1.

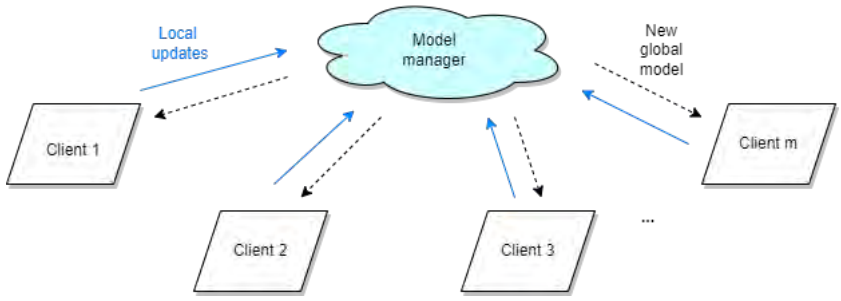


Figure 2.1: Overview of federated learning

A possible choice is for all components of $\vec{\alpha}$ to be $1/m$, in which case all clients have the same influence. If the client data sets are of very different sizes, an alternative choice giving weight $\alpha_u = n_u/n$ to the u-th client might make sense. Also, in case a client is found malicious, her α_u value can be set to 0 to exclude her contributions from the aggregate. This approach to aggregating updates is the most usual one and is known as federated averaging (FedAvg, [3]). See its pseudocode in Protocol 1.

Protocol 1: FedAvg

Initialize model parameters θ^0 and distribute them among clients;
while *termination condition not met* **do**
  foreach *client* $u \in S$; // S: set of m clients
  do
    $\delta_u^t \leftarrow \frac{\rho}{n_u} \sum_i \nabla L(\{x_i^u, y_i^u\}, \theta^{t-1})$;
    Send δ_u^t to the model manager;
  end
  $\vec{\alpha} \leftarrow AttackerDetection(\theta^{t-1}, \{\delta_u^t\})$;
  $\theta^t \leftarrow \theta^{t-1} + \eta \sum_u \alpha_u \delta_u^t$;
  Distribute θ^t among clients;
end

2.2.1 Attacks to federated learning

Despite the fact that FL appears secure, it cannot by itself fully provide the levels of privacy and security required by current distributed systems. The following are major attacks and threats that are relevant to FL settings [8, 9]:

- Byzantine attacks: The adversary's goal in these attacks is to stop the convergence and effectiveness of the global model.
- Poisoning attacks: In these attacks, malicious users inject fake training data with the aim of corrupting the learned model. They can be classified into two types:
  - Data poisoning attacks: They involve tampering with the client's training data to produce undesirable outcomes. One common example is the label-flipping attack, where the labels of honest training examples of one class are flipped to another class while the features of the data are kept unchanged.
  - Model poisoning attacks: They aim to poison local model updates before sending them to the server or insert hidden backdoors into the global model. One example of these attacks are backdoor attacks.
- Inference attacks: In these attacks, a user is able to infer sensitive information about a database by analyzing locally computed updates.

2.3 Notions of fairness

ML systems use sensitive attributes, which cannot be ignored or removed because they are correlated with other features and help understand the data. If not properly handled, these sensitive attributes

may cause ML models to be biased in some way [10]. Model bias stems from biased training data and/or from the decision algorithms themselves. In [11], the authors list some potential causes of bias in ML systems:

- A first type of bias in ML is inherited from humans, most likely the designers or programmers of the ML system. This type of bias typically surfaces when unfair judgments are made because the designer/programmer is influenced by a characteristic that is actually irrelevant to the matter at hand, typically a discriminatory preconception about members of a vulnerable group. Thus, one form of bias is a learned cognitive feature of a person, often not made explicit[12].
- A second form of bias is also human-inherited, due to the fact that human judgments are often not rational [13].
- A third form of bias is present in data that exhibit systematic error, *e.g.*, "statistical bias". ML trained on such data would not only fail to recognize the bias, but would incorporate it in its decisions.

Research on bias-aware ML techniques can be separated into three stages: pre-processing, in-processing, and post-processing, which correspond to a typical ML model construction pipeline.

- Pre-processing: These techniques try to transform the training data so that the underlying discrimination is removed *i.e.*, the prediction is independent of the sensitive attribute [14, 15, 16]. Often, fairness criteria will be used to filter out individual records in the training data sets that do not conform to such criteria. In cases in which the whole dataset is accessible and can be modified, such as in centralized learning scenarios, then pre-processing mechanisms can be used.

- In-processing: Such methods try to modify or adapt the machine learning algorithms in order to account for and remove discrimination during the model training process, according to some fairness criteria. In cases in which the training data are not accessible, but the learning procedure for a machine learning model can be altered (such as in an FL scenario), then in-processing can be used, either by incorporating changes into the objective function or imposing a constraint. Our first contribution falls in this category.
- Post-processing: These methods are performed after training by accessing a holdout set that was not involved during the training of the model. If the learning process must be treated as a black box without any ability to modify the training data or the learning algorithm, then only post-processing can be used. In post-processing, the labels initially assigned by the black-box model get reassigned based on some fairness criteria.

To avoid the existing bias in the ML systems and achieve fairness, numerous fairness notions have been defined in the literature. There have essentially been two sorts of fairness definitions: group fairness notions, which make up the vast majority of them, and individual fairness notions. The first type is about partitioning a population into groups based on protected attributes (*e.g.*, gender or ethnicity) and aims to achieve some statistical measure to be equal across groups. Individual fairness is based on the principle that any two individuals who are similar should be classified similarly [17]. Similar individuals are determined by the choice of the metric on individuals. Although [17] does not distinguish between discriminatory and sensitive attributes, it is possible to encode the distinction into the chosen metric.

To define these metrics, we begin with some notation:

Let $A \in \{0, 1\}$ be a sensitive attribute (*e.g.*, gender or race), $Y \in \{0, 1\}$ be the target decision variable and $\hat{Y} := \hat{y}(X, A) \in \{0, 1\}$ be

the binary predictor produced by an ML algorithm as a prediction of Y.

We assume X, A, Y are generated from an underlying distribution D, *i.e.*, $(X, A, Y) \sim D$. We will consider the binary classification problem with a single sensitive attribute for simplicity, *i.e.*, a beta-binomial distribution D over X. When working with multiple sensitive attributes, a Dirichlet-multinomial distribution can be used.

We denote individuals for whom $y = 1$ as being members of the "positive" class and individuals for whom $y = 0$ as being members of the "negative" class. For instance, if an algorithm predicts whether an individual will get a loan, y will be 1 for the individuals who are granted the loan (and who are therefore members of the positive class) and 0 for individuals who are denied it (and are thus members of the negative class). Whereas in this scenario the "positive" outcome $y = 1$ corresponds to the desirable outcome, this need not always be the case.

With this notation in hand, we now proceed to define and discuss several fairness criteria that commonly appear in the literature [18, 19, 17]. All of the criteria presented below can also be assessed conditionally by further conditioning on some covariates in the feature vector X.

Traditional fairness-aware classification aims to learn a mapping $f : X \to Y$ that accurately maps instances x to their correct classes without discriminating between protected and non-protected groups. The discrimination is assessed in terms of some fairness measure. It is very important to select the right type of fairness; otherwise, the wrong metric can lead to some harmful decisions. We start with metrics for group fairness:

- *Demographic parity or statistical parity.* Regardless of the sensitive attribute, the probability of a positive outcome should be the same for each group:

$$P(\hat{Y} \mid A = 0) = P(\hat{Y} \mid A = 1).$$

- *Predictive parity, a.k.a. outcome test.* A classifier satisfies this definition if the subjects in the protected and unprotected groups have equal positive predictive value (PPV), that is:

$$P(Y = 1 \mid \hat{Y} = 1, A = 0) = P(Y = 1 \mid \hat{Y} = 1, A = 1).$$

- *False positive error rate balance, a.k.a. predictive equality.* A classifier satisfies this definition if the subjects in the protected and unprotected groups have an equal false positive rate (FPR):

$$P(\hat{Y} = 1 \mid Y = 0, A = 0) = P(\hat{Y} = 1 \mid Y = 0, A = 1).$$

- *False negative error rate balance, a.k.a. equal opportunity.* A classifier satisfies this definition if the subjects in the protected and unprotected groups have an equal false negative rate (FNR):

$$P(\hat{Y} = 0 \mid Y = 1, A = 0) = P(\hat{Y} = 0 \mid Y = 1, A = 1).$$

  The error rate balance is also closely connected to the notions of equalized odds and equal opportunity as introduced in [20].

- *Equalized odds, a.k.a. disparate mistreatment.* This is a slightly more complex version of the above-mentioned equal opportunity. A classifier satisfies this metric if the subjects in the protected and unprotected groups have an equal true positive rate (TPR) and equal FPR. This means that the probability of a person in the positive class being correctly assigned a positive outcome and the probability of a person in a negative class being incorrectly assigned a positive outcome should both be the same for the protected and unprotected group members:

$$P(\hat{Y} = 1 \mid A = 0, Y = y) = P(\hat{Y} = 1 \mid A = 1, Y = y),$$
$$y \in \{0, 1\}.$$

Equal opportunity and equalized odds both assume that the training data are correct.

- Calibration. Outcomes should be independent of the sensitive attributes conditional on the risk score. The notion of calibration features among the widely accepted and adopted standards for empirical fairness assessment. While predictive parity and calibration look like very similar criteria, well-calibrated scores can fail to satisfy predictive parity at a given threshold.

We now review criteria based on individual fairness.

- *Fairness through awareness.* Any two individuals who are similar with respect to a similarity (inverse distance) metric defined for a particular task should receive a similar outcome.
- *Fairness through unawareness.* In this case, it is assumed that if we are unaware of sensitive attributes when making decisions, our decisions will be fair. This notion is consistent with disparate treatment, which requires not to use the sensitive attribute. The main limitation of this criterion is that there can be many highly correlated features that are proxies of the sensitive attribute. Thus, just removing the sensitive attribute may not be enough.

2.4 Distribution of data

The fact that training data are often non-i.i.d. among clients is a challenge faced by FL that also has fairness ramifications. In this

setting, the distribution of the local data of each client is not representative of the distribution of the global data (those that would be obtained if all the clients' local data were pooled). Non-i.i.d. data make it difficult for FL to learn models that are as good as those obtained with centralized learning.

Non-i.i.d.-ness can be measured by the differences between the gradients obtained by the clients on their respective local data and the gradients of the global model. For a non-negative real value δ [21], δ-non-i.i.d. data in FL can be characterized with the condition

$$||\nabla f_u(\theta) - \nabla f(\theta)|| \leq \delta, \forall u, \tag{2.1}$$

where θ are the model parameters, $\nabla f_u(\theta)$ are the gradients obtained by client u after a local training phase, and $\nabla f(\theta)$ are the global model gradients. Expression (2.1) limits to δ the difference in the distributions of the gradients of individual users and those of the global model.

2.5 Types of data

When dealing with data sets, the data type is crucial in determining the pre-processing strategy that would yield the right results for a particular set, or the class of statistical analyses that should be applied to acquire those results. In tabular data sets consisting of records and attributes, there are two types of attributes: qualitative and quantitative. Qualitative data are further classified into nominal or ordinal, whereas quantitative data are either discrete or continuous. We describe these types next:

- Qualitative or categorical data describes the object under consideration using a finite set of discrete classes. It means that this type of data cannot be counted or measured easily using numbers and therefore divided into categories. Subtypes:

  - Nominal data in statistics are not quantifiable and cannot be measured through numerical units, *e.g.*, gender, marital status, or ethnicity.
  - Ordinal data express some kind of sequential order by indicating a position within a scale. Although ordinal values can be viewed as numbers, arithmetic operations on them do not make sense in general (*e.g.* averaging two education levels can yield a decimal number that does not correspond to any education level).

- Quantitative or numerical data can be expressed in numerical values, which can be operated arithmetically. Subtypes:
  - Discrete data take values in a countable and typically finite range, *e.g.*, age.
  - Continuous data can take any value within a range in the real numbers, *e.g.*, the height of a person.

2.6 Microaggregation

Microaggregation is a perturbative method for statistical disclosure control of quantitative microdata. It is used to protect the privacy of individuals by aggregating similar data points. It works by grouping together data points that are similar and then replacing each point with the average of the group. This process makes it difficult for an individual to be identified from the data set, as their data is mixed in with the data of others. By aggregating the data points, microaggregation helps ensure that only the necessary data is shared and that individuals are protected from having their information revealed. For example, instead of publishing exact age or salary data, microaggregation can be used to group individuals into ranges or categories. This ensures that personal data is not revealed, while still providing meaningful insights. The method was introduced in [22] for numerical

data, and [23, 24] extended it for categorical data. Microaggregation is based on two steps:

1. *Partition.* The records in the original data set are partitioned into a number of clusters, each of them containing at least k records (the minimum cluster size) and no more than $2k - 1$ records. To minimize information loss in the following step, records in each cluster should be as close to one another as possible.

2. *Aggregation.* An aggregation operator is used to compute the centroid (*e.g.* the arithmetic mean value) of the records in each cluster. Then the records in the cluster are replaced by their centroid.

2.7 Gaussian mixture models

Consider a data set $\{x_j\}_{j=1}^m$ of points in $\mathbb{R}^d$, representing client-provided model updates, where each dimension would correspond to a continuous demographic attribute.

Recall that the maximum-likelihood estimates of the mean μ and covariance Σ parameters of a multivariate Gaussian model[1]

$$q_{\mu,\Sigma}(x) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma}} e^{-\frac{1}{2}\|x-\mu\|^2_{\Sigma^{-1}}}$$

match their respective sample moments. This means that we may view these moments as an expectation and covariance under a probability

[1]We adhere to the standard notation $\|x\|_G$ for the norm corresponding to an inner product $\langle x, y\rangle_G$ with Gramian matrix G, that is, $\langle x, y\rangle_G = x^{\mathrm{T}} G y$ and $\|x\|_G = x^{\mathrm{T}} G x$. More explicitly, this general inner product is the quadratic form $\langle x, y\rangle_G = \sum_{i,j} g_{ij} x_i y_j$ of the vector components. As the Gramian is customarily positive definite, its Cholesky decomposition $G = LL^{\mathrm{T}}$, with L lower triangular, enables us to write $\langle x, y\rangle_G = \langle Lx, Ly\rangle$ in terms of the canonical inner product, and similarly for $\|x\|_G = \|Lx\|$. This gives an efficient method to compute the probabilities of a large number of multivariate Gaussian samples.

mass function p_X representing the relative frequencies of the observed samples, typically uniform and therefore equal to $1/m$ in the case of continuous alphabets, with samples that may now be construed as a random vector X distributed according to p_X. Precisely, the maximum likelihood estimates are

$$\hat{\mu}_{\mathrm{maxL}} = \mathrm{E}_{p_X} X = \tfrac{1}{m} \sum_{j=1}^{m} x_j \text{ and}$$

$$\hat{\Sigma}_{\mathrm{maxL}} = \mathrm{Cov}_{p_X} X = \tfrac{1}{m} \sum_{j=1}^{m} (x_j - \mu)(x_j - \mu)^{\mathrm{T}}.$$

Under the simple assumption of uncorrelated, identically distributed dimensions and after convenient scaling, we may concordantly restrict the covariance parameter Σ to be the identity matrix, which finally leads us to conclude that the probability density $q_{\mu,I}(x)$ of any given point x is a decreasing function of the Euclidean norm $\|x - \mu\|$ with respect to the mean parameter μ. Simply put, the higher the distance, the lesser the probability. Incidentally, the ever-present optimization criterion in engineering of minimization of mean square error may be often understood as the disguised maximization of likelihood under a normal model.

On the other hand, without the symmetry restriction on Σ, our criterion becomes formally equivalent to the identification of outliers by means of their Mahalanobis distance to the centroid μ of the data set, a distance that fully characterizes the probability density of a given observed sample. This distance will thus offer no loss in statistical information and constitute an adequate metric for outlier detection, provided that the Gaussian model is deemed appropriate.

This probabilistic estimation perspective lends itself to the sophistication of outlier detection by means of refining our choice of parametric estimation of probability density functions. In the context of

client model updates, maximum-likelihood fitting of multimodal probability densities will enable us to identify both natural demographic clusters and anomalous dispersions. With such multimodal methods, outliers will not merely be points far from a single centroid, but statistically unlikely combinations of weights and biases away from several clusters, characterized with overlapping, smooth densities around several centroids, automatically identified as we maximize the likelihood of the parametric fit.

A particularly strong candidate, recognized by its suitability in a wide variety of scenarios, is the Gaussian mixture model (GMM), a probabilistically weighted combination of Gaussian components, each with its own mean and covariance. Representing the observed samples and the component index as random variables X and Y, respectively (one continuous, the other discrete), the GMM is a generative model with observation X and hidden label Y in which X given $Y = y$ is conditionally normally distributed with mean $\mu(y)$ and covariance $\Sigma(y)$. The weights of the components are the marginal probabilities $q_Y(y)$ of the component indices y. Therefore,

$$\begin{aligned} q_X(x) &= \mathrm{E}_Y\, q_{X|Y}(x|Y) = \sum_y q_Y(y) q_{X|Y}(x|y) \\ &= \sum_y q_Y(y) \frac{1}{\sqrt{(2\pi)^d \det \Sigma(y)}}\, e^{-\frac{1}{2}\|x-\mu(y)\|^2_{\Sigma(y)^{-1}}}. \end{aligned}$$

This more expressive model comes with the challenge of selecting not only the means and covariances but also the weights, jointly and in accordance with the maximum-likelihood criterion.

The joint estimation of the mean, covariance, and weight parameters for the GMM is typically accomplished via the EM algorithm [25, 26, 27], guaranteed to monotonically converge to local optima under mild conditions [28, 29, 30]. Because of local convergence, the

end result is ultimately driven by the initialization [31] of the algorithm. A simple, common choice for the mean parameters resorts to the k-means method, which itself requires initialization, for instance in the form of the k-means++ method [32]. As is usually the case for model hyperparameters, k-fold cross-validation permits selecting an appropriate number of mixture components algorithmically. Cholesky factorization of the covariance matrices may help efficiently compute the probability densities involved. The maximum likelihood objective may be logarithmically calculated as a cross-entropy.

The EM algorithm is an iterative method to find the maximum-likelihood estimates for model parameters in the presence of latent (hidden) variables.

To start, we need to pick the hyperparameter K [2] that decides how many Gaussians we want in our model. The algorithm has the following two steps:

- E-step: For each point x_i, compute the probability that it belongs to cluster k. This can be written in terms of probability and using Bayes' theorem

$$r_{ik} = \frac{\mathcal{N}(x_i; \mu_k, \Sigma_k) \cdot \pi_k}{\sum_{j=1}^{k} \mathcal{N}(x_i; \mu_j, \Sigma_j) \cdot \pi_j}.$$

  This is just the normalized probability of each point belonging to one of the K Gaussians weighted by the mixture distribution (π_k).
  The value r_{ik} will be higher when the point is assigned to the right cluster and lower otherwise.

- M-step: After the E-step, we go back and update the π, μ, and

[2] This is non-trivial since it is not known how many clusters we have. One could vary K until obtaining useful results.

Σ values. We want to maximize the likelihood function across all these values. These are updated in the following manner:

1. The new density is defined by the ratio of the number of points in the cluster and the total number of points:

$$\pi_k = \frac{\text{number of points assigned to cluster}}{\text{total number of points}}$$

$$= \frac{1}{m} \sum_i r_{ik}.$$

Next, we need to estimate the Gaussians.

2. The mean and the covariance matrix are updated based on the values assigned to the distribution, in proportion to the probability values for the data point. Hence, a data point that has a higher probability of being part of that distribution will contribute a larger portion. This yields the following updates for the mean vector μ_k and the covariance matrix Σ_k:

$$\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}};$$

$$\Sigma_k = \frac{\sum_i r_{ik}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i r_{ik}}.$$

Based on the updated values generated from this step, we calculate the new probabilities for each data point and update the values iteratively. This process is repeated in order to maximize the log-likelihood function, which can be defined as

$$\ell(\pi, \mu, \Sigma) = \sum_i \log(\sum_k \mathcal{N}(x_i; \mu_k, \Sigma_k^2) \cdot \pi_k).$$

2.8 DBSCAN

In [33], a new notion called density-based cluster was introduced, whereby clusters of any shape can be identified in data sets containing noise and outliers. The goal of density-based spatial clustering of applications with noise (DBSCAN) is to identify dense regions, which can be measured by the number of objects close to a given point.

DBSCAN requires two parameters:

- Epsilon (Eps): maximum radius of the neighborhood around a point.
- Minimum points ($MinPts$): minimum number of points in the Eps-neighborhood of a point. This Eps-neighborhood of a point p can be defined as $N_{Eps}(p) = \{q \in D | dist(p, q) \leq Eps\}$, where D is the total set of points.

Any point p in the data set with a neighbor count at least $MinPts$ is marked as a *core point*. A point p is a *border point* if the number of its neighbors is less than $MinPts$, but it belongs to the Eps-neighborhood of some core point. If a point is neither a core point nor a border point, then it is called a *noise point* or an outlier. Figure 2.2 illustrates these points.

To understand the DBSCAN algorithm, the following definitions are needed once parameters Eps and $MinPts$ have been set:

- Direct density-reachability: A point q is directly density-reachable from a core point p if $q \in N_{Eps}(p)$.
- Density reachability: A point q is density-reachable from a core point p if there is a set of core points $p_1, \cdots, p_{n-1}$, such that p_{i+1} is directly density-reachable from p_i for $i = 1$ to $n - 1$, where $p_1 = p$ and $p_n = q$.

- Density connected: A point q is density-connected to a point p if there is a point s such that p and q are density-reachable from s.

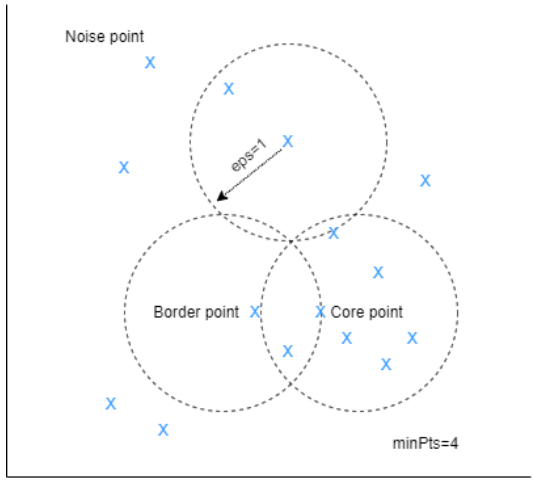


Figure 2.2: DBSCAN clustering

The DBSCAN algorithm is presented in Algorithm 2 and works as follows. First, we arbitrarily select a point x in the data set X. Then, we retrieve all points that are density-reachable from x with respect to Eps and $MinPts$. If x is a core point, form a cluster with the retrieved points. Otherwise, mark x as a noise point. Iterate through the remaining unvisited points in the data set.

Those points that do not belong to any cluster are treated as outliers or noise. One limitation of DBSCAN is that it is sensitive to the choice of parameters, especially if clusters have different densities. If Eps is too small, a cluster whose point-to-point distances are greater than Eps will be taken as noise. In contrast, if Eps is too large, clusters whose inter-cluster distance is less than Eps may be merged together.

Algorithm 2: DBSCAN

```
Input: X, ϵ, minPts
Output: A set of clusters
foreach unvisited point x ∈ X do
    mark x as visited
    N ← find neighbors N of x
    if |N| < minPts then
        label(x) ← noise
    end
    c ← next cluster label
    label(x) ← c
    foreach x′ ∈ N do
        if x′ not visited then
            mark x′ as visited
            N′ ← find neighbors N′ of x′
            if |N′| < minPts then
                N = N ∪ N′
            end
        end
    end
    if x′ ∉ c then
        c ← c ∪ x′
    end
end
```

2.9 Generative adversarial networks

GANs are a type of ML algorithm used to generate new synthetic data. They work by having two neural networks work together: a generator network and a discriminator network. The generator network creates data, while the discriminator network evaluates them and determines their authenticity. The two networks then compete in an adversarial process, with the goal being for the generator network to create data that are indistinguishable from the "real" data in the training set.

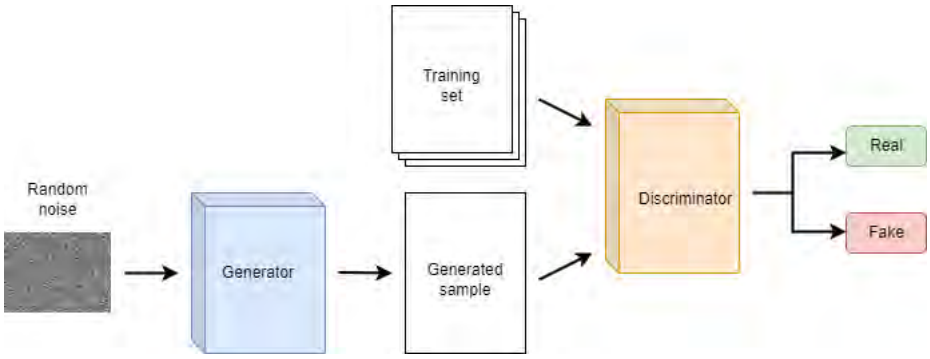


Figure 2.3: Architecture of a Generative Adversarial Network

GANs were first introduced by Goodfellow *et al.* in [34]. They have a variety of applications, but their most common usage is the generation of synthetic images.

The generator G takes as input a random noise vector z and outputs an image $z = G(z)$. The discriminator D receives as input either a training image or a synthesized image from the generator and outputs a probability distribution $D(X)$ over possible image sources. The discriminator is trained to maximize the log-likelihood it assigns to the correct source:

$$L = E_x[log(D(x))] + E_z[log(1 - D(G(z)))].$$

The generator is trained to minimize the second term in the expression above.

2.10 Adversarial examples

Adversarial examples are special inputs to machine learning models constructed intentionally to confuse the model so that it misclassifies the given input [35]. Adversarial examples can cause serious issues in terms of security, but also safety. A noteworthy example of the potential safety implications of adversarial examples is to cause ML

models guiding self-driving cars to misclassify traffic signals. Adversarial examples can be used to attack any machine learning system, from image recognition to fraud detection. Adversarial examples are closely related to counterfactual examples: in [36], counterfactuals are described as adversarial examples in which the changes introduced to the original inputs are interpretable by humans. Thus, the misclassification of counterfactual examples can provide some insights into how the model works internally. Since the changes to the original input are interpretable, the reasons for giving a different classification result than what was expected are also interpretable. Therefore, counterfactual examples are a mechanism for ML interpretability or explainability.

Theoretically, adversarial examples are typically defined as inputs x', where the differences between x' and non-adversarial inputs x are minimal under a distance metric $d(\cdot)$ (*e.g.*, $d(\cdot)$ can be the L_p distance), whilst fooling the target model f. Generally, adversarial examples seek to satisfy

$$d(x', x) < \epsilon \text{ and } \hat{y}(x') \neq \hat{y}(x),$$

where ϵ is a small constant that bounds the magnitude of the perturbations, and $\hat{y}(\cdot)$ denotes the predicted label of a classifier model.

Chapter 3

Fair Detection of Poisoning Attacks in Federated Learning on Non-i.i.d. data

3.1 Introduction

In this work, we explore the tensions between data privacy, partially achieved by the use of federated learning, model robustness against label-flipping attacks, and fairness in classification tasks. As outlined previously, federated learning is vulnerable to poisoning attacks, and in particular to label-flipping attacks. Mechanisms to protect against these attacks are based on filtering outlying model updates. However, it is not known *ex ante* whether these outliers come from attackers or from benign clients whose data are genuinely different from those of the majority of clients, either because the data are

non-i.i.d. or because those benign clients belong to a minority group. Thus, attack protection mechanisms in the literature provide model robustness at the cost of classification fairness. We propose mechanisms to better model the updates provided by the clients, by finding similarities among outliers that can indicate the existence of minority groups and by only discarding those updates which are completely isolated. Other tensions among desirable properties of ML models are explored in [37].

3.2 Contributions

No honest client ought to be discriminated in FL due to the genuine attribute values of the person represented by the client when interacting with other clients or the model manager. In other words, all honest clients should be able to contribute to the training process, because this is the best way to obtain not only fair but also good-quality decision models. Note that ignoring minority groups in the training process decreases the quality of the learned global model.

However, as introduced above, being inclusive with respect to minorities often clashes with the ability to detect attacks against FL models. A common detection approach is for the model manager to compute the Euclidean distance between each of the client-provided model updates and the average of such updates, and then discard as potentially malicious any update too far from the average, according to some threshold or rule. In the presence of non-i.i.d. data, or when some of the clients represent individuals from minority groups, this approach might lead to treating genuinely different individuals as potential attackers. This would not only be unfair to minorities, but would result in a biased model.

Our aim is to strike a balance between anti-poisoning and diversity accommodation. By including diverse clients, we aim at making it possible to learn less discriminatory machine learning models. Thus,

the contributions in this chapter are:

- A first method to distinguish minority members from attackers based on microaggregation. [22]. Clients who identify themselves as belonging to a minority group announce some relevant attributes to their peers, such as their *gender*, their *sexual orientation*, or their *ethnicity*. From these attributes, the peers carry out a clustering process via collaborative microaggregation. In this way, the majority group and the minority groups are clustered separately. After that, an FL model is trained for each cluster. Since peers have been already clustered according to some of their attributes, outliers within clusters are likely to be attackers because their updates are unusual even for a minority group. Finally, a weighted aggregation of the different cluster-level models is computed, where the weights are proportional to the sizes of the clusters.

- A second method where we use Gaussian mixture models to characterize the distribution of the client-provided updates and classify outliers in a more sophisticated way than just relying on the distance to an average client update. In the presence of minority groups that differ from the majority group in some attributes, but that are homogeneous within themselves, we expect this approach to label honest individuals from minority groups as non-malicious.

- A third method predicated on density-based clustering. Specifically, we use the DBSCAN algorithm to identify clusters of any shape among the client updates. In the FL setting the assumption is that the objectives for all clients approximate the global objectives. However, this is not the case with non-i.i.d. data. DBSCAN can help correctly characterize the distribution of updates from clients with non-i.i.d. data.

3.3 Fair detection of poisoning attacks in FL

Several solutions have been proposed in the literature to detect attacks or abnormal behaviors in machine learning [38, 39]. In the specific context of FL, where the model manager has access to the individual updates from the clients, the following classes of attack detection methods have been proposed:

- *Detection of malicious clients via model metrics.* The model manager can reconstruct the individual updated models for every client u and compare the model performance metrics, such as accuracy or loss, against a validation data set with respect to the model obtained by aggregating all updates except that of client u. The model manager can mark it as anomalous and possibly discard any client updates that degrade the model performance according to some rule or threshold. Note that the model manager needs a suitable validation data set, which may not always be available in the FL scenario. Moreover, re-evaluating the model accuracy after each update is extremely costly, and introduces an unacceptable overhead in the FL process.

- *Detection of malicious clients via update statistics.* A very common and natural approach for the model manager is to observe the statistics of the magnitudes of the updates [40]. The model manager can compute how much the distributions of distances in successive iterations change, for example using the Kullback-Leibler divergence metric. In a scenario with colluding malicious clients, these might have enough influence on the computed centroid to render the previous countermeasures ineffective. To gain additional protection, the model manager can compute the centroid as a median rather than as an average. The median is more robust in front of outlying updates submitted by malicious clients. More costly alternatives are presented in [41], where anomalous clients are detected by generating low-dimensional

surrogates of model weight vectors, and in [42], in which a spectral anomaly detection is performed by the model manager.

A decentralized approach based on update statistics is presented in [43]. A client's model update is considered legitimate if its distance to the centroid of all client updates is roughly between the first and the third quartiles of the set of distances between all client updates and the centroid.

- *Krum aggregation.* The authors of [44] propose an aggregation function that is resilient against a certain number of malicious clients. This function is called Krum. The authors show that averaging does not stand Byzantine attacks, while Krum does. An important advantage of Krum is its (local) time complexity $\mathcal{O}(m^2 \cdot d)$, which is linear in the dimension of the updates. The authors also evaluate a variant of Krum, Multi-Krum, which interpolates between Krum and averaging.

- *Coordinate-wise median.* In [40], a median-based distributed algorithm is proposed that selects the coordinate-wise median instead of the coordinate-wise average. Since the median is a more robust statistic than the mean (*i.e.*, it is less influenced by outliers), the obtained global model is less influenced by potential malicious peers.

- *Coordinate-wise trimmed mean.* Also in [40], a second distributed algorithm is proposed, called coordinate-wise trimmed mean, that can achieve order-optimal error rate under weaker assumptions than the coordinate-wise median algorithm.

In the approaches above, updates that are statistical outliers departing from a global aggregate model are considered malicious. However, it may also be the case that honest clients have genuinely outlying local data and therefore generate genuinely outlying updates. This may be a consequence of the clients belonging to a minority group.

There is a growing interest in the development of fair models for machine learning. In federated settings, [45] propose DITTO, a multi-task learning framework, to address the competing constraints of accuracy, fairness, and robustness in FL. The authors of this work define fairness as each client achieving equal test performance on the federated model. In [46], a collaborative fair federated learning framework (CFFL) is proposed. In this work, fairness is achieved by adjusting the performance of the models allocated to each participant based on their contributions. Also, [47] aims at achieving group fairness in FL. The authors mimic the centralized fair learning setting by very frequently exchanging information for each local update, rather than for each round of local training.

Several works have dealt with non-i.i.d. data in a federated learning setting. As prior studies show, decentralized learning algorithms lose significant model accuracy in the non-i.i.d. setting. In [1], the authors propose a strategy to improve training on non-i.i.d. data by creating a small subset of data which are globally shared among all the edge devices. However, this relies on a substantial amount of public data being available for a given task. In [48], the authors propose federated augmentation (FAug), where clients collectively train a generative model, thereby augmenting their local data towards yielding an i.i.d. data set. The authors of [49] analyze the convergence of federated averaging on non-i.i.d. data and establish a convergence rate of $\mathcal{O}(\frac{1}{T})$ for strongly convex and smooth problems, where T is the number of rounds of local SGD updates.

The commonly used FedAvg [3] makes no special adjustments when encountering non-i.i.d data and therefore suffers from a deterioration in the accuracy of FL [50]. This performance degradation can chiefly be attributed to weight divergence of the local models resulting from non-i.i.d data.

A systematic study on local model poisoning attacks to FL is of-

fered in [51], including the attacks mentioned above. The authors simulate FL with different non-i.i.d. training data distributions. They generalize two defenses against data poisoning attacks, which are effective in some cases but not in others; this highlights the need for new defenses against local model poisoning. For further background on attacks and defenses in FL, see the surveys [52] and [53]. The methods we introduce in the next sections depart from the state of the art in that they aim at properly managing updates originated by clients in minority groups.

3.4 Fair attack detection methods

To evaluate the performance of the trained model, the fairness notions of Section 2.3 can be readily applied to centralized model training. However, with non-i.i.d data in FL, low levels of fairness are likely.

To address this problem, one must pay attention to the distribution of outlying updates. If these are concentrated, then this could signal a minority, rather than attackers. Fairness comes from differentiating attackers from minorities, so that the latter can avoid rejection of their updates.

3.4.1 Fair attack detection based on microaggregation

In this section, we introduce our microaggregation-based approach for fair detection of attacks in federated learning.

In our approach, we are interested only in the partition step, whereby similar clients will be clustered together based on their demographic attributes. The superiority of microaggregation over standard clustering for our purposes lies in that the former ensures that clusters will have at least size k. In this way, we avoid training models for clusters that are too small. Note that it is impossible to detect any outliers if too small clusters are allowed.

We propose the solution in Protocol 3 based on collaborative microaggregation to distinguish malicious clients from genuine outliers/minority group members, which we will call in what follows *protected clients*.

Protocol 3: AttackerDetectionMicro

1 Each protected client publishes, together with her pseudonym, demographic attribute values that characterize her as a member of a minority group;
2 The model manager and protected clients engage in decentralized microaggregation as per Protocol 4 to microaggregate protected clients according to their published attributes;
3 **for** *each cluster* C *of the microaggregation* **do**
4 Compute the centroid c_C of the updates sent by clients in C;
5 Let λ_C be the set of clients $\mathcal{P}_u \in C$ such that the distance $dist_u$ from $\mathcal{P}_u$'s update δ_u^t to the centroid of the updates is not an outlier, more precisely

$$\lambda_C = \{\mathcal{P}_u \in C | Q1 - \tau \times IQR \leq dist_u \leq Q3 + \tau \times IQR\}, \tag{3.1}$$

where $Q1$ and $Q3$ are, respectively, the first and the third quartile of the set of distances, $IQR = Q3 - Q1$ is the interquartile range and τ is a tolerance parameter;
6 **if** *a client* $\mathcal{P}_u \in \lambda_C$ **then**
7 $\alpha_u = 1/m$ (or n_u/n);
8 **else**
9 $\alpha_u = 0$;
10 **end**
11 **end**

In Line 1 of Protocol 3, the demographic attributes that characterize a minority might for example be $\{Sex = female, Age = young, Ethnicity = black\}$. In the microaggregation called in Line 2, parameter k must be taken large enough so that outliers can be distinguished within a group of k, and a collusion of k clients or of a significant fraction of k clients is unlikely. In Line 7, assigning a nonzero

weight α_u to P_u's update means accepting the update as legitimate (because it is similar to most updates in $\mathcal{P}_u$'s cluster C). In contrast, in Line 9 assigning $\alpha_u = 0$ means discarding the update (because it is too outlying even for the minority group represented by C).

Protocol 4: Decentralized microaggregation

Input: Clients $\mathcal{P}_1, \ldots, \mathcal{P}_m$; microaggregation parameter k;
Output: A partition of clients $\{C_i\}_{i=1}^{p}$;

1 Let qi_u be demographic attribute values of $\mathcal{P}_u$, for $u = 1, \ldots, m$;
2 Each $\mathcal{P}_u$ publishes qi_u;
3 The model manager uses a microaggregation algorithm based on the quasi-identifiers $qi_1, \ldots, qi_c$ to obtain clusters C_1, C_2, …, C_p, such that $k \leq |C_j| \leq 2k - 1$;
4 The model manager publishes C_1, C_2, …, C_p;
5 Each $\mathcal{P}_u$ can compute the above clusters and verify they are correct, and check that the cluster $C_{\mathcal{P}_u}$ where qi_u belongs contains k or more quasi-identifiers;

Note that microaggregation attempts to create clusters such that the published attributes of protected clients in each cluster are maximally similar. Therefore, if clients within a cluster are similar, it is natural to expect that the updates they send are also similar. As a consequence, if an update differs very much from the others, it is not unreasonable to treat the client having contributed it as malicious.

To create homogeneous clusters in an efficient way, in Protocol 4 we use the maximum distance to average vector (MDAV) algorithm, detailed in Algorithm 1, which is the most widely used microaggregation algorithm [24].

MDAV is a heuristic algorithm that clusters records in a data set so that each cluster is guaranteed to contain at least k records. At each iteration, two records are selected: the record x_r farthest from the average record $\bar{x}$ of the data set and the record x_s farthest from x_r. Then, a cluster is formed with x_r and its closest $k - 1$ records,

Algorithm 1: MDAV microaggregation algorithm

Input: Data set R, microaggregation parameter k;
Output: A partition of R, with sets of minimum size k;

while $|R| \geq 3k$ **do**
  Compute the average record $\bar{x}$ of all records in R;
  Consider the most distant record x_r to $\bar{x}$;
  Find the most distant record x_s from x_r considered in the previous step;
  Form two clusters around x_r and x_s, respectively, one containing x_r and the $k-1$ records closest to x_r and the other cluster containing x_s and the $k-1$ records closest to x_s;
  Take as a new data set R the previous data set R minus the clusters formed around x_r and x_s in the last instance of Step 5;
end
if *there are between* $3k-1$ *and* $2k$ *records in* R **then**
  Compute the average record $\bar{x}$ of the remaining records in R;
  Find the most distant record x_r from $\bar{x}$;
  Form a cluster containing x_r and the $k-1$ records closest to x_r;
  Form another cluster containing the rest of records;
else
  Form a new cluster with the remaining records;
end

and another cluster with x_s and its closest $k-1$ records. The records in both clusters are removed from the data set in the next iteration.

3.4.2 Fair attack detection based on Gaussian mixtures and expectation-maximization

In this section, we propose a second approach to tackle the problem of fair attack detection in federated learning. It is based on Gaussian mixture models (GMM) and the expectation-maximization (EM) algorithm.

Gaussian mixture models are probabilistically weighted combinations of Gaussian components, each with its own mean and covariance.

Mixture models, in general, are better suited than single distributions at modeling populations where differences between sub-populations exist. We leverage this property of Gaussian mixture models to capture the differences among different sub-populations (*e.g.*, minorities) while still being able to determine whether some data points are too far from the distribution modeling the population.

The expectation-maximization algorithm takes the number of Gaussians to model the data, K, and iteratively finds for each Gaussian $k \in \{1, \ldots, K\}$ its weight π_k, its mean μ, and its covariance matrix Σ_k. Given these parameters, we are able to compute how likely each point is to belong to the mixture of Gaussians.

Algorithm 2 shows how we use the expectation-maximization algorithm to detect potential malicious updates in federated learning aggregation.

This algorithm is used at each global learning step, that is, at the time of aggregating local updates. Once the model manager receives all updates from clients, it fits a GMM to the received updates using the expectation-maximization algorithm. Then, each individual update is evaluated according to the log-likelihood that it follows the

derived distribution. Those updates with a log-likelihood below a parameter τ (*i.e.*, those updates that are significantly different from the rest) are flagged as potentially malicious and discarded.

Algorithm 2: AttackerDetectionEM

Input: Client updates $\{\delta_u^t\}_{u=1}^m$

$\pi, \mu, \Sigma \leftarrow EM(\{\delta_u^t\}_{u=1}^m)$;
for u *in* S **do**
  $\ell \leftarrow log(\sum_k \mathcal{N}(\delta_u^t; \mu_k, \Sigma_k) \cdot \pi_k)$;
  if $\ell < \tau$ **then**
    $\alpha_u = 0$;
  else
    $\alpha_u = 1/m$ (or n_u/n);
  end
end

3.4.3 Fair attack detection based on DBSCAN

In this section, we propose a third approach to tackle the problem of fair attack detection in federated learning. It is based on a commonly used data clustering algorithm, *i.e.* density-based spatial clustering of applications with noise (DBSCAN).

In Algorithm 3 we show how DBSCAN can be used for attacker detection in federated learning.

The model manager fits the model to the updates and goes through each individual update to check if it is a noise point. If the latter happens, then the model manager flags the update as malicious.

Algorithm 3: AttackerDetectionDBSCAN

Input: Set of clients S, Client updates $\{\delta_u^t\}_{u=1}^m, Eps, MinPts$

$Core, Border, Noise \leftarrow DBSCAN(\{\delta_u^t\}_{u=1}^m, Eps, MinPts)$;
for u *in* S **do**
  if $\delta_u^t \in Noise$ **then**
    $\alpha_u = 0$;
  else
    $\alpha_u = 1/m$ (or n_u/n);
  end
end

3.5 Experimental results

We conducted experiments to examine the effectiveness of our attack detection mechanisms in FL with minority groups and non-i.i.d. data. To that end, we chose three publicly available data sets, namely (i) the Adult Income data set [54], (ii) the Athletes data set [55], and (iii) the Bank Marketing data set [56].

In the next sections, we describe these data sets and the preprocessing we conducted on them to emulate both client bases including minority groups and client bases with non-i.i.d. data.

3.5.1 Data sets, preprocessing, and baseline scenarios

Here, we present a summary table (Table 3.1) of the data sets we use in our experiments, along with how we compute the initial baseline metrics.

Table 3.1: Characteristics of data sets.

	Adult	***Athletes***	***Bank Marketing***
# Records	45,222	206,165	45,211
Sensitive attr.	Race	Height and Country	Marital status
Balanced	No	Yes	No
Attributes	14	17	17
Class label	Income	Height	Term deposit

For the three data sets, we first cleaned missing values and identified (sensitive) attributes that split the population of individuals in each data set into majority and minority groups.

To measure potential biases in the data sets, we trained a federated learning baseline model for each of them and we computed performance metrics. In all three cases, the baseline models were built using Keras and consisted of a multilayer perceptron (MLP) with two hidden layers using the ReLU activation function. Since we were training binary classifiers, the output layer used a sigmoid activation function. We used binary cross-entropy as the loss function and the Adam optimizer with a learning rate $3e-4$. We split the data randomly in same-size sets and trained the models using the FedAvg 1 algorithm, with no attackers and no attacker detection mechanisms. All results were computed using random 75-25% train-test splits, and we trained the models for 100 epochs each time. The resulting metrics are provided as the average of several training procedures.

After this evaluation, we prepared the data sets for further experiments in a federated scenario with fair malicious client detection. We considered three different kinds of clients: majority clients, minority clients, and malicious clients. We followed the approach of [3]. The step-by-step process was to first sort the data, then divide the data into equally sized shards, and finally assign each of the shards to a different client.

The next subsections provide details on these procedures for each of the data sets.

Adult Income data set

The Adult data set has been typically used to train classifiers that predict whether an individual earns more or less than $\$50,000$ per year, according to a set of demographic attributes, and with two class values: $\leq$ \$50K (class 0) and $>$ \$50K (class 1). The two classes are distributed as follows: $34,014$ individuals (75.21%) earn up to $\$50K$ per year and $11,208$ individuals (24.78%) earn more.

The race attribute has 5 distinct values: (i) White: $38,903$ (86.02%); (ii) Black: $4,228$ (9.34%); (iii) Asian-Pac-Islander: $1,303$ (2.88%); (iv) Amer-Indian-Eskimo: 435 (0.96%); and (v) Other: 353 (0.78%).

We observe that White and Asian-Pacific-Islander are more likely to be in the $>\$50K$ class than the rest, with over 25% of the observations of these two in that class. In comparison, only around 12% of black individuals belong to this class. Therefore, we found that this data set had a bias towards White and Asian-Pacific-Islander individuals, so we first wanted to establish whether a machine learning model trained with these data will also show this bias.

To that end, we first set to finding which proportion of black individuals were misclassified as earning less than $\$50K$, in comparison to individuals from other ethnicities. The whole data set is clearly imbalanced towards the low-earning class, and so we expected a certain FNR (taking the class $<\$50K$ as class 0) across all ethnicities. We wanted to know if this FNR was balanced across all ethnicities.

First, we recoded the 'Black' individuals in the variable 'race' as 0 and the rest as 1, who were the 'nonblack' individuals. This resulted in $4,228$ black individuals and $40,994$ nonblack individuals.

Then, we trained an MLP as described above.

The training set consisted of 33,916 samples and the test set had 11,306 samples.

Table 3.2 shows the baseline scenario results for Adult.

Table 3.2: Performance measures for the Adult baseline scenario

	Black	***Nonblack***
Accuracy	87.31%	90.18%
FNR	0.1203	0.0881
FPR	0.0797	0.0702
ROC AUC	0.80	0.82

From Table 3.2, we can confirm that the model is biased to favor the 'nonblack' individuals since their FNR is smaller (9.5%) than for the 'Black' individuals (12.21%), *i.e.*, black individuals are incorrectly assigned to be in the low income category in a bigger proportion than nonblack individuals.

Next, we prepared the Adult data set for a federated learning scenario with malicious clients. Adult was split into 50 client shards, as follows:

- 30 shards contained records that only included nonblack individuals, *i.e.*, $race = 1$. These corresponded to the majority clients. Each shard consisted of 1,500 records, of which 1,400 were reserved for training and 100 for testing.
- 19 shards contained records of black individuals, *i.e.*, $race = 0$ across the two income classes. These shards corresponded to the minority clients. Each shard consisted of 1,500 records, of which 1,400 were reserved for training and 100 for testing.
- The remaining shard contained 55% of rich black individuals and 45% of low-income black individuals (3,694), whose labels had

been flipped to high-income. This shard represented a malicious client, trying to make the model misclassify low-income black individuals as high-income.

Our objective was to detect and remove those updates computed on poisoned records without contributing to the bias against the minority clients. That is, without punishing the genuine high-income black individuals.

Athletes Data set

The second data set contains information on all the athletes that have competed in any of the Olympic games from Athens 1896 to Rio 2016.

We replaced all missing values in the 'Medal' attribute with 'No Medal', and, after dropping the rest of missing values, we were left with 206, 165 records.

For our study, we used 'Country' and 'Height' as sensitive attributes. We made a new column 'Country_height', which has the values 'SouthAsian', labeled as 0 and 'NSA', labeled as 1. The first class included countries from south Asia and south east Asia present in the data set (Indonesia, Vietnam, Philippines, Malaysia, Sri Lanka, Thailand, Singapore, India, Pakistan, Maldives, Afghanistan, Bangladesh, Bhutan, Nepal, Brunei, Cambodia, Laos, Myanmar, Japan) for which the data shows people are more likely to have a height attribute below the data set median height (175.0 cm). The second value was the non-South Asian countries, which included the rest of the countries.

We considered 'tall' (class 1) those athletes with height 175.0 (the data set median) or above (110, 618 athletes) and 'short' (class 0) those athletes with height below 175.0 (95, 547). This was the classification task.

We were interested in studying the biases in models trained on these data, by focusing on male South Asian athletes as the protected minority.

First, we created a new data set with only male athletes, where $7,851$ were from South Asian countries and $131,603$ were from non South Asian countries. Then, we trained an MLP as described above, in which the training set consisted of $104,590$ samples and the test set of $34,864$ samples.

This was the baseline scenario, whose metrics we show in Figure 3.3.

Table 3.3: Performance measures for the Athletes baseline scenario

	SouthAsian	***NonSouthAsian***
Accuracy	89.42%	92.81%
FNR	0.0794	0.0701
FPR	0.0566	0.0513
ROC AUC	0.82	0.83

Then, we prepared the data set to be evaluated in a federated learning scenario with fair malicious client detection as described above.

In this case, we split the data into 90 shards, again, with three different kinds of clients:

- 60 shards contained records of non South Asian athletes, *i.e.*, 'Country' = 1. These corresponded to the majority clients. Each shard consisted of $1,500$ records, of which $1,400$ were reserved for training and 100 for testing.
- 29 shards contained records of South Asian athletes, *i.e.* 'Country' = 0. These shards corresponded to the minority clients. Each shard consisted of $1,500$ records, of which $1,400$ were reserved for training and 100 for testing.

- The last shard contained both tall South Asian athletes (25%) and short South Asian athletes (75%) whose labels had been flipped to tall. This shard is assigned to the attacker.

Again, our purpose was to detect the attacker and discard its updates without affecting the performance of the model or causing South Asian athletes to be misclassified in a larger proportion than in the baseline scenario.

Bank Marketing data set

The last data set we used is related to direct marketing campaigns (phone calls) of a Portuguese banking institution. Its classification goal is to predict whether the individual will subscribe to a term deposit ('yes' = 0, 'no' = 1).

It is an imbalanced data set with the class 1 ('no', that is, not subscribed) being the majority class (88.7%).

In this case, we used the $Marital-status$ attribute as the sensitive attribute. The majority of the individuals are married (60.19%) and these are the most approached ones. The rest are singles (28.29%) or divorced (11.52%).

To find out whether this imbalance caused bias in classification, we first labeled 'Divorced' individuals in the variable 'Marital-status' as 0 and the rest as 1, which are the 'non-divorced'. $5,207$ records belonged to divorced individuals and $40,004$ belonged to individuals who are not. Then, we trained an MLP as described above, with a training data set of size $33,908$ and a testing data set of size $11,303$.

We show the metrics for this baseline scenario in Table 3.4.

To prepare the data set for experiments with malicious clients, we split it in 50 shards as follows:

Table 3.4: Performance measures for the Marketing baseline scenario

	Divorced	***NonDivorced***
Accuracy	85.74%	88%
FNR	0.1102	0.1063
FPR	0.0787	0.0709
ROC AUC	0.79	0.80

- 30 shards contained records of non-divorced clients, *i.e.*, 'Marital-status' = 1. These corresponded to the majority clients. Each shard consisted of $1,500$ records, of which $1,400$ were reserved for training and 100 for testing.
- 19 shards contained records of divorced individuals. These shards corresponded to the minority clients. Each shard consisted of $1,500$ records, of which $1,400$ were reserved for training and 100 for testing.
- 1 shard of 1,500 records included 622 divorced individuals who had not subscribed to a term deposit. The label for 90% of those 622 records was flipped from 'no' to 'yes'. This shard represented a malicious client trying to poison the model into misclassifying non-subscribers to term deposits as subscribers.

3.5.2 Detection of malicious updates

We compared four approaches to detect malicious updates on the generated data sets:

1. *Baseline.* In the baseline experiment we trained a federated learning model using a distance-based method to detect outliers as in [43].

   In summary, an average update was computed from all client-provided updates. Then, the Euclidean distance between ev-

ery individual update and the average update was computed. All updates whose distance fell outside the bounds in Expression (3.1) with $\tau = 1.5$ were considered malicious and thus discarded.

2. *Microaggregation.* The second experiment modeled FL with microaggregation-based attack detection. We used different parameters for the experiments to show how varying them affects the metrics. The local learning steps varied among 1, 2 and 5, and for parameter k we chose 1, 3 and 5. *Note that microaggregation with $k = 1$ is actually the baseline case (no clusters).*

3. *GMM.* This experiment tested the GMM-based approach.

   We used the GMM implementation in Scikit-learn [57], with parameters $K = n_components = 3$ (3 mixture components) and $covariance_type = "full"$ (each component had its own general covariance matrix).

   Any update whose log-likelihood fell below $\tau = -20,000$ was considered malicious and was discarded. We used different parameters to see how this affected the metrics. The local learning steps were the same as in the previous experiment, the mean μ took values 0 and 2, and the covariance σ^2 took values 1 and 4. To get the optimal number of clusters, we used the Bayesian Information Criterion (BIC) function. The optimal value is the one that minimizes BIC.

4. *DBSCAN.* The last experiment was similar to the previous one, but using Algorithm 3 to detect malicious updates.

   The algorithm used the DBSCAN implementation provided in Scikit-learn with parameters $Eps = 0.5, 3, 5$, and $minPts$ depending on the data set as follows:

   - Adult Income data set: $minPts = 5, 15, 28$.

- Athletes data and Bank Marketing data sets: $minPts = 5, 18, 34$.

Values for *Eps* other than 0.5 and values for $MinPts$ other than 5 were selected following the procedure in Section 4.2 of [58].

3.5.3 Performance measures and discussion

In Table 3.5 we count the number of good updates sent by genuine minority clients but flagged as malicious, for each of the four approaches and for each of the three data sets. The numbers are cumulative over all epochs (100 epochs were used for all approaches). Clearly, the three methods we propose misclassified genuine minority updates as malicious in a smaller proportion than plain federated learning for the three studied data sets. In particular, the method based on microaggregation offered the best results among our three proposals.

Table 3.5: Number of genuine minority updates misclassified as malicious.

	Baseline	***Microaggr.***	***GMM***	***DBSCAN***
Adult	183	102	131	126
Athletes	385	213	227	234
Bank Marketing	271	158	172	164

Further, we used Accuracy and ROC AUC as performance metrics for majority, minority, and attacker clients. The basic objective of our mechanisms was to increase these performance metrics for both majority and minority clients while ensuring attackers achieved worse results. Additionally, we used the above mentioned Predictive Equality (PE) and Equal Opportunity (EO) metrics to detect the presence of unfairness towards the majority or the minority. According to these two metrics, the closer PE and EO to 0, the fairer is a method.

Results are summarized in the following tables. Tables 3.6, 3.7

and 3.8 report the above metrics for the microaggregation experiment with different parameters k and learning steps LS. Tables 3.9, 3.10 and 3.11 report the above metrics for the GMM experiment with different mean, covariance and learning steps. Finally, Tables 3.12, 3.13 and 3.14 report the above metrics for the DBSCAN experiment with different Eps, $minPts$, and learning steps.

The results show that all methods perform comparably or slightly better than the baseline scenario (microaggregation with $k = 1$) in terms of accuracy and ROC AUC. These results indicate that our methods allow the models to better capture the differences present between the majority and minority groups, while still being able to discard malicious updates which, in all cases, show worse performance measures than those from legitimate users.

Additionally, all methods reduce in most cases the differences between majority and minority groups with respect to the baseline scenario, which can be observed with the EO and PE.

Regarding the different parameters, we can observe in the results for microaggregation that the accuracy with $k = 3$ is better than with $k = 1$ (baseline). However, further increasing k to 5 decreases accuracy. A plausible explanation is that larger values of k yield larger clusters, which entails some information loss and thus a performance degradation. Thus, $k = 3$ seems best for accuracy, and it also yields the best values (closer to 0) for fairness metrics PE and EO between majority and minority groups.

In the case of GMM, the results improve as we increase the mean. When the means are too low, then the maximum-likelihood of the model fits Gaussians that may encompass legitimate users not distinguishable from malicious ones.

For the last method, with low $minPts$, the outliers are more clear. This is because, with a higher number of $minPts$, smaller clusters will

be incorporated into the larger ones, making it difficult to differentiate between majority and minority groups. Moreover, the accuracy and ROC AUC are better when minPts are lower.

Also, for the three methods, taking $LS = 2$ local learning steps appears as a better choice than $LS = 1$ or $LS = 5$.

Again, this allows us to conclude that our proposed outlier detection mechanisms are capable of distinguishing between genuine minority groups and attackers. In particular, the microaggregation-based method achieves the best performance in most cases. This was to be expected because microaggregation implements a finer-grained assessment of inter-client likeness not only to a prototypical majority, but to prototypical minorities. In this way, minority groups are properly (and fairly) considered, and only true outliers within these minority groups are discarded.

Table 3.6: Performance measures for Adult Income data set with different microaggregation parameters. 'B' stands for black (minority clients), 'NB' for non-black (majority clients), 'Att' for attacker clients.

Parameters		**Accuracy**			**ROC AUC**			**PE**	**EO**
k	LS	B	NB	Att	B	NB	Att	—	—
1	1	0.9003	0.9307	0.7712	0.82	0.83	0.68	0.0323	0.238
1	2	0.9125	0.9362	0.8298	0.82	0.84	0.72	0.0271	0.0205
1	5	0.9072	0.9251	0.7809	0.83	0.83	0.71	0.0308	0.0251
3	1	0.9395	0.9591	0.7421	0.84	0.85	0.72	0.0097	0.0061
3	2	0.9594	0.9873	0.8406	0.86	0.87	0.73	0.0022	0.031
3	5	0.9775	0.9849	0.7421	0.89	0.90	0.78	0.0035	0.0029
5	1	0.92	0.9401	0.8103	0.84	0.83	0.73	0.0106	0.0095
5	2	0.9261	0.9387	0.8164	0.83	0.84	0.71	0.0182	0.0206
5	5	0.9122	0.9394	0.8027	0.82	0.83	0.72	0.0213	0.0285

Table 3.7: Performance measures for Athletes data set with different microaggregation parameters. 'SA' stands for South Asian (minority clients), 'NSA' for non-South-Asian (majority clients), 'Att' for attacker clients.

Parameters		**Accuracy**			**ROC AUC**			**PE**	**EO**
k	LS	B	NB	Att	B	NB	Att	—	—
1	1	0.9227	0.9546	0.8115	0.83	0.85	0.71	0.0118	0.0241
1	2	0.9395	0.9591	0.802	0.82	0.85	0.72	0.0172	0.0283
1	5	0.9282	0.9487	0.8146	0.82	0.84	0.71	0.0105	0.0199
3	1	0.9582	0.9721	0.7903	0.87	0.89	0.69	0.0081	0.0065
3	2	0.9689	0.9984	0.8049	0.88	0.9	0.72	0.0037	0.0103
3	5	0.9508	0.9714	0.8021	0.87	0.89	0.71	0.0073	0.0091
5	1	0.9372	0.9522	0.8241	0.85	0.86	0.73	0.011	0.0183
5	2	0.9388	0.9564	0.8173	0.84	0.86	0.72	0.0256	0.0174
5	5	0.9261	0.947	0.8153	0.84	0.85	0.72	0.0234	0.0201

Table 3.8: Performance measures for Bank Marketing data set with different microaggregation parameters. 'D' stands for divorced (minority clients), 'ND' stands for non-divorced (majority clients), 'Att' for attacker clients.

Parameters		**Accuracy**			**ROC AUC**			**PE**	**EO**
k	LS	B	NB	Att	B	NB	Att	—	—
1	1	0.9106	0.9212	0.7572	0.81	0.83	0.68	0.0147	0.0132
1	2	0.9163	0.9387	0.7681	0.83	0.84	0.72	0.0212	0.0263
1	5	0.9201	0.93	0.7657	0.83	0.83	0.72	0.0178	0.0224
3	1	0.9695	0.9691	0.7521	0.86	0.87	0.74	0.0098	0.0113
3	2	0.9738	0.9888	0.7806	0.87	0.89	0.75	0.0096	0.0077
3	5	0.9572	0.9614	0.7521	0.87	0.88	0.74	0.0103	0.0094
5	1	0.9362	0.9584	0.7662	0.86	0.86	0.73	0.0152	0.188
5	2	0.9344	0.9482	0.78	0.85	0.86	0.73	0.0183	0.0205
5	5	0.9272	0.9439	0.7638	0.85	0.85	0.72	0.0175	0.0226

Table 3.9: Performance measures for Adult Income data set with different GMM parameters

Parameters			**Accuracy**			**ROC AUC**			**PE**	**EO**
μ	σ^2	LS	B	NB	Att	B	NB	Att	—	—
0	1	1	0.9117	0.9398	0.7251	0.83	0.83	0.72	0.042	0.0158
0	1	2	0.9295	0.9401	0.7288	0.83	0.84	0.71	0.0301	0.0242
0	1	5	0.9272	0.9464	0.73	0.83	0.84	0.71	0.0323	0.0168
0	4	1	0.939	0.9591	0.7321	0.84	0.84	0.70	0.0285	0.0315
0	4	2	0.9208	0.9446	0.74	0.83	0.83	0.71	0.0273	0.0205
0	4	5	0.9384	0.9516	0.7461	0.84	0.85	0.72	0.0207	0.0186
2	4	1	0.9473	0.9628	0.7582	0.87	0.89	0.73	0.0096	0.105
2	4	2	0.9578	0.9763	0.7534	0.88	0.91	0.74	0.0062	0.0091
2	4	5	0.9455	0.9682	0.7495	0.87	0.89	0.73	0.0071	0.0112

Table 3.10: Performance measures for Athletes data set with different GMM parameters

Parameters			**Accuracy**			**ROC AUC**			**PE**	**EO**
μ	σ^2	LS	B	NB	Att	B	NB	Att	—	—
0	1	1	0.9109	0.9423	0.7762	0.83	0.84	0.69	0.0289	0.0213
0	1	2	0.9154	0.9365	0.7864	0.84	0.84	0.7	0.0162	0.0197
0	1	5	0.9277	0.9488	0.7742	0.83	0.83	0.7	0.0256	0.0301
0	4	1	0.9293	0.9491	0.7821	0.84	0.84	0.71	0.0143	0.0106
0	4	2	0.9207	0.9456	0.7895	0.83	0.85	0.71	0.0187	0.0259
0	4	5	0.9156	0.9327	0.7759	0.83	0.83	0.7	0.0175	0.0201
2	4	1	0.9476	0.9702	0.8143	0.86	0.89	0.73	0.0084	0.0107
2	4	2	0.9587	0.9765	0.8278	0.87	0.89	0.73	0.0051	0.0038
2	4	5	0.9443	0.9631	0.8109	0.87	0.88	0.72	0.0096	0.012

Table 3.11: Performance measures for Bank Marketing data set with different GMM parameters

Parameters			**Accuracy**			**ROC AUC**			**PE**	**EO**
μ	σ^2	*LS*	B	NB	Att	B	NB	Att	—	—
0	1	1	0.8906	0.9273	0.8064	0.82	0.84	0.69	0.0359	0.0401
0	1	2	0.9159	0.9365	0.7929	0.83	0.84	0.7	0.0225	0.0362
0	1	5	0.9066	0.9284	0.7942	0.84	0.83	0.7	0.0274	0.0207
0	4	1	0.9224	0.9391	0.7831	0.84	0.84	0.71	0.0148	0.0262
0	4	2	0.9282	0.9256	0.7995	0.83	0.85	0.71	0.0197	0.0211
0	4	5	0.9144	0.9397	0.8059	0.83	0.84	0.7	0.025	0.0193
2	4	1	0.9376	0.9551	0.8142	0.86	0.89	0.73	0.0123	0.0085
2	4	2	0.9554	0.9672	0.8178	0.87	0.89	0.73	0.011	0.0152
2	4	5	0.9437	0.9614	0.8109	0.87	0.88	0.72	0.0097	0.0104

Table 3.12: Performance measures for the Adult Income data set with different DBSCAN parameters

Parameters			**Accuracy**			**ROC AUC**			**PE**	**EO**
Eps	*minPts*	*LS*	SA	NSA	Att	SA	NSA	Att	—	—
0.5	5	1	0.9632	0.9861	0.7994	0.89	0.91	0.72	0.0062	0.0031
0.5	5	2	0.9777	0.9636	0.7885	0.9	0.9	0.71	0.0045	0.0081
0.5	5	5	0.9592	0.9622	0.7839	0.89	0.9	0.71	0.0093	0.0106
3	15	1	0.94	0.9546	0.8047	0.85	0.85	0.7	0.0126	0.0174
3	15	2	0.9584	0.9603	0.8021	0.85	0.86	0.69	0.0117	0.0109
3	15	5	0.9401	0.9514	0.7962	0.84	0.85	0.7	0.0183	0.0196
5	28	1	0.9225	0.9347	0.8139	0.83	0.84	0.7	0.0241	0.0273
5	28	2	0.9332	0.9289	0.8016	0.84	0.84	0.71	0.0191	0.0262
5	28	5	0.9187	0.9311	0.7812	0.83	0.84	0.72	0.0285	0.0306

Table 3.13: Performance measures for the Athletes data set with different DBSCAN parameters

Parameters			**Accuracy**			**ROC AUC**			**PE**	**EO**
Eps	*minPts*	*LS*	SA	NSA	Att	SA	NSA	Att	—	—
0.5	5	1	0.9505	0.9682	0.8082	0.88	0.89	0.76	0.0105	0.0138
0.5	5	2	0.9602	0.9688	0.7943	0.89	0.9	0.75	0.0078	0.0063
0.5	5	5	0.9558	0.9592	0.8031	0.88	0.89	0.76	0.0097	0.0115
3	18	1	0.9372	0.9546	0.8147	0.85	0.85	0.71	0.0167	0.0199
3	18	2	0.9434	0.9501	0.8104	0.85	0.86	0.7	0.0202	0.0186
3	18	5	0.9386	0.9427	0.8175	0.84	0.83	0.7	0.0231	0.0295
5	34	1	0.9269	0.9335	0.8268	0.83	0.84	0.72	0.0282	0.0334
5	34	2	0.9037	0.9284	0.82	0.82	0.83	0.7	0.0227	0.0285
5	34	5	0.8948	0.9005	0.8293	0.82	0.82	0.73	0.0312	0.0294

Table 3.14: Performance measures for the Bank Marketing data set with different DBSCAN parameters

Parameters			**Accuracy**			**ROC AUC**			**PE**	**EO**
Eps	*minPts*	*LS*	SA	NSA	Att	SA	NSA	Att	—	—
0.5	5	1	0.9522	0.9568	0.8121	0.88	0.89	0.72	0.0061	0.0074
0.5	5	2	0.9641	0.9759	0.8195	0.89	0.88	0.71	0.0094	0.0103
0.5	5	5	0.9548	0.9607	0.805	0.88	0.88	0.71	0.0114	0.0101
3	18	1	0.9476	0.9532	0.8048	0.85	0.86	0.7	0.0182	0.0192
3	18	2	0.94	0.9503	0.8021	0.85	0.86	0.7	0.0174	0.0152
3	18	5	0.9366	0.9407	0.7932	0.84	0.85	0.7	0.0252	0.0138
5	34	1	0.913	0.9328	0.8039	0.82	0.84	0.7	0.0322	0.0285
5	34	2	0.9174	0.9349	0.7963	0.83	0.84	0.71	0.0387	0.0393
5	34	5	0.9021	0.9136	0.7812	0.82	0.82	0.71	0.0372	0.0414

Finally, from the related work approaches mentioned in Section 3.3 for non-i.i.d. data in FL, we took [1] and its implementation[1] to enhance our approach in the non-i.i.d. setting.

We implemented our three methods on top of their approach and measured their performance on the Adult data set. Table 3.15 shows the results, where FL_Zhao is the baseline (microaggregation with $k = 1$) on top of [1]. For all the methods, the local learning steps are the same, $LS = 1$. The parameter k for microaggregation is 3. In the second method, GMM, we use $\mu = 0$ and $\sigma^2 = 1$. And for the last one, DBSCAN, the parameters used are $\epsilon = 0.5$ and $minPts = 5$. We observe that the results improve for all three methods and show similar behavior when compared to our previous experiments. In particular, we see that the microaggregation protocol performs the best out of the three.

Table 3.15: Performance measures for Adult Income data set with [1] approach.

	FL Zhao	***Microaggr. Zhao***	***GMM Zhao***	***DBSCAN Zhao***
Accuracy	0.8108	0.9075	0.9023	0.864
PE	0.0195	0.009	0.0103	0.0110
EO	0.0351	0.0192	0.0201	0.0273

3.6 Summary

In this chapter, we have dealt with the problem of distinguishing abnormal/malicious behaviors from legitimate ones in federated learning. We focus on scenarios with clients having legitimate minority data, whose updates are likely to be classified as outlying/malicious by the standard attack detection mechanisms proposed in the literature. To make progress towards fair attack detection, we propose three different methods, one based on microaggregation, another based on the Gaussian mixture model, and the third one based on

[1] https://github.com/yjlee22/FedShare

DBSCAN.

To evaluate and compare the performance of these methods, we computed standard evaluation metrics, namely accuracy, ROC AUC, PE, and EO. Our results indicate that the microaggregation method is especially effective at differentiating malicious model updates from normal (even minority) model updates. This results in improvements in all observed evaluation metrics. From a more qualitative perspective, our approach avoids discriminating minority groups.

Chapter 4

Measuring Fairness in Machine Learning Models via Counterfactual Examples

4.1 Introduction

The existing research addressing the topic of fairness in machine learning has focused on how to measure and evaluate fairness (or, equivalently, bias) in models.

In [59], the authors introduced a causal, individual-level, definition of fairness, called counterfactual fairness, which states that a decision is fair toward an individual if it coincides with the one that would have been taken in a counterfactual world in which the protected (a.k.a. sensitive) attributes were different[1].

[1]Vulnerable minorities to be protected from biased decisions are formed by

More formally, consider a classifier f with protected attributes A, remaining inputs X, and output Y. Then f is counterfactually fair if, for any values $X = x$ and $A = a$,

$$\Pr(Y_{A\leftarrow a} = y|X = x, A = a) = \Pr(Y_{A\leftarrow a'} = y|X = x, A = a)$$

for all y and for any value a' attainable by A.

In relation with group fairness, counterfactual fairness is complementary to the group fairness notion of equality of odds, which demands equality of true positive rates and true negative rates for different values of the protected attribute.

Any of the metrics mentioned in Section2.3 can be used to calculate disparities in data across groups, but many of them cannot be balanced across subgroups at the same time. As a result, one of the most crucial components of measuring bias in the model is understanding how fairness should be defined for a certain scenario.

In this work, we are going to merge the concept of group fairness with individual fairness, in the sense that the proposed method protects the individuals of any minority against any bias in the model.

Contributions

The purpose of this work is to detect any bias in the ML models targeting any individual, regardless of the data type used in the model, by leveraging the use of the counterfactual examples. The key contributions in this chapter are:

1. The proposed method detects the bias in the ML model with regard to the model behavior and the training data, unlike the previous work [59] that solely targets the bias in the data sets.

those individuals having certain values for the protected attributes. For example, a protected attribute could be Ethnicity, and a vulnerable minority could be that formed by individuals with Ethnicity='black'.

2. The method we are proposing is based on the concept of counterfactual examples, which provide bias detection regardless of the data type (tabular, images, text, etc.). This offers the model developers a straightforward way to measure the fairness of their developed models.

4.2 Measuring fairness in ML models via counterfactual examples

Ensuring the algorithms used in machine learning are free from discriminatory biases has spurred researchers to define an enormous assemblage of new work seeking to better understand and prevent discrimination. The existing research addressing the topic of fairness in ML has focused on how to measure and evaluate fairness (or, equivalently, bias) in models. There are several fairness definitions based on group fairness or individual fairness. Group fairness compares statistics of protected groups, whereas individual fairness compares outcomes for comparable subjects. For instance, [19] survey more than twenty measures of fairness; however, no clear indication exists as to which measure is most suitable for classification tasks.

The work in [17] introduced the definition of individual fairness, which contrasts with group-based notions of fairness [20], [60] that require demographic groups to be treated similarly on average.

Later, [59] introduced a causal, individual-level definition of fairness, called counterfactual fairness, which states that a decision is fair toward an individual if it coincides with the one that would have been taken in a counterfactual world in which the protected attributes were different.

In [61], by utilizing adversarial examples for data augmentation, the authors implemented a prototype application to solve the algorithmic bias problem. In order to obtain a fair data set in which the

distribution of bias variables is balanced, they apply adversarial attacks to generate examples containing information of the bias variable as the enhanced data.

In [62], the authors develop a GAN for fairness. Their architecture is developed for low-dimensional tabular data and only applies to demographic parity, whereas our work is geared towards high-dimensional image data and individual fairness.

Another work that measures fairness based on GANs is [63]. The authors propose an adversarial approach, inspired by GANs, in which a sanitizer is learned from data representing the population. In this work, local sanitization has been considered to reach algorithmic fairness.

4.3 Measuring fairness via counterfactual examples

In this section, we propose a method to measure fairness in ML models. Our approach relies on generating counterfactual examples. The technique used to generate those counterfactuals may differ according to the input data type.

4.3.1 Measuring fairness in tabular data

We want to measure the fairness of any ML model, that is, its lack of bias against any particular minority. To this end, we need to guarantee that the ML model is not making its predictions based on a protected attribute. This process should be automated in the sense that it is not enough to change the attribute value and monitor the outcome of the model. At the same time, this process should be model- and data-related, because the bias might be in the training data themselves, or in the model learned during the training.

In order to measure the fairness of an ML model f, for a specific

data-record x, taking into consideration the protected attributes A, we have to meet the counterfactual fairness definition described in Section 4.1. We propose an automated approach that creates counterfactual examples for each record in the data set. The proposed method uses adversarial examples as a means to create those counterfactual examples. To create an adversarial example, the proposed method targets only the attribute with the highest effect on the ML model prediction in an attempt to alter its predicted label. As a result, if the changed attribute was one of the protected attributes, this indicates that the model is not fair to this record.

Algorithm 4 describes the proposed method to generate the counterfactual examples that are used to measure the fairness of the model. Given an ML model f, a maximum allowed value ϵ_{max}, and an input record x containing n attributes, the proposed method generates the counterfactual example x_* by changing at most $c \leq n$ attributes from x, where c is the number of protected attributes. First, to set the target classification label y_* we desire x_* to be classified into, we compute *probs*, that is, the probability vector f outputs for x. Then, we set the desired class label y_* to be the index of the second most probable class in *probs*. Choosing this class makes it easier for the proposed method to find the desired x_* with small changes to x. Note that the user can also set y_* to any class label she wants. Then, to select the c attributes which have the highest effect on the model prediction, we compute the gradient of the loss between the model output $f_u(x)$ and the desired output y_* with regard to the attributes of the input record. After that, we take the L_1-norm of the computed gradient ∇_x. Subsequently, we identify the c attributes with the highest L_1-norms as the attributes to be changed when generating x_*. This is done using a weighting vector w which contains 0s for the unchanged attributes and 1s for the changed attributes. Using this vector allows us to change only c attributes while creating x_*. Afterwards, we keep repeating the gradient descent step

$$x_* = x - w \cdot \epsilon \cdot \frac{\partial}{\partial x}\mathcal{L}(f(x), y_*), \tag{4.1}$$

with regard to w until one of two following conditions is satisfied: i) an adversarial example x_* is obtained that fools f into labeling it as y_* or ii) the maximum value ϵ_{max} is reached for ϵ. Note that we start with a small $\epsilon = 0.05$ and we increase it by 0.05 at each step. Once we create the adversarial example x_*, we compare it with x to identify the changed attributes. If one or more of the changed attributes belong to A, the set of the protected features, then this indicates bias in the model.

Algorithm 4: Creating the counterfactual example

Input: Trained model f, record x, maximum allowed ϵ_{max}, maximum number of attributes to be changed c.
Output: Counterfactual example x_*

$probs \leftarrow$Get_Probabilities($f(x)$);
$y_* \leftarrow argmax(probs, 2)$;
$n \leftarrow$ Number of attributes in x;
$|\nabla_x| \leftarrow abs(\frac{\partial}{\partial x}\mathcal{L}(f(x), y_*))$;
$w \leftarrow$ Zero vector of length n;
$idxs \leftarrow$ Indices of the highest c values in $|\nabla_x|$;
for $idx \in idxs$ **do**
  $w[idx] = 1$;
end
$x_* \leftarrow x$;
$\epsilon \leftarrow 0.05$;
while $f(x_*) \neq y_*$ *and* $\epsilon \leq \epsilon_{max}$ **do**
  $x_* \leftarrow x - w \cdot \epsilon \cdot \frac{\partial}{\partial x}\mathcal{L}(f(x), y_*)$;
  $\epsilon \leftarrow \epsilon + 0.05$;
end
if $f(x_*) = y_*$ **then**
  Return x_*;
else
  Return $\emptyset$;
end

4.3.2 Measuring fairness in image data

Measuring fairness in image classifiers is more challenging than measuring fairness in tabular data classifiers because the attributes in image data are not self-explanatory. To tackle this challenge, we propose a method that generates counterfactual examples for image data by leveraging the cycle GANs described in 2.9.

In the proposed method, we use two generators and one discriminator consisting of a trained binary classifier, as shown in Fig.4.1. Both generators create images to fool the trained image classifier model. However, each generator can only fool the model into one specific outcome. Generator A creates counterfactual images to be classified by the classifier into class a, whereas generator B's counterfactuals are to be classified into class b. After training both generators, they can be used to create counterfactual examples to detect any bias in the classifier. Any image m from class a that the classifier misclassifies into class b is passed through generator A to generate the counterfactual m_*, which will be classified into class a. By comparing m and m_*, we can notice whether they differ in any of the discriminatory attributes (those related to race, such as dark skin or dark hair). Similarly, an image m' from class b misclassified into class a is passed through generator B to create the counterfactual m'_*.

4.4 Empirical results

In this section, we evaluate the performance of the proposed approach on two ML tasks: tabular data classification and image classification. For each task, we trained a baseline model with the original data set and a biased model after we did some alterations to the data set. In both data sets, the baseline and biased models had the same architecture. First, we show the performance of the models, and then we evaluate the proposed fairness measures.

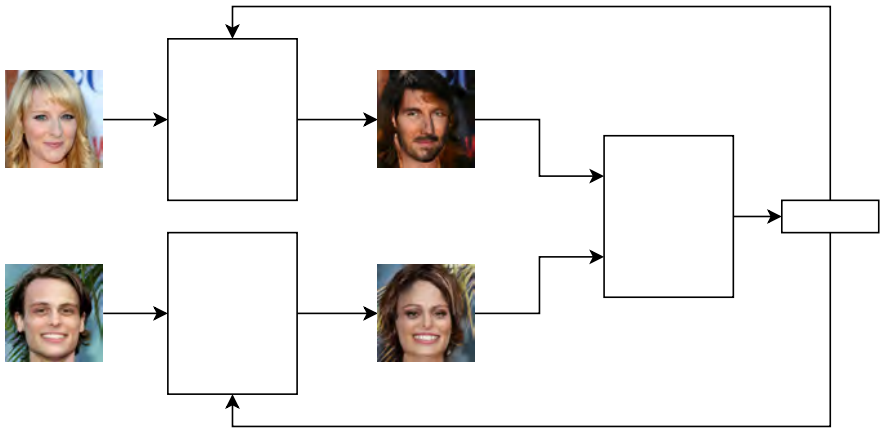


Figure 4.1: Training the generator of the counterfactual examples

4.4.1 Experimental setup

Data sets and provider models. We evaluated the proposed approach on two data sets:

- **Adult** (more details in Section 3.5). From the attributes of this data set, we dropped the final weights (*fnlwgt, capital-loss, and capital-gain*), which reveal too much information to the model, and the *education* attribute, which is redundant with *education-num*. We also encoded categorical attributes as numbers. The class label is the attribute *income*, which classifies records into either $>$ 50K or $\leq$ 50K. We used 80% of the data as training data, and the remaining 20% as validation data.

- **CelebA**[2] is a binary classification data set from Kaggle. It consists of RGB images of male and female faces with a training set of 17,943 female images and 10,057 male images. The validation set contains 2,000 images evenly divided into the two

[2] `https://www.kaggle.com/dataset/504743cb487a5aed565ce14238c6343b7d650ffd28c071f03f2fd9b25819e6c9`

Table 4.1: Architectures of the models used in the experiments for the Adult and the CelebA classification data sets. $C(3, 32, 3, 0, 1)$ denotes a convolutional layer with 3 input channels, 32 output channels, a kernel of size 3×3, a stride of 0, and a padding of 1; $MP(2, 2)$ denotes a max-pooling layer with a kernel of size 2×2 and a stride of 2; and $FC(18432, 2048)$ indicates a fully connected layer with 18,432 inputs and 2,048 output neurons. We used $ReLU$ as an activation function in the hidden layers; lr stands for learning rate.

Data set	Model architecture	Hyper-parameter
Adult	FC(12,100), FC(100,100), FC(100,2)	lr = 0.001, epochs = 10, batch = 128
CelebA	C(3,32,3,0,1), C(32,64,3,1,1), MP(2,2),C(64,128,3,0,1), C(128,256,3,1,1), MP(2,2), C(256,512,3,0,1), C(512,512,3,1,1), MP(2,2), C(512,256,3,0,1), C(256,256,3,1,1), MP(2,2), FC(16384,2048), FC(2048,1024), FC(1024,512), FC(512,128), FC(128,32), FC(32,2)	lr = 0.001 epochs = 10 batch = 64

classes. Since the images have large sizes of 1024 × 1024 pixels, we first resized them to 256×256 pixels in order to train our models faster.

The architectures of the models used in the experiments are shown in Table 4.1. These models were used in a previous paper [64]. For all the experiments, the binary cross-entropy loss function and the Adam optimizer [65] were used.

Biased models training data. The two data sets were balanced in terms of sensitive data. To this end, we modified as follows both training data sets in order to train the biased model.

- **Adult:** To cause the model to be biased against the attribute *gender* , we selected 45% of the *females* whose *race* was *black*, and we changed their income into $\leq$ 50k. Also, we selected 45% of the *males* whose *race* was *black*, and we changed their income into >50k.
- **CelebA:** To train a biased gender classification model, we separated from the female training data the images containing dark

skin. In order to do so, we used the unsupervised cluster K-means [66], while keeping track of some cherry-picked images to monitor the outcome of the cluster. The desired class consisted of 1904 images. We changed the class of 60% of those images from female into male.

Evaluation metrics. We used the following evaluation metrics to measure the performance of the trained surrogate models and the generated explanations:

- *Accuracy*: Number of correct predictions divided by the total number of predictions. We used this metric to measure and compare the performance of the baseline and the biased models.
- *ROC curve*: The receiver operating characteristic (ROC) is a graph plotting the false positive rate (FPR) in the abscissae and the true positive rate (TPR) in the ordinates. It shows the performance of a classification model at all classification thresholds. The AUC (area under the ROC curve) measures the two-dimensional area underneath the ROC curve, giving values from 0.0 (when the model predictions are 100% wrong), to 1.0 (when the model predictions are 100% correct). AUC is scale-invariant and classification-threshold invariant. In some cases, it is a better metric than accuracy.

4.4.2 Results

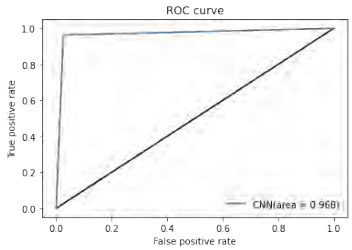
Accuracy and ROC-AUC score

In Table 4.2, we show the accuracy of the baseline and biased model when evaluated on the full evaluation data set, for both data sets. The baseline models were more accurate than the biased models on both data sets, even though the biased models' accuracies were not much lower.

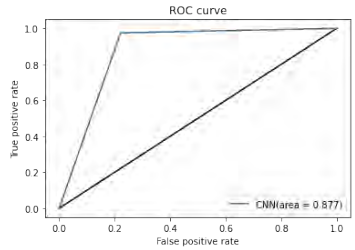
Table 4.2: Accuracy of the baseline and biased models evaluated on the full evaluation data set, on both data sets

Data set	Baseline model	Bias model
Adult	82.80%	79.67%
CelebA	98.52%	86.42%

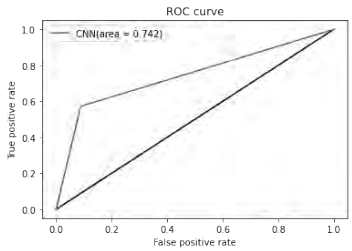
In Fig. 4.2 the ROC curves are presented for both models on the two data sets. The results shown are consistent with those in Table 4.2, where both the baseline models and the biased models display a similar performance, even though there are small differences in the curves between both models.



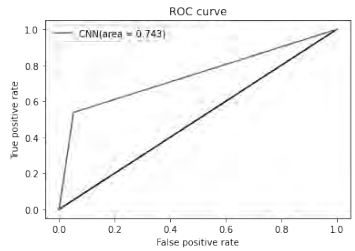
(a) ROC curve of the baseline model on the Adult data set



(b) ROC curve of the biased model on the Adult data set



(c) ROC curve of the baseline model on the CelebA data set



(d) ROC curve of the bias model on the CelebA data set

Figure 4.2: ROC curves of the baseline and biased models on both data sets

To better understand the performance differences between the baseline and the biased models, we evaluated them on modified data sets, which consist of the samples belonging to the minorities in each data

Table 4.3: Accuracy of the baseline and biased model evaluated on the targeted versions of Adult and CelebA.

data set	Baseline model	Bias model
Targeted Adult	81.90%	0.54%
Targeted CelebA	99.60%	1.67%

set. For the Adult data set, the modified set consisted of all the records with the *race* value equal to *black*, and for the CelebA dataset, we took a modified data set consisting of all female images with dark skin. We call those modified sets the targeted data sets. Table 4.3 reports the accuracy of the models when they were evaluated on the targeted sets. For both data sets, the baseline model accuracy was similar to that in Table 4.2, while the performance of the biased model was very poor: it was around 1% in both data sets. This illustrates that the biased models were precisely biased against the individuals in the targeted data sets.

Fairness of the trained models

- **Adult data set:** We generated a counterfactual example for each record in the validation portion of the Adult data set. First, we considered the attribute *gender* as the protected attribute. Then, we used Algorithm 4 with $c = 1$, to restrict the changes to only one attribute when generating the examples. After that, we computed the number of cases where the counterfactual example was created by changing the protected attribute. We found that for the bias model, this happened 1,637 times, whereas for the baseline model, this happened only 12 times. This indicates that for the baseline model, the protected attribute was not essential to the prediction. On the other hand, the protected attribute had a very high impact on the predictions made by the biased model. We present two of the counterfactual examples generated by the proposed method in Table 4.4. Record 1's income was classified as $> 50K$ by

Table 4.4: Two examples of records from the Adult data set. Symbol '"' indicates that the value of the attribute did not change during the creation of the counterfactual example.

Features	age	workclass	educational -num	marital -status	occupation	relationship	race	gender	hours -per -week	native -country	income
Original record 1	38	State-gov	13	Married	Protective-serv	Husband	Black	Male	52	United-States	>50K
Baseline recommendation	"	"	9	"	"	"	"	"	"	"	$\leq$ 50K
Biased model recommendation	"	"	"	"	"	"	"	Female	"	"	$\leq$ 50K
Original record 2	43	Local-gov	14	Unmarried	Tech-support	Not-in-family	Black	Female	47	United-States	$\leq$ 50K
Baseline recommendation	"	"	"	"	"	"	"	"	60	"	$>$ 50K
Bias-model recommendation	"	"	"	"	"	"	"	Male	"	"	$>$ 50K

both models. The first counterfactual example which was created based on the baseline model shows that we can change the original record's prediction to $\leq 50K$ by decreasing *educational level* from 13 to 9. This recommendation from the proposed method seems logical in the sense that less education yields less income. The second counterfactual example which was created based on the biased model recommended changing the *gender* from *male* to *female*, which shows the model learned a gender bias (the income of males is higher than the income of females). On the other hand, record 2's income was classified as $\leq 50K$ by both models. In this case, the counterfactual created by the baseline recommended increasing the *hours-per-week* from 47 to 60, which is also reasonable where more working hours usually means more payout. In contrast, the counterfactual created by the biased model recommended changing the attribute *gender* but this time from *female* to *male*, because the model learned that males earn more.

- **CelebA data set:** In order to generate the counterfactual examples for the CelebA data set, we selected the female images that were classified into the label male by the biased model and into the label female by the baseline model, because those images were targeted when training the biased model. Fig. 4.3 presents five examples of the counterfactual created by the pro-

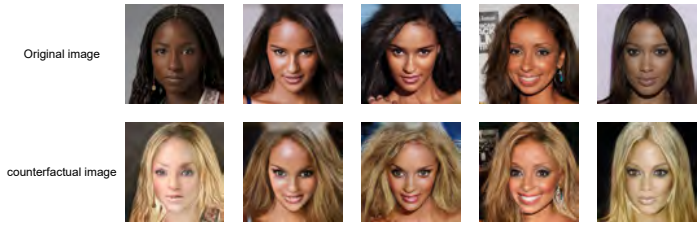


Figure 4.3: Five examples of the counterfactual examples created by the proposed method.

posed method. In all the images, the proposed method made the skin lighter, and the hair color blond. Those counterfactuals were classified as females by both models. Thus, our method detected that the biased model classified every face image with dark skin and black hair as male.

4.5 Summary

In this chapter, we have examined fairness in the scenario of binary classification for two types of input data, tabular and image data. The proposed method is based on generating counterfactual examples to measure fairness in ML models. In the case of tabular data, we used adversarial examples to create the counterfactuals. To achieve this for image data, we used GANs as a generator for the counterfactuals. Our experiments confirm that the proposed method can detect any bias in the model against protected attributes.

Chapter 5

Conclusions

This chapter gives a conclusion over the analysis of the results and discuss the potential future work that can be done on these subjects.

5.1 Contributions

In this thesis, we have developed methods for fair and robust ML. We want to ensure that no minority in the data set is unfairly impacted by the model's prediction. To this end, we have proposed methods that tackle this situation. To differentiate members of minority groups from potential model poisoners while performing robust aggregation of updates, we have presented three approaches. These are based on clustering algorithms: microaggregation, GMM, and DBSCAN. Also, we have studied the differences between the cases of i.i.d. and non-i.i.d. data. To measure fairness in generalized ML models, we have proposed a method based on generating counterfactual examples. In the case of tabular data, we made use of adversarial examples to create these counterfactual examples. In this way, we can create scenarios that are similar to real-life situations, with minimal differences to force

our models to make an incorrect prediction. Also, counterfactuals are useful for testing the robustness of our models. When dealing with image data, we have leveraged GANs to generate counterfactual examples. They also show the robustness of our models. The results confirmed that the biased models were precisely biased against the individuals in the targeted datasets.

5.2 Future work

We outline potential avenues of further research for the two main areas investigated in the thesis (Chapters 3 and 4, respectively).

- Our experimental work has considered a single protected attribute and binary classification. Using a single protected attribute if there are multiple protected attributes in a data set could exacerbate the actual underlying bias. Therefore we plan to work with multiple protected attributes.

- In Chapter 3, we have restricted to the detection of label-flipping attacks. Another future direction is to apply similar approaches to protect against other kinds of poisoning attacks, such as collusion attacks, while taking the fairness of the classification tasks into account.

- The recent literature mostly focuses on group fairness, and it seems that individual fairness definitions are neglected by researchers. In Chapter 4, we have used both types of fairness. However, an important future direction would be to perform more studies on the de-biasing methods capable of reducing both the individual and the group biases in classification.

- We plan to implement a new measure that guarantees the fairness of the image classifier models, since there is a limitation in the current literature on this topic.

- Furthermore, we intend to automate the comparison process between the original data and the counterfactual example.

5.3 Publications

The publications supporting the content of this thesis are listed below:

1. Singh, A. K., Blanco-Justicia, A., Domingo-Ferrer, J., Sánchez, D., & Rebollo-Monedero, D. (2020, November). "Fair detection of poisoning attacks in federated learning". In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 224-229). IEEE.
   This publication supports Chapter 3.

2. Singh, A.K., Blanco-Justicia, A. & Domingo-Ferrer, J. "Fair detection of poisoning attacks in federated learning on non-i.i.d. data". *Data Mining and Knowledge Discovery* (2023). `https://doi.org/10.1007/s10618-022-00912-6`.
   This publication supports Chapter 3 and is the journal extension of the previous publication.

3. Haffar, R., Singh, A. K., Domingo-Ferrer, J., & Jebreel, N. (2022). "Measuring fairness in machine learning models via counterfactual examples". In *International Conference on Modeling Decisions for Artificial Intelligence (MDAI 2022)* (pp. 119-131). Springer.
   This publication supports Chapter 4.

Bibliography

[1] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[2] F. Blix, S. A. Elshekeil, and S. Laoyookhong, "Designing gdpr data protection principles in systems development," *Journal of Internet Technology and Secured Transactions (JITST)*, vol. 6, no. 1, 2018.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[4] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[5] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[6] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Confer-*

ence on Artificial Intelligence and Statistics. PMLR, 2020, pp. 2938–2948.

[7] A. Narayanan, "Translation tutorial: 21 fairness definitions and their politics," in *Proc. Conf. Fairness Accountability Transp., New York, USA*, vol. 1170, 2018.

[8] M. S. Jere, T. Farnan, and F. Koushanfar, "A taxonomy of attacks on federated learning," *IEEE Security & Privacy*, vol. 19, no. 2, pp. 20–28, 2020.

[9] M. Benmalek, M. A. Benrekia, and Y. Challal, "Security of federated learning: Attacks, defensive mechanisms, and challenges," *Revue des Sciences et Technologies de l'Information-Série RIA: Revue d'Intelligence Artificielle*, vol. 36, no. 1, pp. 49–59, 2022.

[10] B. Friedman and H. Nissenbaum, "Bias in computer systems," *ACM Transactions on information systems (TOIS)*, vol. 14, no. 3, pp. 330–347, 1996.

[11] S. Barocas and A. D. Selbst, "Big data's disparate impact," *Calif. L. Rev.*, vol. 104, p. 671, 2016.

[12] R. Binns, "Fairness in machine learning: Lessons from political philosophy," in *Conference on Fairness, Accountability and Transparency*. PMLR, 2018, pp. 149–159.

[13] D. Kahneman, *Thinking, fast and slow*. Macmillan, 2011.

[14] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning fair representations," in *International conference on machine learning*. PMLR, 2013, pp. 325–333.

[15] C. Louizos, K. Swersky, Y. Li, M. Welling, and R. Zemel, "The variational fair autoencoder," *arXiv preprint arXiv:1511.00830*, 2015.

[16] K. Lum and J. Johndrow, "A statistical framework for fair predictive algorithms," *arXiv preprint arXiv:1610.08077*, 2016.

[17] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd innovations in theoretical computer science conference*, 2012, pp. 214–226.

[18] P. Gajane and M. Pechenizkiy, "On formalizing fairness in prediction with machine learning," *arXiv preprint arXiv:1710.03184*, 2017.

[19] S. Verma and J. Rubin, "Fairness definitions explained," in *2018 ieee/acm international workshop on software fairness (fairware)*. IEEE, 2018, pp. 1–7.

[20] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," *Advances in neural information processing systems*, vol. 29, 2016.

[21] X. Zhang, M. Hong, S. Dhople, W. Yin, and Y. Liu, "Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data," *arXiv preprint arXiv:2005.11418*, 2020.

[22] J. Domingo-Ferrer and J. M. Mateo-Sanz, "Practical data-oriented microaggregation for statistical disclosure control," *IEEE Transactions on Knowledge and data Engineering*, vol. 14, no. 1, pp. 189–201, 2002.

[23] V. Torra, "Microaggregation for categorical variables: a median based approach," in *International Workshop on Privacy in Statistical Databases*. Springer, 2004, pp. 162–174.

[24] J. Domingo-Ferrer and V. Torra, "Ordinal, continuous and heterogeneous k-anonymity through microaggregation," *Data Mining and Knowledge Discovery*, vol. 11, no. 2, pp. 195–212, 2005.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[26] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the em algorithm," *SIAM review*, vol. 26, no. 2, pp. 195–239, 1984.

[27] A. Roche, "Em algorithm and variants: An informal tutorial," *arXiv preprint arXiv:1105.1476*, 2011.

[28] C. J. Wu, "On the convergence properties of the em algorithm," *The Annals of statistics*, pp. 95–103, 1983.

[29] D. Nettleton, "Convergence properties of the em algorithm in constrained parameter spaces," *Canadian Journal of Statistics*, vol. 27, no. 3, pp. 639–648, 1999.

[30] I. Naim and D. Gildea, "Convergence of the em algorithm for gaussian mixtures with unbalanced mixing coefficients," *arXiv preprint arXiv:1206.6427*, 2012.

[31] S. Michael and V. Melnykov, "An effective strategy for initializing the em algorithm in finite mixture models," *Advances in Data Analysis and Classification*, vol. 10, no. 4, pp. 563–583, 2016.

[32] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[33] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "Density-based spatial clustering of applications with noise," in *Int. Conf. Knowledge Discovery and Data Mining*, vol. 240, 1996, p. 6.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[35] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[36] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harv. JL & Tech.*, vol. 31, p. 841, 2017.

[37] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 070–16 084, 2020.

[38] A. George and A. Vidyapeetham, "Anomaly detection based on machine learning: dimensionality reduction using pca and classification using svm," *International Journal of Computer Applications*, vol. 47, no. 21, pp. 5–8, 2012.

[39] M. Gander, M. Felderer, B. Katt, A. Tolbaru, R. Breu, and A. Moschitti, "Anomaly detection in the cloud: Detecting security incidents via machine learning," in *International Workshop on Eternal Systems*. Springer, 2012, pp. 103–116.

[40] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," *arXiv preprint arXiv:1803.01498*, 2018.

[41] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," *arXiv preprint arXiv:1910.09933*, 2019.

[42] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[43] J. Domingo-Ferrer, A. Blanco-Justicia, D. Sánchez, and N. Jebreel, "Co-utile peer-to-peer decentralized computing," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE, 2020, pp. 31–40.

[44] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 119–129.

[45] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6357–6368.

[46] L. Lyu, X. Xu, Q. Wang, and H. Yu, "Collaborative fairness in federated learning," in *Federated Learning*. Springer, 2020, pp. 189–204.

[47] W. Du, D. Xu, X. Wu, and H. Tong, "Fairness-aware agnostic federated learning," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 2021, pp. 181–189.

[48] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[49] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[50] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4387–4398.

[51] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[52] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[53] A. Blanco-Justicia, J. Domingo-Ferrer, S. Martínez, D. Sánchez, A. Flanagan, and K. E. Tan, "Achieving security and privacy in federated learning systems: survey, research challenges and future directions," *arXiv preprint arXiv:2012.06810*, 2020.

[54] D. Dua and C. Graff, "Uci machine learning repository," http://archive.ics.uci.edu/ml, 2017.

[55] R. H. Griffin, "120 years of Olympic history: Athletes and results," https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results, 2018.

[56] P. C. S. Moro and P. Rita, "Bank marketing dataset," https://archive.ics.uci.edu/ml/datasets/Bank+Marketing, 2014.

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[58] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[59] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," *Advances in neural information processing systems*, vol. 30, 2017.

[60] A. Chouldechova, "Fair prediction with disparate impact: A study of bias in recidivism prediction instruments," *Big data*, vol. 5, no. 2, pp. 153–163, 2017.

[61] Y. Zhang and J. Sang, "Towards accuracy-fairness paradox: Adversarial example-based data augmentation for visual debiasing," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 4346–4354.

[62] D. Xu, S. Yuan, L. Zhang, and X. Wu, "Fairgan: Fairness-aware generative adversarial networks," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 570–575.

[63] U. Aïvodji, F. Bidet, S. Gambs, R. C. Ngueveu, and A. Tapp, "Local data debiasing for fairness based on generative adversarial training," *Algorithms*, vol. 14, no. 3, p. 87, 2021.

[64] R. Haffar, J. Domingo-Ferrer, and D. Sánchez, "Explaining misclassification and attacks in deep learning via random forests," in *International Conference on Modeling Decisions for Artificial Intelligence*. Springer, 2020, pp. 273–285.

[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[66] G. H. Ball and D. J. Hall, "Isodata, a novel method of data analysis and pattern classification," Stanford research inst Menlo Park CA, Tech. Rep., 1965.



UNIVERSITAT
ROVIRA i VIRGILI