Applied Intelligence
https://doi.org/10.1007/s10489-022-03435-1

Explaining predictions and attacks in federated learning via random forests

Rami Haffar[1] · **David Sánchez**[1] · **Josep Domingo-Ferrer**[1]

Accepted: 23 February 2022

Abstract

Artificial intelligence (AI) is used for various purposes that are critical to human life. However, most state-of-the-art AI algorithms are black-box models, which means that humans cannot understand how such models make decisions. To forestall an algorithm-based authoritarian society, decisions based on machine learning ought to inspire trust by being *explainable*. For AI explainability to be practical, it must be feasible to obtain explanations systematically and automatically. A usual methodology to explain predictions made by a (black-box) deep learning model is to build a surrogate model based on a less difficult, more understandable decision algorithm. In this work, we focus on explaining by means of model surrogates the (mis)behavior of black-box models trained via federated learning. Federated learning is a decentralized machine learning technique that aggregates partial models trained by a set of peers on their own private data to obtain a global model. Due to its decentralized nature, federated learning offers some privacy protection to the participating peers. Nonetheless, it remains vulnerable to a variety of security attacks and even to sophisticated privacy attacks. To mitigate the effects of such attacks, we turn to the causes underlying misclassification by the federated model, which may indicate manipulations of the model. Our approach is to use random forests containing decision trees of restricted depth as surrogates of the federated black-box model. Then, we leverage decision trees in the forest to compute the importance of the features involved in the wrong predictions. We have applied our method to detect security and privacy attacks that malicious peers or the model manager may orchestrate in federated learning scenarios. Empirical results show that our method can detect attacks with high accuracy and, unlike other attack detection mechanisms, it can also *explain* the operation of such attacks at the peers' side.

Keywords Explainability · Attack detection · Machine learning · Federated learning · Surrogate model · Random decision forests

1 Introduction

The past two decades have witnessed major advances in artificial intelligence (AI) systems. Deep learning (DL) models are the cornerstone of many modern AI systems due to their ability to solve many complex tasks such as computer vision, natural language processing or speech recognition [1, 2]. Training accurate DL models requires two basic resources: i) high computational capabilities and ii) a large amount of training data. The first requirement is relatively easy to meet, but obtaining a large amount of training data is not trivial for several reasons, among which privacy concerns stand out. Indeed, personal smart devices such as smartphones are a conspicuous trove of training data, but such data are personal and sensitive. To allay the concerns of individuals about their privacy and encourage them to share their data for training DL systems, the federated learning (FL) framework has emerged [3].

Federated learning [4] is a decentralized machine learning technique that aggregates local models trained by a set of peers on their own private data to obtain a global model. More specifically, a model manager first sends an initial machine learning model to the peers; then, each peer

✉ Rami Haffar
rami.haffar@urv.cat

David Sánchez
david.sanchez@urv.cat

Josep Domingo-Ferrer
josep.domingo@urv.cat

[1] Department of Computer Engineering and Mathematics, CYBERCAT-Center for Cybersecurity Research of Catalonia, Universitat Rovira i Virgili, Av. Països Catalans 26, 43007 Tarragona, Catalonia, Spain

Published online: 13 April 2022

computes an update of the model based on her private data and sends the update back to the manager; after that, the manager aggregates these updates to obtain a new global model, which is sent back again to the peers. The process is repeated until the model converges.

Since private data do not leave the peers' devices, FL provides more intrinsic privacy to the participating peers than other approaches that require uploading the peers' data to a central server. Another advantage of FL systems is that the learning effort is distributed among the peers, instead of being centralized in a single entity.

However, like centralized deep learning, federated deep learning produces unexplainable (black-box) models, which makes it difficult for humans to understand how the model makes decisions. This lack of transparency is problematic both for the individuals affected by the decisions and for the developers who train black-box models:

- Individuals are affected by a growing number of automated decisions: credit rating, loan granting, insurance premiums, medical diagnoses, job selection, etc. While transparency measures are being implemented by public administrations worldwide, there is a danger that automated decisions become a ubiquitous black box. To protect citizens, explainability requirements are beginning to show up in legal regulations and ethics guidelines, such as article 22 of the European Union's General Data Protection Regulation (GDPR) [5], which states the right of citizens to an explanation on automated decisions on them, the European Commission's Ethics Guidelines for Trustworthy AI [6], which insist that the organizations making automated decisions be prepared to explain them at the request of the affected citizens, or the IEEE report on ethically aligned design for intelligent systems [7]. Furthermore, the recent EU proposal for a regulation on artificial intelligence (nicknamed Artificial Intelligence Act [8]) also emphasizes the explainability requirement.
- Developers would like to know how predictions are made by the black-box algorithm to make sure the algorithm takes the relevant features into account during the training phase. It may happen that wrong features are used to make predictions, such as in the well-known example of [9] where the animal is classified as a wolf if the image has a snow background and as a husky dog if the image has a grass background, because all the wolves in the training pictures were displayed in a snowy landscape and the huskies were not.

For explanations on black-box models to be practical and scalable, their generation must be automated. A standard methodology to generate explanations for predictions made by black-box AI models, such as DL models, is to build a surrogate model based on simpler and more understandable machine learning algorithms. However, the methods proposed in the literature only address the centralized black-box setting. Besides, many methods are either too simple (and, therefore, cannot suitably approximate the performance of black-box models) or too complex (and thus fail to deliver really comprehensible explanations).

1.1 Contributions and plan of this paper

The originality of our proposal lies in using random decision forests [10] as black-box FL model surrogates in order to accomplish two tasks. The first task is to create explanations for the predictions of the black-box FL model. Even though decision trees are commonly used as explainability tools for DL models in the literature (see Section 3.1), *ours is the first attempt to use random forests of limited depth to explain the (wrong) predictions of black-box models trained in a decentralized setting.* The second –and most relevant– task consists in leveraging the random forest surrogates to detect security and privacy attacks against FL model training. Again, even though a variety of FL attack detection mechanisms have been proposed in the literature (see Section 3.2), *our approach is novel in that we do not only detect the attack, but we also explain its operation at the peer's side*.

The purpose of most attacks targeting FL is to affect the model predictions, and this usually results in lower model accuracy. Therefore, we focus on wrong predictions by the FL model, because they may signal possible attacks. The empirical results we report demonstrate the effectiveness of our proposal at detecting and explaining attacks against FL.

The remainder of this paper is organized as follows. Section 2 gives background on FL, summarizes the possible attacks it may suffer, and reviews the desirable properties of surrogate models. Section 3 discusses related works devoted to explaining black-box models by means of surrogates, and the countermeasures to attacks in FL proposed so far. Section 4 describes our surrogate model based on random decision forests. Experimental results on the explainability performance of the surrogate and on the attack detection accuracy are reported in Section 5. Finally, in Section 6 we gather conclusions and sketch future research lines.

A preliminary version of this work was presented in the conference paper [11]. Whereas that work was devoted to explaining centralized deep learning models, here the focus is on federated learning and on the detection and explanation of security and privacy attacks. As a result, Sections 1 and 3.1 have been substantially rewritten; furthermore, Section 2 (except Sections 2.2), 3.2, 4 and 5 are new.

2 Preliminaries

In FL a model manager (a.k.a. server) uses a learning algorithm such as *FedAvg* [12] to learn a global shared model θ by cooperating with m participating peers. At the beginning, the server initializes an ML model, for example a neural network, with parameters θ^0, and asks the m participating peers to update the model by training it on their local data with a set of predefined hyper-parameters. The main hyper-parameters are the number of local epochs e, the local batch size bs and the learning rate α. After that, following the prescribed learning algorithm, at epoch t each peer p among the m participating peers uses her private data set D_p to train θ^t locally, and sends the resulting local model θ_p^{t+1} or the computed gradient δ_p^{t+1} back to the server. Once the server receives the local updates, it computes the global model θ^{t+1} for epoch $t+1$ by averaging the received updates. This process is iteratively repeated for a fixed number of epochs or until the global model converges.

FL is particularly vulnerable to security and privacy attacks because the server has no control over the participating peers. For example, a malicious peer may spoil the learning process by sending out bad updates to the model manager. Ideally, honest peers should be able to *detect* and *understand* such attacks, because they may significantly alter the expected behavior of the black-box model.

2.1 Attacks in federated learning

Federated learning, due to its distributed nature, is vulnerable to a number of security attacks orchestrated by malicious peers:

- *Byzantine attacks*: These attacks try to prevent the model from converging. Lack of convergence may be caused by transmission errors, attackers that alter updates from other peers, or malicious peers that submit random updates [13].
- *Poisoning attacks*: These attacks aim at misleading the global model into misclassifying a specific set of inputs. For example, in a recommendation system trained via federated learning, a possible aim would be to cause the recommendation system to suggest a specific item or to recommend it more often than due [14].

  In [15], a powerful poisoning approach based on transferable adversarial examples has been proposed: a surrogate model is leveraged to create adversarial examples that can poison a black-box deep learning model with only API access (*i.e.* without knowing its structure or having access to the data it has been trained on).
- *Label-flipping attacks*: In these attacks, the attacker is assumed to have access to a part or to all of a peer's training data. The attacker uses this access to alter (some of) the labels in the training data [16].

The model manager may also be attacked by external entities or may even mount attacks to break the privacy of the contributing peers:

- *Security attacks against the model manager*: Since the model manager receives all local updates, an external attacker who takes control of the model manager gains access to all updates and can manipulate the aggregation process [17]. Thus, the attacker can easily carry out Byzantine or poisoning attacks.
- *Privacy attacks by the model manager*: A powerful and hard-to-detect privacy attack was proposed in [18]. The authors use a multi-task Generative Adversarial Network (GAN) for Auxiliary Identification, called mGAN-AI. The typical GAN [19] consists of two parts: the generator G and the discriminator D. G is trained to generate data from the same distribution of a target data, while D is trained to classify the output of G as a real or a fake data sample. In mGAN-AI, the updates (or the model) sent by a peer are used as a discriminator, so that the model manager can take advantage of it to generate data from the same distribution as the data owned by a participating peer. This may break the privacy of the peer's data.

2.2 Requirements of surrogate models

As discussed in [20], the main properties of the explanations on the predictions of a black-box model via a surrogate model are the following:

- **Accuracy:** The accuracy of the explanations of the surrogate model should be proportional to the accuracy of the black-box model. Low-accuracy explanations can be accepted in case the black-box model is also inaccurate.
- **Fidelity:** The performance of the resulting surrogate model should be as close as possible to that of the black-box model. In other words, the accuracy loss of the surrogate model compared to the black-box model should be minimal.
- **Consistency:** The performance of the surrogate model should stay unchanged for any black-box model trained on the same data.
- **Degree of importance:** The explanation should identify the important features in the data.
- **Comprehensibility:** The explanations given by the surrogate model ought to be comprehensible to humans. Depending on the target users, more or less complex explanations can be acceptable, but in general short explanations are more desirable.

The proposed method has been tested according to the above properties (see Section 5).

3 Related work

3.1 Explaining black-box models

Several methods have been proposed in recent years to make centralized black-box DL models explainable. We focus here on proposals relying on a (simpler) surrogate model that is intrinsically explainable. In [9], an explanation method named LIME is proposed to explain the predictions of any machine learning classifier by learning a linear support vector machine (SVM). To do so, LIME creates new data samples by adding small perturbations to the features of the data sample. Then, LIME labels each obtained sample using the black-box model. Finally, it trains a separate surrogate model for each obtained sample. Therewith, LIME learns the relation between the predictions of the black box and the changes in the values of the features. LIME has recently been used in several works [21–23] to obtain interpretable AI systems.

The authors of [24] also use a linear model to explain the predictions of a black-box model on individual inputs. Their method explains the predictions of a model through the contributions of individual features based on fundamental concepts of coalitional game theory. Although linear models are intrinsically explainable, they cannot encompass the complexity of black-box models. Moreover, it is not possible to obtain linear models with acceptable accuracy when they are trained on a small subset of data, such as the data held by a participant in decentralized learning.

In [25], the authors build a surrogate model that augments the black-box decision and gives explanations by means of a Monte Carlo algorithm. They use a system based on decision rules to find the best explanation and they consider that the explanations are simple logical statements. The authors of [26] propose a system for medical scoring that uses Bayesian rule lists to extract understandable global predictors as decision lists. A decision list is comprised of a series of if-then statements that discretize the whole feature space into a set of simple and directly understandable decision statements. Decision rules or decision lists, being of linguistic nature, are easy to understand. However, the possible rules from which explanations would be built should be defined beforehand and one cannot have a rule for every individual input.

In [27] and [28], the authors use decision trees to explain the behavior of the predictions of a black-box model for individual inputs. Both works aim at centralized deep learning models, and even though [28] limits the depth of the trees, these may still be too hard to understand due to the number of features that should be encompassed for large models.

Unlike the works mentioned thus far, which deal with interpreting black-box DL models in the centralized setting, we specifically focus on explainability in the (decentralized) FL setting.

In this respect, [29] proposes using a federated forest as an intrinsically explainable alternative to black-box neural models in decentralized learning. In the federated forest, each participant builds random decision forests according to her own data. After that, the manager builds the global forest from the individual trees sent by participants. However, the aggregated forest captures less information than black-box DL models, which results in lower accuracy.

In contrast to the previous work, we use random forests as a means to *explain* black-box DL models in decentralized/federated learning, rather than as *replacement* of DL models. This allows retaining the accuracy of the underlying learning task. We also show that appropriately selecting the depths of the trees in the forest enables simultaneously achieving explainability on the one side and accuracy close to that of black-box models on the other side. Moreover, we leverage surrogates not only to explain decisions, but also to detect attacks and explain the roots of such attacks *at the peers' side.*

3.2 Countermeasures against attacks in federated learning

Security attacks by malicious peers can be detected (and filtered) by the model manager provided that the manager has access to the individual updates. Several methods to detect malicious updates have been proposed in the literature:

- *Detection of malicious peers via model metrics*: This class of methods use a custom validation set to assess the updates sent by the participating peers. The performance metrics (*e.g.*, accuracy) of the model as updated by each peer are computed on the validation set. The model manager filters out the peer updates that yield poor performance before aggregating the remaining updates to obtain the new global model.
- *Detection of malicious peers via update statistics*: As the model converges, the magnitudes of the gradients δ_p^{t+1} tend towards zero. This attack detection method checks the magnitudes of the updates and identifies as anomalous those updates outside a given range. Usually, this range is related to the interquartile range of the set of updates sent by the participating peers during training. However, since the gradients become close to zero only when models approach convergence, it is not realistic to apply this methodology throughout

the training phase [30–32]. In [33], a different way of filtering out label-flipping attacks by analyzing updates is presented. The authors propose to construct a graph of all the updates sent by the participants. This graph is built according to the parameters of each update, and Euclidean distances are calculated to identify updates that are not similar to the majority, which are filtered out. This method, however, requires a large number of costly operations per epoch.

– *Outlier-excluding aggregation*: Other countermeasures, such as Krum aggregation [34] or the coordinate-wise median [35], consist in aggregation mechanisms that exclude outlying values (which are regarded as probably malicious updates). However, these countermeasures may reject updates from honest peers whose private data distribution legitimately differs from that of the majority.

Other measures to prevent or filter out attacks exist, but they rely on cryptography, secure channels and trusted hardware[36], thereby imposing significant deployment requirements.

Some recent research works [37, 38] combine federated learning with blockchain technologies. The idea is to save global model parameters and weights in an unchangeable blockchain ledger, in order to ensure the security of the global ML model. However, although the blockchain can save the current model, it cannot protect against bad updates uploaded by malicious participants. Furthermore, very substantial computational power is needed to create each block in the blockchain.

Even though the security countermeasures discussed above can filter out malicious updates and thereby increase the model accuracy, they can only be implemented at the server side, and they do not explain the operation of the attack, that is, the data features modified by it. In this work, we use surrogate models built by the peers themselves not only to detect but also to *explain* potential security attacks.

Regarding privacy attacks by the model manager, these can be proactively prevented by the peers themselves by distorting their updates via differential privacy [39] or by securely aggregating updates of several peers before sending them to the server [40]. However, differential privacy severely deteriorates the accuracy of the learned model [41], whereas secure aggregation is incompatible with countermeasures against security attacks because it hides from the model manager the individual updates provided by the clients [42]. As an alternative, in this work we use our surrogate models to allow peers to detect potential privacy attacks mounted by the model manager. If peers can detect privacy attacks in this way, they do not need to resort to preventative measures based on update distortion or aggregation, thereby circumventing the aforementioned accuracy and security shortcomings of such measures.

4 Constructing explainable surrogate models via random forests

To fulfill the properties listed in Section 2.2, we need to build a surrogate model that can provide information about the black-box model predictions with high accuracy while keeping complexity at bay. Random decision forests [10] are a promising solution because they are able to provide accurate predictions and the decision trees they build are intrinsically understandable [28].

Whereas a single decision tree becomes too deep to be understandable if it has to make accurate decisions based on a large number of features, random forests can offer accuracy in case of many features by using several trees that have limited depth and stay thus understandable. Random decision forests also satisfy consistency because the surrogate model does not depend on the internal structure of the black-box model. Since forests are built from the actual data, they can explain the predictions of any black-box model trained on the same data.

Another distinguishing trait of random forests is that each tree in the forest focuses on a different subset of features. Our method allows deterministically choosing the trees/feature sets that have the highest influence on the predictions that match the predictions of the black-box model, thereby yielding more accurate explanations.

In FL systems, the training data are held by the participating peers. Therefore, the peers themselves can build surrogates of the global FL model by leveraging the updated global model they receive at each epoch and their own data. This will allow peers to obtain explanations of the global model's predictions. Moreover, since peers can get explanations at each epoch, they can also observe significant changes in the model's behavior, which can indicate attacks orchestrated by malicious entities. Thus, *random forests can equip peers both with explanations and attack detection capabilities.*

The proposed method for extracting explanations of the predictions of the black-box model is formalized in Algorithm 1. Peers interested in creating interpretations of black-box predictions follow the learning protocol and, at the same time, they test the black-box model with their own data. To this end, they divide the data they own into two parts. They use the first part to build the random decision forest in the first epoch and to train the black-box model at each epoch. They use the second part of the data to test the updated black-box model they receive at each epoch, and to obtain explanations (by means of the random forest they built) on the predictions the black-box model may give.

We concentrate specifically on the wrong predictions made by the black-box model, which may be an indication of malicious manipulations and attacks. For each wrong prediction of the black-box model on the peer's test data, the algorithm scans the forest and stores a vector with the feature importances of each tree in the forest whose prediction matches the wrong black-box prediction. As mentioned above, the forest diversity makes it very likely to find trees that match the (wrong) black-box predictions. Finally, the algorithm averages all the stored feature importance vectors to obtain a vector containing the average importance of every feature in causing wrong predictions.

Algorithm 1 Computation by each peer of the importance of features in the black-box model's wrong predictions.

Input: Data set X owned by peer P_X, epoch number $Epoch\_No$, black-box model $Black\_Box$ obtained via FL at the current epoch;
if $Epoch\_No == 1$ **then**
  $Train\_X, Test\_X \leftarrow Split\_Train\_Test(X)$;
  $Forest \leftarrow Build\_Random\_Forest$ $(Number\_trees, Max\_depth, Train\_X)$;
else
  Retrieve $Test\_X$ and $Forest$ computed in the first epoch;
end if
$Score\_Black\_Box \leftarrow Evaluate\_Black\_Box(Black\_Box, Test\_X)$;
$Feature\_Importances\_List \leftarrow \{\}$;
for each $Sample$ in $Test\_X$ **do**
  if $Predict(Black\_Box, Sample)$ not correct **then**
    for each $Tree$ in $Forest$ **do**
      if $Predict(Black\_Box, Sample) == Predict(Tree, Sample)$ **then**
        $Feature\_Importances\_List.Append(Tree.Feature\_Importances)$;
      end if
    end for
  end if
end for
$Feature\_Importances \leftarrow Average(Feature\_Importances\_List)$;
$Updated\_Black\_Box \leftarrow Train(Black\_Box, Train\_X)$;
Return $Score\_Black\_Box, Forest, Feature\_Importances, Updated\_Black\_Box$.

In Algorithm 1, feature importances are computed according to [43], as follows. First, let us define the notion of impurity of a data set, which is a measure of the homogeneity of the values in it. Impurity can be measured in several ways, including Gini impurity and Shannon entropy. In particular, the Gini impurity of a data set is

$$C = \sum_{\ell=1}^{L} f_\ell (1 - f_\ell),$$

where L is the number of output classification labels, and f_ℓ is the relative frequency of values belonging to the ℓ-th label. Clearly, if all the values in the data set correspond to the same label, then $C = 0$; the more diverse the values, the higher C. Now, given a decision tree, the importance of each node j in it is

$$ni_j = w_j C_j - \sum_{k \in \text{Children}_j} w_k C_k,$$

where w_j is the weighted number of samples reaching node j, C_j is the Gini impurity of the samples reaching node j, and Children_j are the children nodes of node j. Thus, the higher the homogeneity gain of a node, the more important it is, where the homogeneity gain is the reduction of impurity between the input set of the node and its output subsets (those that go to its children nodes). In other words, an important node is one that "neatly" classifies the samples that reach it. Then, the raw importance of each feature i is

$$fi_i = \frac{\sum_{j:\text{ node } j \text{ splits on feature } i} i_j}{n} \sum_{k \in \text{ all nodes}} ni_k.$$

Finally, the normalized feature importance of each feature i is a number between 0 and 1 computed as

$$normfi_i = \frac{fi_i}{\sum_{j \in \text{ all features}} fi_j}.$$

From now on, when we mention feature importances we will refer to normalized feature importances.

By relying on feature importances, Algorithm **??** attempts to detect whether a malicious manipulation or an attack happened during the training of the federated black-box model. The idea is that the algorithm compares the changes in the feature importances across the training process. Particularly, starting from the second epoch, the peer that built the surrogate model calculates the changes in the feature importances between every two consecutive epochs. In this way, she can monitor whether there are big changes in the values of the feature importances, which may indicate that an attack affecting the performance of the black-box model is in progress. To automate the attack detection task, she needs to compute the average of those changes from the start of the training to the current epoch and use the successive averages as baselines for comparison at each epoch.

Then, starting from the third epoch, Algorithm 2 checks the changes in feature importances and compares them with the average changes over the training epochs up to the current epoch. Since the global model is updated by all

peers at every training epoch, we expect small changes in feature importances. However, if the current changes are greater than the threshold α, where $\alpha > 1$, this may be an indication that an attack has happened. Parameter α controls how strictly Algorithm 2 reacts to changes in the feature importances; the smaller α, the more sensitive the algorithm is to small changes. Due to the way the training of the model naturally evolves at each single iteration (where small changes are expected), it is not recommended to set $\alpha < 1.2$. Also, if $\alpha > 2$, the algorithm will only detect the attacks after the model has been significantly affected. Thus, we recommend values within the range $1.2 \leq \alpha \leq 2$.

Algorithm 2 Detection by each peer of attacks on the federated black-box model.

Input: $Feature\_Importances[1 : Max\_Epochs][1 : Max\_Features]$, $Epoch\_No$;
$i = Epoch\_No$;
$Targeted\_Features \leftarrow \{\}$;
if $i \geq 2$ **then**
  $Changes\_Feature\_Importances[i] = |Feature\_Importances[i] - Feature\_Importances[i-1]|$; ▷ The operands in this subtraction are vectors with $Max\_Features$ components
  if $i > 2$ **then**
    $Average\_Overall\_Changes\_per\_Epoch = AVG(Changes\_Feature\_Importances[1 : i])$; ▷ $Average\_Overall\_Changes\_per\_Epoch$ is a vector with $Max\_Features$ components
    $Total\_Feature\_Change\_per\_Epoch = SUM(Average\_Overall\_Changes\_per\_Epoch)$; ▷ $Total\_Feature\_Change\_per\_Epoch$ is a scalar obtained as the sum of the components of $Average\_Overall\_Changes\_per\_Epoch$ over all features and it represents the mean of the total feature importance changes over the epochs until the current one
    $Total\_Current\_Changes = SUM(Changes\_Feature\_Importances[i])$;▷ Scalar containing the sum of all feature importance changes in the current epoch
    if $Total\_Current\_Changes \geq \alpha \times Total\_Feature\_Change\_per\_Epoch$ **then**
      for j *in* $Changes\_Feature\_Importances[i]$ **do**
        if $Changes\_Feature\_Importances[i][j] \geq \beta \times Total\_Current\_Changes$ **then**
          $Targeted\_Features.append(Feature[j])$;
        end if
      end for
    end if
  end if
end if
Return $Targeted\_Features$.

On the other hand, by looking at changes at the feature level, the algorithm is able to find out which features were attacked: it selects those features having an impact on changes in the feature importance greater than β, where $\beta < 1$ is the threshold to select the affected feature according to the weight of the single feature in the total changes in the feature importance. The set of selected features explains how the attack operated. Since the total weight of changes in the feature importances is normalized to 1, we recommend setting $0.15 \leq \beta \leq 0.4$, which means that a feature should have an influence on the total changes of at least 15% to be considered altered. This weight can be tuned according to the number of features.

5 Experimental results

In this section, we report experimental results on the explanatory usefulness of our surrogate model and on the ability of our algorithms to detect attacks against FL based on two different real data sets, one them numerical and the other containing a mix of categorical and numerical attributes. The source code of the reported experiments is available for reproducibility purposes.[1]

5.1 Data sets

We considered the following data sets:

– *Numerical data set*: We used the "PAMAP2 Physical Activity Monitoring" (a.k.a. Activity) data set from the UCI Machine Learning Repository.[2] This data set contains continuous measurements of 3 inertial body sensors (placed on the arm, chest, and ankle) and a heart-rate monitor worn by 9 subjects who performed 18 different activities such as walking, cycling, watching TV, etc. First, as recommended by the releasers of the data set [44], we discarded the transient activity (*e.g.* going to the next activity location). Second, for simplicity, we mapped the various types of activity into two categories indicating whether the activity involved displacement or not (*e.g.* walking and cycling were mapped to "displacement" and watching TV to "not displacement"). As a result, we obtained a data set containing 1,942,872 records of which 1,136,540 records were labeled as "displacement" and 806,332 as "not displacement". Each record contained 54 numerical attributes corresponding to timestamp, label, heart rate, and 17 sensor data feeds for each of the 3 inertial sensors. Given an unlabeled record, the classification task consisted in deciding whether

[1] https://github.com/RamiHaf/Explainable-Federated-Learning-via-Random-Forests

[2] https://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring

the subject was performing at that instant an activity involving physical displacement.

– *Data set with a mix of categorical and numerical attributes*: We used the Adult data set, which is a standard data set hosted in the UCI Machine Learning Repository.[3] Adult contains 48,842 records of census income information and it includes 14 attributes reporting both numerical and categorical values. For each categorical attribute, we recoded categories as numbers to obtain a numerical version of the attribute.

In the experiments below, we used 100 peers, except in attack detection, where we tried various numbers of peers (30, 50 and 100). For the two data sets, 70% of the data were randomly split into as many disjoint shards as the number of peers and each peer was assigned a different shard. Specifically, for 100 peers, each shard from Activity consisted of 13,600 records and each shard of Adult consisted of 342 records.

The model manager kept the remaining 30% of the data set for validation purposes. This validation data set was used by the model manager at the end of each epoch to test the performance of the model. At the same time, since the model was not trained on these data, they could be used for the final evaluation of the model.

Each peer P_X then further split her shard X into two parts: 70% of the data were used as $Train\_X$ for locally training the surrogate random forest, whereas the remaining 30% were used as $Test\_X$ to validate the black-box model and compute the feature importances (as per Algorithm 1).

5.2 Results on the explainability of the predictions of the black-box model

To test the desirable properties of surrogate models introduced in Section 2.2, we employed 6 federated black-box models with different internal structures. We built them by using the Keras[4] library [45]. The first model (*Black-box1*), which was employed in [28] on the same data sets, had two hidden dense layers with 64 and 50 neurons, respectively. The second model (*Black-box2*) had three hidden layers, each with 100 neurons. The third (*Black-box3*) contained five hidden layers, with 100, 50, 30, 20 and 10 neurons, respectively. The fourth (*Black-box4*) also had five hidden layers, with 10, 20, 50, 20 and 6 neurons, respectively. The fifth model (*Black-box5*) had three hidden layers with 10, 6 and 4 neurons, respectively. Finally, the sixth (*Black-box6*) had seven hidden layers with 10, 40, 80, 30, 10, 6 and 4 neurons, respectively. We ran these models for 10 epochs. We used the Adam optimization algorithm [46], with batch size 32 and learning rate 0.001, with five local epochs for each peer. Using six different black-box models allowed us to test the fidelity and accuracy of the proposed method, and also to compare the explanations it provided on different model architectures performing the same task. At the peers' side, we built random forests using the command RandomForestClassifier[5] from the sklearn library with 1000 trees of maximum depth 5 and average size 62 nodes.

In what follows, we report the experiments we carried out and the results we obtained on each of the properties listed in Section 2.2.

5.2.1 Fidelity and accuracy

Table 1 compares the accuracy of the six federated black-box models after converging and the random forest surrogate built by one peer on a shard from each of the two data sets introduced above. Table 1 also reports the fidelity score of the surrogate model vs. each of the black-box models.

To train the black-box models, we used 70% of records of the corresponding full data set, whereas the random forests were trained on 70% of the shard of that data set held by the peer. Since the shards were created by randomly splitting the full data set, they can be assumed to consist of independent and identically distributed data, and to be representative of the full data set.

The models with simpler architecture attained higher accuracy on the Activity data set: *Black-box1*, with only 2 hidden layers, achieved the highest accuracy (96.55%); *Black-box2* and *Black-box5*, with 3 hidden layers each, reached accuracy 93.32% and 93.11%, respectively; *Black-box3*, *Black-box4* and *Black-box6*, with 5, 5 and 7 hidden layers, respectively, achieved lower accuracy than the simpler models. On the other hand, for the Adult data set, *Black-box4*, with its 5 hidden layers having each a small number of neurons, attained the highest accuracy (87.56%); *Black-box3*, with the same architecture as *Black-box4* but with more neurons per layer, achieved the second-highest accuracy (84.14%); the rest of the models offered similar accuracy values, between 82.15% and 83.73%.

The fidelity reported in Table 1 is the absolute difference between the accuracy of each black-box model and the accuracy of the surrogate model. The smaller the fidelity score, the closer the performance of the surrogate model to the performance of the respective black-box model. For both data sets, the fidelity score of the surrogate model was at most 6.71%, which indicates that the surrogate model offered good fidelity to the black-box models.

[3] http://archive.ics.uci.edu/ml/datasets/Adult

[4] https://keras.io/

[5] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Table 1 Accuracy of the black-box models and the surrogate random forest for the Activity and Adult data sets

	Activity data set		Adult data set	
model name	Accuracy	Fidelity	Accuracy	Fidelity
Black-box1	96.55%	5.85%	82.28%	1.43%
Black-box2	91.32%	0.62%	82.15%	1.3%
Black-box3	89.33%	1.37%	84.14%	3.29%
Black-box4	88.58%	2.12%	87.56%	6.71%
Black-box5	93.11%	2.41%	83.73%	2.88%
Black-box6	90.74%	0.04%	83.51%	2.66%
Surrogate model	90.7%	N/A	80.85%	N/A

The fidelity of the surrogate model to each black-box model is also displayed (a smaller score indicates more fidelity)

Table 3 Accuracy of the *Black-box1* model after removing the features with the highest importance on the wrong predictions, as identified by the proposed method and LIME, in comparison with the accuracy after removing random subsets of features. Subset1 and Subset2 are example random subsets. Subset1 consists of *(magnetometer_z_chest, magnetometer_z_ankle)* for the Activity data set, and *(native-country, workclass)* for the Adult data set. Subset2 consists of *(acceleration_16_z_ankle, acceleration_6_y_chest, acceleration_16_x_hand)* for the Activity data set, and *(relationship, occupation, marital-status)* for the Adult data set

Data set	Activity data set	Adult data set
Proposed method	99.67%	84.15%
LIME	97.38%	80.37%
Subset 1	95.65%	79.72%
Subset 2	94.75%	78.39%
Average accuracy over 15 removed random subsets of features	95.22%	79.16%

5.2.2 Degree of importance

Our approach specifically focuses on computing the importance of features in the cases where the black-box model made a wrong prediction.

We compared the explanations provided by our method with those obtained with the LIME method [9] mentioned in Section 3. LIME creates new samples of the data (as we do with adversarial examples) by adding small perturbations to the features of the data samples. Then, for each sample LIME intends to explain, it labels the newly created data samples using the black-box model, and trains a linear SVM surrogate model to learn the relation between the value of the features and the corresponding prediction of the black-box model. It classifies the features into two labels: one contains the features that support the prediction of the black box and the other the features that contribute against it. Feature importances are computed by calculating the mean and standard deviation for the values of the feature in the original sample and the new perturbed samples and discretizing them into quartiles.

Table 2 reports the ten highest feature importances identified by our method (Algorithm 1) on the Activity and Adult data sets, and the corresponding importances computed by LIME.

We can see that both LIME and our method agree on the features having the greatest importance. The difference is that, whereas LIME gives a dominant weight to just a single feature (*magnetometer_z_chest* for the Activity data set and *capital_gain* for the Adult data set), our method prioritizes a set of features (*gyroscope_z_ankle*, *magnetometer_z_chest* and *acceleration_6_x_ankle* for the Activity data set, and *age*, *educational-num*, *capital-gain* and *hours-per-week* for the Adult data set).

Since no approach in the literature allows comparing two explanation methods and since our proposed method is only

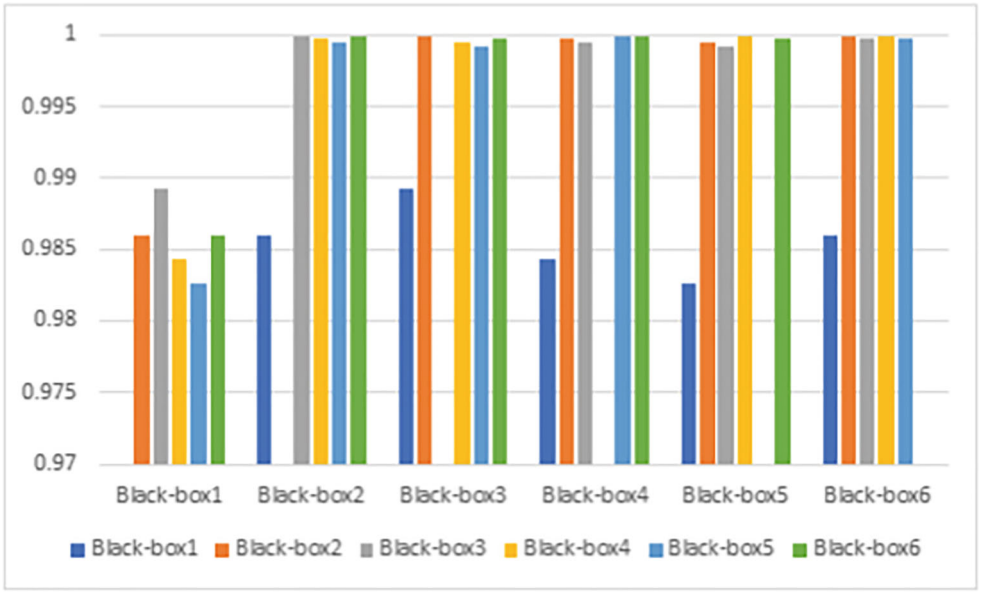
Table 2 Ten highest feature importances for wrong *Black-box1* predictions computed by a peer using Algorithm 1 on the two data sets, and corresponding importances computed by the LIME algorithm

Activity data set			Adult data set		
Feature	Proposed method	LIME	Feature	Proposed method	LIME
gyroscope_z_ankle	12.38%	3%	*age*	18.17%	8%
magnetometer_z_chest	11.43%	17%	*educational-num*	15.63%	10%
acceleration_6_x_ankle	8.13%	2%	*capital-gain*	15.24%	68%
acceleration_16_x_ankle	7.63%	0%	*hours-per-week*	14.16%	7%
magnetometer_x_hand	6.15%	3%	*marital-status*	9.33%	2%
gyroscope_x_ankle	5.26%	2%	*relationship*	8.19%	3%
acceleration_6_y_chest	4.19%	1%	*occupation*	6.47%	1%
acceleration_16_y_chest	3.90%	0%	*race*	4.86%	0%
acceleration_6_z_chest	3.60%	2%	*workclass*	4.41%	3%
magnetometer_z_ankle	3.53%	4%	*native-country*	1.61%	4%

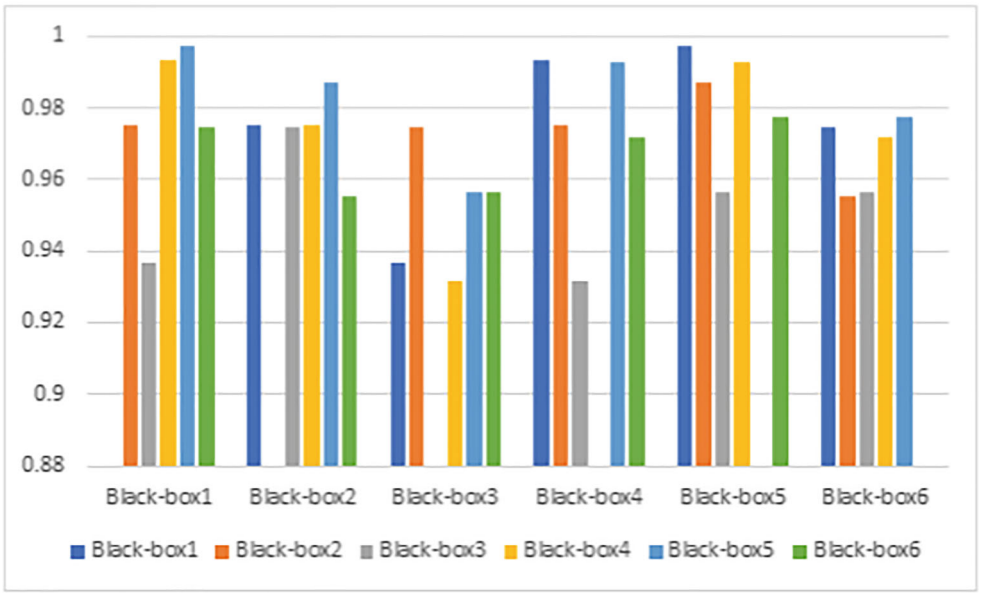
concerned with investigating the reasons of the black-box wrong predictions, we devised an indirect comparison. Our idea was to evaluate the accuracy benefits obtained after removing the features identified by the two explanation methods (ours and LIME) as most important in wrong predictions.

To this effect, we retrained and tested the accuracy of the *Black-box1* model on modified versions of the data sets after removing the features identified by each explanation method as being the most important for wrong predictions (see Table 2); feature removal affected only the input layer of the black-box model, but the rest of its structure was maintained. LIME identified just one clearly dominant feature (with importance above 10%). Thus, we removed the single most important feature identified by LIME on both data sets. In contrast, our method identified two features in Activity and four features in Adult with importance above 10%, which we removed. Afterwards, to confirm that the above feature removal was meaningful, we compared the accuracy values obtained by removing the above most important features with the accuracy values obtained by removing random subsets of features.

Table 3 reports the resulting accuracy values for the two data sets after removing: (i) the most important



(a) Feature importance correlations for the Activity data set



(b) Feature importance correlations for the Adult data set

Fig. 1 Pearson correlations between the feature importances obtained by the surrogate for each pair of black-box models on the Activity and Adult data sets

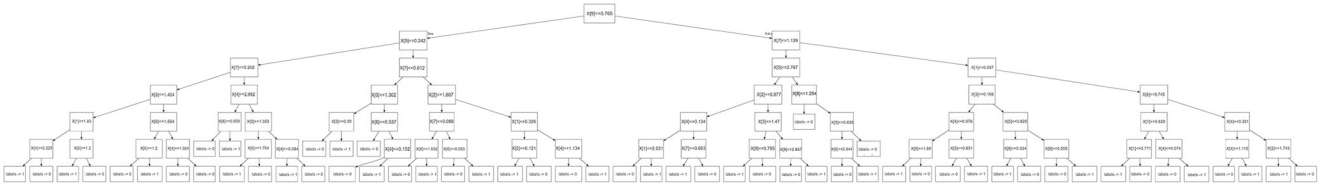


Fig. 2 Structure of a tree from a random forest with depth limit 6 built on a synthetic data set

features detected by both methods, and (ii) random subsets of features. For both data sets, removing features based on the explanations produced by our method resulted in better accuracy than removing features based on LIME explanations. We also see that removing the features we identified had a significant effect on the model accuracy (improvement from 96.55% in Table 1 to 99.67% in Table 3 for the Activity data set, and from 82.28% in Table 1 to 84.15% in Table 3 for the Adult data set).

In contrast, removing random subsets of features worsened the model accuracy with respect to Table 1. These results suggest that our method yields explanations that are not only meaningful, but more meaningful than those offered by LIME.

5.2.3 Consistency

The surrogate model should consistently perform on any black-box model trained on the same data set. To measure the consistency of the surrogate model, we computed the Pearson correlation between the set of feature importances obtained by the surrogate for each pair of black-box models.

Figure 1 reports the correlation values for each pair of models. We can see that the correlation is very high (above 0.93) for all pairs and perfect in many of them, which indicates that the relative importances obtained by the surrogate are consistent across models.

5.2.4 Comprehensibility

The outcomes of the surrogate model should be understandable by humans. In addition to the feature importances associated with wrong predictions, our method produces a collection of trees with limited depth that are easy to interpret. In particular, to obtain a visual representation of the behavior of the black-box model, we can select one or several trees that agree with the predictions of the black-box model. Figure 2 shows the structure of a tree selected from a random forest limited to depth 6 built on a synthetic data set with 10 features. In contrast, if we build a surrogate consisting of a single tree covering the whole set of features in the data set, we obtain the very large structure shown in Fig. 3, which has 20 as its maximum depth.[6] Notice that in this comparison we are just interested in the size of the trees, rather than in their content. It is clear that trees in the random forest, being smaller and shallower, are easier to understand while still being representative of the behavior of the black-box model (because we can choose the trees that best agree with the black-box model).

5.3 Results on the attack detection performance

In this section, we evaluate the performance of our method at detecting security and privacy attacks on FL. We considered attacks conducted by other peers or by the model manager himself. Moreover, we compared the performance of our method with other attack detection mechanisms described in Section 3.2.

5.3.1 Detection of security attacks

To test the effectiveness of our method at detecting security attacks initiated by malicious peers (or even by the model manager in case an attacker took control over her), we applied Algorithm 2 on the side of a peer during the training of the federated *Black-box1* model on the Adult data set. We took $\alpha = \frac{3}{2}$, which is a neither too sensitive nor too insensitive attack detection threshold. We considered the three types of security attacks described in Section 3.2: Byzantine attack, poisoning attack and label-flipping attack. We ran several experiments with different numbers of participants: 30, 50, and 100 peers. In all cases, the attack was initiated by a single peer in the fourth epoch and ended in the seventh epoch. In Byzantine attacks, malicious peers replaced 40% of the data they had with random values in the same range as the original data. In a poisoning attack, the attacker targeted three data features, namely *relationship*, *marital-status* and *capital-gain*, by replacing the original values of these features in 60% of the data samples with random values in the same range. In a label-flipping attack,

[6] We have used a synthetic data set with 10 features for this illustration, because using the Activity or the Adult data sets would yield a standalone tree too large and deep to depict. *E.g.* for the Adult data set we would get a tree with depth 41.

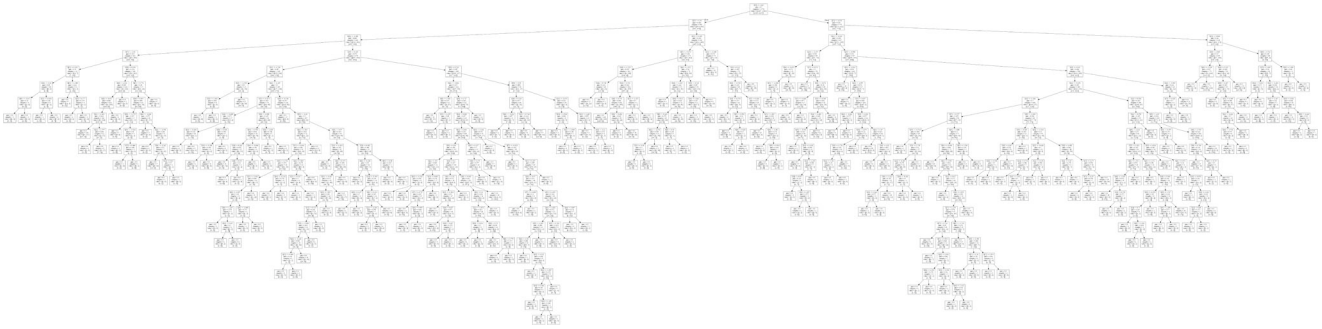


Fig. 3 Standalone decision tree covering the whole set of features of the same synthetic data set used in Fig. 2

the attacker flipped label 0 to label 1 in 50% of the samples that had label 0.

Table 4 reports the difference between the average feature importances at each epoch and the importances of the features that were attacked. The proposed method detected attacks if there was a significant increase in such difference between two consecutive epochs. In most cases, this significant difference manifests one epoch after the attack begins, that is, the attack begins in the fourth epoch and is detected in the fifth epoch. However, in some cases, such as the Byzantine attack with 50 peers or the poisoning attack with 100 peers, the attack was detected in the sixth epoch. The reason is that the influence of the attack may not be immediately noticeable if it is conducted by a single peer in a large network. In other cases, the proposed method detects the security attacks in post-attack epochs, such as the ninth or tenth epochs. This is because, once the attack is over, model training continues on the correct data and the weights of the model undergo a big shift toward correct predictions. This facilitates post-attack detection.

Each time the proposed method detects an attack, it reports the features that are most suspicious of having been attacked. We considered that a feature has been attacked if the change in its importance accounts for more than 25% of the total changes in the feature importances. Thus, we used $\beta = \frac{1}{4}$ as the threshold for detecting the attacked features. Note that *Byzantine and label-flipping attacks* did not aim at specific features, but rather at impairing the overall system performance. Yet, these attacks were more effective on some features than on others, and our method reports the most perturbed features. In the case of the *poisoning attack*, however, three specific features were attacked, and our method reported two out of the three every time: with 30 and 100 participating peers, *marital-status* and *capital-gain* were correctly detected, whereas, with 50 peers, *capital-gain* and *relationship* were correctly detected.

We then calculated the performance of our method at detecting attacks on the two data sets and for all types of attacks and network configurations described above. For each case studied, we considered different numbers of attacking peers; since the majority of the peers involved were regarded as honest, we took the number of attackers to be between 1 and $\frac{m}{2} - 1$, where m was the number of peers participating in the training. We used a threshold $\alpha = \frac{3}{2}$ for all the experiments. We compared the performance of our method against the performance of malicious peer detection via update statistics described in Section 3.2, even though this countermeasure can only be implemented at the server side. The reason for choosing the latter method is that it is also based on distances and it does not require a lot of computing power. We considered the detection to be correct if the attack was detected while it was taking place and one malicious peer was correctly reported as the attacker. The duration of the attacks was four epochs, starting at the fourth epoch, and ending at the seventh epoch. We calculated the percentage of the cases where the two methods being compared correctly detected suspicious activities.

Table 5 reports the detection rate of our proposed method and the method based on update statistics:

- For the Activity data set, the detection rate of our method was low for Byzantine attacks, because these attacks did not affect the accuracy of the model on this data set, and hence were less noticeable. However, for poisoning and label-flipping attacks, the detection rate of our method was above 90%.
- For the Adult data set, the detection rate was over 90% for all numbers of peers and attacks, except for 100 peers under Byzantine attacks, where detection was only 75.51%.

In general, the detection performance of our method improved when there were fewer participating peers in the network. The reason is that the effect of a single altered update is more noticeable when fewer peers are involved in the training process.

Boldface numbers in Table 5 indicate the best detection rate for a given number of peers, data set and attack configuration. We can see that our method outperformed the detection of malicious peers via the update statistics approach for all attacks and for all numbers of participants

Table 4 Differences between average feature importances and importances of attacked features for different attacks on the Adult data set

	Byzantine attack			*Poisoning attack*			*Label-flipping attack*		
	30 peers	50 peers	100 peers	30 peers	50 peers	100 peers	30 peers	50 peers	100 peers
Epoch 2	0.022	0.011	0.017	0.019	0.032	0.026	0.014	0.024	0.017
Epoch 3	0.005	0.017	0.012	0.004	0.003	0.004	0.003	0.001	0.006
Epoch 4	0.001	0.003	0.007	0.001	0.0003	0.001	0.004	0.004	0.002
Epoch 5	**0.02**	0.002	**0.021**	**0.018**	**0.029**	0.005	**0.042**	**0.038**	**0.056**
Epoch 6	**0.041**	**0.023**	**0.018**	**0.014**	0.001	**0.017**	0.02	0.004	0.002
Epoch 7	0.004	0.003	0.004	0.007	0.004	**0.032**	0.005	0.013	**0.056**
Epoch 8	0.002	0.005	0.004	0.001	0.0003	0.005	0.012	**0.035**	**0.048**
Epoch 9	0.018	**0.013**	0.009	0.004	**0.036**	0.008	0.003	**0.05**	0.005
Epoch 10	0.004	**0.032**	**0.025**	0.001	0.0007	**0.027**	**0.024**	0.006	0.002
Reported features	*educational-num age*	*capital-gain educa-tional-num capital-loss*	*occupation race*	*capital-gain marital-status*	*capital-gain educational-num relationship*	*capital-gain marital-status occupation*	*age relationship*	*capital-gain age occupation*	*capital-gain workclass*

Values in boldface correspond to the epoch in which the attack was detected

with one exception: Byzantine attacks with 100 peers in the Activity data set. This exception is due to the fact that Byzantine attacks with that number of peers and that data set had little impact on the accuracy of the black-box model, in part because the number of attackers was small compared to the number of peers.

We also computed the false negative rate (FNR) and the false positive rate (FPR) of attack detection. A false negative occurs if the attack is not detected, or none of the attacking peers is reported by the attack detection via update statistics method, or none of the attacked features is reported by our method. To assess the FPR, we tested both methods while training the model without any attack; we repeated the test 50 times for each number of participating peers to calculate an accurate error rate. Figure 4a and b depict the FNR for both detection methods under the three above-mentioned attacks (Byzantine, poisoning and label flipping) on the Activity and the Adult data sets, respectively. The results are consistent with those of Table 5. Even though the FNR under the Byzantine attack was rather high for our detection method, it was lower than with the detection method based on update statistics. On the other hand, under the poisoning and label-flipping attacks, our proposed method exhibited a very low FNR in comparison with the other method. Figure 4c and d depict the FPR for both detection methods, the three attacks and the two data sets. We can see that the attack detection method via update statistics reports a list of malicious peers at every round of the training even if there were no attacks, while the FPR of the proposed method was less than 2% on the Activity data set and less than 18% on the Adult data set. The reason of our FPR being lower on Activity than on Adult is that the former data set has more features, which results in smaller changes in the importance of each feature across the training epochs.

Although our proposed method was more efficient at detecting attacks, it might fail in front of attacks that continue non-stop from the first epoch to the end of training. In such case, the importances of the features would not vary significantly in any single epoch and, therefore, the attack would not be detected by the peers. However, since the updates that an attacker following this pattern would send during the whole training process would be very different from the updates generated by honest peers (especially in advanced stages of training), the server can detect and filter out those malicious updates with state-of-the-art server-side attack detection methods (mentioned in Section 3.2).

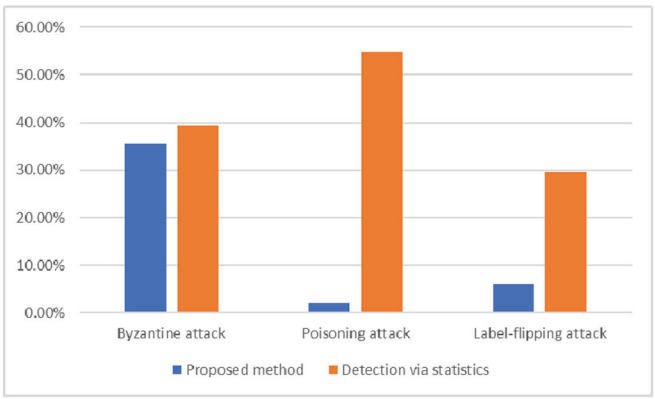
5.3.2 Detection of privacy attacks

To test the ability of our method to detect attacks against privacy, we used the mGAN-AI attack described in Section 3.2. In this attack, the model manager tries to break the privacy of a targeted peer by generating data from

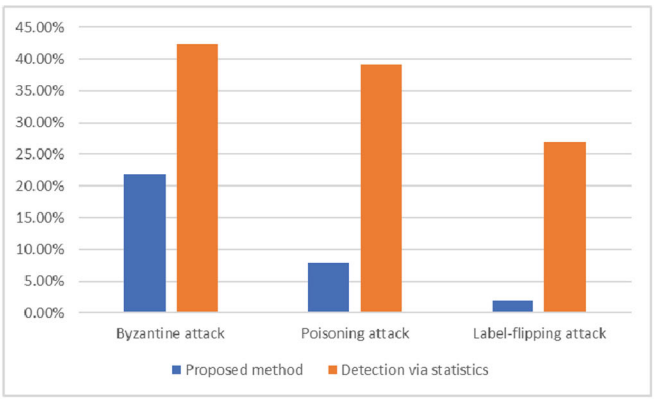
Table 5 Attack detection rate of the proposed method and the update statistics method on the two data sets

	Data set	Attack type	30 peers	50 peers	100 peers
Proposed method		*Byzantine*	**71.42%**	**41.66%**	48.97%
	Activity	*Poisoning*	**100%**	**100%**	**95.91%**
		Label-flipping	**100%**	**91.66%**	**93.87%**
	Adult	*Byzantine*	**100%**	**95.83%**	**75.51%**
		Poisoning	**92.85%**	**91.66%**	**93.87%**
		Label-flipping	**100%**	**100%**	**97.95%**
Detection via statistics		*Byzantine*	64.29%	37.5%	**71.42%**
	Activity	*Poisoning*	28.57%	37.5%	48.95%
		Label-flipping	57.14%	66.66%	63.26%
	Adult	*Byzantine*	85.71%	62.5%	63.26%
		Poisoning	64.28%	50%	61.22%
		Label-flipping	71.42%	75%	77.55%

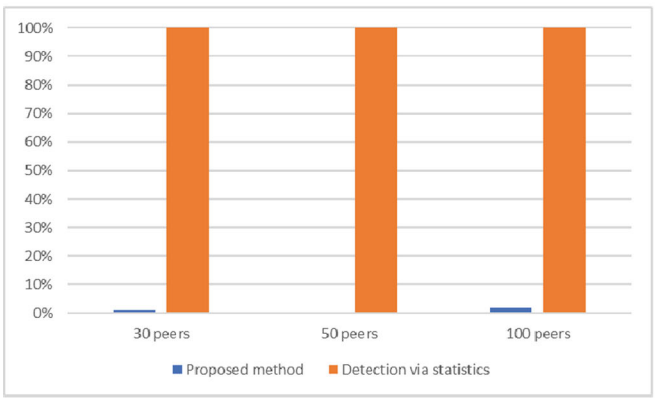
The detection rate of the best-performing method for a certain configuration of data set, attack type and number of peers is depicted in boldface



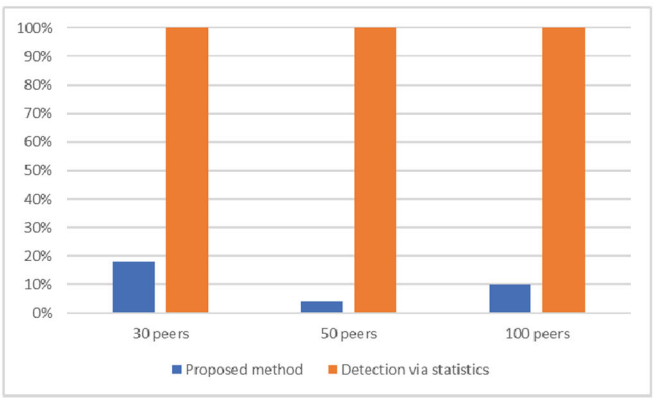
(a) FNR of attack detection on the Activity data set



(b) FNR of attack detection on the Adult data set



(c) FPR of attack detection on the Activity data set



(d) FPR of attack detection on the Adult data set

Fig. 4 FNR and FPR of attack detection with the proposed method and with the method based on update statistics

Table 6 Differences between average feature importances and importances of attacked features for the model manager's mGAN-AI privacy attack on the Adult data set

	30 peers	50 peers	100 peers
Epoch 2	0.037	0.014	0.046
Epoch 3	0.006	0.006	0.017
Epoch 4	0.018	0.004	0.008
Epoch 5	0.0003	0.004	0.005
Epoch 6	0.0001	0.002	0.003
Epoch 7	0.013	0.07	0.019
Epoch 8	0.002	**0.02**	**0.023**
Epoch 9	**0.015**	**0.022**	0.003
Epoch 10	0.002	0.004	0.015
Reported features	*marital-status age*	*marital-status hours-per-week educational-num*	*relationship occupation*

Values in boldface correspond to the epoch in which the attack was detected

the peer's data distribution. We considered the same three scenarios with 30, 50, and 100 peers by training *Black-box1* on the Adult data set and we used the same thresholds $\alpha = \frac{3}{2}$ and $\beta = \frac{1}{4}$ as in the above-mentioned experiments on security attacks.

In this case, the model manager initiates the attack in the sixth epoch, when the network is close to converging. The attack generates data from the distribution of the two features *marital-status* and *relationship*.

In Table 6 we can see the performance of our method at detecting the mGAN-AI attack. For 50 and 100 participating peers, our method detected the attack one epoch after it was initiated (in the seventh epoch). For 30 peers, the attack was detected in the eighth epoch. Our method was able to detect this attack because the model manager modified the model before distributing it to the participants by including the data created by the generator G on the targeted peer. Because of this change, our method was able to detect the manipulated model and make the targeted peer aware of the attack, so that she could decide to stop sharing further updates to avoid disclosing her data. Also, our method reported one valid target feature out of two in each case, which made the information provided about the attack very valuable to understand the attacker's intent.

6 Conclusions and future research

We have presented an approach based on random decision forests that provides explanations for the (mis)behavior of federated black-box models. Our method can detect and explain attacks orchestrated by the federated training participants. We have argued both conceptually and empirically that random forests inherently fulfill the main desirable properties of surrogate explainable models, namely fidelity, accuracy, consistency, degree of importance and comprehensibility. In particular, we have focused on investigating and explaining wrong predictions. To this end, we provide decision trees that explain such wrong predictions and we report the feature importances associated with those predictions. We have shown that this information can be leveraged to detect security and privacy attacks in machine learning, and specifically in federated learning. Our method is able to discover a variety of attacks with high detection rates, and it clearly outperforms specially tailored attack detection mechanisms based on update statistics. Moreover, unlike the countermeasures proposed in the literature, our method not only can detect the attack but can also explain it by identifying the features that seem to have been affected by the attack.

As future work, we intend to test the performance of our approach on data that are not identically and independently distributed. We also plan to apply our method to fully decentralized machine learning scenarios [47].

Acknowledgements Partial support to this work has been received from the European Commission (projects H2020-871042 "SoBigData++" and H2020-101006879 "MobiDataLab"), the Government of Catalonia (ICREA Acadèmia Prizes to J. Domingo-Ferrer and D. Sánchez), and from the Spanish Government (project RTI2018-095094-B-C21 "Consent").

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

1. Deng L, Yu D (2014) Deep learning: methods and applications. Found Trends Signal Process 7(3–4):197–387
2. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
3. Konečnỳ J, Richtárik P (2018) Randomized distributed mean estimation: accuracy vs. communication. Front Appl Math Stat 4:62
4. Konečnỳ J, McMahan HB, Felix XY, Richtárik P, Suresh AT, Bacon D (2016) Federated learning: Strategies for improving communication efficiency
5. Regulation GDP (2016) Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46. Official J Eur Union (OJ) 59(1-88):294
6. European Commission's High-Level Expert Group on Artificial Intelligence: Ethics Guidelines for Trustworthy AI (2019). https://ec.europa.eu/futurium/en/ai-alliance-consultation
7. Shahriari K, Shahriari M (2017) IEEE standard review. Ethically aligned design: a vision for prioritizing human wellbeing with artificial intelligence and autonomous systems. In: 2017 IEEE Canada International Humanitarian Technology Conference (IHTC). IEEE, pp 197–201
8. (2021). Proposal for a Regulation of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain Union legislative acts. https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence
9. Ribeiro MT, Singh S, Guestrin C (2016) "Why should I trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 1135–1144
10. Kam HT (1995) Random decision forests. In: 3rd International Conference on Document Analysis and Recognition, vol 1416. Montréal, Canada, pp 278–282
11. Haffar R, Domingo-Ferrer J, Sánchez D (2020) Explaining misclassification and attacks in deep learning via random forests. In: International Conference on Modeling Decisions for Artificial Intelligence-MDAI 2020. Springer, pp 273–285
12. McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. PMLR, pp 1273–1282
13. Lamport L, Shostak R, Pease M (2019) The Byzantine generals problem. In: Concurrency: the Works of Leslie Lamport, pp 203–226
14. Fang M, Cao X, Jia J, Gong N (2020) Local model poisoning attacks to Byzantine-robust federated learning. In: 29th USENIX Security Symposium (USENIX Security 20), pp 1605–1622
15. Zhong Y, Deng W (2020) Towards transferable adversarial attack against deep face recognition. IEEE Trans Inf Forensic Secur 16:1452–1466
16. Taheri R, Javidan R, Shojafar M, Pooranian Z, Miri A, Conti M (2020) On defending against label flipping attacks on malware detection systems. Neural Comput Appl:1–20
17. Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R (2019) Advances and open problems in federated learning. arXiv:1912.04977
18. Hitaj B, Ateniese G, Perez-Cruz F (2017) Deep models under the GAN: information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security-CCS'17, pp 603–618
19. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2020) Generative adversarial networks. Commun ACM 63(11):139–144
20. Molnar C (2020) Interpretable machine learning. Lulu.com
21. Magesh PR, Myloth RD, Tom RJ (2020) An explainable machine learning model for early detection of Parkinson's disease using LIME on DaTscan imagery. Comput Biol Med 126:104041
22. Torcianti A, Matzka S (2021) Explainable artificial intelligence for predictive maintenance applications using a local surrogate model. In: 2021 4th International Conference on Artificial Intelligence for Industries (AI4I). IEEE, pp 86–88
23. Hakkoum H, Idri A, Abnane I (2020) Artificial neural networks interpretation using LIME for breast cancer diagnosis. In: World Conference on Information Systems and Technologies. Springer, pp 15–24
24. Strumbelj E, Kononenko I (2010) An efficient explanation of individual classifications using game theory. J Mach Learn Res 11:1–18
25. Turner R (2016) A model explanation system. In: 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP). IEEE, pp 1–6
26. Letham B, Rudin C, McCormick TH, Madigan D (2015) Interpretable classifiers using rules and Bayesian analysis: building a better stroke prediction model. Ann Appl Stat 9(3):1350–1371
27. Singh S, Ribeiro MT, Guestrin C (2016) Programs as black-box explanations. arXiv:1611.07579
28. Blanco-Justicia A, Domingo-Ferrer J, Martínez S, Sánchez D (2020) Machine learning explainability via microaggregation and shallow decision trees. Knowl-Based Syst 194:105532
29. Liu Y, Liu Y, Liu Z, Liang Y, Meng C, Zhang J, Zheng Y (2020) Federated forest. IEEE Transactions on Big Data. early access
30. Tukey JW (1977) Exploratory data analysis, vol 2. Reading, MA
31. Domingo-Ferrer J, Blanco-Justicia A, Sánchez D, Jebreel N (2020) Co-utile peer-to-peer decentralized computing. In: 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID). IEEE, pp 31–40
32. Jebreel N, Blanco-Justicia A, Sánchez D, Domingo-Ferrer J (2020) Efficient detection of Byzantine attacks in federated learning using last layer biases. In: International Conference on Modeling Decisions for Artificial Intelligence-MDAI 2020. Springer, pp 154–165
33. Cao D, Chang S, Lin Z, Liu G, Sun D (2019) Understanding distributed poisoning attack in federated learning. In: 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, pp 233–239
34. Blanchard P, El-Mahdi E-M, Guerraoui R, Stainer J (2017) Machine learning with adversaries: Byzantine tolerant gradient descent. In: Advances in Neural Information Processing Systems, pp 119–129
35. Yin D, Chen Y, Kannan R, Bartlett P (2018) Byzantine-robust distributed learning: Towards optimal statistical rates. In: International Conference on Machine Learning. PMLR, pp 5650–5659
36. Chen Y, Luo F, Li T, Xiang T, Liu Z, Li J (2020) A training-integrity privacy-preserving federated learning scheme with trusted execution environment. Inf Sci 522:69–79

37. Kim H, Park J, Bennis M, Kim S-L (2019) Blockchained on-device federated learning. IEEE Commun Lett 24(6):1279–1283
38. Salah K, Rehman MHU, Nizamuddin N, Al-Fuqaha A (2019) Blockchain for AI: Review and open research challenges. IEEE Access 7:10127–10149
39. Wei K, Li J, Ding M, Ma C, Yang HH, Farhad F, Jin S, Quek TQS, Poor V (2020) Federated learning with differential privacy: algorithms and performance analysis. IEEE Trans Inf Forensic Secur 15:3454–3469
40. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp 1175–1191
41. Domingo-Ferrer J, Sánchez D, Blanco-Justicia A (2021) The limits of differential privacy (and its misuse in data release and machine learning). Commun ACM 64(7):33–35
42. Blanco-Justicia A, Domingo-Ferrer J, Martínez S, Sánchez D, Flanagan A, Tan KE (2021) Achieving security and privacy in federated learning systems: survey, research challenges and future directions. Eng Appl Artif Intell 106:104468
43. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830
44. Reiss A, Stricker D (2012) Introducing a new benchmarked dataset for activity monitoring. In: 16th International Symposium on Wearable Computers. IEEE, pp 108–109
45. Gulli A, Pal S (2017) Deep learning with Keras. Packt Publishing Ltd
46. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015. Conference Track Proceedings, San Diego. arxiv:1412.6980
47. Lian X, Zhang C, Zhang H, Hsieh C-J, Zhang W, Liu J (2017) Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In: Advances in Neural Information Processing Systems, pp 5330–5340

Rami Haffar received his master's degree in information security and artificial intelligence from Universitat Rovira i Virgili, Tarragona, Catalonia, in 2019. He is a Ph.D. candidate and FPI Predoctoral grant holder at the Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Tarragona, Catalonia.

David Sánchez is a Serra Hunter Professor and an ICREA-Acadèmia Researcher at Universitat Rovira i Virgili, Tarragona, Catalonia. He received his PhD degree in computer science from the Technical University of Catalonia, Barcelona, Catalonia, in 2008. He has participated in several national and European-funded research projects and he has authored several papers and conference contributions. His research interests include data semantics, ontologies, data privacy, and security.

Josep Domingo-Ferrer is a Distinguished Professor of Computer Science and an ICREA-Académia Researcher at Universitat Rovira i Virgili, Tarragona, Catalonia, where he founded and leads CYBERCAT-Cybersecurity Research Center of Catalonia. He received his PhD degree in computer science from the Autonomous University of Barcelona in 1991. He also holds MSc degrees in Computer Science (1988) and Mathematics (1995). His research interests are in security and privacy technologies and, more generally, in ethics by design in computing. He is an IEEE Fellow and an ACM Distinguished Scientist.