UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Membership Inference Attack on random forest regression models

Mitgliedschaftsableitungsangriff auf zufällige Gesamtstruktur-Regressionsmodelle

Bachelorarbeit

im Rahmen des Studiengangs
IT-Sicherheit
der Universität zu Lübeck

vorgelegt von
Krasen Heinemeier

ausgegeben und betreut von
Prof. Dr. Esfandiar Mohammadi

Lübeck, den 20. Januar 2022

Erklärung

Ich erkläre hiermit an Eides statt, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

(Krasen Heinemeier)
Lübeck, den 20. Januar 2022

Abstract Machine learning has become a widely used technique in the field of artificial intelligence. One of the most common algorithms is the Random Forest Classification and Regression Algorithm, which uses a series of decision trees to classify the input data and give a prediction of the correct output label. Such models require large amounts of data. By default, the data sets used for this contain some confidential information, some of these models are also publicly available. Recent attempts such as the Model Inversion Attack by Fredrikson et al. [FJR15] or the Membership Inference Attacks by Shokri et al. [SSSS17] have shown that there is a privacy problem. This could lead to sensitive data from such datasets no longer being protected.

However, these attempts do not exploit the peculiarities of random forest models. There are biases in the individual trees that we want to use for an attack. Our hypothesis is to use these biases to create a membership inference attack, with the biases being a strong indicator of information disclosure.

Our approach is a white box attack against a random forest model, where we also have access to a background knowledge. This background knowledge is a data set similar to the original training data set and is used to measure the deviation on each split to gain the bias on the whole tree. Our experiments show that it is possible to precisely identify individual elements. This indicates that it is possible to gain information about the data set used to train the random forest model.

Kurzfassung Maschinelles Lernen ist zu einer weitverbreiteten Technik im Bereich der künstlichen Intelligenz geworden. Einer der gebräuchlichsten Algorithmen ist der Random-Forest-Klassifizierungs- und Regressionsalgorithmus, der eine Reihe von Entscheidungsbäumen verwendet, um die Eingabedaten zu klassifizieren und eine Vorhersage über das korrekte Ausgabelabel zu geben. Solche Modelle erfordern große Datenmengen. Standardmäßig enthalten die dafür verwendeten Datensätze einige vertrauliche Informationen, einige dieser Modelle sind auch öffentlich zugänglich. Neuere Versuche wie der Model Inversion Attack von Fredrikson et al. [FJR15] oder die Membership Inference Attacks von Shokri et al. [SSSS17] hat gezeigt, dass es ein Datenschutzproblem gibt. Dies könnte dazu führen, dass sensible Daten aus solchen Datensätzen nicht mehr geschützt sind.

Diese Versuche nutzen jedoch nicht die Besonderheiten von Random-Forest-Modellen. Es gibt Biases in den einzelnen Bäumen, die wir für einen Angriff nutzen wollen. Unsere Hypothese ist, diese Biases zu nutzen, um einen Mitgliedschaftsrückschlussangriff zu erzeugen, wobei die Biases ein starker Indikator für die Offenlegung von Informationen sind.

Unser Ansatz ist ein White-Box-Angriff auf ein Random-Forest-Modell, bei dem wir auch Zugriff auf ein Hintergrundwissen haben. Dieses Hintergrundwissen ist ein Datensatz, der dem ursprünglichen Trainingsdatensatz ähnlich ist, und wird verwendet, um die Abweichung bei jeder Teilung zu messen, um die systematische Abweichung für den gesamten Baum zu erhalten. Unsere Experimente zeigen, dass es möglich ist, einzelne Elemente genau zu identifizieren. Dies weist darauf hin, dass es möglich ist, Informationen über den Datensatz zu gewinnen, der zum Trainieren des Random-Forest-Modells verwendet wird.

Acknowledgements

First, I would like to thank my supervisor for the thesis, Prof. Esfandiar Mohammadi, who answered my questions even late at night and accompanied this long process.Thanks to my family and friends, who were there even during difficult times and without whom this work would have never been completed. An exceptional thanks go to Freddy, who put up with me even if I couldn't have done it myself. Thank you, to everyone who helped in any way to write this thesis. Thanks very much!

Contents

1. Introduction

1.1. Motivation

In this thesis, we will provide a white-box membership inference attack on a machine learning model based on the random forest regression algorithm. This attack aims to cause a privacy leakage and gain information about the data set which was used to train the random forest model. As these kinds of models are used in different fields like medicine or finance, the underlying data sets could reveal some sensitive information about the persons or groups that submitted this data.

By default, these models do not reveal significant information about the underlying data sets, it will therefore show whether the information required for our hypothesis can be read from the models. As models, we will use random forest regression models. A random forest is based on multiple decision trees, which split up on certain attributes to process the input and gather information for the most likely output. We assume that we have access to the outputs of the random forest model and the data set, as well as information about the attributes and splits inside the model. We suspect these splits may reveal information.

We are looking at a privacy problem here, since an information leakage would mean that the privacy of the persons questioned would no longer be guaranteed. Publicly accessible models must meet the requirements of privacy to be securely accessible. In theory, the model should not reveal information about itself; however, we assume that it is possible to gather information about the data set and data points, that are in the data set, can be distinguished from other data points.

To gain information, we measure deviation, consistency and the coincidence for the single splits by providing a second data set. This functions as background knowledge for information gathering. In contrast to other approaches, which are discussed in section 2.1, our approach does not require any additional models or attempt to mimic the behaviour of an existing model. There is thus no need for previous calculations or time-consuming storage or calculations of many individual models.

To evaluate our theoretical considerations, we implemented a program based on our algorithm that outputs whether a single data point breaks differential privacy or not. Our experiments could not

provide evidence to consistently distinguish member and non-member elements, but individual elements could be predicted very well. For some elements, our accuracy was about 0.8, he mean accuracy was around 0.53 over all used data sets. However, there were also poor accuracies, down to 0.2. With further refinements in our approach, a more precise attack could be possible.

1.2. Structure of the thesis

In the next chapters, we will describe how we developed the algorithm and used it in our attack. First, we will look at other membership inference attacks, and what their approaches were to attack a random forest. We will compare them to our approach and how our attempt is better for our hypothesis.

Next we will begin with the basics of machine learning random forest models and basic concepts that are needed for our attack. We are interested in the decision trees and how they decide and make the splits based on the data points. Decision trees have some characteristics that might be helpful so that they can be used in our attack.

Then we take a look at the membership inference attacks and differential privacy. We use differential privacy to determine whether there is a breach of privacy; therefore, we have to give a definition of it.

After explaining the basics, we can look at the theory for our hypothesis. We use the information from each split in the random forest to draw conclusions about the individual elements to make a membership inference attack. We put the theory into practice by implementing a practical solution for our hypothesis. With this, we show how our experiments with a data set works, what we found and how our attack performed.

2. Background

2.1. Related Work

Machine Learning is widely applicable in different fields. Therefore, other attacks have been presented, that aim to attack machine learning models.
Example: Model Inversion Attack: A Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures [FJR15] is a black-box attack. The basic idea is to assume every possible value for a specific feature and then completes the feature factor to calculate a weighted probability estimate for the specific feature. This attack can identify the sensitive feature, but is not applicable for larger data sets with many unknown features, because of the calculation of the unknown features.

Example: Membership Inference Attack I: The Membership Inference Attacks Against Machine Learning Models [SSSS17] is a black-box membership inference attack approach, where they used multiple so-called shadow models to train an attack model which can predict the probability of the membership of an element in the training data set of the target model. These shadow models were trained on similar data sets to the training data, one shadow model for each output class of the target model. They then used supervised training, so that the attack model learns to distinguish between members and non-members of the shadow models. To test if a data point is a member, the attack model is given the label of the data point and the prediction of the target label for this data point. The attack model then returns the membership probability. Their experiments purposed an understanding how machine learning models leak information and how their attack can be used to evaluate the selection of a certain model. However, the shadow models have to be specially trained, each output class has its own shadow model. This can mean a very high computational effort.

Example: Membership Inference Attack II: SocInf: Membership Inference Attacks on Social Media Health Data With Machine Learning [LWP+19] is another black-box membership inference approach, and it is quite similar to the Shokri et al. [SSSS17]. Instead of training shadow models for the output classes, they try to mimic the target model with a generative adversarial network (GAN), which consists of two neural networks. These models learn to mimic the target model's

behaviour. In their conclusion, they point out that their approach can attack the membership on different models training data sets. However, as they synthesize their data, it is probably not suitable for data sets with a higher number of unknown features. Also, their attack is only evaluated against supervised learning models.

The problems of these attacks are that they are not suitable on larger data sets and unknown features due to their black-box approach or their brute-force like approach for calculating the needed information. The approach of Shokri et al. [SSSS17] circumvents the problems. However, much computation is needed for the shadow models. This could become a problem if there are many classes.

2.2. Problem Statement

This thesis wants to show that random forest regression models have privacy leakage, which make it possible to detect data points from the training data set. The basis is a Chosen-plaintext attack (CPA). We consider a set of labelled data records and assume that a random forest regression algorithm is used to train a classification model. We assume that the attacker has white-box access to the model and can obtain any information about the model, especially about the splits of the single decision trees.

The attacker knows the inputs and has access to a data set that is similar to the original training data set, with same attributes and same distribution as the training data set. He knows the attributes, that are in the data set and which attributes were used for the splits. He has no access to the training data set.

The attacker can choose any data point and insert it into the random forest model, obtain the path of the element through every single decision tree, and measure the deviation between the model split and the target point split. The attack succeeds if the attacker can distinguish between a data point that was a member of the training data set and a data point that was a not a member of the training data set.

2.3. Machine Learning Basics

Machine Learning (ML) is part of the field of artificial intelligence. The goal is to gain knowledge through experience, as the statistic model should be able to predict labels and make decisions based on the before seen training data [GBC16]. Therefore, the training data influences the predictions

and decisions of the model. The Random Forest is an ML regression and classification algorithm. As the name suggests, a random forest is structured like a forest consisting of multiple trees. The trees are called decision trees. To explain how a random forest works, it is important to know how a decision tree works.

Decision Tree

A decision tree is a model to make decisions about a given data point. One data point has different attributes, which are used to predict a label for the data point. Like a normal tree, it consists of a root, nodes and leaves, but is drawn upside down. When the model is trained, it builds itself up from the top (root) to the bottom (leaves). In the root, the whole data set is used to calculate the split value for the root node. To achieve this, the best possible split is calculated by using a cost function and assuming the best attribute and value for the split. Definition 2.1 defines the Gini Impurity, we will use this as the cost function, as it is the most common one.

Figure 2.1 shows the structure of a decision tree. Beginning at the root, a data point would follow a path through the tree to one leaf node. On each node, depending on the data point's attribute value, the data point would either follow the left or the right branch to the bottom. Coming down to the leaf node, the decision tree outputs a prediction, which label fits best for the data point. The split values and the attributes are interesting for our approach, as the split value depends on the training data, we are going to use this for our attack. We assume that the attribute value for a non-member of the training data is further away from the split value than the value for a member of the training data.

Definition 2.1. (Gini-Impurity). Given C as the number of classes and $p(c_i)$ as the probability of picking a data point with class i, the Gini Impurity G is calculated as

$$G = \sum_{i=1}^{C} p(C_i) \cdot (1 - p(C_i)) \tag{2.1}$$

Bias–variance tradeoff

In the context of decision trees, a bias means that the predictions of the model tend in one specific direction, not using all information that the training data is giving to it. Variance means that the

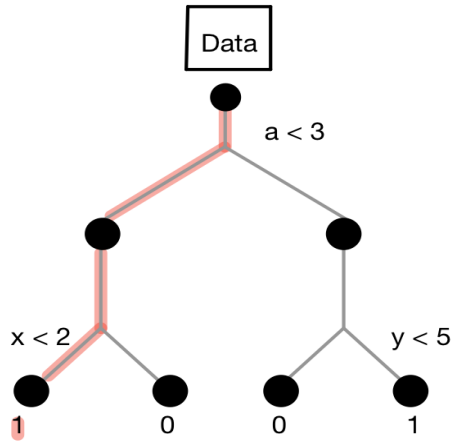


Figure 2.1.: Basic structure of a decision tree

model learns too many things from the training data, including too much noise into the model. A decision tree can be too shallow, that results in a high bias. That means the tree is underfitting. In the opposite, a decision tree can be too deep, that results in a high variance, which means that the tree is overfitting. A tree that overfits is very precise in predicting the labels of its own training data, but lacks in precision when predicting data that it has never seen before. On the other hand, an underfitting decision tree constantly over- or underpredicts the value of the target point.

In easier words, the less complex the model, the smaller the variance and greater the bias. The more complex the model, the lower the bias and greater the variance. This results in the bias-variance tradeoff. The goal is to have the perfect complexity to achieve the lowest possible bias and variance. One possibility would be hyperparameter tuning the parameters of the tree, like the maximum depth per tree or the minimum of samples for a leaf. Another way to circumvent this problem is using the random forest.

Random Forest

A random forest is an ensemble learning technique, where several decision trees are trained with different samples of the training data (bagging) and the result is determined by a majority decision. As described before, a decision tree can overfit or underfit the data, if the hyperparameters are not tuned correctly. To avoid the hyperparameter tuning, a random forest model is built up with low bias, high variance decision trees, i.e. deep decision trees. The decision trees are combined to give the result, and each decision tree is trained on a sample of the training dataset, this leads to the random forest being generally non-overfitting. Additionally, as the trees have low biases, the random forest also does not underfit.

2.4. Membership Inference Attack

We want to develop a membership inference attack. The goal of a membership inference attack is to gain information about the training data set and ultimately gain evidence that a data point was used in the training set.

Definition 2.2. [TLG+18] Given an adversary A, a data point x, a machine learning model M and it's training data set D and a white-box access, the adversary would be successful if he can say with a high degree of reliance that x was in D and used to train the model M or not.

The basic idea of this type of attack is based on the behaviour of machine learning models to perform better on data they know, as mentioned in section 2.3.

2.5. Differential Privacy

In machine learning, private data sets are used to train the models. To train the best possible models, requests to the data records must be answered as precisely as possible. This contrasts with the privacy of the people who provided their data for the data set, as it should be impossible to draw any conclusions about the persons from the data sets. For this purpose, the mathematical model of Differential Privacy (DP) was created to provide privacy protection. The task of differential privacy is to enable inquiries as precisely as possible while disclosing as little as possible about the data sets used.

Definition 2.3. (ε-Differential Privacy, short: ε-DP)[Dwo06]. A randomized function M gives ε-differential privacy if for all data sets D_0 and D_1 differing on at most one element, and all $F \subseteq Range(M)$, if

$$P_r[M(D_0) \in F] \leq exp(\varepsilon) \cdot P_r[M(D_1) \in F] \tag{2.2}$$

2.6. Mathematical Definitions and other Preliminaries

To derive the mathematical solution for our approach, we have to define some preliminaries and notations that we will use in the thesis.

Definition 2.4 (Law of total probability)**.** Given two events X and Y, the conditional probabilities $P_r[X|Y]$ and the probability of the conditional event Y, the probability of $X, P_r[X]$, is defined as:

$$P_r[X] = P_r[X|Y] \cdot P_r[Y] + P_r[X|\neg Y] \cdot P_r[\neg Y] \tag{2.3}$$

Sets as a test [DR+14]

A set can be seen as the description of a property or a test. Given some test T over elements in some set A, the set $S_T := \{\in A | T(o)$ succeeds $\}$ contains all elements for which the test succeeds. Reversely, given a set $S \subseteq A$, we can define a test

$$T_S(o) := \begin{cases} \text{succeed} & \text{,if } o \in S \\ \text{fail} & \text{,otherwise} \end{cases} \tag{2.4}$$

Hence, tests over A and subsets of A can equivalently be used to describe a test on elements of A. We can describe all tests over the possible outputs of a mechanism, M with input D as subsets $S \subseteq [M(D)]$ of the support $[M(D)]$ of the mechanism M applied to D. In the cryptography literature, instead of such sets, typically the output of attackers is considered. Such attackers can, however, be seen as tests. Hence, quantifying over all attackers also characterizes all possible (computable) tests.

Probability for x in data set.

As we defined in 2.2, our data sets only differ on one element. That means that for our data set D, assuming we do not know anything about the dataset, the element x either is in the dataset D or is not in D. For this:

$$D = \begin{cases} D' & \text{with prob. } \frac{1}{2} \\ D' \cup \{x\} & \text{with prob. } \frac{1}{2} \end{cases} \tag{2.5}$$

Cumulative Distribution Function (CDF)

The CDF describes the probability that a random value of a basic set is less than or equal to a certain value. We will use the CDF to measure the randomness for an observation. A high value would suggest a coincidence, while a low value would indicate not coincidental.

$$F_X(x) = P(X \leq x) \text{ for all } x \in \mathbb{R} \tag{2.6}$$

Consistent and Inconsistent

Given a decision tree D_T, let sv_i be the split value for the i-th split in D_T, av_i the value of the target element for the attribute that is used to make the split and *bksv* the split value we calculated with the background knowledge. Then, *av* is defined as:

$$av = \begin{cases} \text{consistent} & \text{if } bksv \leq sv \leq av \text{ or } av \leq sv \leq bksv \\ \text{inconsistent} & \text{else} \end{cases} \tag{2.7}$$

2.7. Hypothesis

Machine learning models work better with data they already know. Our hypothesis is that a data point that was in the training data is more consistent with the model, than a data point that was not in the training data. Therefore, the deviation should be measurable on the individual splits to distinguish between members and non-members. In section 3.1, we will provide theoretical considerations that underpin our attack.

3. Methodology and Approach

3.1. Methodology

We want to describe an adversary, that can attack a random forest model by breaking the differential privacy. As described in section 2.5, a usable random forest model should fulfil the requirements of differential privacy.

Derivation

Differential Privacy 2.2 states that a model F is ε - DP, if

$$\forall F : \frac{P_r[M(D') = F]}{P_r[M(D' \cup \{x\}) = F]} \leq e^{\varepsilon} \tag{3.1}$$

We transposed the equation from the definition of ε - differential privacy definition 2.2, so that e^{ε} is the upper bound of the fraction.

$$= \frac{P_r[M(D) = F | x \in D]}{P_r[M(D) = F | x \notin D]} \leq e^{\varepsilon} \tag{3.2}$$

With Bayes law follows:

$$= \frac{\frac{P_r[x \in D | M(D)=F] \cdot P_r[M(D)=F]}{P_r[x \in D]\} = \frac{1}{2}}}{\frac{P_r[x \notin D | M(D)=F] \cdot P_r[M(D)=F]}{P_r[x \notin D]\} = \frac{1}{2}}} \leq e^{\varepsilon} \tag{3.3}$$

Because x can either be in the dataset D or not, we can use our assumption 2.5. In the end, we can shorten this fraction:

$$= \frac{P_r[x \in D | M(D) = F]}{P_r[x \notin D | M(D) = F]} \leq e^{\varepsilon} \tag{3.4}$$

As mentioned, we want to create an adversary Adv who can attack this definition of differential privacy in our model. So for our adversary, the definition would look like the following:

$$\forall Adv : \frac{P_r[Adv(M(D)) = 0 | x \in D]}{P_r[Adv(M(D)) = 0 | x \notin D]} \tag{3.5}$$

$P_r[Adv(M(D)) = 0 | x \in D$ means that the adversary would output 0, if $x \in D$.

$$= \frac{\frac{P_r[x \in D | Adv(M(D))=0] \cdot P_r[Adv(M(D))=0]}{P_r[x \in D]\} = \frac{1}{2}}}{\frac{P_r[x \notin D | Adv(M(D))=0] \cdot P_r[Adv(M(D))=0]}{P_r[x \notin D]\} = \frac{1}{2}}} \tag{3.6}$$

$$= \frac{\overbrace{P_r[x \in D | Adv(M(D)) = 0]}^{A}}{\underbrace{P_r[x \notin D | Adv(M(D)) = 0]}_{\neg A}} \leq e^{\varepsilon} \tag{3.7}$$

From a mathematical point of view, we can describe the attacker using our assumption section 2.6. The attacker can be seen as a test. The set S_T contains the cases in which the attacker was able to successfully identify the elements. We now have to reformulate the probabilities for A in terms that we can observe in a random forest model. For better understanding, we will define two events. We are looking for the probability of $P_r[A = x \in D | M(D) = F]$ and the probability of $P_r[\neg A = x \notin D | M(D) = F]$. Let $P_r[B]$ be the probability that x is consistent (2.7) and $P_r[\neg B]$ the probability that x is inconsistent (2.7). With the law of total probability, (2.3) follows that $P_r[A]$ can be written as:

$$\begin{aligned}
P_r[A] =& P_r[A | \text{x is consistent}] \cdot P_r[\text{x is consistent}] && (3.8)\\
&+ P_r[A | \text{x is inconsistent}] \cdot P_r[\text{x is inconsistent}] && (3.9)\\
=& P_r[A|B] \cdot P_r[B] + P_r[A|\neg B] \cdot P_r[\neg B] && (3.10)\\
P_r[\neg A] =& P_r[\neg A | \text{x is consistent}] \cdot P_r[\text{x is consistent}] && (3.11)\\
&+ P_r[\neg A | \text{x is inconsistent}] \cdot P_r[\text{x is inconsistent}] && (3.12)\\
=& P_r[\neg A|B] \cdot P_r[B] + P_r[\neg A|\neg B] \cdot P_r[\neg B] && (3.13)
\end{aligned}$$

Deviations can be consistent or inconsistent. We also define whether a deviation is random, consistent or inconsistent. We formalize this as x is in the 95th percentile, i.e. CDF(x) < 0.025 if the split is to the left of the expected value and 1 - CDF(x) < 0.025 if the split is to the right of the expected value. So, we approximate the probability distribution of the splits with a normal

distribution. We can now describe the single probabilities as follows:

$$\begin{aligned} P_r[A|B] = P_r[A|B|\text{ x is consistent by chance}] \\ \cdot P_r[\text{x is consistent by chance}] \\ + P_r[A|B|\text{x is consistent not by chance}] \\ \cdot P_r[\text{x is consistent not by chance}] \end{aligned}$$

The probability that x is consistent by chance is defined as the probability of whether x is in the 95 percentile of the random variable $SPLIT_{arr}$: $P_r[\text{x is consistent by chance}] := P_r[cdf_{SPLIT_{att}}(x) < 0.025]$ where $cdf_{SPLIT,att}(x) := P_r[SPLIT_{att} < x]$. So, there is no randomness in the event "x is consistent by chance": $P_r[\text{x is consistent by chance}] \in \{0,1\}$. We simply have to conduct the test $cdf_{SPLIT_{att}}(x) < 0.025$.

P_r [A | B | x is consistent not by chance] = 1 as this is our hypothesis, if x is a member a consistent, that has to be not by chance. P_r [x is consistent not by chance] is the counter probability for x is consistent by chance. Therefore, it must be 1 - cbc, and it will also be calculated with the CDF.

$$\begin{aligned} P_r[A|\neg B] = P_r[A|\neg B|\text{ x is inconsistent by chance}] \\ \cdot P_r[\text{x is inconsistent by chance}] \\ + P_r[A|\neg B|\text{x is inconsistent not by chance}] \\ \cdot P_r[\text{x is inconsistent not by chance}] \end{aligned}$$

If x is inconsistent by chance, x can be a member or not be a member, the data gives no evidence about it, so we assume that $P_r[A|\neg B] = \frac{1}{2}$. The probability that x is inconsistent by chance is given by the CDF function we use. $P_r[A|\neg B|$ x is inconsistent not by chance] = 0 as this is our hypothesis, if x is inconsistent not by chance, then x has to be a non-member. $P_r[$ x is inconsistent not by chance] is the counter probability for x is inconsistent by chance. Therefore, it must be 1 - cbc, and it will also be calculated with the CDF.

$$\begin{aligned} P_r[\neg A|B] = P_r[\neg A|B|\text{x is consistent by chance}] \\ \cdot P_r[\text{x is consistent by chance}] \\ + P_r[\neg A|B|\text{x is consistent not by chance} \\ \cdot P_r[\text{x is consistent not by chance}] \end{aligned}$$

If x is consistent by chance, x can be a member or not be a member, the data gives no evidence about it, so we assume that $P_r[\neg A|B] = \frac{1}{2}$. The probability that x is consistent by chance is given by the CDF function we use. $P_r[\neg A|B|$ x is consistent not by chance] = 0 as this is our hypothesis, if x is consistent not by chance, then x has to be a member. P_r [x is consistent not by chance] is the counter probability for x is consistent by chance. Therefore, it must be 1 - cbc, and it will also be calculated with the CDF.

$$\begin{aligned} P_r[\neg A|\neg B] = P_r[\neg A|\neg B| \text{ x is inconsistent by chance}] \\ \cdot P_r[\text{x is inconsistent by chance}] \\ + P_r[\neg A|\neg B|\text{x is inconsistent not by chance}] \\ \cdot P_r[\text{x is inconsistent not by chance}] \end{aligned}$$

If x is inconsistent by chance, x can be a member or not be a member, the data gives no evidence about it, so we assume that $P_r[\neg A|\neg B] = \frac{1}{2}$. The probability that x is inconsistent by chance is given by the CDF function we use. $P_r[\neg A|\neg B|$ x is inconsistent not by chance] = 1 as this is our hypothesis, if x not inconsistent by chance, then x has to be a non-member. $P_r[$ x is inconsistent not by chance] is the counter probability for x is inconsistent by chance. Therefore, it must be $1-$ cbc, and it will also be calculated with the CDF.

We can now use these probabilities to describe the term (2.3).

$$\begin{aligned} P_r[A|x \text{ is consistent}] =& (P_r[A|B| \text{ x is consistent by chance}] && (3.14) \\ & \cdot P_r[\text{x is consistent by chance}] \\ & + P_r[A|B|\text{x is consistent not by chance}] \\ & \cdot P_r[\text{x is consistent not by chance}]) \\ =& (\frac{1}{2} \cdot \text{cbc} + 1 \cdot (1 - \text{cbc})) && (3.15) \end{aligned}$$

$$\begin{aligned} P_r[A|x \text{ is inconsistent}] =& (P_r[A|\neg B| \text{ x is inconsistent by chance}] && (3.16) \\ & \cdot P_r[\text{x is inconsistent by chance}] \\ & + P_r[A|\neg B|\text{x is inconsistent not by chance}] \\ & \cdot P_r[\text{x is inconsistent not by chance}]) \end{aligned}$$

$$= (\frac{1}{2} \cdot \text{cbc} + 0 \cdot (1 - \text{cbc})) \tag{3.17}$$
$$= (\frac{1}{2} \cdot \text{cbc}) \tag{3.18}$$

$$\begin{aligned} P_r[\neg A|x \text{ is consistent}] = &(P_r[\neg A|B| \text{ x is consistent by chance}] \\ &\cdot P_r[\text{x is consistent by chance}] \\ &+ P_r[\neg A|B|\text{x is consistent not by chance}] \\ &\cdot P_r[\text{x is consistent not by chance}]) \end{aligned} \tag{3.19}$$
$$= (\frac{1}{2} \cdot \text{cbc} + 0 \cdot (1 - \text{cbc})) \tag{3.20}$$
$$= (\frac{1}{2} \cdot \text{cbc}) \tag{3.21}$$

$$\begin{aligned} P_r[\neg A|x \text{ is inconsistent}] = &(P_r[\neg A|\neg B| \text{ x is inconsistent by chance}] \\ &\cdot P_r[\text{x is inconsistent by chance}] \\ &+ P_r[\neg A|\neg B|\text{x is inconsistent not by chance}] \\ &\cdot P_r[\text{x is inconsistent not by chance}]) \end{aligned} \tag{3.22}$$
$$= (\frac{1}{2} \cdot \text{cbc} + 1 \cdot (1 - \text{cbc})) \tag{3.23}$$
$$= (\frac{1}{2} \cdot \text{cbc} + (1 - \text{cbc})) \tag{3.24}$$

The probabilities of $P_r[B]$ and $P_r[\neg B]$ can be observed by the data, so they are either $P_r[B] = 1$ and $P_r[\neg B] = 0$ or $P_r[B] =$ and $P_r[\neg B] = 1$. At this point, we know the probabilities $P_r[A|B]$, $P_r[A|\neg B]$, $P_r[\neg A|B]$, $P_r[\neg A|\neg B]$, $P_r[B]$ and $P_r[\neg B]$. This can be put in formula (3.10) and (3.13):

$$P_r[A] = P_r[A \mid B]P_r[B] + P_r[A \mid \neg B]P_r[\neg B] \tag{3.25}$$
$$= (\frac{1}{2} \cdot \text{cbc} + (1 - \text{cbc})) \cdot P_r[B] + (\frac{1}{2} \cdot \text{cbc}) \cdot P_r[\neg B] \tag{3.26}$$

$$P_r[\neg A] = P_r[\neg A \mid B]P_r[B] + P_r[\neg A \mid \neg B]P_r[\neg B] \tag{3.27}$$
$$= \frac{1}{2} \cdot \text{cbc} \cdot P_r[B] + (\frac{1}{2} \cdot \text{cbc} + 1 - \text{cbc}) \cdot P_r[\neg B] \tag{3.28}$$

As we want to put these into formula (3.4):

$$P_r[x \in D | M(D) = F] = (\frac{1}{2} \cdot \text{cbc} + (1 - \text{cbc})) \cdot P_r[B] + (\frac{1}{2} \cdot \text{cbc}) \cdot P_r[\neg B] \tag{3.29}$$

$$P_r[x \notin D | M(D) = F] = \frac{1}{2} \cdot \text{cbc} \cdot P_r[B] + (\frac{1}{2} \cdot \text{cbc} + 1 - \text{cbc}) \cdot P_r[\neg B] \tag{3.30}$$

These probabilities can be put into (3.4), this results in:

$$\frac{P_r[M(D) = F | x \in D]}{P_r[M(D) = F | x \notin D]} \tag{3.31}$$

$$= \frac{P_r[x \in D | M(D) = F]}{P_r[x \notin D | M(D) = F]} \tag{3.32}$$

$$= \frac{P_r[A \mid B] P_r[B] + P_r[A \mid \neg B] P_r[\neg B]}{P_r[\neg A \mid B] P_r[B] + P_r[\neg A \mid \neg B] P_r[\neg B]} \tag{3.33}$$

$$= \frac{(\frac{1}{2} \cdot \text{cbc} + (1 - \text{cbc})) \cdot P_r[B] + (\frac{1}{2} \cdot \text{cbc}) \cdot P_r[\neg B]}{\frac{1}{2} \cdot \text{cbc} \cdot P_r[B] + (\frac{1}{2} \cdot \text{cbc} + 1 - \text{cbc}) \cdot P_r[\neg B]} \tag{3.34}$$

To use formula 3.34 for our attack, we also need to define the random forest model and the decision trees.

Random Forest

$$M(D) = F = (f_1, f_2, ..., f_q)$$

We use the dataset D in our function M to model the random forest F. The random forest consists of multiple decision trees $f_1, f_2, ..., f_q$. We can represent this as independent applications of M':

$$M'(D) = f_i : P_r[M(D) = (f_1, f_2, ..., f_q)] \tag{3.35}$$

$$= \prod_{i=1}^{q} P_r[M'(D) = f_i] \tag{3.36}$$

That means the function M' uses the dataset D' to model f_i. So, the probability that $M(D) = F$ can be written as the sum of probabilities for every single tree in the forest.

Decision Tree

$$M(D') = f_i = (s_1, s_2, ..., s_q) \tag{3.37}$$

As we can represent the random forest as an independent set, we can also represent a tree f_i as an independent product of splits $s_1, s_2, ..., s_q$. In this case, just a sample of the overall dataset D' is used.

$$M'(D') = s_i : P_r[M(D') = (s_1, s_2, ..., s_q)] \tag{3.38}$$

$$= \prod_{i=1}^{q} P_r[M'(D') = s_i] \tag{3.39}$$

As defined in 3.38 the tree can be represented as an independent product of splits and a random forest can be defined as a product of trees 3.35. We use this simplification for our attack by calculating the ε-differential-privacy test 3.34 for each individual split. The test for the tree is the product of the individual splits, for the random forest it is the product of the individual trees.

By Chance or Not By Chance

In simple words, we want to measure the deviation between our target point and the target model split value. A deviation from the split value can either happen by chance or not by chance. For this, we define 2σ as the limit value. If a value deviates by more than 2σ, this deviation did not happen by chance. If it deviates by less than 2σ, this is considered to be by chance. This limit of 2σ is a hard limit, it could possibly be too high or too low, but for this experiment it is initially a good limit.

Summary

We derived a mathematical solution for a membership inference attack and defined an equation to test the differential privacy of a model.

Definition 3.1. (ε-Differential Privacy Test)

$$\frac{(\frac{1}{2} \cdot \text{cbc} + (1 - \text{cbc})) \cdot P_r[B] + (\frac{1}{2} \cdot \text{cbc}) \cdot P_r[\neg B]}{\frac{1}{2} \cdot \text{cbc} \cdot P_r[B] + (\frac{1}{2} \cdot \text{cbc} + 1 - \text{cbc}) \cdot P_r[\neg B]} \leq e^{\varepsilon} \tag{3.40}$$

3.2. System Design

Our attack requires some specifications. First, we will use a data set as background knowledge. This data set is similar to the training data set, it has the same attributes and the values are evenly distributed. This data set then can be used to train a specified random forest model. We have to specify this model, so it fits our requirements. We also need some values to calculate the ε-differential privacy. We have to calculate the probability, if an element was consistent or inconsistent and if it was consistent by chance or not. We also need to calculate the Gini Impurity for the splits to calculate the best possible split and some splits based on samples of the background knowledge.

General Procedure

Before starting the attack itself, we have to train a random forest model. For that, our data set is separated into three parts. One part is the training data set, one is the test data set, and the last one is our background knowledge. We used 60% of the original dataset as training data, 20% as test data and the other 20% as background knowledge. We also required the random forest to be built with five trees. For the evaluation, we will train different models on the same training data sets. As splitting criterion, the Gini score is used. The model requires a criterion to choose the split value on every node. The most common ones are the information gain and the Gini Impurity. We choose the Gini Impurity, shown in Algorithm 1, as it is the standard criterion and it doesn't require the attack algorithm to compute a logarithm.

Before the attack begins, a random element is selected to serve as target element. This element will be removed from the data set as it will not be used for calculations made with the background knowledge during the attack. The elements will be added to the training data set if the element should be a member, else it will not be added. To be able to measure the values at the individual splits, we will first go through each tree in the random forest individually. To accomplish this, we make a list with the individual nodes of the respective tree. This list only collects those nodes that our target element would visit on the way through the tree. All other nodes are not relevant for our calculations.

After assembling the list for each node, the best possible split value is calculated. At this point, the background knowledge comes in. As the background knowledge mimics the training data set, we use it to calculate the best possible split value with the Gini score. The Gini score divides the data set into two parts, one contains the data points whose values are smaller than the eventual split value and the other one the values which are greater than the split value. Then the Gini Impurity

can be calculated for both sides. For that purpose, we also need the labels for the data points in the background knowledge.

We want to ensure that this split value was not an anomaly. This can be circumvented by making samples out of our background knowledge and calculate a split value based on these samples. We can use these sample-based split values to calculate the deviation between them and verify if the target element is consistent or inconsistent by chance, shown in Algorithm 2. This background split value can be used to measure if the target element is consistent or inconsistent with the original split value. A target element can be seen as inconsistent if it would cause the split value to deviate from the background split value. For example, if the target element's value is 5, the split value is 6 and the background split value is 7, the target element would have caused a deviation of the split value from the best possible split. Else, the target element would be seen as consistent.

Afterwards, for every split, independently whether it was consistent or inconsistent, it will be calculated whether the split value was consistent/inconsistent by chance or not. We assume any deviation of the target value from the split value greater than double the standard deviation σ as not random. For simplicity, the whole process is shown in Figure 3.1.

Algorithm 1 Gini

```
for value in bk[att] do
    divide bk in to sets s1, s2
    s1 ← bk[att] =< value
    s2 ← bk[att] > value
    // calculate gini impurity for s1, s2
    for class i in s1/2 do
        G_i ← p(i) * (1 − p(i))
        G_s1/2* = G_i
    end for
    // calculate weighted gini
    if weighted gini > weighted_gini_value then
        split_value ← value
        weighted_gini ← weighted_gini_value
    end if
end for
```

Here, the sample split values can be used to measure the standard deviation σ. With that done, we now know all values for our formula 3.34 to calculate the value for the split. The ε-differential privacy value for one tree is the product of all splits of that tree. The final ε-differential privacy value for the whole model is the product of all trees in the random forest model.

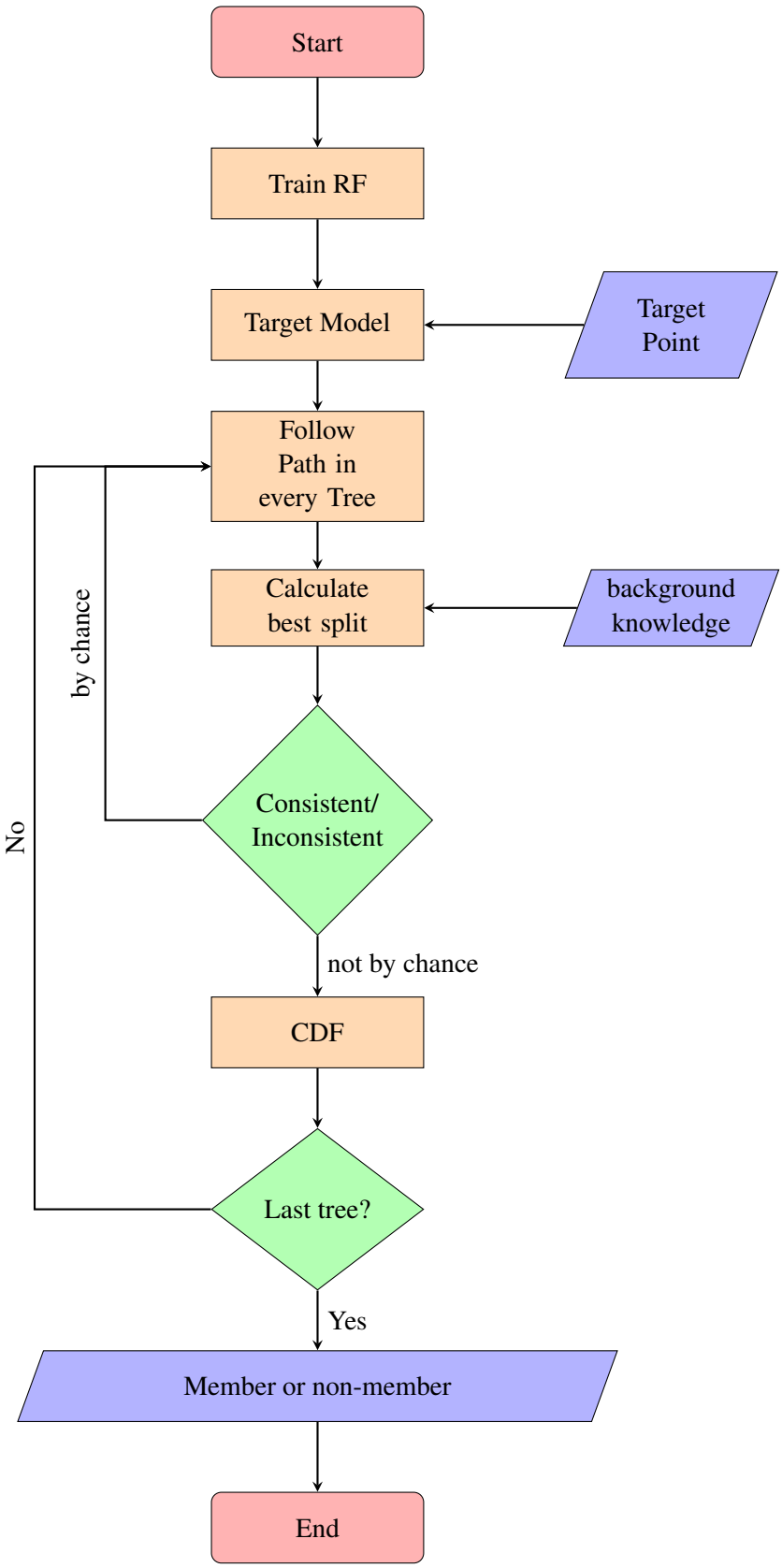


Figure 3.1.: General procedure

Algorithm 2 by chance or not by chance

```
if av < (sv - 2σ) then
    calculate CDF
else if av > (sv + 2σ) then
    calculate 1 - CDF
else av by chance
end if
```

3.3. System Implementation

Our program consists of 796 lines of code. This implementation is not the most efficient, one could possibly increase the speed with more efficient structures. Python was used as the programming language, but there are much faster languages that could possibly implement the attack better. For our random forest model, we use the python library Scikit-learn. Scikit-learn is a free python library for machine learning, based on NumPy and SciPy. It can be used for various machine learning techniques, in our case regression models.

In the ensemble package, Scikit-learn offers a Random.Forest.Classifier. This classifier can be specified with different parameters. The relevant parameters are *n_estimators*, *criterion*, *max_depth* and *max_features*. *max_depth* and *max_features* determine the depth of all trees in the random forest. As we want the trees to be as deep as possible, we set *max_features* to None and *max_depth* to default, which is also None, so the trees grow to maximum depth. The *criterion* parameter will be set to Gini Impurity, as we choose this to be the split criterion. We will build or random forest model with five estimators. Five estimators are enough to take advantage of a random forest while not overwhelming our limited computing power. The random forest classifier has the method fit(X, y) to build a random forest based on the data set and the labels.

Sklearn also offers the *train_test_split* method to split our data set into the training data set and the test data set. The training data set is then used to train the random forest. With the random forest model, the actual attack can begin. The model is given to our membership inference method, along with a target point, the background knowledge and the labels. First, the method goes through the tree and makes a list of all nodes. Then, starting at the root node, the background knowledge split value is calculated. With this split value, it can be measured whether the target element is inconsistent or not. Either way, afterwards, the method calculates split values based on samples of the background knowledge. These splits are needed to determine if a target element was consistent or inconsistent by chance or not. If a split is randomly consistent or inconsistent, is decided based on the deviation of the target element from the target model split value. This procedure is done for every node in every tree.

4. Experiments

4.1. Experimental Setup

To prove our hypothesis, we will test our attack on different data sets. The three used data sets are all different from each other and vary in number of attributes and elements. Also, we use various samples and different partitions of the data sets for the evaluation. This enables us to rule out the possibility of accidentally using a certain division of the data sets, which could falsify the results of our experiments. In addition, the three data sets are of different sizes, which is why we can observe how the size of the data sets and the resulting greater variation in data affects our attack.

Data Sets

We choose three different data sets for our experiments. The first one is a heart disease data set from the UCI [DG17]. This data set originally contained 75 attributes, but for experiments were cut down to 14. The last one, called "target", are the labels for our binary classification. The second data set is a diabetes data set [Sig90]. The labels for our binary classification is the last attribute "Outcome". We chose these two datasets to test our attack on smaller datasets. There may be many similar data points in these datasets, we want to check if this affects our attack.

The third one is depression data set from the University of New South Wales [Lov95]. We chose this data set because it is very diverse due to its large number of attributes and elements and may have many unique elements. This could make the attack easier as these elements are easier to distinguish. However, it could also result in the models having less bias. This data set require some preparation to be applicable for our approach.

Depression Anxiety Stress Scales (DASS)

The Depression Anxiety Stress Scales (DASS) [Lov95] is a questionnaire of the University of New South Wales in Australia. It is a self report of 42 questions and consists of 39 775 entries.

First, the answers were scaled to [0,1], so that only a distinction is made between not applicable (original scale 1) and applicable (original scale 2-4). Afterwards, the data was split into features and possible class labels which should be predicted. The questions of the class labels should be answered positively by a possible depressed person or a person with much negative emotional stress.

The questions to be predicted are: *I felt sad and depressed, I felt I wasn't worth much as a person, I felt that life was not worthwhile, I felt I was pretty worthless, I felt terrified, I could see nothing in the future to be hopeful about, I found it difficult to work up the initiative to do things*. Two tests were then carried out to select the questions that could be particularly well predicted. In the first test, a random forest classifier was used to predict all selected answers as a multiclass problem.

The overall accuracy was 51%. Therefore, a second test was implemented. Here, an individual random forest classificator was learned for each class. We use the f1-score to measure the accuracy of the model. The question with the highest f1 score (94%) is: *I felt that life was not worthwhile* and is therefore used for further processing. Furthermore, the not numeric questions were removed from the data, as these were not necessary for our model.

Metrics

All attacks were run locally on our own machine so that we can tune the configuration and test different settings. An adversary would be seen as successful if he can break the definition of differential privacy (3.1) and therefore determine if an element was a member of the training data set or not. An element is seen as a member if the differential privacy value is greater than e^{ε}. We will set ε to zero. That means, an adversary would see every bit of information as clear evidence for a membership inference. Also $\varepsilon = 0$ offers a high level of privacy and can be seen as the benchmark for our attack. This also simplifies our test, it is simply checked which probability is greater. If the probability for member is higher, then our program's prediction is member, if the probability for non-member is higher, the program predicts non-member. To evaluate the attack, we will use the accuracy metric, as it is a standard metric for classification models. To use this properly, we also have to make sure that the labels of the target points are evenly distributed between member and non-member.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

We will test our attack evenly on members and non-member of the respective data set, so that an adversary that would randomly guess the membership would have a baseline accuracy of 0.5.

Parameters

We have different parameters that could or could not influence our approach. The first one is the number of estimators in our random forest model. As a random forest is built on decision trees that slightly overfit, a lower number of estimators would mean that our model would more behave like a single tree. A higher number should minimize the overfitting.

We will set the estimators to five, as the calculation with more estimators would take too much time with our computation power. We will set the split criteria to gini impurity, as it is the default criteria as well as the most common one. We will also experiment with different distributions between test, training data and background knowledge to avoid a certain distribution that could falsify our results. The parameters can be set in the sklearn Random-Forest-Classifier-Method.

```
def fit_random_forest(x, y):
        rf = RandomForestClassifier(n_estimators=5, n_jobs=-1,
        max_features=None, criterion="gini")
        rf = rf.fit(x, y)
        return rf
```

k-fold Cross-Validation

k-fold cross-Validation is a validation procedure to evaluate machine-learned algorithms. This procedure helps to tune the hyperparameters of the estimators, in order to train the best possible model on the data set. We can use this technique to train different models on different data samples and evaluate our attack on these models. The standard procedure is meant to use the k-folds of the training data and train a model on k-1 folds and test it on the remaining set while finding the best parameters. In our experiments, we will shuffle the data set and split it randomly into k=5 sets.

Three sets will be used as training data, one will be the test data and the last one will be the background knowledge. Then a model is trained with the training data set. The respective model is evaluated with the test data set, to get an accuracy score. We use the score later in the evaluation to determine any connections between the quality of the model and the accuracy of our attack. The background knowledge is used for the attack with the respective model. One run of the attack with a certain partitioning of the data set and one model that has been trained with this division is called a configuration. We will target 120 random elements and test our attack with 16 different configurations.

4.2. Experimental Implementation

Experiments

The data sets were prepared as described in section 4.1. The model estimators will be set to five, the criterion is gini impurity and the *max_depth* is set to None.

parameter	value
n_estimators	5
criterion	gini impurity
max_depth	None

We want to measure three different values for the experiments, the categorization in every single round, the number of times how often an item was categorized and the number of times the items were consistent or inconsistent. These values are saved in three lists so that they can be output in a plot at the end. In every round, the model will output its categorization of the target element. We will save the categorization in every round and calculate the accuracy at the end.

```
categorization = []
categories = []
consistency = []
```

The experiments are carried out in 16 rounds. One round is called a configuration, consisting of a target model and a randomly shuffled data set. One model is trained per lap, the data set is shuffled new in each lap. Each target data point is evaluated per round using the one model. To ensure we don't take a specific sample that falsifies the experiments, we shuffle the data randomly Algorithm 3. After that, we split the dataset into two parts, one for the training/test part and the other on as background knowledge. We use Algorithm 4 to split the training/test set into the training and the

Algorithm 3 Shuffle the data

```
dataset = shuffle(data)
train_test_data <- 4/5 of the dataset
back_data <- 1/5 of the dataset
```

test set. As mentioned in section 4.1, we use k-folds for our evaluation. As the training/test set is 4/5 of the original data set, the test data set will be 1/4 of the training/test set. Altogether, the training data set is 3-folds of the original data set, the test set and the background knowledge are 1-fold. To ensure that we use an even number of members and non-members, we use Algorithm 5 to append the target elements 50% of the time to the training data set.

Algorithm 4 Test-Train-Split

```
x, y <- split(train_data, label)
x_train, x_test, y_train, y_test <- train_test_split(x, y)
```

Algorithm 5 Mix member and non-member evenly

```
if i in loop is even then
    append target element to train_data
end if
```

After we prepared the data set, we use Algorithm 6 to train the model on the training data set and safe them to use them for the attack evaluation. This happens for every configuration.

Algorithm 6 Train the model

```
rf <- fit_random_forest(x_train, y_train)
calculate train_score and test_score
```

Our attack then can be evaluated with every target element, as shown in Algorithm 7.

Algorithm 7 Membership Inference

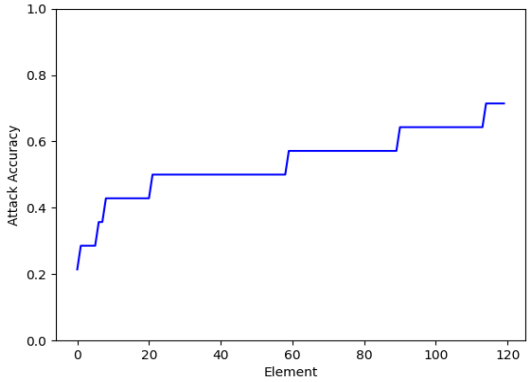
```
for every target element do
    for rf in rfs do
        membership_inference(target, rf)
        Save intermediate results in the respective lists
    end for
end for
calculate accuracy
```

5. Results

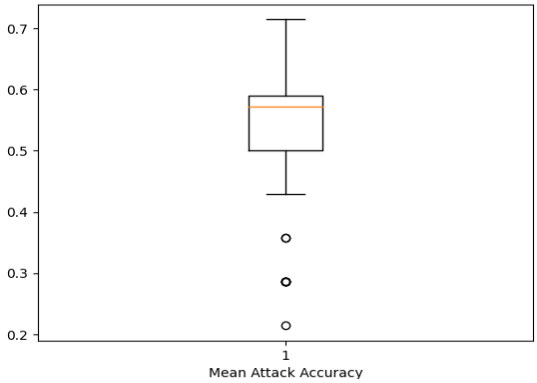
5.1. Data Analysis

DASS

Figure 5.1a for the depression detection data set shows the accuracy for the membership inference over all target points. The mean accuracy is just over 0.5, more precisely 0.54. Most elements have an accuracy between 0.5 and 0.6, with 30 elements the accuracy is even better than 0.6, the best accuracy is 0.7. About 20 elements have an accuracy below 0.5, the lowest accuracy is about 0.2. Figure 5.1b shows a box plot, the box marks the first and the third quantile of the accuracy of all target points. The median (yellow line) is quite higher than the mean. Figure 5.2a shows how the individual elements were classified. The elements have high true negative and false negative values. The true positive values are lower, the elements were most rarely rated as false positives. The test accuracy of the models was always above 0.88.

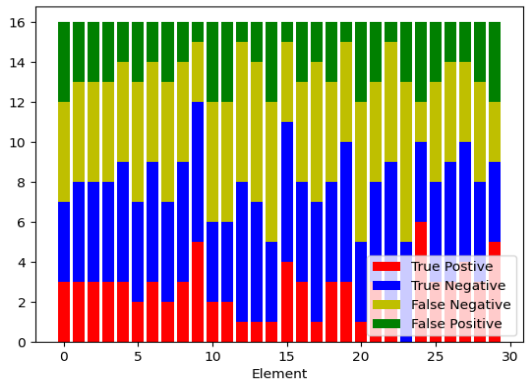


(a) Accuracy over the target element's. Most elements have an accuracy between 0.5 and 0.6, the highest accuracy is nearly 0.7, while the lowest is nearly 0.2

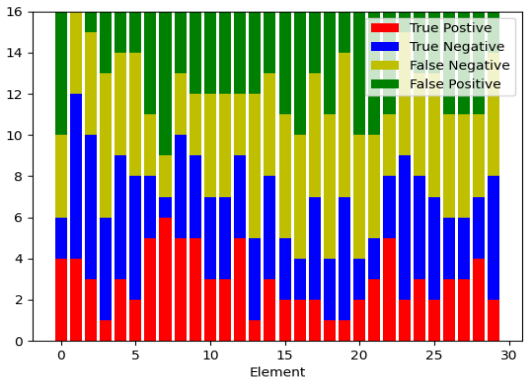


(b) The mean accuracy of the attack is at 0.54. The median is slightly higher. Most outliers have a lower accuracy, these elements pull the mean down, the data bias is more up to a better accuracy.

Figure 5.1.: Sorted Accuracy of our attack and a box plot over all accuracy values for the depression detection data set.



(a) The target elements of the depression data set have high false negative and true negative values, little to no true positives. The remaining categorizations are in appendix A.1



(b) The target elements of the heart data set have overall evenly mixed values, with partly extreme values in each category. The remaining categorizations are in appendix A.2

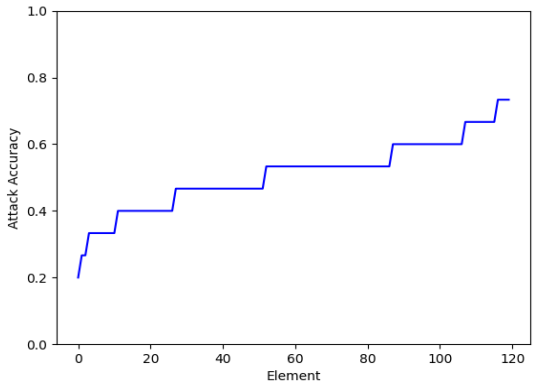
Figure 5.2.: Categorization of the single target elements of the depression data set 5.2a and the heart data set 5.2b

Heart Data set

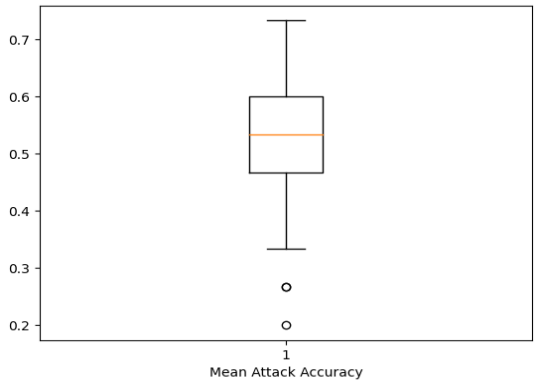
Figure 5.3a for the heart data set shows the accuracy for the membership inference over all target points. The mean accuracy is just over 0.5, more precisely 0.51. Most elements have an accuracy between 0.4 and 0.6, with ten elements the accuracy is even better than 0.6, the best accuracy is 0.7. About 15 elements have an accuracy below 0.4, the lowest accuracy is 0.2. Figure 5.3a shows a box plot, the box marks the first and the third quantile of the accuracy of all target points. The median (yellow line) is barely higher than the mean. Figure 5.2b shows how the individual elements were classified. The elements have high false negative values, while the true positive and true negative values are lower, the elements were most rarely rated as false positives. The test accuracy of the models was always above 0.88.

Diabetes data set

Figure 5.4a for the diabetes data set shows the accuracy for the membership inference over all target points. The mean accuracy is just over 0.5, more precisely 0.55. Most elements have an accuracy between 0.5 and 0.6, with 40 elements the accuracy is even better than 0.6, the best accuracy is nearly 0.9. About 18 elements have an accuracy below 0.4, the lowest accuracy is 0.25. Figure 5.3a shows a box plot, the box marks the first and the third quantile of the accuracy of all target points. The median (yellow line) is barely higher than the mean. Figure 5.5 shows how the individual elements were classified. The target elements were categorized relatively evenly,



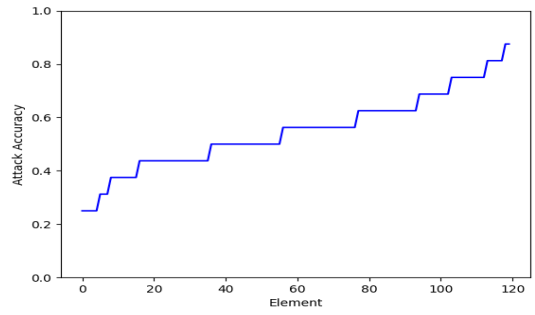
(a) Accuracy over the target element's in the heart data set. The mean accuracy is the lowest of the three data sets, with 0.51.



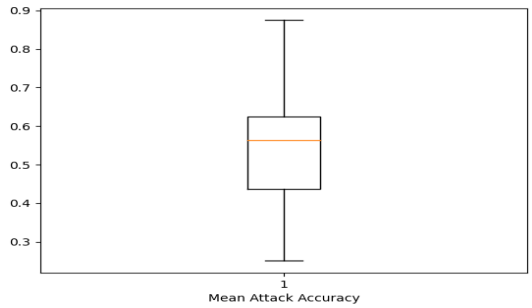
(b) The mean accuracy of the attack is at 0.51. The median is slightly higher. The outliers may cause a lower mean accuracy here.

Figure 5.3.: Sorted Accuracy of our attack and a box plot over all accuracy values for the heart data set.

tending to rank items as true negative and false negative the most. The target points in this data set have the highest true positive values. The test accuracy of the models was always above 0.88.



(a) Accuracy over the target element's in the diabetes data set. Target points in this data set have the highest accuracy in our experiments.



(b) The median is nearly 0.6. The data indicate that better accuracy is possible. The median confirms this.

Figure 5.4.: Sorted Accuracy of our attack and a box plot over all accuracy values for the diabetes data set.

5.2. Experimental Results

Our hypothesis was that we can break differential privacy and gain information about any data point out of the random forest. An element that was actually a member should more often be consistent with the split value of the random forest. Therefore, a non-member element should be more inconsistent. Our data analysis shows that the attack can sometimes successfully distinguish

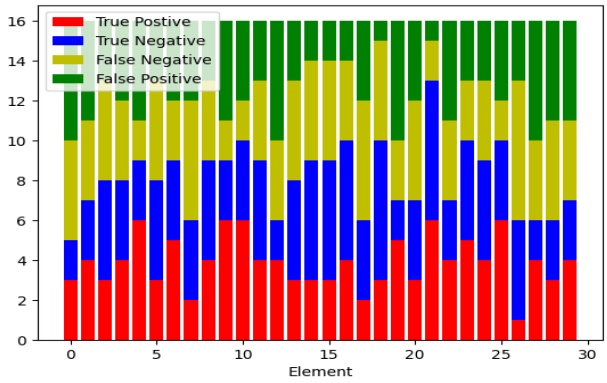
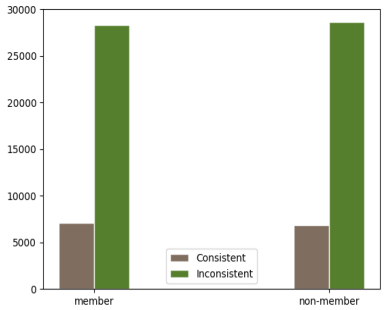


Figure 5.5.: Shows how often the elements were correctly or incorrectly categorized in the diabetes data set. The elements have the highest true positive rates of all data sets. The remaining categorizations are in appendix A.3

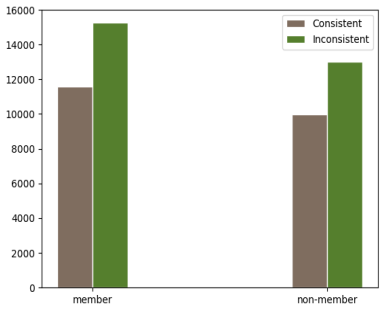
between member and non-member elements. The accuracies of our attacks are above average with 0.51, 0.54 and 0.55. Looking at the individual data sets, the accuracy for most elements is in the range between 0.5 and 0.6. For some elements, the accuracy is even as high as 0.7, with a peak of 0.9. Our attack was constantly unable to categorize a few elements correctly, with accuracy going down to 0.2. The observations of the categorizations confirm these measurements. However, we could not confirm the idea of our hypothesis that member elements are more often constant than non-member elements, as Figure 5.6 shows. Amazingly, even member elements are more often inconstant than constant.

5.3. Interpretation

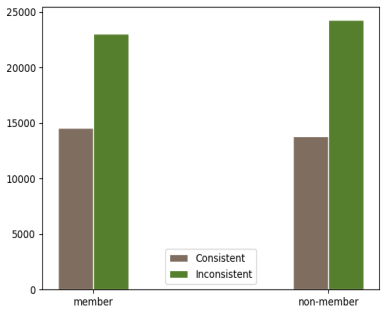
We have hypothesized that it is possible to break differential privacy for random forest machine learning models by calculating the deviation of individual elements in a model. Although our experiments do not provide unequivocal proof of our claim, the data suggest that it may be possible to distinguish individual data points from one another. In each data set, there were data points that had a very high accuracy. So, the main idea of our hypothesis could be promising. However, it is questionable whether this can be measured via the constancy and inconsistency of the data points, our data do not provide any evidence for this. But it should be noted again that our attack does not have a reliable accuracy, a random guesser would only have a slightly worse accuracy.



(a) The depression dataset target point shows a high discrepancy between consistent and non-consistent data points, however, member and non-member are relatively equally often inconsistent.



(b) Members and non-members are most often inconsistent, but here the difference to the consistent splits is not as great as with the other two data sets.



(c) Both member and non-member are mostly inconsistent, the number of consistent splits is a little more than half. Nevertheless, the accuracy is greatest with this data set.

Figure 5.6.: Difference between members and non-members for the depression data set 5.6a, the heart data set 5.6b and the diabetes data set 5.6c.

6. Conclusion

Machine learning will continue to grow and find new applications. However, due to the sensitive data required for this, it is still necessary to protect privacy. This thesis provided an approach for a membership inference attack through which it might be possible to privacy leakage on random forest regression models. The basic idea was that elements that were part of a test dataset would behave differently than elements that weren't. For this purpose, the mathematical basis was laid here, based on which further experiments and theories can be tested.

We have also developed a program to test our hypothesis and attack random forest models. As our experiments have shown, it is possible to build a working attack on such models based on our hypothesis. However, the accuracy is not yet good enough to speak of a reliably functioning attack. We could not confirm the argument of our hypothesis that non-members are more often inconsistent. Future work could possibly get more precise results based on our observations and improve our attack.

6.1. Future Work

We chose a model by scaling the number of classes to two. The assumption is that this may have generalized data that is important for our approach, and thus could not provide any evidence for our hypothesis. Shokri et al. [SSSS17] indicate in their work that their results differ significantly between models with two classes and models with several classes. The models with multiple classes are more susceptible to their attack.

Trees that tend to overfit are also more susceptible; therefore, future work could show the same for our attack. We also simplified the dependencies in our model by assuming that each split in a tree is independent of the split before. That is not the case, each split depends on the splits happened before. This must be considered in future work.

Future work could also use a better algorithm and/or more computational power to run our attack with more random forest estimators, to evaluate the influence of more estimators. We set ε to zero as our upper bound to measure the differential privacy provided by every model. This means that

an attacker would assume that the smallest piece of information would be successful. It would have to be checked how a change in this barrier affects the attack, and whether better results can be achieved as a result.

A. Appendix

This appendix contains the categorizations for the other target points of the individual data sets, whose were evaluated in section 5.1.

Depression data set

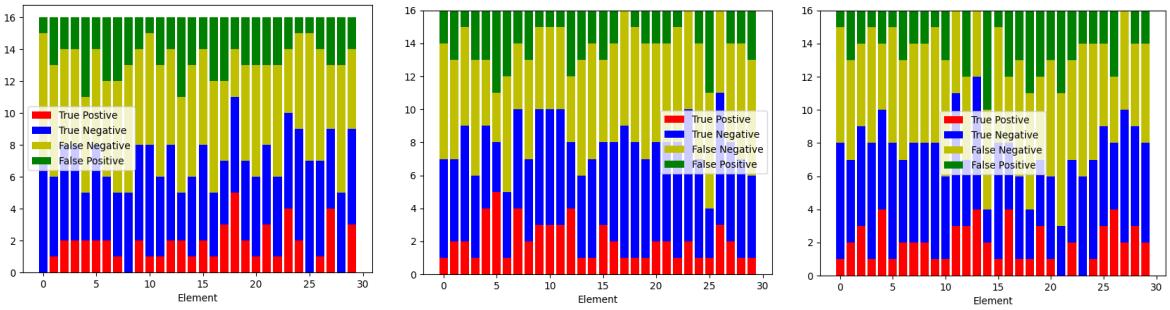


Figure A.1.: The remaining categorizations of the depression data set.

Heart data set

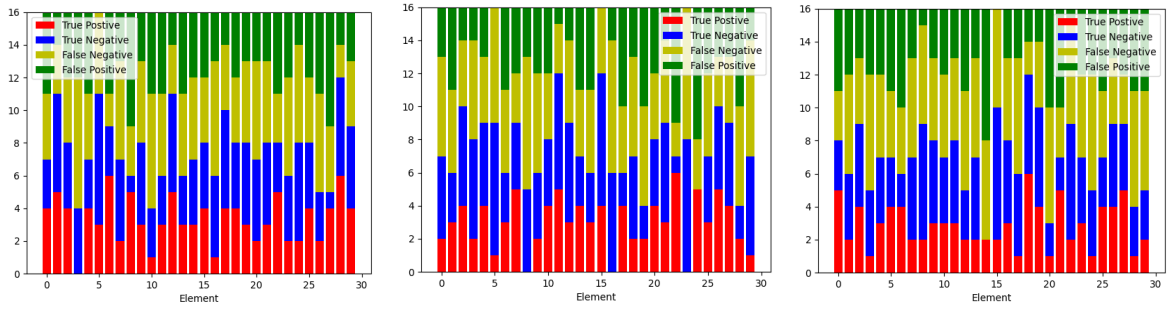


Figure A.2.: The remaining categorizations of the heart data set.

Diabetes data set

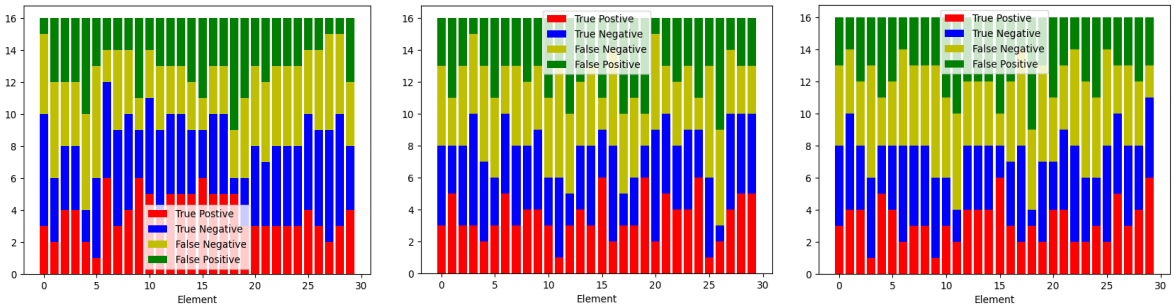


Figure A.3.: The remaining categorizations of the diabetes data set.

Bibliography

[DG17] "DUA, Dheeru ; GRAFF, Casey": *UCI Machine Learning Repository*. `http://archive.ics.uci.edu/ml`. Version: 2017

[DR+14] DWORK, Cynthia ; ROTH, Aaron u. a.: The algorithmic foundations of differential privacy. In: *Found. Trends Theor. Comput. Sci.* 9 (2014), Nr. 3-4, S. 211–407

[Dwo06] DWORK, Cynthia: Differential privacy. In: *International Colloquium on Automata, Languages, and Programming* Springer, 2006, S. 1–12

[FJR15] FREDRIKSON, Matt ; JHA, Somesh ; RISTENPART, Thomas: Model inversion attacks that exploit confidence information and basic countermeasures. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, S. 1322–1333

[GBC16] GOODFELLOW, Ian ; BENGIO, Yoshua ; COURVILLE, Aaron: Machine learning basics. In: *Deep learning* 1 (2016), Nr. 7, S. 98–164

[Lov95] LOVIBOND, S.H. LOVIBOND, P.F ; SYDNEY: PSYCHOLOGY FOUNDATION (Hrsg.): *Manual for the Depression Anxiety Stress Scales. (2nd. Ed.)*. Sydney, Australia: Sydney: Psychology Foundation, 1995

[LWP+19] LIU, Gaoyang ; WANG, Chen ; PENG, Kai ; HUANG, Haojun ; LI, Yutong ; CHENG, Wenqing: Socinf: Membership inference attacks on social media health data with machine learning. In: *IEEE Transactions on Computational Social Systems* 6 (2019), Nr. 5, S. 907–921

[Sig90] SIGILLITO, Vincent: *Pima Indians Diabetes Database*. `https://www.kaggle.com/mathchi/diabetes-data-set`, 1990. – [Online; accessed 01/05/2022]

[SSSS17] SHOKRI, Reza ; STRONATI, Marco ; SONG, Congzheng ; SHMATIKOV, Vitaly: Membership inference attacks against machine learning models. In: *2017 IEEE Symposium on Security and Privacy (SP)* IEEE, 2017, S. 3–18

[TLG+18] TRUEX, Stacey ; LIU, Ling ; GURSOY, Mehmet E. ; YU, Lei ; WEI, Wenqi: Towards demystifying membership inference attacks. In: *arXiv preprint arXiv:1807.09173* (2018)