Lund University/LTH

Master thesis

EDAL01

Federated Learning Used to Detect Credit Card Fraud

Authors:
Madeleine Jansson
Måns Axelsson

Supervisors: Rasmus Ros (LTH), Mehmood Alam Khan (IBM) and Peter Forsberg (IBM)

Examiner: Volker Krüger (LTH)

June 23, 2020

Abstract

Credit card fraud is a common problem for consumers and financial institutions all over the world. They are loosing billions of dollars every year. Thus, an effective Fraud Detection System (FDS) is important to minimise the loss for financial institutions and cardholders. A common solution to detect fraud is to use machine learning algorithms as they help to predict future outcomes and recognise patterns by analysing massive quantities of data. In order to get a well performing model a large dataset is required, but an issue with credit card transaction datasets are that they are skewed, i.e. there are significantly fewer samples of fraudulent than legitimate transactions. Moreover, due to the data privacy and security associated with credit card transaction datasets, banks and other finical institutions are usually not allowed to share their transaction data. These problems together make it difficult for the centralised FDS to learn the patterns of fraud and, hence, to detect them. In this thesis, we propose a framework for training of a fraud detection model with federated learning, i.e. a machine learning setting where multiple entities collaborate in solving a machine learning problem under the coordination of a central server or service provider. With this approach financial institutions can collectively reap the benefits of a shared model, which has seen more fraud than each bank alone, without sharing the dataset with each other. Hence, the sensitive information of the cardholders is protected. Our result of this thesis indicates that the federated model (Federated Averaging) can perform and even outperform the centralised model (Multi Layer Perceptron) when trying to detect credit card fraud.

Keywords: Credit card fraud · Skewed dataset · Federated learning

Preface

This master thesis project was carried out during the spring of 2020 at the Faculty of Computer Science at Lund University in cooperation with International Business Machines Corporation (IBM) in Kista. It is part of our master program Machine Intelligence and the particular topic was chosen since it suits not only our own interests but also the program.

In this section we would like to take the opportunity to thank the people that made this project possible. Firstly, a big thanks to our supervisors Rasmus Ros from LTH and Mehmood Alam Khan as well as Peter Forsberg from IBM. They have all supported us during the work by sharing their expertise, helped us with necessary materials and not least helped to steer us in the right direction when needed. Secondly, we would like to thank the people at IBM that were kind enough to offer their support this spring.

Contents

1 Introduction

In the article Fraud: 'A Serious Problem' written by Tracy Kitten [27] she captures how society and especially the financial institutions suffer from fraud.

> "Fraud continues to hamper financial institutions around the globe, and the problems will only worsen, unless banks and credit unions put forth effort and dollars to mitigate risk."

Financial institutions have carried out and are carrying out extensive research in order to prevent and detect fraud, regardless of type. Fraud is however a complex concept as it involves different behaviours and tactics which constantly changes. One area of fraud which is very common nowadays and where a lot of research has been carried out is fraud within the banking industry, and more precisely credit card fraud [2]. This thesis will focus on this type of fraud.

Today, credit card fraud is a frequent problem for consumers and banks all over the world and costs banks billions of dollars every year [46]. One of the driving factors to this is that the electronic commerce has grown rapidly and has resulted in more purchases being made over the internet. This in turn means that credit card details are available on the internet to a further extent than before and makes it easier for fraudsters to access the details. To combat fraud there are mainly two mechanisms used, where one mechanism is designed to *prevent* fraud and the other to *detect* fraud [2]. When developing computer systems to detect fraud it is common to use machine learning (ML) as a tool since it is an efficient way of teaching a system how to make accurate predictions when fed with data [16]. Different machine learning algorithm are used depending on the data and what kind of fraud the model should detect.

Credit card transaction datasets from banks do not only contain highly sensitive information but the fraudulent transactions are also usually in minority compared to the non-fraudulent transactions, i.e. the dataset i skewed [10]. Both of these issues introduces challenges when building a fraud detection system. The highly sensitive information raises questions such as who can see the information and what can be shared. The problem of skewed data introduces a challenge of how to train the model in the best way without over- or underfitting it. When aiming to have a model that performs well in detecting credit card fraud it is necessary to have a metric that describes the models ability of doing so. Working with a skewed dataset there are a few measurements that are better suited than others to measure such performance. [3]

Since the data from banks contains highly sensitive information it is traditionally stored and used locally. In other words, banks access and use its own data to train various machine learning models to be able to detect fraudulent behaviours. This is what is referred to as a centralised machine learning approach since not only the data is stored locally but the models are as well. Today, this machine learning approach is usually used in the financial industry due to its efficiency of handling massive quantity of data and extract patterns [2]. As mentioned, there are plenty of different machine learning algorithms and one model which has shown good results in other research papers for the classification problem, in this case to classify fraudulent or non-fraudulent transactions, is the Multi Layer Perceptron (MLP) [23].

A problem with the centralised approach is, however, that different banks might have seen different types of fraudulent transactions which would limit their ability to detect new types of fraudulent transactions. A possible solution to this would be for banks to collaborate in order to share between them all types of fraudulent transactions that they have encountered. However, such collaboration is a sensitive area since the banks do not want its competitors to know how much, or what type of fraud they are exposed to. The rather new machine learning approach called federated learning could however be a feasible solution to this problem [34]. The difference between the centralised and the federated machine learning approach, for this problem, is that there would not only be one bank involved in the training of the fraud detection model but several. The process of the federated learning approach is that each bank trains a beforehand determined model and then passes up encrypted information to a central model. The outcome will be one central model which have been collectively trained by way of averaging the results from each bank's encrypted information. Finally, this central model is shared with each bank and hopefully the banks are now able to detect fraudulent transactions to a wider extent. When performing this federated learning no sensitive information is shared, just as in the centralised approach. The final federated model will have seen more fraudulent transactions than then the centralised model trained by an individual bank.

In this thesis the Multi Layer Perceptron is used as a baseline model. Then, the MLP is modified to the federated setting which is referred to as Federated Averaging (FedAvg) [62]. The scope of this thesis is to investigate if the Federated Averaging model is suited for fraud detection in a credit card transaction dataset by studying limitations, privacy and security concerns for Federated Averaging and comparing the results to the centralised Multi Layer Perceptron model.

The outline of this thesis is as follows. Section 2 contains a thorough background of what fraud is, where it might happen, how to combat it by using machine learning techniques as well as issues and challenges in connection therewith. Further, Section 3 will present an overview of existing research in this field. In Section 4 the dataset used for training and evaluation of the models are presented. The proposed technique for FDS is detailed in Section 5. The results of our experiments are presented in detail in Section 6 and then discussed in Section 7. In Section 8 suggestions are made for future work in relation to the topic of this thesis. Finally, a conclusion of our work is presented in Section 9.

1.1 Research Question

The main aim of this thesis is to investigate the federated learning technique and to see where it can be applied. The thesis question is: how well does the Federated Averaging model detects credit card fraud transactions in contrast to the centralised Multi Layer Perceptron model?

1.2 Contribution

The main contribution of this master thesis project was to give IBM a better understanding of how federated learning works and in what areas this learning technique can be applied. In our thesis, Federated Averaging has been applied to the banking industry where the aim was to detect credit card fraud. The federated model showed progress on the credit card transaction dataset obtained from Kaggle [25].

2 Background

In order to understand the issues and challenges when trying to detect credit card fraud it is important to know what fraud is, how it is used within the banking industry and also how it can be detected. Sections 2.1–2.1.2 contain an explanation of fraud, machine learning as a tool to detect fraud and different areas of fraud. Then, in Sections 2.1.3–2.2 issues and challenges when working with machine learning for credit card fraud detection are explained. Finally, Sections 2.3–2.4 explains how to detect credit card fraud by applying the centralised and the federated machine learning approaches.

2.1 Fraud

Fraud is a crime where the purpose is to appropriate money. According to The Association of Certified Fraud Examiners [36] (ACFE), fraud is defined as:

> "The use of one's occupation for personal enrichment through the deliberate misuse or misapplication of the employing organization's resources or assets."

ACFE [55] divides fraud into two categories, namely, *internal*, an employee commits fraud against his/her organisation, or *external* fraud against a company, which involves a wide range of different schemes. Regardless of the nature of the fraud this crime has a huge impact on the economy, laws and individuals in the society. Extensive effort has been put into preventing fraud in the financial industry and today there are several fraud detection mechanisms to prevent and detect fraudulent behaviour. As stated by Abdallah et al. [2] these mechanisms are included in the *Fraud Prevention Systems* (FPS) and *Fraud Detection Systems* (FDS). Neither FPS nor FDS seem to be defined terms but rather umbrella terms used by the market to describe certain mechanisms for prevention and detection of fraudulent behaviour.

Starting with FPS, which is the first layer of protection, in place to secure the technological systems against fraud. Abdallah et al. [2] states that the mechanisms in this layer restrict, suppress, destruct, destroy, control, remove or prevent the occurrence of cyber-attacks in hardware and software systems. Such mechanisms are, for instance, firewalls, encryption algorithms, electronic signature and many more.

Moving on to FDS, which is the next layer and which actually tries to detect fraudulent activity as they enter the system. The two main approaches in FDS are, according to Sorournejad et al. [50], *anomaly* based fraud detection and *misuse* based fraud detection.

Anomaly based fraud detection means that we study the behaviour of each client and when abnormal behaviour occurs the system will raise an alarm for suspected fraud. This approach builds on data mining, which involves statistical, mathematical, artificial intelligence and machine learning techniques to extract possible valuable information from large datasets. In contrast to anomaly based fraud detection, misuse based fraud detection is fixed in its setting. That is, fraudulent behaviours are learnt beforehand by the system, thus, the behaviours will be the same for all clients. This means that if any client performs a pre-learned behaviour or equivalent the system will alarm for suspected fraud. This is an approach which utilises rule-based statistics to reveal suspicious events.

Figure 1 visualises the protection system mechanisms for combating cyber based fraud and is inspired by Abdallah et al. [2].

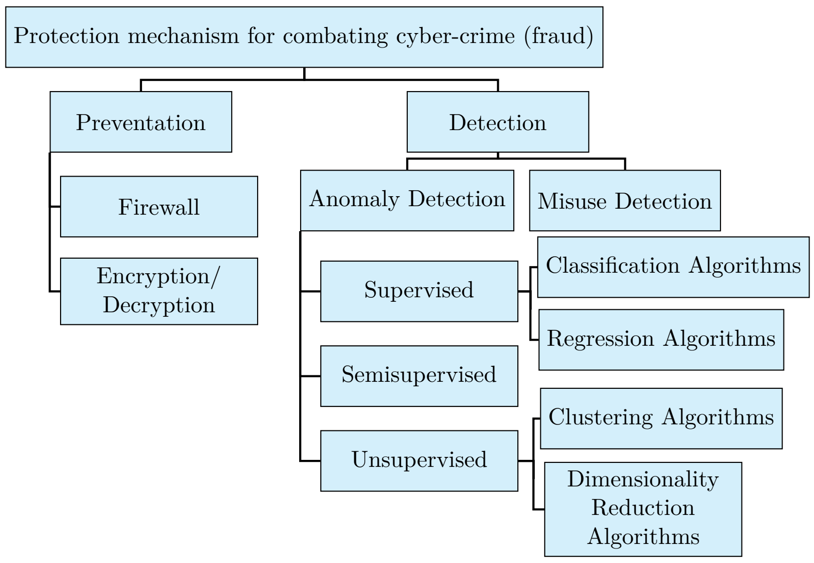


Figure 1: Protection system mechanisms for combating cyber-crime.

Abdallah et al. [2] claims that the most common strategy to detect fraud is to apply pattern recognition where the patterns contain information about the cardholders behaviour such as transaction amount, time gap since last purchase, day of the week, item category, customer address, etc. Any new spending behaviour is compared to the previous ones and if the behaviour seems to be inconsistent with the cardholder's profile it will be considered as suspicious.

Further, it is stated by Abdallah et al. [2] that, misuse detection systems are rarely used since neither legitimate nor fraudulent behaviour is static. Instead, they argue, that the anomaly based fraud detection systems are more common due to three main advantages: (i) fraud patterns are automatically obtained from data; (ii) by extracting patterns of fraudulent behaviour the system can alarm when detecting similar behaviour and as a consequence these can be prioritised; (iii) not previously detected fraudulent behaviours might be detected.

It is clear that, fraud detection systems build on machine learning techniques, namely, supervised learning, unsupervised learning and semi-supervised learning. These techniques are discussed further in Section 2.1.1.

2.1.1 Machine Learning for Fraud Detection

By using machine learning, i.e. the science of getting computers to act without being explicitly programmed to do so, it is possible to get rather accurate results when predicting risk and abnormal behaviour in a dataset, such as credit card fraud [16]. The reason for this is that ML enables the prediction of future outcomes and recognition of patterns by analysing massive quantities of data.

As mentioned there are different machine learning techniques and the four main categories are as follows. First, *supervised learning* which means that the training data passed into the algorithm is labeled. Second, *unsupervised learning* which means that the training data passed into the algorithm is not labeled. Third, *semi-supervised learning* which is a technique that is a mixture of supervised and unsupervised learning, i.e. some labeled and some unlabeled training data is passed into the algorithm. Finally, there is also *reinforcement learning* which is a learning method that interacts with its environment by producing actions and discovering errors or rewards [16].

When implementing machine learning algorithms to detect credit card fraud the classification problem, in this case to classify fraudulent or non-fraudulent transactions, can be categorised as either supervised, unsupervised or semi-supervised learning depending on how the dataset has been prepared. It is claimed by Abdallah et al. [2] that the most common data mining method to build credit card FDS is by applying a classification model, which is a type of supervised learning technique.

2.1.2 Fraud Areas

According to Abdallah et al. [2] fraud is almost always present when money is involved in any technological system. Further, they have mapped the most common areas of fraud to be bank, insurance, telecommunication and internet marketing fraud, see Figure 2.

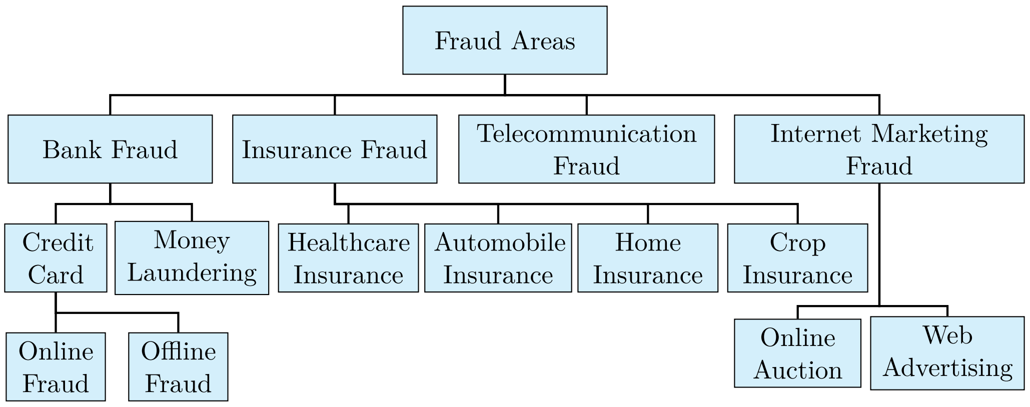


Figure 2: The most common areas of fraud according to [2].

Out of the four main areas presented in Figure 2 bank fraud is the most researched one [2] and credit card fraud is something that most people are aware of. Today credit cards are widely used as they are convenient to bring, in contrast to cash, and as they make it is easy to track spendings. Even though there exist several authorisation techniques, such as signatures, credit card number, identification number, cardholder's address etc., it is still possible for fraudster to hack your card. According to Patidar et al. [42], credit card fraud can be divided into two categories: *offline credit card fraud* which means that the credit card is stolen by the fraudsters and then used; *online credit card fraud* which means that the cardholder's details are taken either by skimming, site cloning, credit card generators or phishing.

This thesis will focus on bank fraud and more precisely online credit card fraud.

2.1.3 Issues and Challenges with Fraud Detection Systems

There are lots of challenging issues when working with fraud detection systems but usually four main topics are discussed: (i) *concept drift*; (ii) *skewed class distribution*; (iii) *large amount of data*; (iv) *support of real time detection*.

Firstly, concept drift is, according to Dal Pozzolo et al. [10], a problem that occurs when the model is trained and has learnt a certain pattern of the customer or imposter but then the behaviour changes. That is, the model is not sufficiently dynamic and does not learn as fast as the behaviour might change. Therefore, it is of great importance for the FDS to distinguish and classify fraudulent behaviour as well as legitimate transactions efficiently.

Secondly, skewed class distribution is, as described by Dal Pozzolo et al. [10], one of the most critical issues faced by FDS and refers to the heavily imbalanced data. There are different approaches to solve this problem, such as data level approaches and algorithmic level approaches that are presented in Section 2.1.4.

Thirdly, the large amount of data and its high dimensionality makes the process of data mining and detection very complex [18]. Therefore, it is common to apply data reduction approaches, including dimensionality and numerosity reduction. Principal Component

Analysis (PCA) is a common dimensionality reduction approach and is explained in Section 2.1.5.

Lastly, the problem of real time detection shows that it is important for the system to detect fraud at an early stage to be able to stop it or to be able to take actions swiftly. Different techniques have been deployed to enhance the real time detection, amongst others, Very Fast Decision Tree (VFDT) [35] and Self-Organization Map (SOM) [44].

This thesis will only deal with (ii) skewed class distribution and (iii) large amount of data.

2.1.4 Skewed Data

In a dataset with credit card transactions the amount of fraudulent transactions are significantly less then the amount of non-fraudulent transactions which leads to, as described by Dal Pozzolo et al. [12] a *skewed* dataset, i.e. big differences in the number of data points for each class. The dataset in this thesis is heavily skewed which can be seen in Section 4.1.

According to Sorournejad et al. [50], skewed datasets are problematic as they cause the detection of fraudulent transactions to be very difficult and imprecise. This is mainly because running the classification model, i.e. a classifier, on skewed data could either result in unbalanced performance on the different classes [5] or a classifier could even completely ignore the minority class [2]. Following the terminology of Krawczyk [28] there are two main approaches to solve this problem, namely, *data level approaches* and *algorithmic level approaches.*

Starting with data level approaches, this is described as pre-processing of the dataset in order to make it more balanced before feeding it into the classifier. This, in turn, makes it easier for the classifier to classify the different classes correctly. Furthermore, the data level approaches can be divided into two subcategories; *undersampling* and *oversampling.*

The process of undersampling down-sizes the majority class by removing some of its observations until the dataset is balanced [2]. Chawla et al. [8] describe two different undersampling techniques; *Random Undersampling* (RUS) and *Direct Undersampling.* RUS means that data from the majority class is removed randomly. While in direct undersampling, the observations that are removed from the majority class are not randomly removed, hence, they are known.

The oversampling technique is based on oversampling the minority class in order to get more observations of that particular class. However, balancing the dataset in this way might result in overfitting the model on the minority class as this class is now up-sampled, i.e. has more data points than before. Last et al. [29] states that, the most commonly used oversampling technique is called *Synthetic Minority Oversampling Technique* (SMOTE). The main idea of SMOTE is defined by Bowyer et al. [7] as oversampling in the feature space by producing synthetic minority data points in the proximity of the observed ones. This is done by creating vectors between one sample and its K closest neighbors in the feature space with the same class. Then, new samples are generated from the minority class on these K vectors, see Figure 3 inspired by [56]. According to Brownlee [29], however, a problem with SMOTE is that synthetic datapoints are created without taking into account the majority class. This results in possible ambiguous examples if there exist an overlap of the classes.

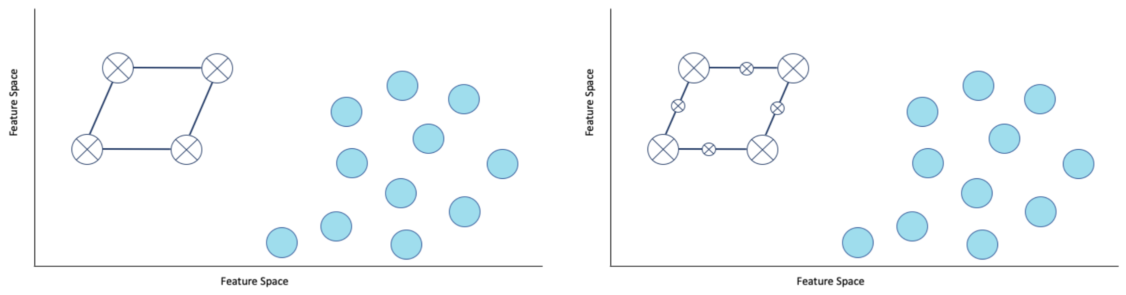


Figure 3: The filled data points belongs to the majority class while the big points with a cross belongs to the minority class. In both the left and right hand side figure $K = 2$ since each data point in the minority class creates vectors to its two closest neighbours within the same class. *Left:* Before SMOTE has been applied to the dataset. *Right:* After SMOTE has been applied to the dataset and new samples are generated from the minority class on these K vectors, visualised as small data points with a cross.

In contrast to data level approaches, algorithmic level approaches do not pre-process the data. Instead Krawczyk [28] outlines the algorithmic approach as a tool to modify the model, either by using cost sensitive learning, explained further in Section 2.2, or by adapting the classification algorithms to be able to handle minority class detection, i.e. *One-Class Learner.*

Figure 4, which is a modified example of the one in [2], provides an overview of the different balance approaches that can be applied to handle skewed data.

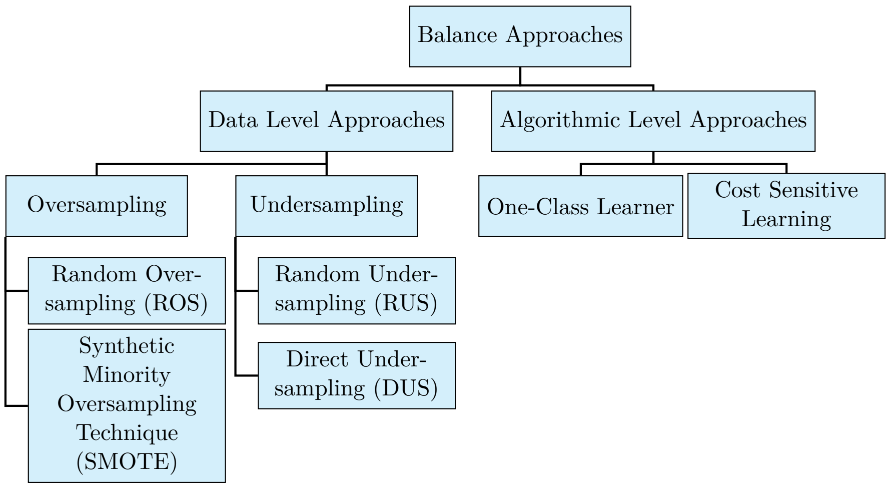

Balance Approaches
Data Level Approaches
Algorithmic Level Approaches
Oversampling
Undersampling
One-Class Learner
Cost Sensitive Learning
Random Oversampling (ROS)
Synthetic Minority Oversampling Technique (SMOTE)
Random Undersampling (RUS)
Direct Undersampling (DUS)

Figure 4: Balance approaches to handle skewed data.

Today, Abdallah et al. [2] argue that, data level approaches are preferred to handle skewed data problems as they are easy to implement and do not lead to an increase in computing resources needed or training time. In accordance with [12], SMOTE has been chosen as a suitable data level approach for this thesis.

2.1.5 Principal Component Analysis

As mentioned in Section 2.1.3, a dataset with a large amount of data and high dimensionality, such as a dataset with credit card transactions, can lead to complications. One way to handle the high dimensionality in a dataset is to reduce the number of dimensions, i.e. reduce the amount of features, and one such method is *Principal Component Analysis* (PCA) [18]. By reducing the number of dimensions when using PCA the result is, just like most of the dimensionality reduction methods, that the calculations are faster and also the data mining gets easier since there are less parameters and dimensions to take into account.

Principal Component Analysis is described in [48] as reducing the dimensions of the features in the dataset with the goal to describe it with orthogonal vectors that are positioned in the direction of the highest variance. These vectors are called *principal components* and the first principal component is pointed in the direction with the highest variance, then the variance decreases as the order of the components increases. The higher the order of the vector is the less representative it is of the data. These components are found by computing the covariance matrix of the features and calculating the eigenvectors and eigenvalues. Each eigenvector is a principal component where the eigenvector corresponding to the highest eigenvalue is the first one.

When the principal components have been computed, then, each observation is projected on each vector. Now, instead of expressing the data in terms of its original features it can be expressed in terms of principal components. This, as stated by Patel et al. [41], not only reduces the data dimensionality but also makes it secure since the transformation of the features masks the features nature, thus, the data gets anonymised which helps to preserve the privacy.

2.2 Performance Measurement

To be able to decide how well a machine learning model is, it is of importance to have a good performance measurement which can indicate the correctness and accuracy of the model. When working with a binary classification model, i.e. a binary classifier, there are a few measurements that are suitable and the most intuitive metric that can be used for this problem is a *confusion matrix*. A confusion matrix shows four different outcomes for the predictions, namely: (i) *True Positive*, a fraudulent transaction which is the predicted as fraud; (ii) *False Positive*, not a fraudulent transaction but is predicted as fraud; (iii) *False Negative*, a fraudulent transaction which is predicted as not fraud; (iv) *True Negative*, not a fraudulent transaction and is also predicted as not fraud. In Figure 5 a confusion matrix is shown with each of these outcomes.

		True Condition	
		Condition Positive	Condition Negative
Predicted Condition	Predicted Positive	True Positive (TP)	False Positive (FP)
	Predicted Negative	False Negative (FN)	True Negative (TN)

Figure 5: Confusion matrix with outcomes for a binary classifier.

Using these outcomes the performance measurement called *Accuracy* is defined as

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

and computes the fraction of correctly classified data points. According to Macaraeg [33], however, accuracy is not suited for models trying to detect credit card fraud due to the skewness of the data.

Instead, they recommend using other performance methods in cases with skewed data and one such is *cost sensitive learning*. In a cost sensitive learning method, as stated by Dal Pozzolo et al. [11], each class (or instance) is given a misclassification cost depending on how important each outcome is with the goal to minimise the total misclassification cost. For example, a misclassification of a fraudulent transaction could be more costly for the company than a misclassification of a non-fraudulent transaction. In order to create a good cost sensitive learning method for credit card fraud detection the cost information from banks is needed which is, as per Stolfo et al. [51], hard do obtain. This results in two other methods being more commonly used, namely, *Area Under the Curve Receiver Operating Characteristic* (AUC-ROC) and *Area Under the Precision-Recall Curve* (AUPRC). For both of these measurements the area is computed by using an average of a number of trapezoidal approximations [45].

Firstly, as specified in the article by Brownlee [6], the Receiver Operating Characteristic curve is a plot of *True Positive Rate* (TPR) versus *False Positive Rate* (FPR). True Positive Rate, or *Recall* as it is also called, is defined as

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

which measures the fraction of actual fraudulent transactions that are correctly classified as fraud. In order to define False Positive Rate we first define *Specificity* as

$$Specificity = \frac{TN}{TN + FP}$$

and now, False Positive Rate is

$$FPR = 1 - Specificity = \frac{FP}{FP + TN}$$

which measures the fraction of actual non-fraudulent transactions that are not classified as fraud. Further, Brownlee describes in [6] that the area under the ROC curve, i.e. AUC score, can be used to decide how well the model performs as it measures the discrimination, meaning the models ability to distinguish between the classes. The higher value on the AUC score the better the model is at predicting correctly. To conclude, the AUC-ROC measurement indicates how well the probabilities from the positive classes are separated from the negative classes.

Secondly, Brownlee [6] specifies the Precision-Recall Curve as a plot of *Precision* versus Recall, i.e. TPR. Precision is defined as

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

and measures the fraction of classified fraud that is actual fraud, meanwhile, Recall is found in Equation 1. Similar to the AUC-ROC measurement, the area under the precision curve is studied to determine how well the model performs where a larger area indicates a better model. According to Brownlee, the maximum area under the precision curve is one and this would mean that the model can classify all positive samples (fraud) right, i.e. not classifying any negative as positive.

When dealing with skewed data there are usually a lot of true negatives, i.e. non-fraudulent transactions correctly classified, which might skew the result for the AUC-ROC measurement [14]. As a matter of fact, according to Bronwlee [6], the AUC-ROC measurement is only appropriate to use when the observations are balanced between each class, otherwise, he claims that the AUPRC measurement should be used. Therefore, together with the confusion matrix, the AUPRC measurement will be used in this work as it does not include true negatives and is a better suited measurement for imbalanced datasets.

2.3 Centralised Learning

Until now, fraudulent pattern recognition systems, i.e. Fraud Detection Systems, have been applied in a centralised way to detect credit card fraud. In the centralised approach the machine learning algorithms are fed with locally stored data from one server, producing a model to find patterns based on the information from the given data. This method is used by banks, independently of each other, since they are usually not allowed to share their transaction data, which is due to the data privacy and security associated with credit card transaction datasets.

When working with a centralised approach there are plenty of different algorithms that can do the job of detecting fraudulent transactions, though, some are better suited than others. In order to chose the best suited centralised algorithm for the task of detecting fraudulent transactions different articles has been studied and in particular [23]. In this article different algorithms trying to detect credit card fraud are compared and the model with the best performance was the *Multi Layer Perceptron* (MLP). We have, in accordance with this article, chosen the Multi Layer Perceptron as baseline algorithm for this thesis. The MLP is introduced below in Section 2.3.1.

2.3.1 Multi Layer Perceptron

Artificial neural networks (ANNs) is a collection name for many algorithms that almost all have in common that they are inspired by the construction and functioning of the human brain. Generally, according to Goodfellow et al. [20], ANNs can be described as a class of parameterised functions $f(\mathbf{x}, \mathbf{w})$ constructed by composing linear and non-linear functions as

$$f = \varphi_n \circ f_n \circ \cdots \circ \varphi_k \circ f_k \circ \varphi_1 \circ f_1.$$

Here, f_n is a linear function parameterised by its weights w_n while φ_n is a non-linear function. Usually, the function f_n is referred to as a *layer* and φ_n is known as the *activation function.*

Each layer in an ANN is composed by a set of nodes where the first set of nodes is referred to as the *input layer*, the last set of nodes is the *output layer* and if there are any set of nodes between these layers than they are referred to as *hidden layers*, see the right hand side of Figure 7. Each weight is usually visualised as branches between the layers and depending on how they are connected to the nodes in the model the layers have different properties. For instance, a simple layer is referred to as a fully connected layer which means that the edges between every input node is connected to every output node, hence, the output layer is a linear combination of the input layer.

Furthermore, the simplest ANN model is the single perceptron which only has an input and output layer, see Figure 6. This model takes a weighted sum of the input $\mathbf{x}$ and passes it through a beforehand determined activation function φ_n which then computes the output to be a probability between zero and one [38], i.e. a forward pass have been made.

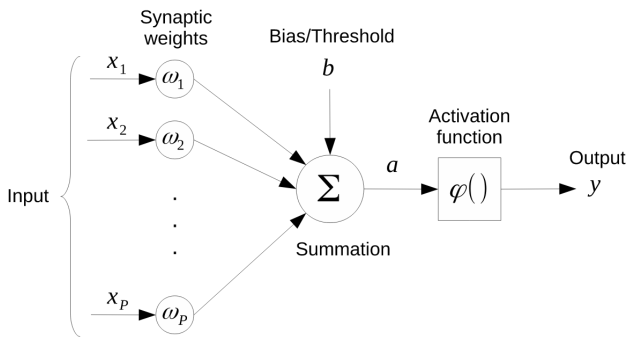


Figure 6: A simple perceptron with an input layer composed by the nodes x_1–x_p and an output layer with only one output y [38].

According to Imane et al. [23], a simple perceptron with p inputs and different weights performs well when the task is to detect linear separable patterns, though, this is rarely the case when aiming to detect credit card fraud. Instead, a more complex ANN model is used, namely, the Multi Layer Perceptron, which has shown to perform well on credit card fraud detection in their article.

In contrast to the simple perceptron, the MLP does not only have p input nodes and m nodes in its output layer, but the network also has hidden layers with an arbitrary number of nodes h [23]. Goodfellow et al. [20, p.5] describes the MLP as a feed-forward network, i.e. there are no recurrent connections in the network, composed of fully connected layers. Now, for each layer an activation function φ is applied and for a larger network the *Rectifier Activation Function* (ReLU) is popular [20, p.189]. Also, when working with a binary classification problem the *Logistic*, also called *Sigmoid*, activation function is applied to the last layer of nodes which then results in the output being between zero and one [20, p.191]. Below are the two activation functions ReLu and Logistic respectively,

$$\varphi(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

and

$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

where $\varphi(x)$ is the activation function which also appears on the left hand side in Figure 7, where it can be seen that the function is applied to every node in each layer.

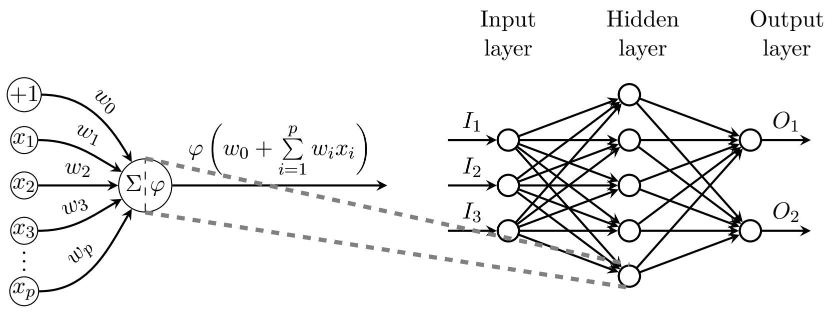


Figure 7: *Left:* A simple perceptron where the input is summed up and set as the argument of the activation function $\varphi(x)$. *Right:* A Multi Layer Perceptron with one input layer (three nodes), one hidden layer (five nodes) and one output layer (two nodes). The arrows between the nodes and neighbouring layer represents the weights, w_k, which are pointed in a forward direction, i.e. it is a feed-forward network.

After a forward pass in the network the weights need to be optimised and this is referred to as training the model. The goal of the optimisation procedure is to minimise the *error function*, also referred to as the *loss function* [23]. The main idea is that the network will eventually reduce its error as the network learns from the training dataset. A common error function for binary classification is the *Cross-Entropy Error* [20, p.177] which is defined as

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\Big(d_n \log y(\mathbf{x_n}) + (1-d_n)\log(1-y(\mathbf{x_n}))\Big)$$

where $\mathbf{w}$ are all weights, d_n is the target value for pattern n and y the prediction. The function can be minimised with respect to the weights by applying *Gradient Descent* [20, p.215],

$$\Delta w_i = -\eta\frac{\partial E}{\partial w_i},$$

where the learning rate η controls the size of the update. This way of updating the weights is called back propagation because, once a forward pass has been done to compare the actual output with the target then a backward pass is done to update the weights. The Gradient Descent algorithm is the most popular when training an MLP [23].

Further, Goodfellow et al. [20, p.290] states that Gradient Descent is not the most efficient optimiser algorithm to use since it only follows the gradient of an entire training set downhill and therefore can suffer from getting stuck in local minimum and regions of small gradients. Instead, they argue that one improvement might be to add a dynamical learning rate, in contrast to the previously fixed one which is set by default, to speed up the minimisation of the error function. The learning rate may be chosen by trial and error, but it is usually best to choose it by monitoring learning curves that plot the objective function as a function of time or training epochs [20, p.291]. In this thesis an optimiser without dynamic learning rate will be used in order to give a fair comparison with the federated model which does not support dynamic learning rate, see Section 4.4. The optimiser we have chosen is *Mini-Batch Gradient Descent* (MBGD) [43], which probably is one of the most used optimisation algorithms for machine learning in general and, in particular, for deep learning [20, p.290].

Mini-Batch Gradient Descent refers to calculating the derivative from each training data instance and compute an immediate update. According to [43], the MBGD optimiser updates the weights by using a small number of patterns, i.e. mini-batches. Formally, MBGD is written as

$$\Delta w_k = \frac{1}{P}\sum_{p=1}^{P}\Delta w_{pk}$$

where Δw_k is the k:th weight to be updated and P is the size of the mini-batch. Every time one mini-batch has been used to update the weights one iteration has been performed [37]. When all mini-batches have been iterated it is said that one epoch has been performed. MBGD has the important property that computation time per update does not grow with the number of training examples, which allows convergence even when the training dataset becomes very large [20, p.292].

Finally, it is important to keep in mind that, when building a large network it might get too specific, i.e. the network gets overfitted, as a consequence of having too many layers and nodes to find features in the training dataset. Therefore, when building the MLP the architecture needs to be carefully chosen and thoroughly tested to avoid both over- as well as underfitting.

An implementation of our baseline model, the Multi Layer Perceptron, is explained in Section 5.1.

2.4 Federated Learning

As mentioned in Section 2.3, centralised learning is a commonly used approach to detect credit card fraud. According to Yang et al. [62], however, FDS that are using this learning technique are prone to be inefficient with a low accuracy rate or raising many false alarms. This being a result of dataset insufficiency, skewed data distribution and limitation of detection time. FDS could probably be greatly improved by collecting more data. One way for banks to do this would be for them to share their data, anonymised, with a central server, so that a machine learning model would be able to see transactions from different banks. Sweeney [52], however, argues that the user privacy still can be at risk even if an "anonymised" dataset is held locally. The article does not explicitly state how the user privacy still would be at risk but one can imagine that the "anonymised" dataset can be reversed engineered. One solution to these problems might be to, instead of using the centralised approach, build an FDS by applying *federated learning* (FL).

The term federated learning was introduced in 2016 by McMahan et al. [34] and they define it as a machine learning setting where many clients, e.g. mobile phones and other edge devices, collaboratively trains a model under the orchestration of a central server. This training is done without sharing any data. Instead, only the updates in the form of model parameters are communicated to the central server.

Additionally, according to Kairouz et. al [26], federated learning is not only of interest for application on edge devices but there is also a growing interest for this method when multiple organisations are seeking to collaborate in order to train a central model. In their paper they name the federated settings defined by McMahan et al. [34] as *cross-device* while they refer to federated learning for multiple organisations as *cross-silo*. According to Kairouz et. al [26], these definitions open up for a broader common definition of federated learning, namely:

> "Federated learning is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each client's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective."

A typical federated training process usually follows the same concept as described by Kairouz et al. [26]. A server orchestrates the training process by repeating the following steps until the training is stopped and convergence has been achieved: (i) *client selection* – the server samples from a set of clients meeting eligibility requirements; (ii) *broadcast* – the selected clients download the current model and training program from the central server; (iii) *client computation* – each selected device locally computes an update of the model by executing the training program; (iv) *aggregation* – the server collects an aggregate of the device update; (v) *model update* – the central model locally updates the shared model based on the aggregated update computed from the clients that participated in the current model.

The applications of FL are many and it can be used to detect credit card fraud and at the same time enable different banks to collaboratively learn a shared model while keeping all their data locally in their own private database. With this approach the central model will see more fraudulent transactions and will as a consequence hopefully be able to classify fraudulent transactions more accurately than the models trained on only one local dataset. The way it works is that a beforehand determined model will be trained locally on each bank's dataset independently of each other and then the learning parameters will be sent to the central server. Now, a central model based on all participants input will be built. Finally, this central model is sent to all participating banks which they can use to detect fraudulent transactions. See Figure 8 for the training procedure in a federated learning framework with banks.

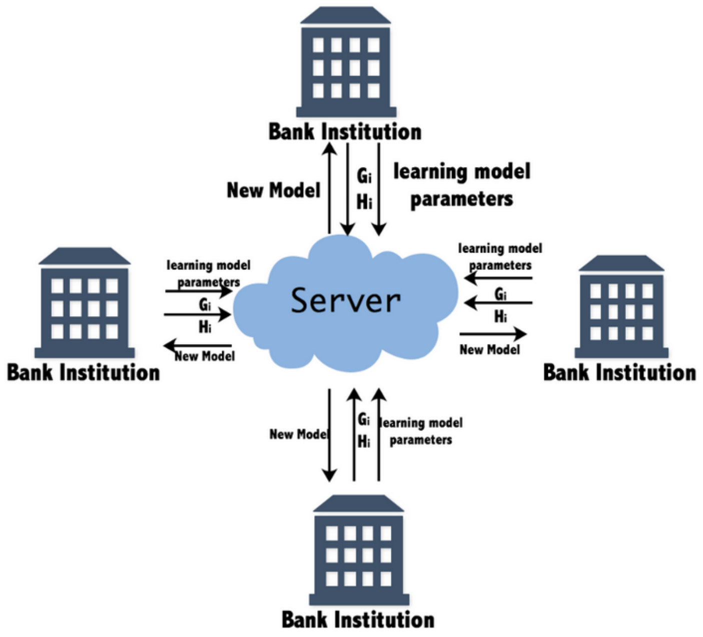


Figure 8: Different banks participating to build a central model by first training a model locally based on their own data and then sharing the encrypted information, weights for the MLP, with the central server in order to build a better central model for all participants [61].

Even though the aforesaid might be a good solution for the banks, other issues may still remain such as that the banks' datsets might differ due to different features and labels or that the datasets are skewed. When a dataset has these properties it is referred to as Non Identical Independent Distributed data, which is explained further in Section 2.4.1.

Moreover, as stated by Kairouz et al. [26], privacy and communication are always first-order concerns in federated learning and it is important to have a clear understanding of where the computation happens as well as what is being communicated. To solve the first question regarding the information flow in the system there are techniques for secure computation. Such techniques will be discussed in more detail in Section 2.4.3. In order to deal with the second issue, concerning how much information about a participating client that should be revealed, there are techniques for privacy-preserving disclosure and such techniques will be discussed in further detail in Section 2.4.4.

There exists different kinds of machine learning algorithms that can be used in a federated environment just as it does for the centralised approach. McMahan et al. [34] has implemented a federated algorithm called *Federated Averaging* which is a type of neural network algorithm. This algorithm will be used in this thesis because it is a federated development of the chosen baseline model, the MLP, which makes it easier to compare their performance to each other and also since the model effectively can be modified to our purpose, i.e. building a federated FDS.

2.4.1 Non Identical Independent Distributed Data

When working with machine learning, the assumption of *Independent and Identical Distributed* (IID) data is often made for the training dataset to imply that the samples are independent and originates from the same generative process, i.e. they are identically distributed [4]. However, when working with federated learning the datasets are typically non-IID [26]. For instance, in the FL setting, the datasets are typically generated locally in different contexts and by different clients.

According to Kairouz et al. [26], the most common source of dependence and non-identicalness in a dataset is due to each client corresponding to a particular user, geographic location and/or time window which leads to significant differences in the data distribution across participants. Further, they argue, that client data can deviate from being IID in different ways and some of them are: (i) *feature distribution skew*; (ii) *label distribution skew*; (iii) *same label, different features*; (iv) *same features, different label*; (v) *quantity skew or unbalancedness*. Most likely, the training dataset in a FL setting contains a mixture of these deviations.

Formally, (i)–(v) can be expressed in terms of probability theory. Consider a supervised leaning task with features x and labels y in addition to the dataset P and different clients i and j. By drawing a sample $(x, y) \sim P_i(x, y)$ from client i's local data distribution, $P_i(x, y)$ can be re-written as $P_i(y|x)P_i(x)$ and $P_i(x|y)P_i(y)$ by using the conditional probability formula. Now, according to Kairouz et al. [26], the differences between (i)–(v) mentioned above can be explained as follows, and a more thorough explanation can be found in [26].

Starting with feature distribution skew, this means that the marginal distribution $P_i(x)$ might vary across clients even though $P_i(y|x) = P_j(y|x)$ for all clients i and j. In other words, there might be differences in personal/organisational characteristics. For instance, in a handwriting recognition domain, persons who write the same words might still have different stroke width, slant, etc. to the words written [26].

Label distribution skew means that the marginal distribution $P_i(y)$ might vary across clients even though $P_i(y|x) = P_j(y|x)$ for all clients i and j. An example of this is clients that are tied to particular geographical regions, then the distribution of labels might vary across different clients — for mobile device keyboards, certain emojis are used by one demographic but not others etc. [26].

Same label different features means that the conditional distributions $P_i(x|y)$ might vary across clients even though $P_i(y)$ is the same. Meaning that, the same label y can have different features x for different clients due to, for instance, cultural differences, weather effects, standards of living, time or seasonal effects. A more describing example of this deviation would be that, images of homes can vary widely around the world, not only their design but also depending on the time of the day etc. [26].

Same features different label means that the conditional distributions $P_i(y|x)$ might vary across clients even though $P_i(x)$ is the same. An example of this could be that the same features might have different labels due to personal preferences. For instance, labels that reflect sentiment or next word predictors have personal and regional variation, etc. [26].

Finally, quantity skew or unbalancedness refers to the vastly different amount of data that the different clients can hold [26].

One problem that is associated with non-IID data, according to Zhao et al. [63], is that the majority of the FL settings rely on the Gradient Descent optimiser (explained in Section 2.3.1), where the assumption of IID training dataset is important to ensure that the gradient is an unbiased estimate of the full gradient. However, as mentioned in this section, this is rarely the case when working with federated learning. Though, they argue, that some federated learning algorithms, such as Federated Averaging, which is further explained in Section 2.4.2, still works with certain non-IID data, which is also shown by McMahan et al. [34].

2.4.2 Federated Averaging

The neural network algorithm referred to as Federated Averaging is similar to the centralised MLP. According to McMahan et al. [34], the main idea for the Federated Averaging algorithm is that for each global epoch of the training a central server chooses a fraction of the participating clients. The central server then sends its model parameters to each of the selected clients. When the training is done locally the clients make one (or more) local weight update, i.e. a local epoch, by applying Mini-Batch Gradient Descent (explained in Section 2.3.1) and eventually their updated weights are sent back to the central server. Now, the central server updates the federated model by averaging all the weight updates from the participating clients and then the process is repeated until convergence or satisfying results are achieved.

Usually, when working with machine learning algorithms the aim is to have as few epochs as possible but still obtain a model that performs well. This is because the amount of epochs are directly correlated to computer power and memory needed. Also, more epochs effect the privacy issue by adding to the privacy budget, see Section 2.4.4. McMahan et al. [34] proposes two approaches to limit the global epochs needed for convergence, i.e. limit the rounds of communications between the central server and the clients, namely, *increase parallelism*, i.e. increase the fraction of clients used, and *increase local computation*, i.e. increase the number of local epochs. When McMahan et al. [34] tested these two approaches in their experiments increase local computation showed the best effect of limiting communication while still keeping the same performance.

Moreover, the averaging of the weight updates in the central server could, according to McMahan et al. [34], be a problem when solving non-convex problems. They argue that in general, averaging models in parameter space can create an arbitrarily bad neural network model. This is because training several models on a non-convex problem can create significantly different models as they find different settings for the optimal weight parameters. Taking the average of models that have big disparity creates a poor central model. However, one method to minimise this problem is, according to the article, to use shared initialisation for the local models. In other words, each client is initialised with the same weights for every global epoch. This is how Federated Averaging is implemented in our thesis since, at the the start of each global epoch the central server initialises the clients with the same weights as its own. In Algorithm 1 the pseudo code for Federated Averaging is shown [34].

Algorithm 1 Federated Averaging. Here, C is the fraction of clients used, K are the clients indexed by k, n is the total number of data points, n_k is the number of data points in client k, P_k is the local dataset, B is the local mini-batch size, E is the number of local epochs and lastly η is the learning rate.

```
Server executes:
Initialise $w_0$
for each round t = 1,2... do
    $m \leftarrow max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    for each client $k \in S_t$ in parallel do
        $w_{t+1}^k \leftarrow$ ClientUpdate($k, w_t$)
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for

ClientUpdate($k, w$): // Run on client k
$B \leftarrow$ (split $P_k$ into batches of size $B$)
for each local epoch $i$ from 1 to $E$ do
    for batch $b \in B$ do
        $w \leftarrow w - \eta \nabla l(w; b)$
    end for
end for
return $w$ to server
```

2.4.3 Secure Computations

Traditionally, cryptography was about concealing content in communication and storage, but today there is more to it, for instance how the information flow in a system should be set up in order to to be secure and preserve privacy. One such technique, that addresses this privacy-preserving issue in machine learning, is *Secure Multi-Party Computation* (MPC) and is a sub-field of cryptography. According to Kairouz et al. [26], the goal for this encryption technique is to create methods for parties to jointly compute an agreed-upon function over their inputs without revealing any additional information. More precisely, Cramer et al. [9, p.32] describes MPC as each client P_i holding a secret input x_i and all clients agreeing to a function f, here a machine learning model, that take n inputs where the goal is to compute the output $y = f(x_1, \ldots, x_n)$, while making sure that the the correct value of y is computed and also that y is the only new information that is released.

An in-practice used MPC tool is *Secret Sharing* which, as argued by Cramer et al. [9], provides a way for each client to spread the secret information, e.g. weights from an MLP model, by splitting a secret key into shares. These shares are then sent across all the participating clients, which means that the clients together hold the complete information, yet no client, except the one who split the key can reassemble it and thus collect the information in full [9, p.36]. There are two commonly used sharing schemes to build a *protocol*, i.e. a set of instructions that the participating clients are supposed to follow in order to obtain the desired result [9, p.34], namely, *Shamir Secret Sharing* and *Additive Secret Sharing* [9, p.112]. In both of these schemes the shares are random elements of a finite field which adds up to the secret and eventually these shares are sent to the central server to compute an output. Now, the security is obvious as the shares look randomly distributed for all non-qualifying sets of shares.

In addition there is also *Homomorphic Encryption* (HE) schemes which have the potential to be a powerful tool for MPC as they have the ability of computing over encrypted data without prior decryption [26]. Thus, the results of the computation remains encrypted which means that the clients would be able to encrypt their data and send it to a central server for computation without ever revealing their information. There are different types of HE ranging from general fully Homomorphic Encryption to more specific ones. Today, data encrypted by secret sharing has homomorphic properties, but HE is not yet fully implemented since the technique has a rather high computational cost compared to other techniques [26]. A lot of research within the area of HE has however been made since researchers have seen great potential in HE for privacy-preserving machine learning [24].

2.4.4 Privacy-Preserving Disclosures

In federated learning the datasets are, as described in Section 2.4, never shared still, according to Geyer et al. [19], sensitive information about the datasets can be accessed through the weights and the model updates sent from the clients as well as from the central server. To minimise this risk the clients used in the training process can be disguised. There are different methods that can be used for this but a common method, according to [26], is *differential privacy*, and in particular *user-level differential privacy.*

The concept of differential privacy is to protect individuals' and clients' privacy by introducing a level of uncertainty into the model. The definition for differential privacy is, according to Cynthia Dwork [15]:

> "Differential Privacy describes a promise, made by a data holder, or curator, to a data subject, and the promise is like this:
>
> You will not be affected, adversely or otherwise, by allowing your data to be used in any study or analysis, no matter what other studies, datasets, or information sources, are available."

In other words, if an outside person gets access to the output but such person cannot tell who or which individual or client was used in the computation than the algorithm is said to be differential private. The user-level differential privacy, in contrast to the regular differential privacy, looks at all data points from a user and not just a single data point [26].

A model can have different levels of differential privacy, which is, as written in [26], quantified by the parameters (ϵ, δ) according to the following mathematical definition.

Definition. *A randomised algorithm A is (ϵ, δ)-differential private if for all $S \subseteq$ Range(A) and for all adjacent datasets D and D' differing one user's data,*

$$P(A(D) \in S) \geq e^{\epsilon} P(A(D') \in S) + \delta,$$

where ϵ, as stated in [30], is the measurement of what the privacy risk is to release sensitive data, also referred to as the *privacy loss* [1]. In other words, this parameter describes how much a person with the output would be able to see of the dataset. The δ parameter is the probability that an unwanted event happens that leaks more data the normally. Small values of ϵ and δ are desired since the smaller the values are the more secure the model is.

As a matter of fact, for every query or weight update sent to the central server leakage of information is possible. Meaning that for each query the privacy loss ϵ accumulates. When a query happens two times the total privacy loss is twice as large as for one. A limit of how much privacy loss that is allowed is, as stated by Abadi et al. [1], called *privacy budget.* Privacy budget is the maximum total privacy loss that can be afforded in the model. When the privacy budget is reached the clients stop sending and receiving updates. The user of the model determines how big a privacy budget is allowed to be through the δ and ϵ parameters.

Approaches to make a federated model subject to differential privacy involves, according to [26], to hide what weight updates were used in the central model as well as making the central model less prone to overfit one client. This is, as described in the article, done by clipping the individual weight updates and adding Gaussian noise to the aggregation in the central server. How much distortion of the weight updates that are needed depends on the level of differential privacy as well as the sensitivity of the model, where sensitivity is the maximum difference in the output from a model from two adjacent datasets. Formally,

$$Sensitivity = \Delta f = max||f(D) - f(D')||_2$$

where D and D' are adjacent datasets differing by a single user's data points [19].

Geyer et al. [19] describes a way to implement differential privacy using a modified Federated Averaging model. The difference here is that, the clients do not send their full weights to the central server, instead, they return the difference between its old and the newly trained weights. This way, the central server adds the average of the differences for each client to its own weights. Also, it does not use a weighted average as is the case for the regular Federated Averaging model. Worth mentioning is that, the result in the article from Geyer et al. [19] shows that in order to preserve privacy a large number of clients are needed in the training process.

In the approach described by Geyer et al. [19], the amount of weight clipping and Gaussian noise is controlled by the number of clients used in each global epoch, the hyper parameter σ as well as the sensitivity which is calculated by taking the median of the difference between the old weights and the new weights for the clients after a global epoch [1]. The limit of how much privacy loss that is allowed is termed by Geyer et al. [19] as δ and when this privacy budget is breached the training stops.

In Algorithm 2 the pseudo code for a modified Federated Averaging model with differential privacy is shown, which has been built on the principles of Geyer et al. [19].

Algorithm 2 Client-side differential private federated optimisation. The Accountant keeps track of the privacy budget (δ) in order to protect privacy. C is the fraction of clients used, K are the clients indexed by k, P_k is the local dataset, B is the local mini-batch size, E is the number of local epochs, η is the learning rate, $\{\sigma\}_{t=0}^{T}$ the variance for the Gaussian Noise, S the sensitivity and Q the threshold for δ.

```
Server execution
Initialise: $w_0$, Accountant($\epsilon, K$)
for each round t = 1,2... do
    $m \leftarrow max(C \cdot K, 1)$
    $\delta \leftarrow$ Accountant($m, \sigma_t$)
    if $\delta > Q$ then
        then return $w_t$
    end if
    $S_t \leftarrow$ (random set of $m$ clients)
    for each client $k \in S_t$ in parallel do
        $\Delta w_{t+1}^k, \xi^k \leftarrow$ ClientUpdate($k, w_t$)
    end for
    $S$ = median$\{\xi^k\}_{k \in S_t}$
    $w_{t+1} \leftarrow w_t + \frac{1}{m}(\sum_{k=1}^{K} \frac{\Delta w_{t+1}^k}{max(1, \frac{\xi^k}{S})} + N(0, S^2\sigma^2))$
end for

ClientUpdate($k, w_t$): // Run on client k
$w \leftarrow w_t$
$B \leftarrow$ (split $P_k$ into batches of size $B$)
for each local epoch $i$ from 1 to $E$ do
    for batch $b \in B$ do
        $w \leftarrow w - \eta \nabla l(w; b)$
    end for
    $\Delta w_{t+1} = w - w_t$
    $\xi = ||\Delta w_{t+1}||_2$
end for
return $\Delta w_{t+1}, \xi$ to server
```

3 Related Work

Today, a lot of different machine learning methods have been proposed to tackle fraud detection. Nearly all credit card FDS are built by using centralised techniques, a few examples are [53, 60, 47, 58, 49] in which both supervised, unsupervised and semi-supervised learning techniques are being used. There is, however, little research and therefore also few articles relating to FDS which has been built by applying federate learning.

The topic of federated learning to detect credit card fraud in the banking industry has, however, been addressed by Wensi et al. [62]. They are proposing a federated Convolutional Neural Network as a suitable ML algorithm for detecting credit card fraud. Though, they are not looking into the privacy concern.

Federated learning is also applied in other business areas, and where this learning technique has shown to be an efficient machine learning technique. For instance, in the healthcare business [59] and when working with keyboard predictions from an edge device such as a mobile phone [21].

4 Methodology

The dataset used in this thesis is introduced in Section 4.1. We have carried out an analysis of the dataset by studying the number of transactions over time, the number of transactions against withdrawal amount and finally the distribution of fraudulent and non-fraudulent transactions. This analysis is also presented in Section 4.1. Further, Section 4.2 contains an explanation of how the dataset has been pre-processed for this thesis and Figure 12 sets out the workflow for pre-processing the data. Moreover, to give a possible practical business perspective an interview has been carried out with an employee from IBM. The interview is described in Section 4.3. Finally, in Section 4.4 we present certain assumptions that have been made as well as certain limitations to the scope of this thesis.

4.1 Dataset

The dataset used in this thesis has been obtained from Kaggle [25] and contains real but anonymised credit card transactions made by European cardholders. The dataset contains 284 807 credit card transactions spread out over two days in September 2013. There is no missing data and out of the 284 807 transactions, only 492 transactions are fraudulent transactions, making the dataset heavily skewed. Further, it contains 30 features and where only two of them are known, namely, amount and time of the transactions. For the rest of the 28 features PCA has been applied due to anonymity reason. Before the data was released to the public each transaction had already been labeled as fraud or not fraud. Thus, there is no known noise in the dataset. See Table 1 for a summary of the dataset.

Total dataset	# fraud	# not fraud	Label not fraud	Label fraud
284 807	492	284 315	0	1

Table 1: Summary of the dataset obtained from Kaggle.

In Figure 9 it can be seen that the time for non-fraudulent transactions have a rather clear periodical pattern that follows day and night cycles, while fraudulent transactions do not.

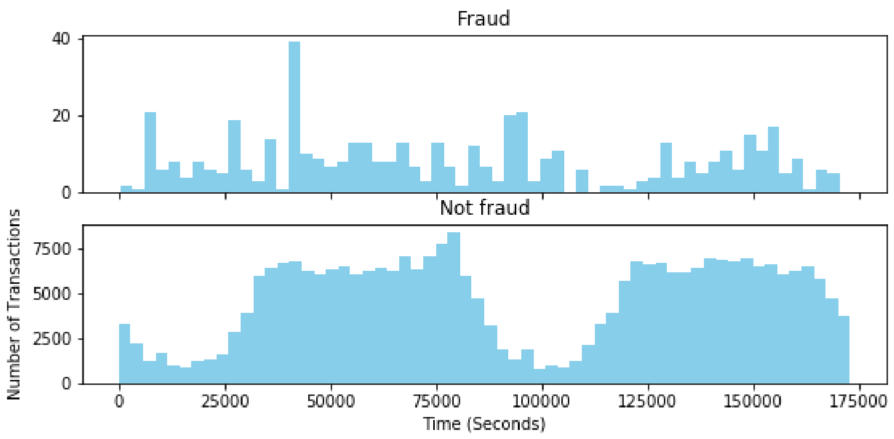


Figure 9: Number of transactions against time for the different classes.

Further, in Figure 10 it can be seen that the fraudulent transactions are usually small while the non-fraudulent transactions occur within a larger span of money.

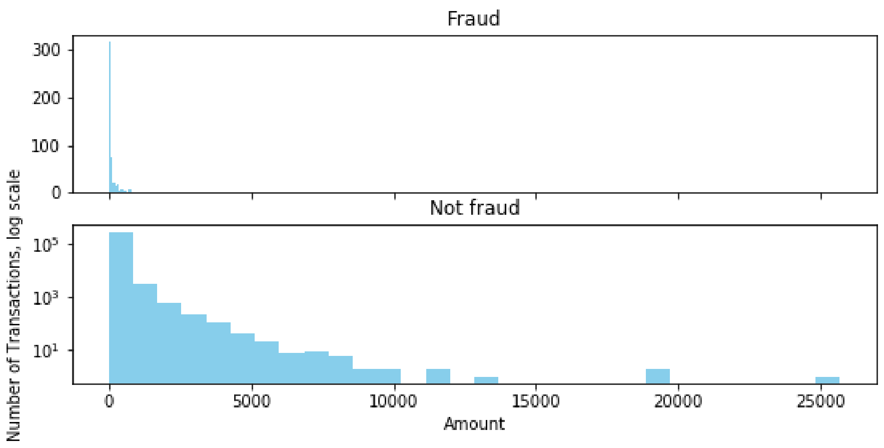


Figure 10: Number of transactions against amount for the different classes.

As a final data analysis the distributions for fraudulent and non-fraudulent transactions in time and amount have been carried out for all features, see Figure 11. It can be seen in the figures that for some features the curves for fraud and not fraud overlaps to a greater extent than for others. The interpretation of this is that, the more overlap of the curves the more difficult it is to extract differences between fraudulent and non-fraudulent transactions. When the distributions do not overlap at all or very little it is easier to find these differences.

This means that, for a model to be able to detect the differences between a fraudulent and non-fraudulent transaction the distributions should have as little overlap as possible.

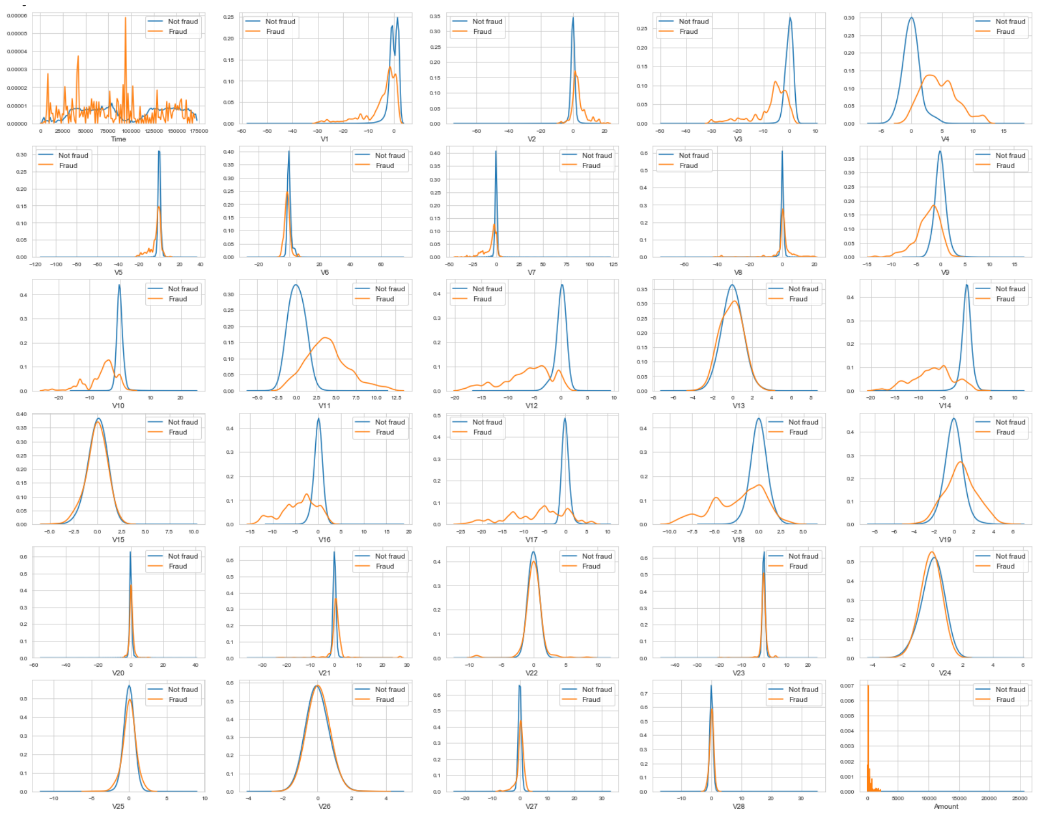


Figure 11: Distributions of fraudulent and non-fraudulent transactions for the different features.

4.2 Data Pre-Processing

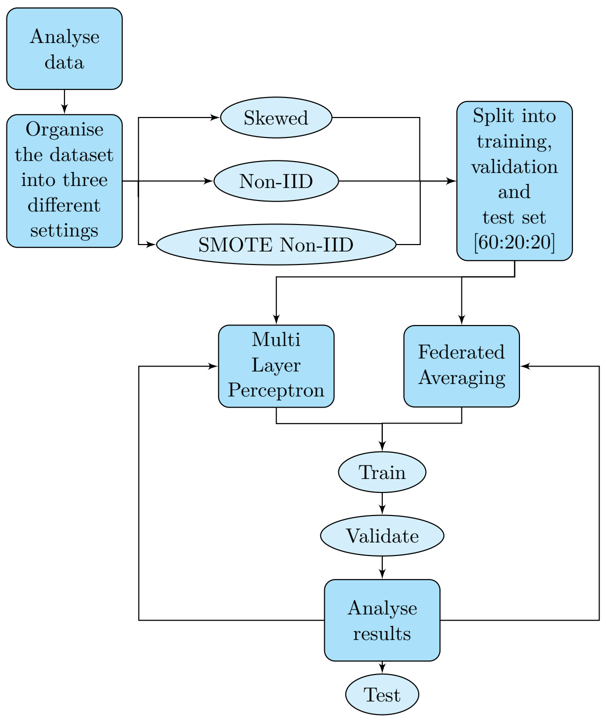


Figure 12: The workflow for pre-processing the data to be able to train the models.

The data is used in three different settings to be able to make three different experiments. The settings used are as follows: (i) no pre-processing of the data, i.e. skewed data; (ii) the data has been split into non-IID data and (iii) oversampling of the non-IID data by applying SMOTE. In all experiments the values of the features have been normalised to be in an interval of [-1,1]. Also, the time feature has been excluded as it is arbitrary from a certain starting point and therefore does not add any value.

In the first experiment, the data is simply split into a training, validation and test set, see Table 2.

Split of dataset	% of dataset	# fraud	# not fraud
Training	60	316	170567
Validation	20	78	56884
Test	20	98	56864

Table 2: Summary of the dataset for the skewed data setting (284 807 transactions).

In the second experiment the dataset is split in a way that makes it non-IID. The non-IID split is made by partitioning the data according to time since the fraudulent transactions do not have any periodic behaviour over time while non-fraudulent transactions do, see Section 4.1. Practically this is done by ordering the dataset after its timeline. Then, the training, validation and test sets are obtained by splitting the ordered dataset, see Table 3.

Split of dataset	% of dataset	# fraud	# not fraud
Training	60	360	170523
Validation	20	57	56905
Test	20	75	56887

Table 3: Summary of the dataset for the non-IID data setting (284 807 transactions).

Finally, in the third experiment the same setting as used in the second experiment has been used and in addition the minority class has been upsampled by applying SMOTE. Here, the fraudulent transactions have been upsampled to 1% of the non-fraudulent transactions. Thus, in this third setting the training dataset has more data points than in the two previous experiments, see Table 4.

Split of dataset	% of dataset	# fraud	# not fraud
Training	60	1705	170523
Validation	20	57	56905
Test	20	75	56887

Table 4: Summary of the dataset for the SMOTE non-IID data setting (286 150 transactions).

For the federated model, the training data was distributed to each client but in different ways depending on the dataset. In the first experiment, the split was distributed equally, meaning that each client had the same amount of transactions. In the other two experiments, each client received a time frame of transactions resulting in each client having different amount of transactions, hence, different distribution.

4.3 Interview – Federated Averaging

IBM's primary interest in this thesis lies towards federated learning as a concept rather than its application within a specific commercial field. We nonetheless wanted to understand and discuss whether there is commercial interest in our research relating to federated FDS within the financial sector, and particularly in the banking industry. For this purpose we conducted an interview with a person at IBM tied to this sector. The interviewer is working with Financial Services Risk and Compliance at IBM and has a solid background within business and finance. One person can obviously not represent the entire financial sector and the banking industry, but the person who we interviewed was the person that we had immediate access to within this field. A natural continuance to this thesis and for further research would of course be to conduct more interviews in order to obtain further insight in the application of federated FDS within the financial sector and the banking industry.

We conducted a semi-structured interview in order to obtain further information about the banking industry as well as the interviewee's perception and thoughts around the application of federated FDS within the said industry. The setting of the interview was a recorded video call which was saved as an audio file. The answers were compiled and reviewed after the interview. Answers which we felt contained unclear aspects were researched further. The outcome of our analysis of the answers obtained during the interview are presented in Section 6.1.

4.4 Assumptions and Limitations

When preparing and working with this thesis we have made certain assumptions. We also had to limit the scope of the thesis due to certain restrictions encountered during our work.

The Fraud Detection System explained in this thesis is built for IBM and not for any particular customer. Hence, we do not know if banks are willing to apply our model in the real world. However, in this thesis we assume that there is an interest for this kind of FDS and that banks would be willing to adapt this system. Furthermore, we assume that the dataset obtained from Kaggle is representative for one or several banks and also that it is correctly prepared, i.e. it is correctly labeled and there is no missing data.

Moreover, when building the model we have assumed that all clients participate on the same terms and contribute with equally important information. Therefore, we computed the updates for the model by using just a simple average instead of a weighted average, as described in Algorithm 1. The simple average has been chosen because the other approach with weighted average generally tends to overfit the model as it puts different weights on different clients which should not be the case when assuming that their contribution is equally important. Also, we have assumed that there are no faulty nodes, for instance, that the communication between the nodes/clients work properly.

In addition to these assumptions there are, as previously mentioned, certain limitations to this work which we are not able to influence. For instance, as this thesis is not made together with a bank we do not have any information about the cost for the banks when a fraudulent transaction is made or detected, hence, we cannot implement a cost sensitive learning as discussed in Section 2.2.

Also, the size of the dataset is a limitation as it is considered to be rather small with only two days of credit card transactions. As a consequence of its small size it does not contain many fraudulent transactions; only approximately 0.0017%. According to an employee, who is building (centralised) fraud detection systems at IBM, the minimum period to collect a credit card transactions dataset in reality is usually three months and, actually, big card processors can have up to as much as 13 months. Though, the employee claims that, for

Proof of Concept (PoC) purposes, a dataset of six to ten weeks is usually enough. This insight shows that the dataset obtained from Kaggle is small.

Furthermore, since PCA has been applied to the dataset from Kaggle before it was released to the public it is difficult to understand why the fraudulent transactions have been classified as fraud, i.e. what is classified as a fraudulent behaviour. As we do not know what is classified as fraud we cannot know if the fraudulent transactions might differ depending on, for instance, geographic location and since this dataset comes from European cardholders only we assume that it limits our model to European banks.

Another limitation is the frameworks that are used for coding the federated FDS. GPU is not yet supported when using the function *VirtualWorker* from PySyft, further explained in Section 5.2, which makes it difficult to run heavy machine learning algorithms without it taking too much time. In addition there is also a risk that the algorithms crash due to insufficient RAM-memory. Moreover, the only optimiser, which is yet supported for federated learning, in the frameworks used is Mini-Batch Gradient Descent, even though Adam would probably be a better optimiser for model as it is better suited for a binary classification problem. Finally, differential privacy as well as secure computation have not been implemented nor do the clients have different IP-addresses which would have made the training scenario more realistic.

5 Implementation of Models

In Section 5.1 the implementation of the centralised MLP is explained and in Section 5.2 the implementation of Federated Averaging is presented. Both implementations have in common that they are coded by using *PyTorch* [17] and *PySyft* [39] frameworks. These frameworks are an extension of the *Torch* [31] library. They are both open source machine learning libraries which aim to enable distributed learning and where PySyft in particular is a framework for encrypted and privacy-preserving machine learning.

5.1 Multi Layer Perceptron

The MLP was built by using the PyTorch class *nn.Module* [40], with three layers. An input layer with 29 nodes, a hidden layer with 15 nodes and an output layer with one node. The activation function was ReLU except for the output layer where Sigmoid was applied. Binary Cross Entropy was the loss function to be minimised by the optimiser Mini-Batch Gradient Descent. The learning rate was set to 0.002 with a batch size of 32 for the skewed data, 128 for the non-IID data and 256 for the SMOTE non-IID data and training was made until the AUPRC value on the validation set converged. See Table 5 for the setting of the MLP.

Layer	Nodes	Activation function	Loss function	Learning rate
Input	29	ReLU	Binary Cross Entropy	0.002
Hidden	15	ReLU	Binary Cross Entropy	0.002
Output	1	Sigmoid	Binary Cross Entropy	0.002

Table 5: Summary of the Multi Layer Perceptron setting.

5.2 Federated Averaging

For the Federated Averaging algorithm in this thesis four clients were created as well as one central server. The choice of just having four clients is due to the size of the dataset used. Also, the number of local iterations was chosen to be two and three since this generated the smallest cross entropy loss on the validation set when compared to one and four local iterations. The pseudo code for Federated Averaging is found in Algorithm 1 in Section 2.4.2.

The four clients were created as virtual machines by using *VirtualWorkers* in PySyft. This function works in such way that each virtual worker can have a dataset and a model without revealing them to the other workers nor to the central server [54]. Moreover, in the central server a simple MLP was created with the same architecture as the centralised MLP, see Table 5.

The training set was distributed through the clients in batches of size 32 for the skewed data, 128 for the non-IID data and 256 for the SMOTE non-IID data by using *FederatedDataLoader* from PySyft and training was made until the AUPRC value on the validation set converged. See Table 6 for the different configurations of the Federated Averaging model.

Model	Total # of clients	C	E
Model 1	4	0.5	2
Model 2	4	0.5	3
Model 3	4	0.75	2
Model 4	4	0.75	3
Model 5	4	1.0	2
Model 6	4	1.0	3

Table 6: The different configurations of the Federated Averaging model made for each dataset. In each model, the number of clients and local iterations, E, used in every global epoch are fixed but the clients that are used, C, were randomly chosen.

6 Results

In this section the results from the interview will be presented. In addition to this the results from the Multi Layer Perceptron and Federated Averaging are presented for the three different dataset settings, namely, skewed data, non-IID data and SMOTE non-IID data respectively.

6.1 Interview – Federated Averaging

Assuming that IBM owns the federated FDS, maintains and distributes it there is, according to our interviewee, a great interest for such product amongst different banks since they all want to prevent and detect credit card fraud. Though, as the banks' data is highly sensitive it is rarely shared between different financial institutions, if not forced by law, but only within the same company group. Therefore, it is important for the banks to have a clear plan of how the collaboration will work, i.e. what information will be shared between the clients and the central server, in order for them to determine if federated learning is a desired approach for them to try to detect credit card fraud.

Actually, the interviewee explained that there is an initiative called the Nordic Know-Your-Customer (KYC) Utility which is a collaboration between six Nordic banks where the aim is to combat financial crime and cut compliance costs by creating a platform with standardised processes for handling KYC data [57]. This initiative indicates that there is an interest from banks to collaborate in order to detect fraud if it can be guaranteed that their data will be kept confidential.

Furthermore, when discussing the application of this federated machine learning approach our interviewee believes that our FDS might be best suitable for banks, or other companies, that are already in a collaboration somehow. This is because, if they are already in a collaboration it might be easier to expand the collaboration since there is already a mutual trust. For instance, our interviewee suggests Sparbankerna as a possible customer since they all are part of the group Svenska Sparbanksföreningen.

6.2 Multi Layer Perceptron and Federated Averaging

In the following subsections the results from the Multi Layer Perceptron and the Federated Averaging will be presented. All subsections have the same layout, namely, they contain a table with different settings for the model and AUPRC score as well as confusion matrices with the results from the Multi Layer Perceptron and Federated Averaging.

6.2.1 Skewed Data

Model	Batch size	Epochs	Total # of clients	C	E	AUPRC
Centralised	32	120	1	1.0	1	0.82282
Federated	32	100	4	0.5	2	0.84192
Federated	32	100	4	0.5	3	0.85772
Federated	32	100	4	0.75	2	0.85473
Federated	32	100	4	0.75	3	0.86824
Federated	32	100	4	1.0	2	0.82824
Federated	32	100	4	1.0	3	0.84768

Table 7: Configuration and Area Under the Precision-Recall Curve (AUPRC) score for the federated as well as the centralised model.

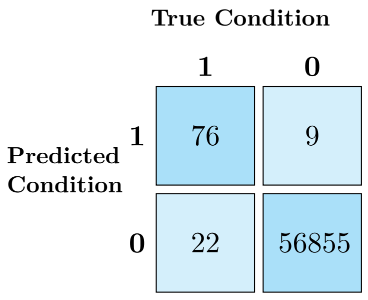

True Condition
1
0
Predicted Condition
1
76
9
0
22
56855

Figure 13: Confusion matrix for the centralised Multi Layer Perceptron applied on the test set.

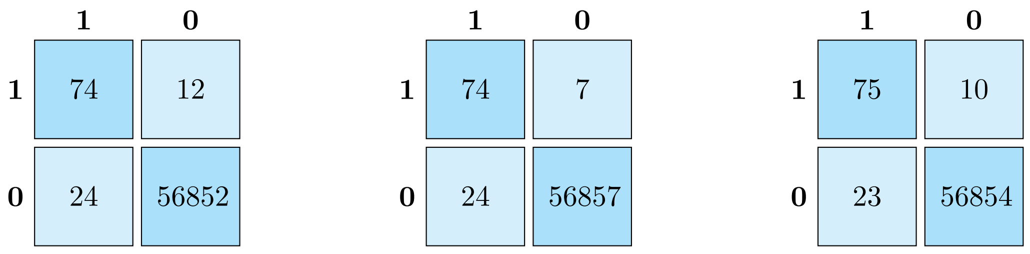

1
0
1
74
12
0
24
56852
1
0
1
74
7
0
24
56857
1
0
1
75
10
0
23
56854

Figure 14: Confusion matrices for Federated Averaging applied on the test set. *Left*: 2 local iterations and 2 random clients. *Centre*: 2 local iterations and 3 random clients. *Right*: 2 local iterations and 4 random clients.

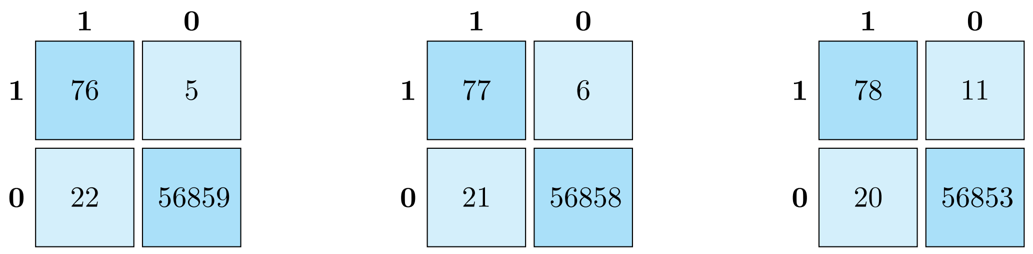


Figure 15: Confusion matrices for Federated Averaging applied on the test set. *Left*: 3 local iterations and 2 random clients. *Centre*: 3 local iterations and 3 random clients. *Right*: 3 local iterations and 4 random clients.

6.2.2 Non Independent Identically Distributed Data

Model	Batch size	Epochs	Total # of clients	C	E	AUPRC
Centralised	128	40	1	1.0	1	0.80593
Federated	128	60	4	0.5	2	0.80264
Federated	128	60	4	0.5	3	0.80723
Federated	128	60	4	0.75	2	0.81348
Federated	128	60	4	0.75	3	0.81566
Federated	128	60	4	1.0	2	0.80732
Federated	128	60	4	1.0	3	0.80926

Table 8: Configuration and Area Under the Precision-Recall Curve (AUPRC) score for the federated as well as the centralised model.

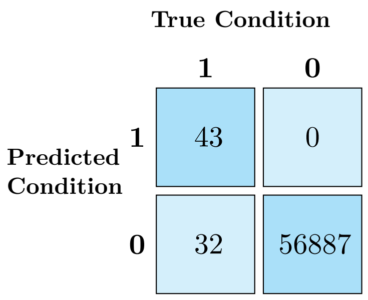
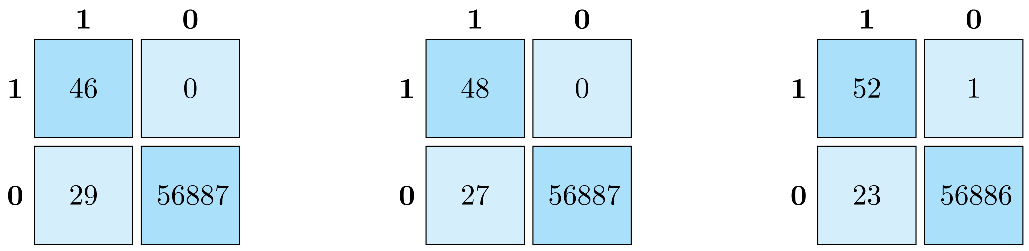


Figure 16: Confusion matrix for the centralised Multi Layer Perceptron applied on the test set.

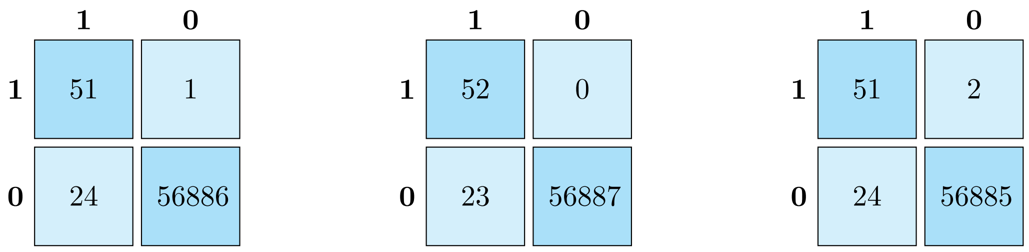


	1	0
1	46	0
0	29	56887

	1	0
1	48	0
0	27	56887

	1	0
1	52	1
0	23	56886

Figure 17: Confusion matrices for Federated Averaging applied on the test set. *Left*: 2 local iterations and 2 random clients. *Centre*: 2 local iterations and 3 random clients. *Right*: 2 local iterations and 4 random clients.



	1	0
1	51	1
0	24	56886

	1	0
1	52	0
0	23	56887

	1	0
1	51	2
0	24	56885

Figure 18: Confusion matrices for Federated Averaging applied on the test set. *Left*: 3 local iterations and 2 random clients. *Centre*: 3 local iterations and 3 random clients. *Right*: 3 local iterations and 4 random clients.

6.2.3 SMOTE Non Independent Identically Distributed Data

Model	Batch size	Epochs	Total # of clients	C	E	AUPRC
Centralised	256	30	1	1.0	1	0.80593
Federated	256	35	4	0.5	2	0.81228
Federated	256	35	4	0.5	3	0.79888
Federated	256	35	4	0.75	2	0.81538
Federated	256	35	4	0.75	3	0.80772
Federated	256	35	4	1.0	2	0.79251
Federated	256	35	4	1.0	3	0.80675

Table 9: Configuration and Area Under the Precision-Recall Curve (AUPRC) score for the federated as well as the centralised model.

		True Condition	
		1	0
Predicted Condition	1	59	45
	0	16	56842

Figure 19: Confusion matrix for the centralised Multi Layer Perceptron applied on the test set.

	1	0
1	54	9
0	21	56878

	1	0
1	57	4
0	18	56883

	1	0
1	54	8
0	21	56879

Figure 20: Confusion matrices for Federated Averaging applied on the test set. *Left*: 2 local iterations and 2 random clients. *Centre*: 2 local iterations and 3 random clients. *Right*: 2 local iterations and 4 random clients.

	1	0
1	56	16
0	19	56871

	1	0
1	57	17
0	18	56870

	1	0
1	57	11
0	18	56876

Figure 21: Confusion matrices for Federated Averaging applied on the test set. *Left*: 3 local iterations and 2 random clients. *Centre*: 3 local iterations and 3 random clients. *Right*: 3 local iterations and 4 random clients.

7 Discussion

One of the goal of the experiments was to see how much performance was sacrificed by having a federated model with individual and secure data instead of having everything centralised. The experiments have been performed several times and also on different computers and yet we have had similar results for each run, in other words, our results are consistent. Looking at the results, true positives in the confusion matrices as well as AUPRC score, for the Multi Layer Perceptron as well as for Federated Averaging, in Section 6.2.1–6.2.3 separately, it shows that the federated model is at pair and sometimes even better than the centralised model, contradictory to our expectations. We based our expectations on the training process of federated learning. The data is spread out between the clients and also in some configurations the federated model might have seen less data than the baseline model if only a fraction of the clients were used during training. Based on this we expected that the baseline model trained on the whole dataset would perform better than the federated model trained on a fraction of the clients, since the central model was updated by taking an average of the participating models.

Moving on to study the result for the different configurations of the federated model, namely, different fractions of random clients used as well as different values on local iterations. In Table 7, 8 and 9 it can be seen that there is only a small difference in the AUPRC score when changing random clients and/or local iterations. Similarly, studying the confusion matrices in Section 6.2.1–6.2.3 the results do not differ much from each other. These small differences in the results make it difficult to draw any definitive conclusions, however, some patterns can be distinguished. One such pattern is that, surprisingly, using all clients in the training does not always give a better result but sometimes using less random clients per epochs is better, which is the case for the non-IID and non-IID SMOTE data. This could be explained by the split of the dataset since one client might have a dataset that is not helping or even impairing the overall model. Moreover, concerning the number of local epochs, the result differs a lot depending on the number of clients and the data split. To be able to achieve the optimal number of local iterations more data and experiments are needed.

Applying SMOTE to the credit card transaction dataset the best performance was achieved by upsampling the fraudulent transactions by 1% of the non-fraudulent transactions. In order to determine this percentage we started by trying to upsample the fraudulent transactions to 50%, then 10% and then gradually decreasing by 1% at the time. For all tests the experiment with 1% upsampling gave the best result in form of correctly classified transactions. Intuitively, this might feel strange as it would mean that more fraudulent transactions in the dataset do not give better performance for the model, instead, the model performs worse. For the higher percentage of upsampling the models classified thousands of the non-fraudulent transaction as fraud, i.e. there were significantly more FP than when upsampling the fraudulent transactions with only 1%. This is due to the nature of SMOTE, as the synthetic fraudulent datapoints are too similar to the non-fraudulent transactions which already present in the dataset. This in turn makes it difficult for the models to predict correctly.

Furthermore, to investigate if SMOTE improved the performance for Federated Averaging the two different data settings, non-IID and SMOTE non-IID data, are compared. The results from these experiments were as expected. Meaning that Federated Averaging trained on the SMOTE non-IID data got better in detecting fraud but worse at distinguishing the classes when compared to when it was trained on non-IID data. By computing the percentage share of true positive from the confusion matrices in Figure 17, 18 for the non-IID data and Figure 20, 21 for the SMOTE non-IID data it can be seen that the percentage of TP is higher for the model trained on SMOTE non-IID data than for the one trained on non-IID data. In other words, the model trained on the SMOTE non-IID data is better at detecting fraud than the one trained on the non-IID data. On the other hand, looking at the AUPRC score in Table 8 and 9 it can be seen that the model trained on non-IID data became better at separating the classes than the one trained on SMOTE non-IID data, since the average AUPRC for the federated model is higher for Table 8 than for Table 9. It is logical that these are the results since there in the SMOTE non-IID data exists more data points from the fraudulent class than it does in the non-IID data. This in turn results in that the model trained on the SMOTE non-IID data is overfitted to the upsampled class, hence, it has difficulties to distinguish what is fraud and what is not.

To summarise, the results indicate that Federated Averaging can perform and even outperform the Multi Layer Perceptron on our dataset. The techniques for applying federated learning algorithms are available and could, as seen be applied to combat online credit card fraud detection. However, the configurations for Federated Averaging have only small effects on the outcome, therefore, more data is needed to determine what the optimal settings are.

8 Future Work

Today, a lot of different machine learning methods have been proposed to tackle fraud detection. Nearly all credit card FDS are built by using centralised techniques, a few examples are [53, 60, 47, 58, 49] in which both supervised, unsupervised and semi-supervised learning techniques are being used. There is, however, little research and therefore also few articles relating to FDS which has been built by applying federate learning but some possible future works are presented below.

A major improvement in evaluation of the Federated Averaging would be to work with a bank and obtaining more data without anonymised features as well as cost-based information to enable cost sensitive learning. All of which would make the model more realistic. Further, the model could also be more businesslike by implementing it in a more realistic real-life system like Docker [13] as well as to use different computers, i.e. different IP-addresses, instead of *VirtualWorkers*.

In addition to these improvements, there are other ways to develop and improve the model's performance. For instance, two issues and challenges that are not addressed in this thesis are concept drift and online learning. It would also be an improvement to implement secret sharing and differential privacy, as has been explained in Section 2.4.3

and Section 2.4.4 respectively.

Moreover, as stated in Section 4.4 we assume that there are no faulty nodes and that all nodes are equally important, but, looking at the result it seems to be better not to use all clients since some client's dataset might not be as important as others for the training process. Therefore, implementing a weighting system for nodes would be a suggestion for future work. Further suggestions might also be to look into other sampling techniques as discussed in Section 2.1.4.

Besides the Federated Averaging model, Liu et al. [32] proposes a privacy-preserving machine learning model called Federated Forest which is a lossless learning model of the traditional random forest method. Their main goal is not to implement this model in the banking industry but it certainly has the properties for it and would therefore be an interesting future work.

There is another rather new machine learning setting named gossip learning [22], which is a setting that could be interesting to study as part of any future work. Gossip learning is similar to federate learning and is described in [22] as a federated learning method without the central server. This learning technique would enable the training to be even more decentralised since no outside party would be needed to set up and own the central server.

9 Conclusion

To conclude, federated FDS enables banks to collaboratively reap the benefits of a shared model, which has seen more fraud than each bank alone, without sharing the dataset with each other. Hence, the sensitive information of the cardholders is protected. The techniques for applying federated learning algorithms are available and could, as seen in this thesis, be applied to combat online credit card fraud detection.

This project was, however, not made in collaboration with a bank nor was the credit card transaction dataset in a representative size for building a federated FDS. Therefore, the results should be construed and viewed in the afore said. One important result though was that federated learning and in particular Federated Averaging seems to be a suitable algorithm to use for credit card fraud detection since the results indicate that Federated Averaging can perform and even outperform the Multi Layer Perceptron on our dataset.

Moreover, it can also be concluded that there is an interest from banks to apply this type of FDS to combat fraud. Though, there are a few aspects that are crucial to them in order for them to join a collaboration with other banks. These aspects mainly relate to data security and privacy. There is extensive research within this area and today there is secure computations as well as privacy-preserving methods that can be implemented in federated algorithms in order to keep the security and privacy for participating banks.

References

[1] Abadi Martin, Chu Andy, Goodfellow Ian, McMahan Brendan H., Mironov Ilya, Talwar Kunal and Zhang Li. Deep Learning with Differential Privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*, October 2016. Doi: `10.1145/2976749.2978318`.

[2] Abdallah Aisha, Maarof Mohd A. and Zainal Anazida. Fraud detection system: A survey. *Journal of Network and Computer Applications*, 68(2):90–113, June 2016. Doi: `10.1016/j.jnca.2016.04.007`.

[3] Ahemad Faizan. Selecting the Right Metric for Skewed Classification Problems, March 2019. Url: `https://towardsdatascience.com/selecting-the-right-metric-for-skewed-classification-problems-6e0a4a6167a7`. Accessed: 2020-04-28.

[4] Analytics University. IID Assumption & Machine Learning Models, 2018. Url: `https://www.youtube.com/watch?v=MlqsXYuvClw`. Accessed: 2020-04-23.

[5] Batista Gustavo E., Carvalho Andre C. P. L. F. and Monard Maria-Carolina. Applying One-Sided Selection to Unbalanced Datasets. *Proceedings of the Mexican International Conference on Artificial Intelligence - MICAI*, pages 315–325, December 2006. Doi: `10.1007/10720076_29`.

[6] Brownlee Jason. How to Use ROC Curves and Precision-Recall Curves for Classification in Python, August 2018. Url: `https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python`. Accessed: 2020-02-20.

[7] Chawla Nitesh V., Bowyer Kevin, Hall Lawrence and Kegelmeyer Philip W. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, January 2002. Doi: `10.1613/jair.953`.

[8] Chawla Nitesh V., Japkowicz Nathalie and Kotcz Aleksander. Editorial: Special Issue on Learning from Imbalanced Data Sets. *ACM SIGKDD Explorations Newsletter*, 6, June 2004.

[9] Damgård Ivan B. Cramer Ronald and Nielsen Jesper B. *Secure Multiparty Computation and Secret Sharing.* Cambridge University Press, August 2015. Doi: `10.1017/CBO9781107337756`.

[10] Dal Pozzolo Andrea, Boracchi Giacomo, Caelen Oliver, Alippi Cesare and Bontempi Gianluca. Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015.

[11] Dal Pozzolo Andrea, Boracchi Giacomo, Caelen Olivier, Alippi Cesare and Bontempi Gianluca. Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–14, September 2017. Doi: `10.1109/TNNLS.2017.2736643`.

[12] Dal Pozzolo Andrea, Caelen Olivier, Le Borgne Yann-Aël, Waterschoot Serge and Bontempi Gianluca. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, 41:4915–4928, August 2014. Doi: `10.1016/j.eswa.2014.02.026`.

[13] Docker. Docker. Url: `https://www.docker.com/`. Accessed: 2020-04-23.

[14] Draelos Rachel, March 2019. Url: `https://glassboxmedicine.com/2019/03/02/measuring-performance-auprc/`. Accessed: 2020-02-21.

[15] Dwork Cynthia and Roth Aaron. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.*, 9(5):3–4, August 2014. Doi: `10.1561/0400000042`.

[16] Expert System Team. What is Machine Learning? A definition, March 2017. Url: `https://expertsystem.com/machine-learning-definition/`. Accessed: 2020-03-06.

[17] Facebook AI Research Lab. PyTorch. Url: `https://pytorch.org/`. Accessed: 2020-04-28.

[18] Geeks for Geeks. Introduction to Dimensionality Reduction, July 2019. Url: `https://www.geeksforgeeks.org/dimensionality-reduction/`. Accessed: 2020-05-08.

[19] Geyer Robin C., Klein Tassilo and Nabi Moin . Differentially Private Federated Learning: A Client Level Perspective. December 2017. Doi: `1712.07557`.

[20] Goodfellow Ian, Bengio Yoshua and Aaron Courville. *Deep Learning.* MIT Press, 2016. Url: `http://www.deeplearningbook.org`.

[21] Hard Andrew, Kiddon Chloé, Ramage Daniel, Beaufays Francoise, Eichner Hubert, Ramaswamy Swaroop, Kanishka Rao, Rajiv Mathews and Augenstein Sean. Federated Learning for Mobile Keyboard Prediction, 2018. Doi: `1811.03604`.

[22] Hegedűs István, Danner Gábor and Jelasity Márk. Gossip Learning as a Decentralized Alternative to Federated Learning. In *Distributed Applications and Interoperable Systems*, pages 74–90, June 2019. Doi: `10.1007/978-3-030-22496-7_5`.

[23] Imane Sadgali, Nawal Sael and Faouzia Benabbou. *Lecture Notes on Intelligent Transportation and Infrastructure – Comparative Study Using Neural Networks Techniques for Credit Card Fraud Detection [p.287–296].* Springer, 2019.

[24] Kamakin Ivan. Current state of MPC in privacy-preserving ML, September 2019. Url: https://medium.com/going-byzantine/current-state-of-mpc-in-privacy-preserving-ml-575486684f4c. Accessed: 2020-03-27.

[25] Kaggle. Credit Card Fraud Detection - Anonymized credit card transactions labeled as fraudulent or genuine, September 2013. Url: https://www.kaggle.com/mlg-ulb/creditcardfraud. Accessed: 2020-02-20.

[26] Kairouz Peter, McMahan Brendan, Avent Brendan, Bellet Aurélien, Bennis Mehdi, Bhagoji Arjun, Bonawitz Keith, Charles Zachary, Cormode Graham, Cummings Rachel, D'Oliveira Rafael G. L., Rouayheb Salim E., Evans David, Gardner Josh, Garrett Zachary, Gascón Adrià, Ghazi Badih, Gibbons Phillip B., Gruteser Marco, Harchaoui Zaid, He Chaoyang, He Lie, Huo Zhouyuan, Hutchinson Ben, Hsu Justin, Jaggi Martin, Javidi Tara, Joshi Gauri, Khodak Mikhail, Konečný Jakub, Korolova Aleksandra, Koushanfar Farinaz, Koyejo Sanmi, Lepoint Tancrède, Liu Yang, Mittal Prateek, Mohri Mehryar, Nock Richard, Özgür Ayfer, Pagh Rasmus, Raykova Mariana, Qi Hang, Ramage Daniel, Raskar Ramesh, Song Dawn, Song Weikang, Stich Sebastian U., Sun Ziteng, Suresh Ananda T., Tramèr Florian, Vepakomma Praneeth, Wang Jianyu, Xiong Li, Xu Zheng, Yang Qiang, Yu Felix X., Yu Han and Zhao Sen. Advances and Open Problems in Federated Learning, December 2019. Doi: 1912.04977.

[27] Kitten Tracy. Fraud: 'A Serious Problem'. *Bank Info Security*, August 2011. Url: https://www.bankinfosecurity.com/interviews/fraud-a-serious-problem-i-1228.

[28] Krawczyk Bartosz. Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence*, 5, April 2016. Doi: s13748-016-0094-0.

[29] Last Felix, Douzas Georgios and Fernando Bacao. Oversampling for Imbalanced Learning Based on K-Means and SMOTE. *Information Science*, October 2018. Doi: 10.1016/j.ins.2018.06.056.

[30] Lee Jaewoo and Clifton Chris. How Much Is Enough? Choosing ϵ for Differential Privacy. In *Information Security*, pages 325–340, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. Doi: 10.1007/978-3-642-24861-0_22.

[31] Leonard Nicholas, Tompson Jonathan, Zagoruyko Sergey, Dundar Aysegul, Jin Jonghoon, Massa Francisco, Canziani Alfredo, Desmaison Alban, Deltheil Cedric and Perkins Hugh. Torch. Url: http://torch.ch/. Accessed: 2020-04-28.

[32] Liu Yang, Liu Yingting, Liu Zhijie, Zhang Junbo, Meng Chuishi and Yu Zheng. Federated forest. *CoRR*, 2019. Doi: 1905.10053.

[33] Macaraeg Randy. Credit Card Fraud Detection, September 2019. Url: https://towardsdatascience.com/credit-card-fraud-detection-a1c7e1b75f59. Accessed: 2020-02-20.

[34] McMahan Brendan, Moore Eider, Ramage Daniel, Hampson Seth and Agüera y Arcas Blaise. Communication-Efficient Learning of Deep Networks from Decentralized Data, February 2016. Doi: `1602.05629`.

[35] Minegishi Tatsuya and Niimi Ayahiko. Proposal of Credit Card Fraudulent Use Detection by Online-type Decision Tree Construction and Verification of Generality. *International Journal for Information Security Research*, 1, March 2013. Doi: `10.20533/ijisr.2042.4639.2013.0028`.

[36] The Association of Certified Fraud Examiners. Report to the nations on occupational fraud and abuse, 2014. Url: `https://www.acfe.com/rttn-introduction.aspx`. Accessed: 2020-02-04.

[37] Olsson Mattias and Edén Patrik. Lecture notes – Feed-forward Neural Networks, 2019. Url: `https://liveatlund.lu.se/departments/theoreticalPhysics/FYTN14/FYTN14_2019HT_50_1_NML__1281/CourseDocuments/Chapt_3.pdf`. Accessed: 2020-02-12.

[38] Olsson Mattias and Edén Patrik. Lecture notes – Introduction to Artificial Neural Networks and Deep Learning, 2019. Url: `https://liveatlund.lu.se/departments/theoreticalPhysics/FYTN14/FYTN14_2019HT_50_1_NML__1281/CourseDocuments/Chapt_Intro.pdf`. Accessed: 2020-02-11.

[39] Openmined. PySyft. Url: `https://github.com/OpenMined/PySyft`. Accessed: 2020-04-03.

[40] Paszke Adam, Gross Sam, Massa Francisco, Lerer Adam, Bradbury James, Chanan Gregory, Killeen Trevor, Lin Zeming, Gimelshein Natalia, Antiga Luca, Desmaison Alban, Kopf Andreas, Yang Edward, DeVito Zachary, Raison Martin, Tejani Alykhan, Chilamkurthy Sasank, Steiner Benoit, Fang Lu, Bai Junjie and Chintala Soumith. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[41] Patel Samir and Amin Kiran R. Preservation of privacy in data mining by using pca based perturbation technique. In *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, pages 202–206, May 2015. Doi: `10.1109/ICSTM.2015.7225414`.

[42] Patidar Raghavendra and Sharma Lokesh. Credit card fraud detection using neural network. *International Journal of Soft Computing and Engineering (IJSCE)*, 1:32–38, January 2011.

[43] Patrikar Sushant. Batch, MiniBatch & Stochastic Gradient Descent, October 2019. Url: `https://towardsdatascience.com/`

batch-mini-batch-stochastic-gradient-descent-7a62ecba642a. Accessed: 2020-05-08.

[44] Quah Jon and Sriganesh Srihari. Real Time Credit Card Fraud Detection using Computational Intelligence. pages 863–868, August 2007. Doi: 10.1109/IJCNN.2007.4371071.

[45] Scikit Learn. sklearn.metrics.average_precision_score. Url: https://scikit-learn.org/stable/modules/generated/sklearn.metric.average_precision_score.html. Accessed: 2020-06-16.

[46] SHIFT. Credit Card Fraud Statistics, February 2020. Url: https://shiftprocessing.com/credit-card-fraud-statistics/. Accessed: 2020-04-28.

[47] Singh Gajendra, Gupta Ravindra, Rastogi Ashish, Chandel Mahiraj and Ahemad Riyaz. A Machine Learning Approach for Detection of Fraud based on SVM. *International Journal of Scientific Engineering and Technology*, 1:194, July 2012.

[48] Smith Lindsay I. A tutorial on Principal Components Analysis, February 2002. Url: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf. Accessed: 2020-02-06.

[49] Sonawane Yogesh B., Gadgil Akshay S., More Aniket E. and Jathar Niranjan K. Credit Card Fraud Detection Using Clustering Based Approach. In *International Journal Of Advance Research And Innovative Ideas In Education*, volume 2, 2016.

[50] Sorournejad Samaneh, Zojaji Zahra, Atani Reza Ebrahimi and Monadjemi Amir. A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective. November 2016. Doi: 1611.06439.

[51] Stolfo Salvatore J., Fan David W., Lee Wenke and Prodromidis Andreas L. Credit card fraud detection using meta-learning: Issues and initial results. In *of AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 83–90, 1997.

[52] Sweeney Latanya. Simple Demographics Often Identify People Uniquely. January 2000. Doi: 10.1184/R1/6625769.v1.

[53] Sweers Tom . Autoencoding Credit Card Fraud. Bachelor thesis, Radboud University, July 2018.

[54] Task Andrew. Secure and Private AI. Chapter: Federated Learning. Accessed: 2020-04-03.

[55] The Association of Certified Fraud Examiners. What Is Fraud? Url: https://www.acfe.com/fraud-101.aspx. Accessed: 2020-05-08.

[56] The Learning Machine. Handling Imbalanced Data SMOTE. Url: https://www.thelearningmachine.ai/smote. Accessed: 2020-05-19.

[57] Wragg Eleanor. Nordic KYC utility takes shape, July 2019. Url: `https://www.gtreview.com/news/europe/nordic-kyc-utility-takes-shape/`. Accessed: 2020-05-04.

[58] Xie Xiaobo, Xiong Jian, Lu Liguo, Gui Guan, Yang Jie, Fan Shangan and Li Haibo. *Generative Adversarial Network-Based Credit Card Fraud Detection*, pages 1007–1014. January 2020. Doi: `10.1007/978-981-13-6508-9_122`.

[59] Xu Jie and Wang Fei. Federated Learning for Healthcare Informatics, November 2019.

[60] Xuan Shiyang, Liu Guanjun, Li Zhenchuan, Zheng Lutao, Wang Shuo and Jiang Changjun. Random forest for credit card fraud detection. In *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, 2018.

[61] Yang Mengwei, Song Linqi, Xu Jie, Li Congduana and Tan Guozhen. The Tradeoff Between Privacy and Accuracy in Anomaly Detection Using Federated XGBoost, 2019. Doi: `1907.07157`.

[62] Yang Wensi, Zhang Yuhang, Ye Kejiang, Li Li and Xu Cheng-Zhong. FFD: A Federated Learning Based Method for Credit Card Fraud Detection. In *Big Data – BigData 2019*, pages 18–32. Springer International Publishing, 2019. Doi: `10.1007/978-3-030-23551-2_2`.

[63] Zhao Yue, Li Meng, Lai Liangzhen, Suda Naveen, Civin Damon and Chandra Vikas. Federated Learning with Non-IID Data, August 2018. Doi: `1806.00582`.