Within the domain of Data Analytics (DA), there is an important field known as Machine Learning (ML), which is embedded in everyday people's lives. A significant topic within this field is Federated Learning (FL). FL occurs when different devices collaborate to build a common model without exchanging their data; instead, the data remains on the devices, and only model updates are sent to a central server where aggregation occurs.

A good example for FL in everyday people’s lives is when phone users are typing a message, and the keyboard predicts words to complete the sentence, a model developed by Google engineers.

A different example where FL is helping society move forward is in the healthcare sector. Patients are using wearable devices to track their movements and help doctors monitor their health conditions.

FL is growing rapidly and is helping the technological and medical sectors build robust machine learning models. This significant development motivates the focus of this thesis.

The primary purpose of this project is to experiment with FL frameworks to evaluate their implementability and develop a functional FL server. Therefore, the research objectives are:

* **To evaluate the implementability of existing FL frameworks.** This section experiments with popular FL frameworks, such as *PySyft, FATE, Flower, FedML* and *TensorFlow Federated* (TFF)*.* by examining their architecture and their applicability to real-world FL scenarios. This evaluation serves as the starting point for the primary research.
* **To develop a FL server**. A practical example of a web Flask FL server will be built, featuring two scenarios: technological and pharmaceutical. Each scenario will run separately, connecting five clients. Each scenario will have its own data, synthetic data for the technological scenario and images for the pharmaceutical scenario. These configurations will be trained, validated, and tested using Neural Networks (NN) that will classify binary outputs. This experiment aims to bridge the gap between popular FL frameworks and real-world FL applications.
* **To compare FL frameworks and the FL server.** This includes discussing the pros and cons each method brings to key industries such as the technological and pharmaceutical sectors. The comparison will address aspects like ease of implementation, scalability, data privacy, and model performance.

This concept was introduced in 2016 by Google engineers (McMahan et al., 2016). FL is a shared model that is trained across multiple devices, often referred to as clients. Each client trains its own local model and sends the updates to a central server, where the updates are aggregated to improve the global model. It is important to note that clients do not exchange their data; the data remains private for each client. This principle drove the design of FL, following the concepts of focused collection or data minimization, which were introduced by the White House in 2013. The intent is to prevent personal data from being sent over the network and potentially being stolen or manipulated by malicious third parties.

Depending on the nature of client FL can be classified in two types cross-device and cross-silo

The clients for cross-device can be mobile devices, edge devices, Internet of Things (IoT) devices, smartphones, tablets, wearables, etc. Figure 2.2.1. illustrates this scenario.  
The characteristics are, the high number of participants it can be thousands to millions of devices, it may have limited processing power and battery life, datasets tend to be small and network bandwidth may be limited. Devices may also connect and disconnect intermittently.

In this scenario, clients can be organizations or institutions such as hospitals, banks, and companies using large data centers. Figure 2.2.2 illustrates a cross-silo ecosystem. Some differences compared to the cross-device scenario include, clients are no longer small devices, there are fewer clients, clients have high computational power and large datasets, and the network is reliable with stable communication.

FL can be categorized according to the distribution of the data held by the clients participating in the modelling. These categories help to understand the different methodologies and use cases for FL.

Horizontal Federated Learning (HFL) or sample-based FL, occurs when different clients have datasets that share the same feature space but differ in the samples they hold (see Figure 2.3.1). A practical example of HFL is when two hospitals in different regions each have patient records with the same features (e.g., age, height, weight, diagnosis) but for different patients. These hospitals can collaborate to train a model to predict disease outcomes without sharing patient data.

Vertical Federated Learning (VFL), or feature-based FL, occurs when different clients have datasets that share the same sample IDs but differ in the feature space. An example of this is a bank and an e-commerce company that have data on the same set of customers. The bank has financial information such as credit scores and loan histories, while the e-commerce company has purchase behaviours and browsing records. By combining their data, they can build a model to predict customer credit without sharing raw data.

Federated Transfer Learning (FTL) is applicable when the datasets of different clients differ in both samples and features, and the overlap between both is minimal. To clarify, a good example is a European pharmaceutical company and a healthcare research institution in China collaborating using FTL. The pharmaceutical company has drug efficacy data, while the healthcare institution has patient health records. Even though they have different types of data, it is possible to train a global model to predict drug effectiveness on certain health conditions.

Terms like FL and Distributed Machine Learning (DML) can create confusion due to their similarities. The main difference lies in the training process: in FL, there is a central server that aggregates updates sent by the clients, whereas in DML, there is no central server; instead, data is spread across different nodes and computations are shared among these nodes. Table 2.4 will help clarify the intricacies of each concept.

A requirement for selecting the FL frameworks was that they must be open source. Open-source frameworks are transparent and trustworthy, developed and maintained by a collaborative community, free to use, and constantly evolving. Additionally, they can be customized to meet users' specific needs. After the selection it was necessary to rank the frameworks. To accomplish this GitHub stats were helpful. Figure 3.1. depicts *PySyft* GitHub repository stats.

Contributors, forks and stars were counted for each framework. These stats were then normalised and finally averaged. Figure 3.2. illustrates the formulas and table 3.1. shows the results being *PySyft* the most popular open-source FL framework.

By creating this ranking, the population for objective one was defined as *PySyft, FATE, Flower, FedML,* and *TFF*. The population was restricted to the top five FL frameworks due to the limited amount of time. As the sampling method is non-probabilistic and the sampling type is judgmental, this approach to ranking the FL frameworks may help mitigate the inherent bias that experimentation has as a primary research methodology and also focus the selection on samples that can represent the entire population. Based on this selection, section 3.1.1. details the frameworks that will be used in the experimentation, and section 3.1.2. lists are the remaining frameworks.

introduced *PySyft* a multi-language library that facilitates secure and private ML. It was developed by the *OpenMined* community with the objective of making FL data science more accessible through Python bindings and user-friendly interfaces. *PySyft* uses libraries like *PyTorch* and TensorFlow with additional capabilities. Comparing it with other frameworks like *TFF* and *PaddleFL*; *PySyft* offers detailed building blocks, allowing developers to implement FL efficiently. Also compared to Flower that supports heterogeneous client environments and offers tools for mobile and edge devices, claiming and advantage over *PySyft* in these aspects.

According to FATE is provided to aid enterprises and institutions in implementing large-scale and distributed collaborative learning with data protection. A number of secure computation protocols and machine learning algorithms are supported within FATE. Through the out-of-box usability and end-to-end building modules and visualization tools, users are able to get their applications up and running with efficiency and effectiveness. It not only offers a distributed platform that supports both stand-alone and cluster deployment but also privacy-preserving *XGBoost*, federated transfer learning, and multi-variate data. *FATE* interacts with users using *FATE-FLow*, which serves as the scheduling system, *FATE-Board*, a visualization tool, and *FATE-Serving*, which is an inference high-performance serving engine. *KubeFATE* is designed by *VMware* to have *FATE* constructed over *Kubernetes* at the data centre, hence an enterprise-managed solution over organizations' distributed infrastructure. It also supports cross-cloud deployment and management through *FATE-cloud*. Second, *FATE* has a security definition in which all parties are honest-but-curious, ensuring that the server learns only aggregated parameters, but not the data of any individual. It guarantees performance that is lossless, which means the algorithms in *FATE* provide comparable accuracy to a centralized solution. *FATE* supports research into the industry communities working together and has been seen as an increasingly business application of interest. Future work in the field will focus on the integration of blockchain functionalities into *FATE*; building lightweight versions of *FATE* for edge deployment and applications; and building new applications using *FATE* in an industrial scenario, such as computer vision and automatic speech recognition.

*FedML* is an open research library and benchmark built for enabling development support and fair comparison in federated learning algorithms. Compared with previous works, it addresses the current limitation of supporting different configurations and computing paradigms for distributed training, mobile on-device training, and standalone simulation. It makes flexible, generic API designs, standardized algorithm implementations, and a comprehensive benchmark dataset available for non-I.I.D. settings. *FedML* is architected into high-level API interactions through its *FedML-API*, whereas the low-level functionality is realized by *FedML*-core to allow convenient implementation of distributed algorithms by users. This library also includes a real-world module for training on smartphones, called *FedML-Mobile*. Using such cryptographic primitives, standardized benchmarks can enforce privacy, security, and robustness, ensuring fair comparisons. *FedML* is designed to encourage community contributions that push the boundaries of what it can do. In design, the critical requirements are met for federated learning research by which researchers can prototype new algorithms and evaluate them on a common fair platform with consistent datasets and experimental settings. The broad support of computing paradigms by the library will make it applicable in different research scenarios, from huge-scale distributed systems to resource-constrained mobile devices. This flexible design of the API allows researchers to extend and customize the library for their specific needs. Standard benchmarks enable trustworthy comparisons of the performance of different algorithms. Moreover, *FedML* is not only robust in terms of privacy and security in FL but also applies advanced cryptographic techniques that ensure user data is secure to the level of model robustness. FedML follows a community-driven approach and is always changing and extending its features. New improvements are regularly updated based on feedback and contributions from global researchers. Such a collaborative effort helps push the frontiers of FL and ensures that *FedML* retains its leading status in research and development.

presented a user-friendly framework, an open-source framework developed to make the implementation and scalability of FL much easier. *Flower's* goal is to bridge the gap between academic research and practical application in real-world FL settings with large-scale experiments and very varied device settings. The big advantage of *Flower*, compared to most other frameworks for simulations, is that it can be used in real deployments with real devices; thus, it is a very good and flexible tool. It has been designed with an architecture supporting most machine learning frameworks, including *TensorFlow*, *PyTorch*, while offering flexible API designs, standardized algorithm implementations, and benchmark datasets for non-IID settings. This way, it has proven to be an excellent tool for experimenting with FL in different configurations and computational paradigms. The important abstractions and functionalities inside Flower are the high-level API interactions in the part represented by *FedML-API*, and low-level functionality in the part represented by *FedML-Core*. This makes it easier for the users to program distributed algorithms. It also has an on-device training capability for smartphones with cryptographic techniques to guarantee privacy, security, and robustness called *FedML-Mobile*. This is a framework motivating the community's contribution continuously to increase the power of it. The architecture of Flower allows a transparent, seamless transition for researchers from simulation to deployment on real devices. With heterogeneous client support and scalable infrastructure, Flower becomes a tool absolutely necessary in the hands of the researcher when FL investigation is performed so that the gap between theory and practice may be addressed.

delve into how *TFF*, an open-source framework, is utilized for machine learning on decentralized data. It has been designed for research and experimentation. Some of the key features are TFF enables FL through low-latency models with less power consumption. The framework uses two layers, the FL learning Application Programming Interface (API) and the federated core (FC) API. The FL API allows developers to implement training and evaluation on existing TensorFlow models through a high-level interface. The FC API integrates TensorFlow with distributed communication operators focusing on computations across distributed systems like mobile phones, tablets, and sensors. Comparing *TFF* to other frameworks, it offers a unique well integrated structure others do not provide this level of integration. *TFF* allows experimenting with new algorithms is not tied to predefined algorithms.

Another innovative framework OpenFL created by Intel Labs and the University of Pennsylvania, OpenFL supports decentralized machine learning models. It allows organizations to train models using data locally without any transfer, and that operates by distributing a global model across various nodes while each organization trains its model locally. Model updates are sent to an aggregator to enhance the global model. This framework is compatible with popular ML frameworks like TensorFlow and PyTorch. In comparison to other frameworks, it stands out due to its open-source nature, TensorFlow Federated or PySyft focus more on academic research applications while OpenFL is focused on real-world applications.

describe NVIDIA FLARE (NF) as an open-source Software Development Kit (SDK) purposefully developed to make it easier for data scientists and researchers to train federated learning models. NF, in support of many collaborators, is applied to create powerful and generalizable AI models by sharing the weights of the models rather than the private data. It is very lightweight and flexible, supporting the scaling of different machine learning frameworks, among which are *PyTorch*, *TensorFlow*, and *XGBoost*. In this way, NF allows researchers to adapt their ML workflow under a federated paradigm and finally achieve secure and privacy-preserving multiparty collaboration through techniques like homomorphic encryption and differential privacy. Some of the key aspects found in NF are high-level APIs of programmable FL workflows, prototyping simulators, and a project management dashboard. It is constructed to support productivity features in the built-in SDK research to deployment simulation to the real-world architecture of NF: multitasking, high availability, server failover, and secure provisioning. In addition, a good application for NF has been found in practice, particularly within the health sector, with regard to predicting clinical outcome for COVID-19 patients and segmenting brain lesions in medical imaging. This paper also presented some of the numerous benefits that a component-based design of NF accrues to make it extensible and customizable, thereby inviting the research community to further develop it.

propose *PaddleFL* is an open-source federated learning framework developed by *PaddlePaddle*, with the purpose of safe collaboration on training machine learning models over a massive amount of devices or organizations without sharing raw data. It provides implementations for different federated learning algorithms and flexible, extensible architecture that will easily plug into different machine learning frameworks. In the aspect of implementations, PaddleFL is an attempt for distributed model training, while it solves the privacy issues in data. In this way, this project will have applications in multiple fields: healthcare and finance, where data security becomes very important. The framework itself has built-in tools for data preprocessing and model training and evaluation under horizontal and vertical federated learning scenarios. It aims to be user-friendly by having complete documentation and examples that can assist the user in quickly getting a foothold. PaddleFL adopts advanced techniques, including homomorphic encryption and secure multiparty computation, into data safety and privacy during the training process. It is also updated and improved all the time by the open-source community, making it a strong candidate for the implementation of federated learning projects.

introduced *Substra* as a framework designed to make machine learning both collaborative and secure. They developed *Substra* to handle the tricky issue of working with sensitive data without compromising privacy. Instead of moving data around, *Substra* keeps it decentralized data stays where it is, and only the necessary algorithms and non-sensitive information are shared. *Substra* uses Distributed Ledger Technology (DLT) to ensure that all operations are secure and traceable. This means there is no need to rely on a central authority to verify the integrity of the data and operations. Originally designed for healthcare applications, *Substra* is flexible enough to work with various data types, algorithms, and programming languages. It supports multiple computation methods, especially those used in Federated Learning. The framework is built on three core principles: collaboration, privacy, and traceability. It brings together data providers and algorithm designers to work on shared goals while keeping data private and secure. Substra manages four key assets: objectives, datasets, algorithms, and models. Each of these assets has specific permissions to control who can access and process them. Computations in *Substra* are coordinated across different nodes, ensuring that data never leaves its original location. The decentralized architecture uses smart contracts to enforce permissions and maintain a tamper-proof ledger of all activities. This makes *Substra* a versatile tool for various collaborative machine learning projects, such as data and algorithm collaborations, data consortiums, and combined training and evaluation efforts.

developed *FLGo* a platform designed to streamline the process of cross-application FL research and enhance shareability among developers. It is a lightweight FL framework aiming to be a customizable solution to suit different applications and data heterogeneity. *FLGo* addresses the gap that exists in current FL frameworks which often make the FL deployment very complex. Some of the key Features, are benchmarks and algorithms, customization, experimental tools, and high degree of shareability. Compared to other frameworks it stands out in, system heterogeneity, high-level API, multi-architecture support, asynchronous operations and customization and flexibility. As a conclusion *FLGo* has been developed with the intention of making FL more accessible to a broader range of developers by simplifying customization and enhancing its shareability. It also aims to bridge the existing gap with conventional machine learning and FL.