Cao, Tien-Dung; Truong, Linh; Truong-Huu, Tram; Nguyen, Tri

**Enabling Awareness of Quality of Training and Costs in Federated Machine Learning Marketplaces**

*Published in:*
2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)

Accepted/In press: 01/01/2022

*Please cite the original version:*
Enabling Awareness of Quality of Training and Costs in Federated Machine Learning Marketplaces

Tien-Dung Cao∗, Hong-Linh Truong†, Tram Truong-Huu‡, Minh-Tri Nguyen†

∗School of Engineering, Tan Tao University, Vietnam
†Department of Computer Science, Aalto University, Finland
‡InfoComm Technology Cluster, Singapore Institute of Technology, Singapore

Email: dung.cao@ttu.edu.vn, {linh.truong,tri.m.nguyen}@aalto.fi, truonghuu.tram@singaporetech.edu.sg

Abstract—The proliferation of data and machine learning (ML) as a service, coupled with advanced federated and distributed training techniques, fosters the development of federated ML marketplaces. One important, but under-researched, aspect is to enable the stakeholder interactions centered around the quality of training and costs in the marketplace and the service models in federated ML training. This paper conceptualizes a federated ML marketplace and proposes a framework to enable the awareness of the quality of training and a variety of costs where both data providers and ML model consumers can easily value the contribution of each data source to ML model performance in nearly real-time. This improves the transparency and explainability w.r.t. the quality and costs for participation in the marketplace. Based on that, we design and implement the quality of training and cost awareness framework for an edge federated ML marketplace. Experimental realistic scenarios show the usefulness of cost and quality details that provides insightful information for various purposes, including potential budget management and training optimization.

Index Terms—ML as a Service, Marketplaces, Cost Evaluation, Federated Learning, Observability

I. INTRODUCTION

One of the key factors for the success of machine learning (ML) wide adoption is the availability of data for training ML models. To date, a large amount of data collected every day by different parties that can be used for ML training is mostly stored at the edges or parties’ on-premise servers and clouds, due to various business conditions and data regulations. The advanced development of the Internet of Things (IoT) further enables data collection combined with ML-based analytics within the edges. These have led to various distributed and federated ML works to leverage distributed data sources. Independently, data marketplaces and ML-as-a-service (MLaaS) [1], [2], [3] are on the rise. The marketplaces and MLaaS typically enable customers to buy not only data but also infrastructural resources (computing resources) and services (MLOps and ML training features) provided by third parties. We advocate the development of marketplaces and service models for federated ML training as we have observed the need to provide richer ecosystems of services for ML beyond MLOps, such as Sagemaker [4], and ML serving platforms, such as Sagemaker [4] and Seldon [5]. A novel federated machine learning marketplace (FMLM) and its relevant service providers would combine several aspects of data marketplaces [6], [7], distributed training, and federating training [8] and MLOps as a service [4], [5].

From the service view, one of the key requirements to use FMLM is the benefit of delivering trained ML models with the expected quality, cost, and time based on the underlying provided data and computing infrastructures. Such a benefit can be represented in terms of quality of training (QoT) characterizing important aspects of ML model training, such as the quality of the trained model (performance), paid cost, and the time to deliver the expected trained model. The costs and QoT enable the awareness of ML model consumers on the trade-off between the quality of the obtained model (e.g., its performance) and the cost paid. They support transparency and explainability on how data and computing resources are used and contributed to ML models, thus motivating the broad participation of different stakeholders in the marketplace. However, before one can optimize what would be the best QoT result and how to achieve it for a specific customer’s training request, we must develop suitable techniques for monitoring attributes of QoT and related costs, especially based on the contribution of provided data in FMLM. To date, there is a lack of studies and frameworks to achieve the above-mentioned techniques. Most existing works focus on deployment and training with experimental data (such as model accuracy and training time) but lack details about real-time training quality based on the contribution of input data, detailed cost, and performance. Data marketplaces and their pricing models [9], on the other hand, focus on data assets/products only. Furthermore, any FMLM would need to capture the quality of the input data provided by data providers for QoT. Although different pricing models have been discussed for ML pipelines [10], to the best knowledge, there exists no framework for services in FMLM and its cost monitoring. This paper makes the following contributions:

- A high-level conceptual architecture of FMLM and its interactions, and characterization of cost components, and detailed cost evaluation processes for FMLM.
- A concrete design and implementation of a real-time cost computation and evaluation of QoT based on the contribution of training data for an edge FMLM.

We present experiments with realistic scenarios to demonstrate the benefit of our solutions.
The rest of the paper is organized as follows. Section II discusses the background and related work. We conceptualize QoT and cost components for FMLM in Section III. Section IV presents a concrete edge FMLM. Experiments are given in Section V. We conclude the paper in Section VI.

II. BACKGROUND AND RELATED WORK

A. Background

**Federated ML:** Federated ML [11] (aka collaborative learning) enables the training of ML models in a distributed manner across multiple training sites holding different datasets. Each training site first trains a local model on its respective dataset. Then a combination approach is used to aggregate the local models to produce a global model. In the view of marketplaces, the data belongs to different providers, each having an appropriate secure computing infrastructure to process data and train ML models without moving data to other places.

**Marketplaces:** Existing marketplaces offer different types of digital assets. Data marketplace [6], [7] and the software marketplace, e.g., Apple App Store and Google Play, are designed for selling data and apps. Unlike software marketplace where seller/buyer just uploads/downloads software applications with a straightforward pricing model, the perspective of data marketplace is more challenging. It is not the only place to upload/download data, but also provide data engineering services [7]. The cost of data is therefore strongly impacted by the data quality and the relevancy of the data to the business context at the time of market transactions. Service marketplaces [4] typically sell software or platforms as a service for storage, computing, or processing data. The marketplace proposed in our work is offering services (i.e., data validating/processing and ML model training) but incorporating selling data assets for the ML training process in the context of federated learning where the privacy of data is a key concern. Recently, pricing models for marketplaces for selling ML models have been discussed [12]. Essentially, it is possible to connect training marketplaces to the ML model marketplace. However, we have not seen such work yet.

**Real-time cost evaluation:** In pay-per-use models, especially in cloud computing with SaaS, PaaS, and IaaS, the cost of using resources and services is calculated in nearly real-time. Depending on the type of services the key metrics could be the number of service invocations, the amount of resource usage, or the volume of data. Such metrics are established also based on billing cycles for certain types of resources, such as CPU usage per minute. In terms of data, it is quite common to price data products based on quantity and quality metrics [9], [13].

B. Related work

**Marketplace requirement and quality of training:** Kumar et al. [14] analyze the difference between AI/ML marketplaces and other conventional online marketplaces, e.g., data marketplace and mobile application marketplace. This work analyzes the expectation of the AI/ML marketplace on technical, economic, and regulatory aspects as well as presents the status of current industrial marketplaces. This work does not focus on the cost of ML models, especially in the context of FL. Cong et al. [10] analyze data cost elements of ML pipelines. They have reviewed the cost models for different ML tasks, e.g., cost of raw data in data collection, cost of data labeling, cost of the individual dataset for FL, and cost of selling ML models on marketplaces. Our work differs from the above works by defining the cost elements based on the concept of QoT, which includes quality of data, the performance of the trained model, training time and resource usage for the training. Furthermore, we focus on building key services for FMLM. A recent work [15] proposed a theoretical model for pricing federated ML models in a broker-centric marketplace. We focus on system design for QoT and cost monitoring.

**Federated ML monitoring and optimization:** Several works have discussed different aspects of data and the performance monitoring of federated ML [16], [17]. In our work, we leverage existing federated ML to demonstrate our support for FMLM. Observability tools for ML, like MLTRACE [18], do not focus on the quality of training but on the basic monitoring features. ML experimental tools provide detailed information on epoch progress and ML model accuracy as well as detailed traces [19]. Research and industrial works in AI/ML as a service [2], [3] mainly provide features for deploying and training ML models and services. These platforms provide monitoring capabilities for the user to manage their training/services only from the perspective of resource consumption. In general, for capturing data for QoT and cost awareness, we leverage similar techniques for training monitoring. However, our framework is designed with different cost components and supports real-time QoT based on the contribution of data and other factors. These include the quantity and quality of data offered by data providers, the performance of the trained model and the amount of resources used for training. Additional alerting mechanisms and automatic training scheduling are also developed in our framework.

III. FEDERATED ML MARKETPLACES AND COST COMPONENTS

A. Stakeholders and interactions

To provide incentives that motivate participation in the marketplace, cost and quality models and evaluation would be needed such that ML model consumers are charged for the training service and the trained ML model while data providers receive a reward for their contribution to the training process,
Data provider (DP): DPs offer various datasets that can be used for training ML models to get rewards (e.g., money). In a marketplace, each dataset must have clear structures and metadata characterizing the data, e.g., number of samples, list of features, and collection timestamps. While the actual data is stored in the storage service under the control of the DPs (e.g., on-premise storage servers or dedicated private clouds to protect the data privacy) that can be retrieved on requests, the metadata must be made available to marketplace consumers for possible usage (e.g., searching and querying) and thus could be stored in the FMLM. Note that current data access and movement techniques simplify a lot of the access to the required data on-demand by using pre-configured connectors between DP’s data services and the FMLM marketplace. In the marketplace, Data Market Management knows metadata about available data and manages connectors to DP’s data services as well as specific and generic data quality evaluation components (Data Evaluation). Thus, it allows Data Processing pipelines to perform suitable data queries and extraction. Technically, Data Market Management can rely on existing data resource tools, feature store and existing metadata (e.g., Linkedin DataHub, Google Data Catalog, and Apache Atlas). The design of Data Market Management is out of the scope of this paper.

Market Customers (MC): MCs need to train their ML models using data offered by DPs. Generally, MCs may only need additional data for training their models to achieve higher model performance. The cost of data usage is based on (i) the amount of data requested and (ii) the quality of the data, which can be defined using various metrics. In the marketplace, to extract data features, perform data processing and train the models, MCs will use the Training Service, which relies on various services provided by other service providers. For example, to prepare the data for the training, Training Service will rely on Data Market Management and Data Processing pipelines to connect to DP’s data services. Within these pipelines, suitable components are integrated to measure the quality and quantity of data used, based on existing data quality components and metadata. It is worth mentioning that while MCs need data, the interactions between MCs and DPs are handled by the marketplace through training request serving. MCs can start with a pre-trained model (e.g., an initialized model or a model that has been trained on the dataset owned by the MCs). MCs then request the training service given specific training parameters such as selected datasets, number of training epochs, etc. The outcome of the training service is a trained model.

Service Provider (SP): SPs offer required services for the end-to-end federated training process. Examples of services include data processing services, training services (e.g., preparing execution environment), computing services (offering computing resources for training models), and deployment services after the models have been trained. Many of such services may be provided by other parties such as computing services and data services (within Computing Infrastructures) based on public cloud providers, while there exist services for ML Deployment and execution, such as based on Kubeflow and Kubernetes. These services are hidden from the MCs as Training Service has the flexibility and the right on using trusted storage and computing infrastructures and deployment services for training. Each service may have a different cost and billing cycle. SPs also provide monitoring and billing services to capture service usage. In the marketplace, the quality of training and costs will be centered on the training. Thus, they are developed for Training Service and for which, the costs from relevant, underlying services will be aggregated by the Training Service and the marketplace such that MCs do not need to interact with those underlying services.

From the view of the marketplace, the Quality of Training (QoT) is defined based on a set of factors, including the quality of trained models (model performance, model metrics), the training time, and monetary costs. The monetary costs are based on various components in which the contribution of the data plays a crucial role. Enabling awareness of QoT and costs means that we must provide detailed factors in QoT and costs at runtime during the execution of the training. On one hand, this allows MCs to have a detailed, transparent, and explainable cost breakdown of the training service with respect to various components. On the other hand, this also allows DPs to receive a reward (e.g., money) for their contribution (how the DPs receive rewards is out of the scope of this work). Even in the case where the trained model does not perform well, DPs can still be aware of their contributions to the data and computing resources used for training.

B. Data usage cost

To train a model, MCs need to request a joined dataset that combines multiple chunks of data requested from different DPs, whereas DPs and the marketplace operator must be aware of detailed costs associated with individual contributions of DPs. Given the metadata of datasets provided by DPs and made available via Data Market Management, an MC can select such datasets and required features. Based on that, chunks are extracted and built up for the MC’s training dataset. While datasets offered by DPs could be heterogeneous in terms of features (i.e., each dataset could have a different number of features), the feature selection process ensures the required training dataset that combines data chunks from different DPs has a consistent feature vector.

1) Cost for data quantity and quality: The data usage cost is calculated based on two components: data quantity and data quality. Data quantity refers to the number of data samples and number of features requested while data quality refers to the...
quality of a dataset for training an ML model. Conceptually, we distinguish two situations:

- Dataset as a whole: Cost factors are for the whole dataset. DPs might calculate the quality of data (QoD) for the whole dataset and set weighted factors regardless of the quality associated with the data used.
- Chunk-based quantity and quality: Cost factors are determined for chunks, the actual amount of data used. While it is easy to determine the number of chunks, it is challenging and costly to determine the quality of a chunk on-demand and DPs might not support this.

There exist many tools for determining data quantity and QoD and we will rely on them for cost evaluation. Furthermore, QoD metrics are not the same for all datasets. There are many metrics that have been discussed extensively in the literature, such as Class overlap, Feature Relevance, Timeliness, Completeness, and Duplicate explained in detail in [20], which can be used to estimate QoD for the data usage cost. Technically, the implementation of data evaluation can be diverse [21].

2) Cost of data based on market contexts: As multiple DPs could provide datasets used for the same ML model, the marketplace becomes a comparative environment in which DPs build their reputation by offering high-quality datasets. We can determine the reputation of a DP or a dataset using diverse information obtained over time from the marketplace. Such information includes but is not limited to:

- The number of times that a dataset has been requested by MCs, as the higher the number of usages may indicate a higher reputation of the dataset.
- The feedback from MCs (i.e., rating and/or comment), can provide more objective opinions about a dataset from a customer’s point of view.
- The impact or suitability of datasets on the business domain, which demonstrates the relevancy of the data to the business.
- The compatibility with other datasets in the same business domain indicates how much those data chunks correlate with each other to improve the model performance.

The cost component based on these types of information can be established based on known strategies in business.

3) DP-provided cost factors: For the goal of cost/quality awareness, we assume that DPs provide cost factors and the framework computes the costs based on the amount of data and data quality. There are many papers discussing the data pricing models that a DP could offer. How and why DPs set their cost factors are not the subject of this work.

C. Training cost

The process to turn a pre-trained ML model into a trained ML model can be qualified through QoT. The key aspect of our FMLM is that the main elements of pricing – rewards and costs – are determined based on the QoT. In this paper, the focus is to break down detailed QoS factors. A trained ML model is an intermediate or final result of a training process that starts from a pre-trained ML model. The quality of the trained ML model is defined by a set of metrics measuring the performance of the model. Depending on the type of the ML model, the performance of ML models can be evaluated using well-known metrics in ML, e.g., Accuracy, F1-Score, Precision, Recall, and Mean Square Error (MSE). FMLM should allow training service providers to select the metrics to evaluate the model performance and QoT. Given a training task, QoT is defined as the improvement of the model performance between the pre-trained model and the trained model obtained after the training task is done. QoT also takes into account other parameters such as training time and the number of training epochs. Obviously, a dataset that helps the model to converge quickly and achieve high performance is preferable. Taking a longer time to converge could lead to higher computing resource costs used for running the training task. However, evaluating the impact of QoD (discussed in Section III-B1) on the improvement of the model before actual training is a complex optimization problem. We assume that the impact of the quality of a specific data chunk on the performance of the model is captured through the trained model model performance improvement, being represented within QoT. Thus, the marketplace charges the MCs based on the QoT.

D. Cost of (third-party) infrastructural resources

When MCs submit a federated training request to train a pre-trained model on multiple data chunks requested from DPs, multiple training tasks will be constructed and submitted to distributed sites that interact with respective DPs. DPs then prepare the data chunks used for training the pre-trained model and also prepare the training environment, which will be executed in a computing platform based on the preference of the DPs. Regardless of whether the training is successful or not (in the view of MCs based on the performance improvement of the pre-trained model presented in the previous section), an amount of computing and network resources have been used for the training, thus incurring additional cost for the computation and network resources. Similar to public clouds, FMLM can charge MCs for computing and network resources using customized billing cycles.

E. Conceptualizing QoS and cost evaluation processes

QoT and cost evaluation are associated with the execution of federated ML. A federated ML pipeline consists of multiple processes involving multiple stakeholders. Thus, composable cost evaluation models will be used. As shown in Fig. 2, we divide a federated ML pipeline into multiple phases including (i) a data selection phase, (ii) a data processing phase, (iii) a model training and optimization phase, and (iv) deployment phase. In each phase, various services are offered by different stakeholders and they are linked with respective cost computation processes depicted in the bottom half of the figure.

In the data selection phase, FMLM provides a searching service that allows MCs to specify a feature list and other searching criteria such as timestamps and the number of samples needed before performing the search on the metadata
Costs and quality evaluation based on model performance

Plugin QoD Evaluation

Data Selection
{Required Features}
{Metadata}

Data Processing
{Qualified Data}

Training
{Trained Model}

Deployment
{ML Service}

Cost evaluation based on market context, QoD, and data quantity

The output of the searching service is the metadata of the data chunks that satisfy the requirement of MCs. The data selection phase also includes an optional service (specified as a plugin depicted in Fig. 2) namely quality of data (QoD) evaluation. If DPs opt-in to use this service to evaluate the quality of data offered by the DPs, evaluation requests will be initialized and sent to DPs for execution. If this service is invoked, MCs will be charged for its usage. The computation of this cost is mostly based on the cost of computing and network resources required for running the service, and an additional upfront cost for the FMLM for the effort of wrapping requests and interacting with DPs.

The output of the data selection phase is the input of the data processing phase. Many services are involved in the data processing phase allowing MCs to explore the data and improve the quality of data before feeding into the training phase. Examples of such services include but are not limited to feature normalization, feature/data exploration, and deletion of value-missed samples. When MCs raise a request for data processing given the metadata of the data chunks to be processed, FMLM instantiates multiple data processing requests and submits them to respective DPs for execution. Based on the metadata provided in the request, DPs clone the data chunks and perform data manipulation. We refer to this data manipulation as a task, which requires computing, network, and storage resources. The cost incurred in the data processing phase includes the upfront cost of the service imposed by the market, and the computing, storage, and network resources incurred at involved DPs.

The metadata of the qualified (processed) data is forwarded to the training service along with a pre-trained model provided by MCs for invocation of the training service. Similar to the data processing service, FMLM instantiates multiple training requests and submits the requests to respective DPs. In our supporting distributed learning execution model, the pre-trained model is sent simultaneously to all involved DPs for certain training epochs. The trained models obtained from respective DPs will be consolidated into an updated global model, which is considered as a new intermediate trained model to be sent to DPs for another training request. We also refer to the training process run by a particular DP as a training task. When a training task is executed, MCs actually use data and thus they will be presented with detailed data usage costs. Furthermore, MCs have to be aware of the cost based on performance improvement of ML models, and also computing and network resource costs. By integrating with external ML services, the trained model can be obtained from the training service for the deployment service, which helps MCs to deploy their model to a cloud infrastructure. MCs can also download the trained model and perform deployment in their in-house infrastructure.

IV. EDGE MARKETPLACE FOR DISTRIBUTED ML TRAINING

A. Architectural overview

To realize our FMLM with a focus on QoT and cost evaluation, we are developing EADRAN (Edge marketplace...
for Distributed AI/ML training), shown in Fig. 3. While orchestration, federated server, and ML training at the edge form our training service, the cost evaluation includes measurement collection, message streaming, and cost computation in nearly real-time. In EADRAN, edge sites are deployed in different places close to the data sources where a DP provides configurations for the edges to access the data. A public centralized site is provided for coordinating ML training. The training pipelines are built with containerized functions and micro-services. The monitoring system captures measurement data and provides data for the cost and quality assessment service, which implements cost computation algorithms. All measurements are captured into messages and sent to a messaging system. For cost/QoT computation, a centralized service receives messages and applies streaming processing to evaluate the cost/QoT and provides the cost/QoT information on-the-fly during the training process. All data related to a single ML training will be correlated through a unique identifier assigned by the orchestrator. We also use isolated resources for the ML training and determine the resource costs based on resource usage.

We instrument training code using QoA4ML probe library\(^5\) to capture metrics of machines (i.e., CPU and RAM usage) and the ML training state (i.e., the performance metrics, training loss value, and training step). We use Redis\(^6\) as a messaging system for communication between the centralized site and the edge sites w.r.t. orchestration requests. We use a combination of RabbitMQ and Apache Kafka as the messaging system to handle the monitoring measurements sent from edge sites to the centralized site for cost computation. We implement a streaming process of cost computation using Spark Streaming Framework. InfluxDB and Grafana are used for storing and visualizing detailed cost components.

B. Establishing cost models

In this section, we present the concrete cost models implemented in EADRAN. Each cost model reflects a cost component presented in Section III. We note that this is a specific setting for edge computing, established based on business scenarios in resource constraint environments and the developing world.

1) Cost model for data quantity and quality: Given \(N\) DPs, each has a dataset that satisfies the requirements of a particular MC. DP \(i\) can provide \(M_i\) data chunks whose size is denoted as \(S_{i,j}\). DP \(i\) specifies the unit cost of data for each chunk, denoted as \(u_{i,j}\). We assume that there are \(K\) third-party evaluators providing tools/services for data quality measurement. DP \(i\) has a weighted value representing the reputation of evaluator \(k\), denoted as \(w_{i,k}\). For data chunk \(j\) provided by DP \(i\), evaluator \(k\) generates a value denoted as \(Q_{i,j,k}\) representing the quality of the data chunk. The total cost due to the quality of data denoted as \(\text{Cost}^{\text{qod}}\), incurred due to the model training request, is formulated as follows.

\[
\text{Cost}^{\text{qod}} = \sum_{i=1}^{N} \sum_{j=1}^{M_i} u_{i,j} + \sum_{k=1}^{K} w_{i,k} \star Q_{i,j,k}.
\] (1)

2) Cost model based on market context: As discussed in the previous section on the conceptualized marketplace, we consider two cost components:

- Manual setting cost component – \(\text{Cost}^{\text{ctx-man}}\), which is updated by DPs: this is based on the DP belief and business model, e.g., the DP can rely on the rating of customers or number of usages. Each DP has its own unit cost of market context-based cost relative to other DPs, denoted as \(u_{i}^{\text{man}}\). We also assume that there are \(K\) context criteria, each being defined by the DP and denoted as \(C_{i,k}^{\text{man}}\). At the beginning, all DPs may start with the same values of \(C_{i,k}^{\text{man}}, \forall i, k\). During the operation of the marketplace, more information will be collected and DPs can update their unit cost, \(u_{i}^{\text{man}}\), and \(C_{i,k}^{\text{man}}\), accordingly. The manual setting cost based on market context is computed as follows:

\[
\text{Cost}^{\text{ctx-man}} = \sum_{i=1}^{N} \sum_{k=1}^{K} u_{i}^{\text{man}} \star C_{i,k}^{\text{man}}.
\] (2)

\(^5\)QoA4ML: https://pypi.org/project/qoa4ml/

\(^6\)https://redis.io/
• For the compatibility degree with other datasets, we define this cost component as $Cost^{ctx}_{compt}$. Each DP specifies its unit cost of compatibility, denoted as $u_{i,j}^{compt}$. The compatibility of the dataset of DP $i$ to the dataset of DP $j$ is defined as $C_{i,j}^{compt}$. The compatibility cost component is computed as follows:

$$Cost^{ctx}_{compt} = \sum_{i=1}^{N} \sum_{j=1}^{N} u_{i,j}^{compt} * C_{i,j}^{compt}.$$  (3)

The total cost presented to an MC based on the market context is: $Cost^{ctx} = Cost^{ctx}_{man} + Cost^{ctx}_{compt}$.

3) Training cost model: Let $K$ be the number of performance metrics used to evaluate the performance improvement of an ML model achieved by a training task based on multiple epochs. Given the training at epoch $t$, we identify $P^{pre}_{i,k,t}$ as the performance in terms of metric $k$ of the pre-trained model tested on the test set of DP $i$ at epoch $t$. If the MCs would like to compute the performance of the pre-trained model on their private test set, $P^{pre}_{i,k,t}$ will have the same value for all the DPs. We also define $P^{post}_{i,k,t}$ as the performance in terms of metric $k$ of the trained model after the training of epoch $t$ is completed. For DP $i$, we define $u_{i,k}^{post}$ as the unit cost for the QoT based on performance metric $k$. The training cost is defined as follows:

$$Cost^{QoT}_{i} = \sum_{i=1}^{N} \sum_{k=1}^{K} u_{i,k}^{QoT} * P^{pre}_{i,k,t} * \max(0, P^{post}_{i,k,t} - P^{pre}_{i,k,t}).$$  (4)

The $\max$ function is based on the assumption that the marketplace does not charge if the performance metric after training would be lower than before. However, the cost of computing and network resources used for training still incurs. The total training cost after $T$ epochs is then defined as $Cost^{QoT} = \sum_{i=1}^{N} Cost^{QoT}_{i}$.

4) Cost model for computing resources: We adopt cost models of cloud platforms to calculate the cost of computing resources. We consider that there are four types of resources that the training of ML models needs, including CPU, GPU, RAM, and DISK (storage space). For training epoch $t$, the amount of resource $j$ used by DP $i$ is defined as $U_{i,j,t}$. Let $d_{i,t}$ be the time duration of training epoch $t$ on the computing infrastructure used by DP $i$. We let DP $i$ define its unit cost of resource $j$, denoted as $u_{i,j}^{res}$. The total computing resource cost for training epoch $t$ is defined as follows:

$$Cost^{res}_{i,t} = \sum_{i=1}^{N} d_{i,t} * \sum_{j \in \{CPU, GPU, RAM, DISK\}} u_{i,j}^{res} * U_{i,j,t}.$$  (5)

We note that for a training epoch, the amount of resources used by the training task may vary over time due to the load. During training, the monitoring system reports those values in time intervals. DPs can compute the amount of resources used by a training task (i.e., $U_{i,j,t}$) by using the average or max of the reported values before feeding into Eq. (5). We take the average of those reported values (in our experiments we observed that the changes in cost were within 5% when we used the max value).

C. Monitoring and evaluating real-time quality and costs

Since the edge networks are protected with numerous policies and configurations, the communication among services needs to be optimized to minimize unnecessary authentication, detect failures (connection loss, software crashes), and self-recover. We adopt messaging protocols (e.g., AMQP, Redis) due to their lightweight and compatibility with various edge infrastructures. We implement an edge orchestration service that could be deployed by DPs in their edge infrastructures and this service is configured to receive orchestration requests from the pubic centralized site via Redis message queues to invoke data processing and training tasks. For each ML model, we reserve an AMQP message queue for monitoring measurements from multiple edge training tasks, each being executed by an edge site. At the public centralized site, we implement a Kafka - Spark server to handle monitoring measurements and provide real-time analysis at a large scale as the number of models and clients increases.

Besides, ML applications can be developed based on various libraries and tools that are constantly being updated, leading to conflicts and failures in deployment. While many ML frameworks offer pre-defined models or support building customized models, they implement different training/predicting functions and operating mechanisms. This makes the automation of the training process on the heterogeneous edge notably challenging. To this end, we support a training program template where we adopt the dynamic loading mechanism to allow ML developers to import their models into the training program template by declaring several specific modules and functions. The program template integrates several probes to monitor the training process and system resource usage at configurable intervals, covering system healthiness. To prevent conflicts, the model’s dependencies/tools/libraries need to be reported and constantly updated with appropriate versions. Thereby, we can containerize its runtime environment for different platforms/hardware architectures at the edge.

Listing 1: Example of a measurement reported

Listing 1 shows an example of monitoring data collected from different edge sites in the JSON structure. These messages are combined with the static information stored in the database such as unit costs, QoD, and machine profile to
compute various cost components as well as the total cost. These costs are saved back to the database and put in the message queue for other functionalities of the marketplace such as cost computation shown in Fig. 4.

V. EXPERIMENTS

A. Experimental Infrastructure

We deployed a distributed ML edge testbed for experiments that includes 5 edge sites: 1 in an Asian university, 2 in a European university, 1 in a national cloud data center in Europe, and 1 in Amazon EC2. Overall, the computing resource at each site is fixed but can be elastic with the edge sites being emulated in clouds. Edge sites in clouds are created with minimum resource capabilities to demonstrate edge nodes. We deploy FMLM services by using 2 other computing instances: 1 Amazon EC2 instance for the messaging system (i.e., Kafka) and the federated server, and 1 local machine for running the orchestrator and cost computation (i.e., Spark). We also use cloud services (RabbitMQ, Redis) from CloudAMQP and Redislab, as well as InfluxDB and Grafana clouds.

B. Experiment setting - Fraud Detection

1) Dataset preparation: We use a dataset with ≈1.3 million samples from Kaggle\(^3\). We split this dataset into 10 different smaller datasets to emulate 10 data providers that locate in three regions. We define the regions by using geometry information (i.e., longitude and latitude) provided in the dataset and split them into 3 regions as follows: i) the western region (named $D_{\text{west}}$) with the longitude $< -100$; ii) the northeast region (named $D_{\text{northeast}}$) with the longitude $\geq -100$ and the latitude $< 40$; iii) the southeast region (named $D_{\text{southeast}}$) with the longitude $\geq -100$ and the latitude $\leq 40$. The 10 datasets are produced as follows:

- Four datasets get the majority of their samples (i.e., a random number is greater than 80%) from $D_{\text{west}}$ and the rest from $D_{\text{southeast}}$ and $D_{\text{northeast}}$.
- Three datasets get the majority of their samples (i.e., a random number is greater than 80%) from $D_{\text{southeast}}$ and the rest from $D_{\text{west}}$ and $D_{\text{northeast}}$.
- Three datasets get the majority of their samples (i.e., a random number is greater than 80%) from $D_{\text{northeast}}$ and the rest from $D_{\text{west}}$ and $D_{\text{southeast}}$.

After dividing the original dataset into 10 smaller datasets, we use the code available with the dataset on Kaggle to explore and pre-process separately. We observe that these datasets are very imbalanced. Therefore, we process these datasets in two steps: i) randomly drop 50% of samples of the majority class; ii) use SMOTE technique [22] to augment the minority class to $X\%$ of the majority class where $X$ is a random number in the range [50, 100]. We implement a neural network pre-trained model using Tensorflow to classify the fraud messages. The Flower Framework [23] and a popular federated algorithm, named FedAvg [24], are used for our experiments.

2) Metadata of data sources: To emulate the QoD metrics for the datasets, we select a subset of metrics proposed by Patel et al. [20] and calculate the selected QoD metrics:

- Class Parity: $\text{class\_parity} = \frac{\text{count}(D_{\text{is\_fraud}}=1)}{\text{count}(D_{\text{is\_fraud}}=0)}$ where $D$ is a dataset and $\text{is\_fraud}$ is a feature.
- Feature Correlation (among features): is calculated using Pearson correlation coefficient for $N$ features of dataset $D$ as follows: $\text{feature\_correlation} = 1 - \frac{N}{\sum_{i=1}^{N} \sum_{j=1}^{N} \text{correlation(column}_i, \text{column}_j)}$.
- Feature Relevance (to label): given $fi$ as the feature $i$ and the rest from

\[^3\text{https://www.kaggle.com/code/vibhorgoyal0301/credit-card-fraud-logistic-regression}\]

\[^4\text{https://scikit-learn.org/stable/modules/tree.html}\]

\[^5\text{https://cloud.google.com/compute/all-pricing#general}\]
of QoD, context, performance improvement, and computing resource was set at the same value for all data providers.

1) Experiments with the normal working conditions of training sites: We trained the model with 10 training epochs. At each training epoch, the federated server asks all the available edges (i.e., the training process at the data provider side) to train the model on their own dataset. Fig. 5 demonstrates how detailed costs could be provided to MCs, DPs, and the FMLM operator in real time. MCs can monitor not only the total cost (Fig. 5a) but also the cost by performance improvement of model (Fig. 5b), the cost of computing resource (Fig. 5d) and the contribution of individual DPs (Fig. 5c). Two static cost components (i.e., cost based on QoD and the market context) are included in the total cost and reported in Table I. Given this real-time monitoring, MCs can take actions to optimize the training or reduce the cost. For instance, looking at Fig. 5c, MCs can stop the training at the time 08:38:05 for most of the edges to avoid the further increment of the resource usage cost since the performance of the model is no longer improved after that time. Without doing so, the cost model performance improvement and computing resources keep increasing as shown in Fig. 5b and Fig. 5d. Given this QoT metric (including QoD, cost elements, quality of the model, and training time), the FMLM provider can develop an algorithm to select data providers during training.

2) Experiments with failures at training sites: A common problem in edge computing is that training sites (managed by DPs) can join or leave the platform over time (due to system and network errors). This may lead to different performance behaviors. From the cost/quality awareness perspective, MCs want to know how the performance of the model is affected in parallel by the changing cost of quality of the model and resource usage if a site stops the training. In this experiment, we randomly disconnect some training sites from the platform during training. Fig. 6a shows that the performance of some data providers was very unstable. In this situation, the cost by the model performance improvement does not increase as shown in Fig. 6b. When a training site is disconnected from the platform, the cost incurred by that site is no longer updated as shown in Fig. 6b. For instance, when ds002 fails at 08:30:52), MCs recognize that the performance of ds010 is affected (Fig. 6a at the time 08:31:10). Thus, if needed, the failed training sites can be restarted or new training sites can be provisioned by using the orchestrator.

VI. CONCLUSIONS AND FUTURE WORK

The success of market-based federated/distributed ML needs novel, foundational mechanisms to assess and report the cost/quality of training and corresponding contributions from each distributed data source and training performance contribution to the final model in real time. It not only gives a detailed insight into the transparency and explainability of the quality of training and costs, but also enables a flexible way to start/stop, and select a data source or an infrastructural service while training ML models. This paper has analyzed and defined the cost components of distributed data sources while training ML models in the context of the ML marketplace. In our specific prototype of EADRAN – an FMLM for edge computing – we have demonstrated four generated cost components at the training step including i) cost based on the
TABLE I: Cost setting and static costs with QoD Metrics= (class_parity, feature_correlation, completeness, feature_relevance)

<table>
<thead>
<tr>
<th>Data Providers</th>
<th>QoD Metrics (1)</th>
<th>Context Metrics (2)</th>
<th>Unit Cost -QoD (3)</th>
<th>Unit Cost -context (4)</th>
<th>Data samples (5)</th>
<th>Cost based on QoD</th>
<th>Cost based on context</th>
</tr>
</thead>
<tbody>
<tr>
<td>ds001</td>
<td>(0.919, 0.945, 1.0, 0.616)</td>
<td>(0.5, 0.5)</td>
<td>0.5</td>
<td>10</td>
<td>75206</td>
<td>130.94</td>
<td>2.45</td>
</tr>
<tr>
<td>ds002</td>
<td>(0.949, 0.939, 1.0, 0.615)</td>
<td>(0.5, 0.5)</td>
<td>0.5</td>
<td>10</td>
<td>57224</td>
<td>100.28</td>
<td>1.86</td>
</tr>
<tr>
<td>ds003</td>
<td>(0.739, 0.940, 1.0, 0.618)</td>
<td>(0.5, 0.5)</td>
<td>0.5</td>
<td>10</td>
<td>62939</td>
<td>103.81</td>
<td>2.05</td>
</tr>
<tr>
<td>ds004</td>
<td>(0.759, 0.941, 1.0, 0.619)</td>
<td>(0.5, 0.5)</td>
<td>0.5</td>
<td>10</td>
<td>54776</td>
<td>90.95</td>
<td>1.78</td>
</tr>
<tr>
<td>ds005</td>
<td>(0.660, 0.938, 1.0, 0.617)</td>
<td>(0.5, 0.5)</td>
<td>0.5</td>
<td>10</td>
<td>53817</td>
<td>86.54</td>
<td>1.75</td>
</tr>
<tr>
<td>ds006</td>
<td>(0.579, 0.949, 1.0, 0.619)</td>
<td>(0.5, 0.5)</td>
<td>0.5</td>
<td>10</td>
<td>307498</td>
<td>484.17</td>
<td>10.00</td>
</tr>
<tr>
<td>ds007</td>
<td>(0.690, 0.943, 1.0, 0.620)</td>
<td>(0.5, 0.5)</td>
<td>0.5</td>
<td>10</td>
<td>79918</td>
<td>130.01</td>
<td>2.60</td>
</tr>
<tr>
<td>ds008</td>
<td>(0.590, 0.949, 1.0, 0.619)</td>
<td>(0.5, 0.5)</td>
<td>0.5</td>
<td>10</td>
<td>166451</td>
<td>267.60</td>
<td>5.51</td>
</tr>
<tr>
<td>ds009</td>
<td>(0.880, 0.941, 1.0, 0.621)</td>
<td>(0.5, 0.5)</td>
<td>0.5</td>
<td>10</td>
<td>112294</td>
<td>193.11</td>
<td>3.65</td>
</tr>
<tr>
<td>ds010</td>
<td>(0.930, 0.944, 1.0, 0.620)</td>
<td>(0.5, 0.5)</td>
<td>0.5</td>
<td>10</td>
<td>131600</td>
<td>229.95</td>
<td>4.28</td>
</tr>
</tbody>
</table>

Acknowledgements: The authors wish to acknowledge CSC IT Center for Science, Finland, for cloud resources. The work is also supported by Tan Tao University Foundation for Science and Technology Development under Grant No. TTU.RS.22.102.001.

REFERENCES
