Query-driven Edge Node Selection in Distributed Learning Environments

Tahani Aladwani*, Christos Anagnostopoulos†, Kostas Kolomvatsos‡, Ibrahim Alghamdi§, Fani Deligianni¶,
* † ¶ School of Computing Science, University of Glasgow, UK
‡ Dept. of Informatics and Telecommunications, University of Thessaly, GR
§ Faculty of Science and Arts in Baljurashi, Dept. of Computer Science, Al-Baha University, SA
*tahani.aladwani@glasgow.ac.uk, †christos.anagnostopoulos@glasgow.ac.uk, ‡kostasks@uth.gr,
§ia.alghamdi@bu.edu.sa, ¶fani.deligianni@glasgow.ac.uk,

***Abstract*—Computing nodes in Edge Computing environments share unlimited data. Such data are exploited to locally build Machine Learning (ML) models for applications such as predictive analytics, exploratory analysis, and smart applications. This edge node-centric local learning reduces the need for data transfer and centralization, which is affected by different factors such as data privacy, data size, communication overhead, and computing resource limitations. Therefore, a collaborative learning fashion at the network edge has appeared as a promising paradigm that enables multiple distributed (edge) nodes to train and deploy ML models cooperatively without infringement of data privacy. Nevertheless, the variety, distribution and quality of data vary between edge nodes. Hence, selecting unsuitable edge nodes can have a negative impact on the ML model performances. We have devised (i) an intelligent node selection mechanism per analytics query based on the range of the availability of required training data at the edge and (ii) variants of collaborative learning processes engaging the most suitable nodes for models training and inference. We evaluate the efficiency of our selection mechanism and collaborative learning and provide a comparative assessment with other methods found in the literature using real data. The results showcase that our mechanism significantly outperforms baseline approaches and existing node selection mechanisms in distributed computing environments.**

***Index Terms*—Query-driven node selection, Distributed learning, Data overlapping.**

I. Introduction

Machine Learning (ML) and Deep Learning (DL) models are typically trained using centralized data. This means that all necessary training data should be available on a central data server. However, bringing all of these data into a centralized server is no longer practical due to concerns about data privacy and data volume. On the one hand, in several application domains such as healthcare sector systems (e.g., medicine records/data in hospitals, electronic health record (EHR), and disease-specific or products-specific registries) and banking sectors (e.g., customers information/data), data privacy is regarded as a top priority. These data are not shareable because of ethical, legal, logistical, and administrative barriers [1]. On the other hand, computing resource constraints exist. In particular, when dealing with a large-scale model, we require a huge amount of data to train this model efficiently, and in many scenarios, the organization might lack the computing power to process this data locally [2]. Meanwhile, the quality of ML models depends heavily on the training data volume, quality, and availability [3]. For example, if the data is *less* than what the model requires, we face the challenge of the model's limited generalization. This implies that the model will perform well when it is presented with new data that are similar to the data that was trained on. In our context, the model performance will worsen when it is presented with new data from other nodes. To overcome this problem, a distributed learning framework has been proposed to facilitate access to the data by training a single ML model over disjoint data space by leveraging clients' data and computing resources. The objective is to alleviate concerns about client data privacy while reducing the load on centralized devices and communication bandwidth [4]. In distributed learning context, to make an ML model learn efficiently and become more generalized, we must train it over a set of distributed edge nodes, hereinafter referred to as *participants*. However, in such a setting, not all participants play the same role as evidenced in [5]. This is determined by the amount of data in each participant, the quality of the data, and the rate of data overlap between the query (required task) and what the participant already has [4]. As a result, an important challenge in distributed learning is **how to select the most important subset of participants from all participants to be engaged per issued query.**

In addition, large-scale models and distributed learning models are more likely to *forget* what they have learned from previous participants when they move to new participants with different data distributions. To reduce the impact of this problem on the quality of results and accuracy, we should pay more attention to establish a participants selection mechanism, i.e., **which participants are most appropriate to execute a collaborative learning process according to analytic task requirements?** Furthermore, ML and DL models need a specific amount of data to be trained. Less data could cause an under-fitting problem, and more data could cause an over-fitting problem. To avoid this problem, we also must ensure that the model only picks and trains on the required data. Hence, the third critical question is **how can we select only the required data from each chosen participant, especially when we have limited access to the data?** Inspired by these three challenging questions, in this paper, we propose an edge node selection mechanism for distributed learning environments per issued analytics query. The purpose of performing such kind of edge selection mechanism is to improve the global model performance and reduce the model

drift and model forgetting chances that could happen due to training the model on irrelevant data. The main goal of implementing such kind of edge selection mechanism is to improve the global model performance and reduce the model drift and model forgetting that could happen due to training the model on irrelevant data. Our technical contributions are:

- We introduce a mechanism to determine the participation of nodes per query. Nodes might have similar data patterns, thus, the model only needs to train on enough data. In this case, selecting participants at random may be faster and produce the same results as using a participant selection mechanism. In other cases, however, participants hold disjoint data spaces. We need a selection mechanism to specify which scenario we deal with.
- Our participant selection mechanism is based on data clustering for each participant. We take into account the data representatives (clusters) in each participant that satisfy the query boundaries. Then, based on a novel overlapping rate according to the clusters and amount of data needed per cluster, we contribute with a ranking method, which determines the nodes to be selected.
- Our mechanism suggests the model be trained only on clusters that satisfy the query boundaries over the selected nodes. We contribute with model averaging methods across selected nodes and compare our results with related work in [6] and [7] over real datasets.

II. Related Work & Rationale

The quality of distributed models' performance is based mainly on the quality of selected participants. However, few works have been conducted on the participants' selection mechanisms. In [6], the number of images, image quality, and the number of computation resources in each participant were the criteria to select each participant. In [7], the participant selection criteria are based on selecting participants having different data compared to what the models have learned before to make models more generalized. In [8], participant selection criteria are data quality score, computation score, and communication score to quantify the capabilities of the participant device, while [9],is based on a reward value, which is the sum of this information, battery status, amount of computation, and communication situation. In [10], they proposed Federated Trace to trace the model training and the data distribution on a random set of participants. In [11], the participant selection mechanism based on measuring each participant's contribution in the previous round, contribution here points to the accuracy of the global model before and after aggregating with this participant. Finally in [12], participant selection mechanism based on fairness of selection. Which means each participant had the the same chance to get involved during the training process. Most of the previous works do at least one model training round before applying the participant selection mechanism. This could be time and resource wasting while the server node will drop any participant's model affecting the global model performance negatively.

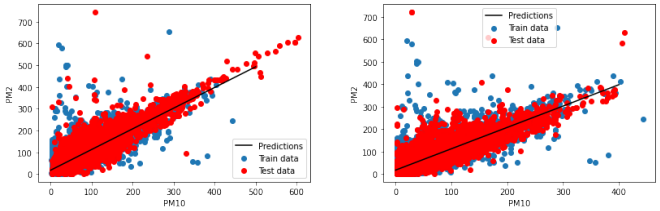
One of the main challenges that can affect the models' performance in distributed learning is the heterogeneity of participants [6]. Each participant can have different data patterns, ranges, and distributions. Also, data patterns and ranges can be similar or correlated in a set of participants, e.g., edge devices have been used to collect weather data from a specific area. In this case, any set of nodes can lead to a trained model with similar quality. Hence, we do not need to consider which participant to select. In more complicated scenarios, for example, we can find a regression between the required variables in two participants, but it is negative in one participant and positive in the other. This is clear evidence that data patterns differ between nodes. Hence, a node selection mechanism is required to determine which node could improve the model performance and which can lead to a model that can *forget* what it has learned. Unfortunately, upon any incoming analytics query [13], we do not have full access to data of all participants; thus, we cannot extract the data space and patterns. In order to extract this knowledge per incoming query, it is deemed appropriate to define a pre-test mechanism to check if participants have a similar or different data patterns and relevant to the current query or not.

Following a previous effort in [7], we devised a rather similar pre-test mechanism. The mechanism begins by initializing a *leader* participant (i.e., the node who wants to build an ML model given an incoming query) as an *independent federation*. This leader participant will build a global model based on the data that it has locally. This data could form a small part compared to what the model needs to reach the required accuracy and the requested data as specified by the incoming query. In this case, the model will seek to federate with other networked participants by engaging all the available participants. The compatibility study is based on testing the resulting model of the leader node against all the available participants. In this situation, we have two scenarios, firstly, all participants give almost similar results to the global model. This means that participants have almost similar data distribution and patterns. Hence, accordingly, selecting participants randomly will provide a baseline solution. As it can be seen in Figure 1, the participant in Fig. 1a has higher priority to be selected compared to the participant that has been selected randomly in Fig. 1b. However, taking into account the model accuracy results in Table I, we expect that both of these participants have similar data, thus, the models built proceed with similar performances, even before looking at Fig. 1 that proof this fact.

TABLE I: Expected loss (prediction error measured) from selecting a participant according to an all-participant selection mechanism and random participant selection mechanism.

Model	All-node selection	Random selection
LR	24.45	24.70

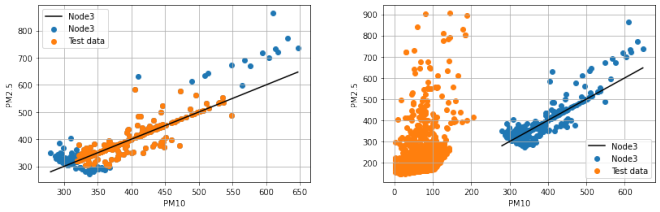
Secondly, when nodes' models result in different losses, that denotes that nodes have very different data distribution and patterns compared to the global model and against each



(a) Participant selected based on all-node selection mechanism.

(b) Participant has been selected randomly.

Fig. 1: Similar participants due to similar data patterns and distributions.



(a) Participant selected based on all-node selection mechanism.

(b) Participant has been selected randomly.

Fig. 2: Heterogeneous (dissimilar) participants due to different data patterns and distributions.

other. As evidenced in Fig. 2a, this node has almost a similar data pattern to the global model, and the expected loss in Table II proves this. In contrast, the node in Fig. 2b has very different data compared to the global model, and the loss in Table II proves this fact as well. In this case, we need to introduce a node selection mechanism to decide which node should the global model considers and which one should the global model avoids. If the model selects the first one, it will be trained on data coming from the distribution. Otherwise, the model would be trained on unrelated and different data, thus, decreasing its predicting capability. In this case, the node selection mechanism should focus more on the nodes that are expected to give a high degree of data overlapping extracted by the underlying data distribution and exclude nodes that have a low degree of data overlapping. And, this reasoning should be taken intro account for each issued analytics query. Therefore, we introduce a *query-driven node selection mechanism* elaborated in the following sections.

TABLE II: Expected loss from selecting heterogeneous nodes.

Model	All-node selection	Random selection
LR	9.70	178.10

III. System Model

A. Overview

In this work, we consider a general framework for distributed learning in an edge computing environment per analytics query. Specifically, our framework has two components:

- **Leader node:** In the distributed learning procedure, an elected leader node serves as the organizer for handling the selection of those nodes deemed appropriate for engagement given a query. It receives queries from applications (e.g., predictive analytical tasks selected over data [14]), selects the best nodes (participants) for each query, collects the trained local models from the participants, and performs model aggregation [6].
- **Participant nodes:** Edge computing nodes are forming groups of nodes that act as participants, which can be chosen to participate in distributed learning per analytics query. Each node n_i has access to its local data and sources, e.g., equipped with a set of built-in sensors used for different purposes (e.g., weather data collected, or photos), has storage capacity for the collected data, and communicate with neighboring networked participants and its assigned leader node.

The interaction process between leader and participants is happening according to the principles of federated learning in [6], [15] and [16]. However, the fundamental difference in our context is that the leader node based on the query data-ranges/boundaries determines the node ranking per query. That is, node ranking is used by the node selection mechanism which is introduced in Section III-C. According to our node selection mechanism, we seek a subset of $n_i \in \mathcal{N}$ nodes/participants that their data *satisfy* the query requirements based on query boundaries. This set will adapt to the distributed learning process as we elaborate in Section IV-B.

B. Problem Fundamentals

Distributed learning builds a model $\mathcal{M}$ on a dataset D to get a function $f = h(w, \mathcal{X})$, where w is the model's weight/parameters and $\mathcal{X}$ is the model's inputs. Fundamentally, the difference here is that D is not centralized; it is distributed among N edges nodes, denoted as $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$, where each $n_k \in \mathcal{N}$ has its own computing capacity $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$. These computing capacities can be used to train an ML model locally. In addition, each $n_k \in \mathcal{N}$ has its own local dataset D_k consisting of m training data samples. Each sample point is represented as $\xi = (x, y)$, where $x \in \mathcal{X}$ is the input to the ML model and $y \in \mathcal{Y}$ is the desired outputs (e.g., class label). Each D_k is represented as $D_k = \{\xi_1, \ldots, \xi_m\}$. This D_k represents a subset of the whole needed training dataset $\mathcal{D} = \cup\{D_k\}_{k=1}^{N}$. Additionally, we assume that all $\mathcal{N}$ edges utilize similar datasets in terms of features, e.g., weather data (humidity, temperature, and pressure) with different data varieties, distributions and patterns. As we mentioned, each edge $n_k \in \mathcal{N}$ has different data amounts and diversity, i.e., different *data space*. These data could be used to train a distributed model $\mathcal{M}$ with more data variety without transmitting them over the network. Training a model $\mathcal{M}$ on one edge node n_k will produce a local/weak model f. Therefore, a model $\mathcal{M}$ needs to pass through a set of edge nodes $n_k \in \mathcal{N}$ to become a *global* model. However, some edges n_k have a low amount of the required data or different distribution compared to what the model $\mathcal{M}$ needs

based on the incoming query. As a result, involving these edge nodes in the training process cannot significantly improve the model $\mathcal{M}$, and may potentially have negative effects. Therefore, selecting the *right* participating edge nodes n_k with the required range is an important factor in our distributed learning framework as follows in the remainder.

C. Edge Nodes Selection Mechanism

In the considered setting, let us assume a set of analytics queries $\mathcal{Q} = \{q_1, q_2, \ldots, q_M\}$, each query in $\mathcal{Q}$ represents an analytic task that needs a specific amount of d dimensional data to be executed. These required data by query q_k are not centralized, i.e., they are distributed across multiple edge nodes. Unfortunately, due to data privacy concerns, nodes are unwilling to share their data with others or a central server, e.g., Cloud. Meanwhile, all nodes in $\mathcal{N}$ are willing to collaborate and benefit from the other nodes' data variety. Also, the nodes $\mathcal{N}$ are heterogeneous in terms of data distributions and spaces. Selecting the not appropriate nodes given a query, however, could degrade the effectiveness of distributed learning process. In order to define the most suitable nodes for each query, we explore the data overlapping ranking between each node's data and query. The data overlapping indicates an estimation of the percentage of data (out of the whole dataset $\mathcal{D}_i$) required for executing a query q_k. We notate the data overlapping between node n_i and query q_k by h_{ik}. Specifically, to quantify h_{ik}, each node has quantized its own data space $\mathcal{D}_i$, e.g., using the k-means algorithm [17], into K clusters. In order to obtain the set of K clusters $\mathcal{K}$, the node minimizes the following quantization loss over the local samples:

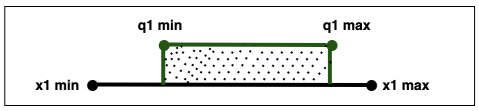
$$\min_{\{u_1,\ldots,u_K\}} \sum_{k=1}^{K} \sum_{j=1}^{m} \|\xi_j - u_k\|^2. \tag{1}$$

Each cluster representative is represented by the d-dimensional vector u_k and m is the number of samples in the local dataset. The main goal of this step is to find the number of clusters (out of K) that have a high overlapping degree with a given query q. The node has almost the same pattern of the required data by the query, which it is expected to increase the chance of the node being selected as a participant in the learning process.

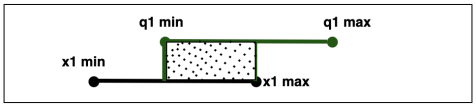
In the following, we exploit the boundaries of each cluster across all the data dimensions (i.e., taking the minimum and maximum values for each dimension per cluster) and the cluster representatives themselves. This will be included in calculating the overlapping rate between each clusters and the current query. Specifically, for each cluster, we find the maximum and the minimum value for each dimension, thus, obtaining the vector $\mathbf{k} = [k_1^{\min}, k_1^{\max}, \ldots, k_d^{\min}, k_d^{\max}]$ across all the dimensions d. This means that we have the associated boundaries/rectangle for each cluster.

Furthermore, we obtain the corresponding query q boundaries/rectangle to measure the overlapping between the query's rectangle and each cluster's rectangle at each dimension. This is done locally in each node. In order to determine the number of clusters that have a high overlapping degree with the query q, we need to match the query boundaries with each cluster's boundaries. Specifically, the analytics query is expressed as the following vector denoting the data boundaries/regions of data requested by the application, i.e., $\mathbf{q} = [q_1^{\min}, q_1^{\max}, \ldots, q_d^{\min}, q_d^{\max}]$. Given this representation, we can obtain the corresponding query q (hyper)rectangle and the (hyper)rectangle for each cluster. Based on these two hyper-rectangles, data overlapping rate h_{ik} can be estimated focusing on the overlapping of the lengths of each dimension in each cluster k with these of the query at node n_i.

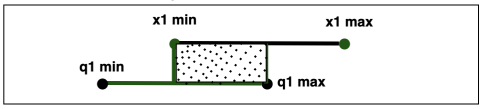
We can identify five overlapping cases between cluster and query per dimension. In the first case, both query boundaries $(q_d^{\min}, q_d^{\max})$ are lower than cluster's boundaries $(k_d^{\min}, k_d^{\max})$, i.e., $(k_d^{\min} < q_d^{\min}$ AND $q_d^{\max} < k_d^{\max})$. Fig. 3a demonstrates that both of the query boundaries for a specific dimension are located inside the cluster's boundaries for that dimension. Hence, the overlapping is represented by the ratio $h_{ik}^d = \frac{q_d^{\max} - q_d^{\min}}{k_d^{\max} - k_d^{\min}}$. In the second case, only the minimum



(a) Both of the query boundaries belong inside the cluster boundaries.



(b) Only the minimum boundary of the query boundaries belongs to cluster boundaries.



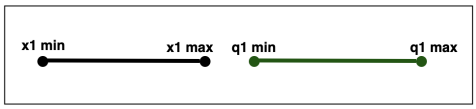
(c) Only the maximum boundary of query belongs to Cluster boundaries.

Fig. 3: Different levels of overlapping for three different cases.

boundary of query belongs to the cluster boundaries, i.e., $(q_d^{\min} < k_d^{\max})$. In this case, the cluster includes data range between $\{q_d^{\min}, k_d^{\max}\}$ as shown in Fig. 3b. That is, the overlapping is the ratio $h_{ik}^d = \frac{k_d^{\max} - q_d^{\min}}{q_d^{\max} - k_d^{\min}}$. In contrast, in the third case, only the maximum boundary of the query belongs to the cluster boundaries. In this case, data availability will be in the range $\{k_d^{\min}, q_d^{\max}\}$ as shown in Fig. 3c. The corresponding overlapping is then $h_{ik}^d = \frac{q_d^{\max} - k_d^{\min}}{k_d^{\max} - q_d^{\min}}$. In the fourth case, there exists zero overlapping, $h_{ik}^d = 0$, when both of the query's boundaries are completely outside the cluster's boundaries. This could happen when $\{q_d^{\min} > k_d^{\max}\}$ as in Fig. 4a or $\{q_d^{\max} < k_d^{\min}\}$ as in Fig 4b, thus, no overlapping. Based on these five cases, the data overlapping h_{ik} between cluster k and query q in node n_i for all the dimensions is:

$$h_{ik} = \frac{1}{d} \sum_{d} h_{ik}^d. \tag{2}$$

Given that for each cluster k in node n_i we have calculated the corresponding data overlapping across all dimensions, h_{ik}, we select the cluster k if and only if the $h_{ik} \geq \epsilon$, given a predefined overlapping threshold $\epsilon > 0$.



(a) Zero overlapping: both of cluster boundaries are lower than the query boundaries.

(b) (Opposite) Zero overlapping: both of the query boundaries are lower than cluster boulders.

Fig. 4: Zero overlapping between query and cluster per dimension.

Then, for each node n_i, we define the corresponding *potential* p_i according to the number of *supporting* clusters $K' \leq K$, whose $h_{ik} \geq \epsilon$, $k = 1, \ldots, K'$, as follows:

$$p_i = \sum_{k=1}^{K'} h_{ik}. \tag{3}$$

Given the potential of a node n_i and the number of supporting clusters $K' \leq K$, we define the *ranking* for a node n_i for the incoming query q as:

$$r_i(q) = p_i \frac{K'}{K} \tag{4}$$

The nodes are sorted w.r.t. their rankings $\{r_1, \ldots, r_N\}$, thus, the leader node can now select the subset of these nodes, i.e, the top-ℓ nodes ($\ell \leq N$) to be acting as the participants for the incoming query q. Note that the number ℓ of the top ranked nodes can be also determined by selecting those node whose ranking exceeds an application-defined threshold $\psi > 0$, i.e., $\ell = |\mathcal{N}'(q)|$ with

$$\mathcal{N}'(q) = \{n_i \in \mathcal{N} : r_i(q) \geq \psi\}, \tag{5}$$

and $|\mathcal{N}|$ denotes the cardinality of the set $\mathcal{N}$. Any node n_i having a relatively high ranking will be involved in the learning process. The leader node can determine the list of the best participants given a query with negligible calculations and communication. The nodes just send to the leader the boundaries of their clusters and the number of the clusters per node, yielding $O(1)$ communication complexity. The leader then can locally determine the list of participants based on the query boundaries, which is of $O(d)$ complexity.

IV. Data Selectivity & Distributed Learning Mechanisms

A. Query-driven Data Selectivity Mechanism

After the leader node determines the most suitable participants given a query, it is noteworthy to mention that not all the data in one selected node could be helpful to the global model. Many models target a specific range of of values over their data, e.g., learning the relation between age range having children with the chance of getting a specific kind of cancer does not require all value ranges about all patients in a hospital; just those with age e.g., between 20 to 50 with children. According to most of the studies, they consider all data in one selected node to build a model. However, in many scenarios, this is not efficient. If we have considered all the data in each selected node, we could get an inaccurate model due to over-fitting factors. To avoid these issues, we need to be specific about the amount of requested data in each node for building a model given a query.

Assume that we have set of selected nodes (participants) $n_k \in \mathcal{N}'(q)$ according to our nodes selection mechanism given a query q. These nodes n_k have different ranking according to the number of clusters that satisfies the threshold condition $\geq \epsilon$. As we have produced in (4), nodes' ranking depends on the number of the supporting clusters out of all clusters. In this part, we build a model over only those supporting clusters that gave a node n_i high potential to be selected. When the model $\mathcal{M}$ is to be trained on participant n_i that has a high ranking r_i, the node has already clustered its own data as discussed in Section III-C and shown in Fig. 5. The model will be trained only over those data belonging to clusters k with $h_{ik} \geq \epsilon$. Accordingly, we will obtain a model $\mathcal{M}$ which has been trained on supporting clusters' data rather than the *whole* data of node n_i. This (training) data selectivity reduces the over-fitting problem especially when the amount of data in each node is huge and increases the expected model accuracy since these selected data are explicitly requested by the incoming query (as inferred by the data overlapping metric), as it will be evidenced in our experiments.

Remark: A question now is raised: *why should we have more than one cluster per node for potential selection?* If we had only one cluster, the cluster boundaries could be expanded and included many unrelated data points given a query. This would lead to a data size similar to the whole node's data size. Note, the model $\mathcal{M}$ is trained in an incremental way inside the node n_i. This means that it works as the mini-batch way across clusters: each cluster represents a mini-batch. Hence, each node produces only one model including all the training obtained by the K' supporting clusters.

B. Distributed Learning Mechanism

This section shows how the leader node updates the global model according to the selected $\ell \leq N$ participants in $\mathcal{N}'(q)$ given a query q. First the leader node sends the initial global model w to each $n_i \in \mathcal{N}'(q)$. Each n_i trains w locally and incrementally only over each supporting cluster's data. Specifically, after $\mathcal{E}$ rounds of local iterations on each supporting cluster, the model considers the data from the second supporting clusters until the last supporting cluster's data. Then n_i updates w to get a local model $w_i^{\mathcal{E}}$. At the end of this step, the leader receives the local model $w_i^{\mathcal{E}}$ from each participant $n_i \in \mathcal{N}'(q)$. Then, we consider two types of aggregation in the

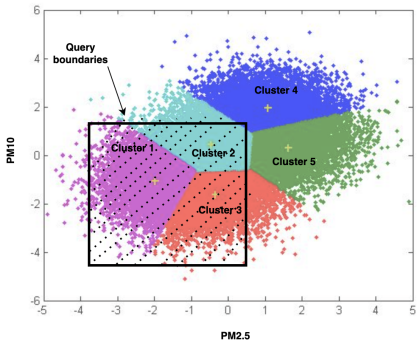


Fig. 5: Representation of the query q space projected onto a participant's data space.

leader node. **Model Averaging**. The leader node aggregates the prediction outcome $\hat{y}_i(q)$ from each received local model $w_i^{\mathcal{E}}$ equally given the query q:

$$\hat{y}(q) \quad = \quad \frac{1}{\ell}\sum_{i=1}^{\ell}\hat{y}_i(q). \tag{6}$$

However, nodes could have different data distributions and ranges. Hence, some models can perform better or worse than other models. It could be unfair to aggregate then the models equally. Accordingly, we propose a weighted average over the predicted outcomes that takes into account the contribution of the models based on the ranking determined given the incoming query. This is the **Weighted Averaging** such that the prediction outcome $\hat{y}_i(q)$ from each received local model is weighted with the relative ranking given the query q:

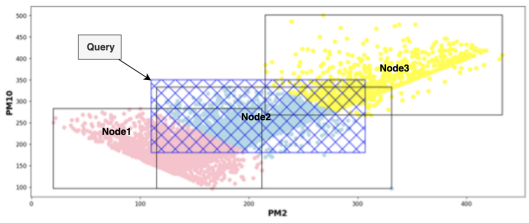
$$\hat{y}(q) \quad = \quad \sum_{i=1}^{\ell}\lambda_i\hat{y}_i(q). \tag{7}$$

where $\lambda_i = \frac{r_i}{\sum_{k=1}^{\ell} r_k} \in (0,1)$ represents the weight for each model w.r.t. ranking with $\sum_{i=1}^{\ell}\lambda_i = 1$.

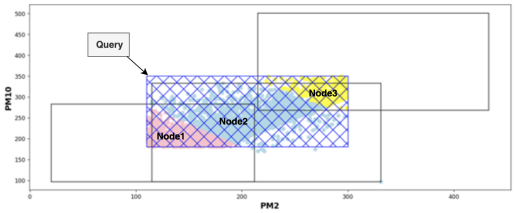
V. Experimental Evaluation

A. Experimental Setup

We assess the performance and efficiency of our mechanisms by conducting our experiment implemented in Python. To set up a realistic environment, we used the publicly available real dataset from the ML Repository (Beijing Multi-Site Air-Quality Data Data Set)[1]. From the dataset, we selected 10 data files; each file contains data collected from different geographical regions with assigned nodes collecting data locally. The number of edge edges N in our experiment is 10. Each data file represents a local dataset for an edge node. For each node, we focused on one import feature and labels to make it easy to track the models' performance when we apply our node selection mechanism. Since our goal in this work is to measure the data impact on models' performance, not improve models. Then, each node quantizes its data using the k-means clustering algorithm. We set up the number of clusters $K = 5$ for all nodes to avoid biases. According to these clusters, we can then obtain node ranking per query. We mainly focus on high-ranking nodes to participate in model construction. In our experiments, we used the Keras library to train models incrementally within each node over the supporting clusters' data for each of the 200 queries issued. Each query has been randomly created over the whole data space based on the dynamic query workload method described in [18]. Accordingly, each query needs a specific range of numerical data to build a prediction model. Some of these queries may have a high rate of overlapping with a large number of nodes, while others may have a high rate of overlapping with a small number of nodes. Meanwhile, the overlapping rate has a high impact on model performance. Fig 6 shows the amount of data needed by a query q, and its availability in 3 different nodes: nodes 1 and 3 have several supporting clusters, while node 2 has all its own clusters as supporting clusters for the considered query. One can observe from Fig 6a, that *all* the data that should be involved in the training process if we had not considered the data clustering. Whereas, Fig 6b shows the amount of data that are actually needed to be involved in the training process from three different nodes if we consider the clusters and query boundaries.



(a) The query space projected onto the available data spaces of 3 nodes.



(b) The *actual* data required by the query projected over the *whole* data space of the 3 nodes.

Fig. 6: The data required by a query compared to the whole available data.

B. Distributed Learning Experiment

We do not target the models themselves; our goal is to set up a suitable environment for models requested to be built over the data subspaces as specified by the incoming queries. That is, we assess how we correctly select the nodes that need to be engaged per query to build models over the

[1]https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data

right data. Therefore, we are targeting the needed data by models, and how we can improve these models performance by selecting the most appropriate participants with their most supporting clusters' data in each node. Upon our proposed nodes selection mechanism, we need to see how this could affect Machine Learning model's performance. In this work, we experiment with the Linear Regression (LR) and Neural Network (NN) models in order to get an insight of how our node selection mechanism can be appropriately adopted in distributed learning environments. The models' tuned hyper-parameters are provided in Table III.

TABLE III: Model Hyper-parameters

Model	LR	NN
Dense	1	64
epochs	100	100
validation split	0.2	0.2
Learning rate	0.03	0.001
activation	relu	relu
Loss	MSE	MSE

C. Models Under Comparison

In this work, we use the mechanisms of Model Averaging and Weighted Averaging to assess our selection mechanism against two models found in the literature: Random selection, where ℓ nodes are selected randomly as in [6] and the Game Theory (GT) selection [7]. In GT, each node builds its own independent local model in advance according to its local dataset. In GT, it is assumed that each node could exchange its model with the other nodes in the same environment. When a node n_k receives the trained model from the leader, it tests the model's performance locally against its data and returns the results to the leader. The leader node targets those nodes that obtained accuracy lower than a threshold. The rationale behind this is that this node could have data with different patterns than the data of the leader node. Therefore, the leader node selects those nodes n_k that have models with low accuracy in order to make the model more general.

D. Evaluation Results

The number of nodes selected per query depends on the underlying data over the nodes, the query boundaries, and the number of supporting clusters. In our experiments, we take the average of the performance metrics across all the issued queries and selected nodes. When we try to measure a distributed model's performance, we look for three main criteria: loss, the time needed to train a model given a query, and the amount of data used for each query. In terms of loss and model accuracy, the Average and Weighted models are aggregation mechanisms that have been used to aggregate models selected according to our mechanism. As we can observe from Figure 7, our proposed mechanism outperforms the Random and GT methods in terms of error rate since they focus on selecting participants according to the matching between participants' data and data needed by the queries. The Random selection is the fastest node selection mechanism. However, in terms of accuracy, it is the worst because it does not take into account the query-specified requested data spaces. Hence, its performance gets worse by increasing the participants' heterogeneity. Therefore, it is the highest loss rate compared to the other mechanisms. The GT mechanism is the slowest since it needs to build models on each participant *before* applying its node selection mechanism. In terms of accuracy, the GT is not introducing a high accuracy when dealing with queries. This is due to the fact that its mechanism on selecting participants takes into account almost the entirely different data spaces compared to what models has seen before, regardless of whether all this variety of data is needed or not by the queries.

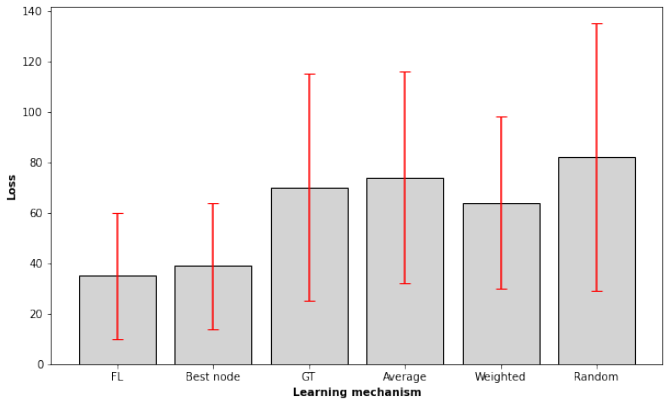


Fig. 7: Average loss of all the models: GT, Random, Averaging, and Weighted.

Regarding the model training time, our node selection mechanism produces models that take less training time over the selected participants. This is attributed to the fact that we only focus on training models on specific supporting clusters in each participant rather than the whole participant's dataset. Therefore, models will no longer need to be trained on all the data during the training process as specified by the incoming queries. As a result, the training time decreases as well. In Fig.8, the green line refers to the required time to train models according to our mechanism, while the blue one represents the time required to train models on the whole participants' datasets, i.e., without taking into account the query (for legibility reasons, only the results of a stream of 20 sequential queries are plotted). Finally, Figure 9 shows the percentage of data needed by each query from all participants when we apply our query-driven mechanism (green bars). In contrast, the gray bars show the amount of data used by each query if we do not consider the queries (for legibility reasons, only the results of a stream of 20 sequential queries are plotted). It is evidenced that the query-driven mechanism is efficient in terms of selecting the most appropriate nodes to build the model per query and, in turn, the node considers appropriate data sub-space to be involved in the model training phase ensuring higher model accuracy. The proposed data selectivity and node selectivity mechanisms yield our mechanism applicable to distributed learning environments.

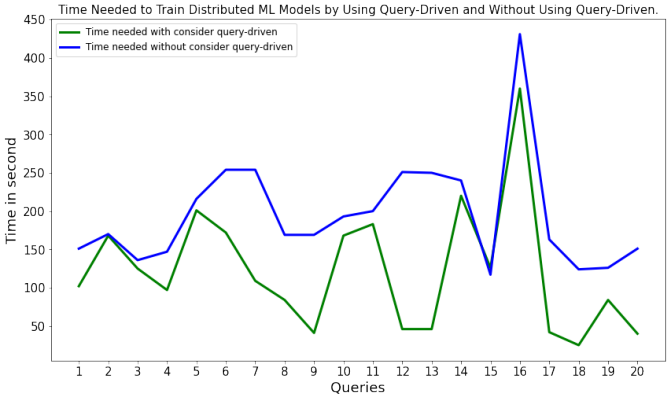


Fig. 8: Required model building time w/wo considering the incoming queries.

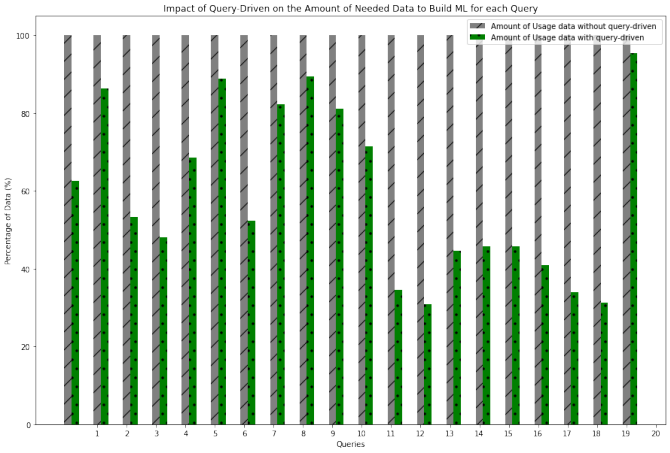


Fig. 9: Percentage of data needed to build models w/wo considering the incoming queries.

VI. Conclusions

We propose a query-driven node selection mechanism in distributed learning environments. We contributed with a mechanism to determine whether participants are heterogeneous since this plays a significant role in identifying the most appropriate set of nodes to be engaged in a model building per analytics query. We studied the data overlapping between each cluster in each participant and the query. This yields information about the most relevant data to be involved in training models in a distributed way by also selecting the most suitable nodes. According to a computationally and communication efficient ranking framework, the model training process is obtained by highly ranked participants per query. Our mechanism minimizes the data needed by each participant to train the model only over the query-driven supporting clusters' data. Our experimental results and comparative assessment showcase that our selection mechanism is deemed appropriate in distributed edge learning environments.

References

[1] K. V. Sarma, S. Harmon, T. Sanford, H. R. Roth, Z. Xu, J. Tetreault, D. Xu, M. G. Flores, A. G. Raman, R. Kulkarni *et al.*, "Federated learning improves site performance in multicenter deep learning without data sharing," *Journal of the American Medical Informatics Association*, vol. 28, no. 6, pp. 1259–1264, 2021.

[2] C. Anagnostopoulos, "Edge-centric inferential modeling analytics," *Journal of Network and Computer Applications*, vol. 164, p. 102696, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804520301703

[3] H. Ludwig and N. Baracaldo, "Federated learning: A comprehensive overview of methods and applications," 2022.

[4] J. Jiang, L. Burkhalter, F. Fu, B. Ding, B. Du, A. Hithnawi, B. Li, and C. Zhang, "Vf-ps: How to select important participants in vertical federated learning, efficiently and securely?" in *Advances in Neural Information Processing Systems*.

[5] Q. Long, K. Kolomvatsos, and C. Anagnostopoulos, "Knowledge reuse in edge computing environments," *Journal of Network and Computer Applications*, vol. 206, p. 103466, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S108480452200114X

[6] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[7] A. Hammoud, A. Mourad, H. Otrok, and Z. Dziong, "Data-driven federated autonomous driving," in *International Conference on Mobile Web and Intelligent Information Systems*. Springer, 2022, pp. 79–90.

[8] R. Saha, S. Misra, A. Chakraborty, C. Chatterjee, and P. K. Deb, "Data-centric client selection for federated learning over distributed edge networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 675–686, 2022.

[9] W. Lee, "Reward-based participant selection for improving federated reinforcement learning," *ICT Express*, 2022.

[10] Z. Zhu and L. Sun, "Federated trace: A node selection method for more efficient federated learning," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 1234–1238.

[11] W. Lin, Y. Xu, B. Liu, D. Li, T. Huang, and F. Shi, "Contribution-based federated learning client selection," *International Journal of Intelligent Systems*, 2022.

[12] T. Huang, W. Lin, L. Shen, K. Li, and A. Y. Zomaya, "Stochastic client selection for federated learning with volatile clients," *IEEE Internet of Things Journal*, 2022.

[13] F. Savva, C. Anagnostopoulos, P. Triantafillou, and K. Kolomvatsos, "Large-scale data exploration using explanatory regression functions," *ACM Trans. Knowl. Discov. Data*, vol. 14, no. 6, sep 2020. [Online]. Available: https://doi.org/10.1145/3410448

[14] K. Kolomvatsos, C. Anagnostopoulos, M. Koziri, and T. Loukopoulos, "Proactive time-optimized data synopsis management at the edge," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 7, pp. 3478–3490, 2022.

[15] N. Harth, C. Anagnostopoulos, H.-J. Voegel, and K. Kolomvatsos, "Local federated learning at the network edge for efficient predictive analytics," *Future Generation Computer Systems*, vol. 134, pp. 107–122, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X22001054

[16] P. K. et. al, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021. [Online]. Available: http://dx.doi.org/10.1561/2200000083

[17] S. Na, L. Xumin, and G. Yong, "Research on k-means clustering algorithm: An improved k-means clustering algorithm," in *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, 2010, pp. 63–67.

[18] F. Savva, C. Anagnostopoulos, and P. Triantafillou, "Aggregate query prediction under dynamic workloads," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 671–676.