H2020 – ICT-13-2018-2019

Machine Learning to Augment Shared Knowledge in Federated Privacy-Preserving Scenarios (MUSKETEER)

Grant No 824988

D6.2 Scalability of machine learning algorithms over every POMs

August 21

Imprint

Contractual Date of Delivery to the EC:	**31 August 2021**
Author(s):	**Ángel Navia-Vázquez (UC3M), Jesús Cid-Sueiro (UC3M), Manuel Vázquez (UC3M), Roberto Díaz (TREE), Jose Manuel Fernández (TREE), Jaime Medina (TREE)**
Participant(s):	**Lucrezia Morabito (COMAU), Susanna Bonura (ENG), Mark Purcell (IBM), Gal Weiss (IBM)**
Reviewer(s):	**Lucrezia Morabito (COMAU), Susanna Bonura (ENG)**
Project:	**Machine learning to augment shared knowledge in federated privacy-preserving scenarios (MUSKETEER)**
Work package:	**WP6**
Dissemination level:	**Public**
Version:	**1.0**
Contacts:	**Roberto Díaz Morales (roberto.diaz@treelogic.com)** **Angel Navia-Vázquez (angel.navia@uc3m.es)**
Website:	**www.MUSKETEER.eu**

Legal disclaimer

The project Machine Learning to Augment Shared Knowledge in Federated Privacy-Preserving Scenarios (MUSKETEER) has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824988. The sole responsibility for the content of this publication lies with the authors.

Copyright

Executive Summary

This deliverable (D6.2 Scalability of machine learning algorithms over every POMs) is the result of task T6.2 (Assessing scalability and computational efficiency of federated privacy-preserving machine learning algorithms) and describes the results obtained after intensive tests carried out to analyse the main technical characteristics of every algorithm over every POM. We have followed the Goal Question Metric methodology described in D6.1, and the main goals are to assess the scalability, computational efficiency and performance of the proposed and implemented schemes in the MUSKETEER Machine Learning Library (MMLL).

Document History

Version	Date	Status	Author	Comment
1	30th July 2021	For internal review	Angel Navia (UC3M), Jesús Cid (UC3M), Manuel Vázquez (UC3M), Roberto Díaz (TREE), Jose Manuel Fernández (TREE), Jaime Medina (TREE)	First draft
2	4th August	Reviewed	Lucrezia Morabito (COMAU)	Interim version
3	5th August	Reviewed	Susanna Bonura (ENG)	Interim version
4	18th August	Changes implemented	Roberto Díaz (TREE), Jaime Medina (TREE)	Interim version
5	22nd August	Changes implemented	Angel Navia (UC3M)	Interim version
6	23rd August	Technical and coordinator review	Mark Purcell (IBM), Gal Weiss	Interim version
7	23rd August	Final version	Jaime Medina (TREE)	Final version

Table of Contents

List of Figures

List of Acronyms and Abbreviations

ABBREVIATION	DEFINITION
AUC	Area Under (ROC) Curve
AWS	Amazon Web Service
CA	Consortium Agreement
DP	Differential Privacy
DOW	Description of Work
DSVM	Distributed Support Vector Machine
FBSVM	Federated Budgeted Support Vector machine
GA	Grant Agreement
GQM	Goal, Questions and Metrics
GPU	Graphical Processing Unit
gRPC	gRPC Remote Procedure Call
HBC	Honest But Curious
HE	Homomorphic Encryption
IDP	Industrial Data Platform
ML	Machine Learning
MLA	Machine Learning Algorithm
MMLL	MUSKETEER Machine Learning Library
MN	Master Node
Non-iid	Non-independently and identically distributed data
PERT	Program evaluation and review technique

PHE	Partial Homomorphic Encryption
POM	Privacy Operation Mode
PP	Privacy Preserving
PPML	Privacy Preserving Machine Learning (a.k.a. Privacy Preserving Data Mining)
RAM	Random Access Memory
REST	REpresentational State Transfer
ROC	Receiver Operating Characteristics
SDP	Secure Dot Product
SMC	Secure Multiparty Computing
WN	Worker Node

1 Introduction

1.1 Purpose

The MUSKETEER project aims at building an Industrial Data Platform (IDP) such that different users can contribute with their own data to solve a given Machine Learning (ML) task without compromising the confidentiality of the data. The core component of the platform providing the capability of training ML models while preserving confidentiality in the data, is named as the MUSKETEER Machine Learning Library [MMLL_2021] and comprises several ML Algorithms developed under different Privacy Operation Modes (POMs), as described in Deliverables D4.5 and D4.7.

As a reminder a brief summary of each POM is presented below:

- POM 1. This POM is designed for scenarios where the final trained model is not private, since at the end of the training every worker node and also the master node have a copy the model. This POM implements a federated-learning framework based on the concept introduced by Google in 2016 (Konečný et al. 2016a).
- POM2. This POM also implements the Federated Machine Learning (FML) paradigm described in POM1, but uses additively homomorphic encryption (Yi et al., 2014) to preserve model confidentiality from the central node.
- POM3. This POM is an extension of POM2 that makes use of a proxy re-encryption protocol to allow that every data owner can handle her/his own private key (Hassan et al., 2019). The aggregator has access to all public keys and is able to transform data encrypted with one public key to a different public key so that all the participants can share the final model.
- POM 4. This POM uses an additively homomorphic cryptosystem to protect the confidentiality of the data and requires the cooperation of a special node named as Crypto Node (CN), which is an object/process providing support for some of the cryptographic operations not supported by the homomorphism (González-Serrano et al., 2017).
- POM 5. This POM is able to operate only with the aggregator and worker nodes. It uses an additively homomorphic cryptosystem to protect the confidentiality of the model (Giacomelli et al, 2017). The data is also protected, since it does not leave the worker facilities, and only some operation results are sent to the aggregator.
- POM 6. This POM does not use encryption; it relies on Secure Two-Party Computation (Cramer and Damgård, 2015) protocols to solve some operations on distributed data such that both mod and data privacy is preserved (Chen et al., 2019).

The deliverable D6.1 described the assessment methodology that has been proposed to characterize the behaviour and main performance indicators of the implemented algorithms in MMLL, from a technical point of view.

The assessment experiments described in this document focus on the evaluation of the ML components from a general point of view, not focusing on any specific application domain, and therefore datasets will not be provided by the use cases. We have, instead, used standard open datasets, such that the experiments can easily be replicated by other researchers and potentially compared with the results obtained with other analogous systems.

1.2 Related documents

As indicated in the PERT diagram below (Figure 1), the results from this deliverable will provide input to WP7 (User Cases), such that they can better decide which POM/algorithms are the most adequate to solve a given task. Although not shown in that PERT diagram, the inputs are mainly defined by the project KPIs and objectives that concern the performance of the implemented Machine Learning algorithms. Deliverables D4.5 and D4.7 describe in full detail the implemented algorithms under the different Privacy Operation Modes, which comprise the MUSKETEER Machine Learning Library (MMLL) [MMLL_2021]. This report focuses on the technical assessment of the developed ML library under the different POMs and this information will help in the selection of models while developing the use cases solutions in the context of WP7.

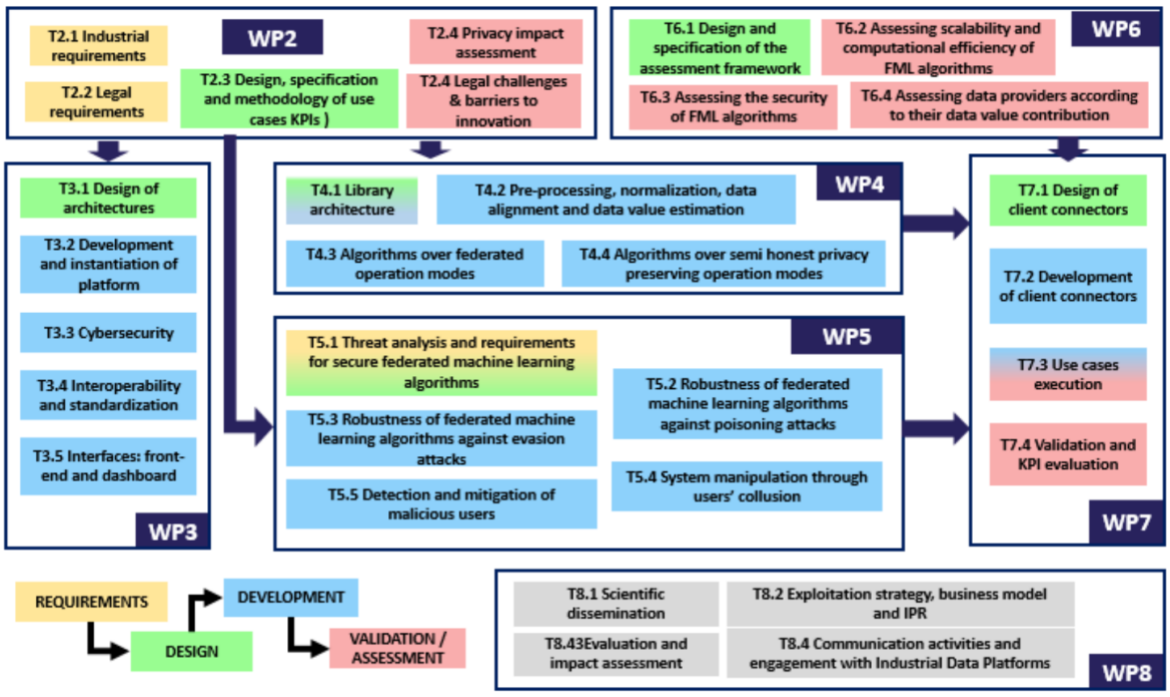


Figure 1. MUSKETEER's PERT diagram

1.3 Document structure

This document is structured as follows:

- The current section (Introduction) presents the purpose of the document, as well as the relationship with other WPs in the project.
- In Section 2 the selected assessment methodology it is provided: Goals, Questions and Metrics [Solingen_1999].
- In Section 3 we describe the collection of open access datasets that have been used throughout this assessment work.
- In Section 4 we describe the experimental setup that has been used to run the tests and collect the needed information to answer the questions defined in the aforementioned GQM methodology.
- In Section 5 we explain the detailed results of the GQM (Goal 1) for POMs 1, 2 and 3.
- In Section 6 we explain the detailed results of the GQM (Goal 1) for POMs 4, 5 and 6.
- In Section 7 we have synthesised a simple questionnaire that helps the end user to select the most appropriate POM for a given task.
- In Section 8 we present the main conclusions.
- Section 9 collects the main references.
- Section 10 (Appendix I): comprises the full collection of figures used to obtain the conclusions for POMs 1, 2 and 3.
- Section 11 (Appendix II): comprises the full collection of figures used to obtain the conclusions for POMs 4, 5 and 6.

2 Goals-Questions-Metrics (GQM) methodology revisited

The Goals-Questions-Metrics (GQM) methodology [Solingen_1999] has been adopted in D6.1 to define the main common evaluation framework to be used to assess the scalability, computational efficiency, performance, security, and data value estimation capabilities of the proposed and implemented Machine Learning Algorithms (MLAs) under the different Privacy Operation Modes (POMs), according to Tasks T6.2, T6.3 and T6.4 in WP6. We briefly summarize here the main GQM characteristics; the interested reader may refer to D6.1 for a deeper understanding of this methodology. Anyhow, in what follows we will briefly revisit

GQM and explain its main characteristics, focusing on the aspects to be evaluated in this report. The GQM approach is usually illustrated using the following Figure.

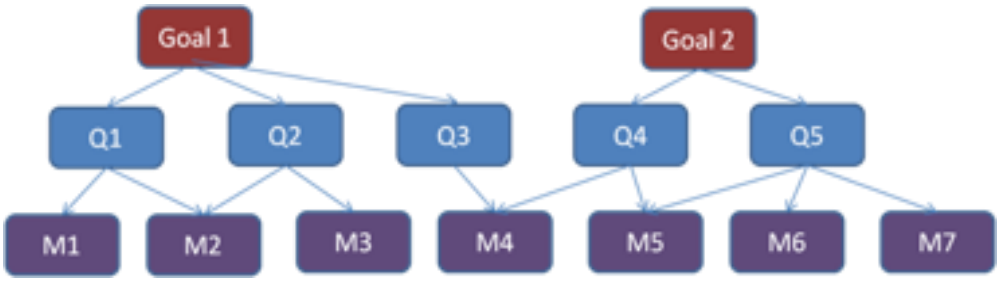


Figure 2. Goals, Questions and Metrics (GQM) paradigm example

This methodology, used in agile environments, allows for identifying and further refining a collection of explicit measurement goals, depicted on the top level of Figure 2. After the goal identification phase, one or more questions can be defined for every goal. Finally, one or more metrics are described to answer those questions. Summarizing:

- "**Goals**" define what the project wants to improve.
- "**Questions**" refine each goal in a more quantifiable way.
- "**Metrics**" indicate the metrics required to answer each question.

In this document we will focus on the Goal G1 described in D6.1 (the results for Goals 2 and 3 will be reported in D6.3 and D6.4, respectively):

Goal 1: Assessing performance, scalability and computational efficiency of MLAs (G1) which has been refined into the following sub-goals:

- **G1.1: Assessing the performance of MLA**
- **G1.2: Assessing the reliability of MLA**
- **G1.3: Assessing the scalability of MLA**
- **G1.4: Assessing the computational efficiency of MLA**

Therefore, the objective is to evaluate the developed MLAs to determine if their general behaviour is as expected, mainly from the point of view of performance:

- Does MMLL provide models as competitive as those obtained with other libraries?
- Does the computational cost of the training procedure grow in a controlled manner?
- Do we need to provide an excessive amount of memory, computational or communication resources for the library to work?

Depending on the MLA under analysis, the question may be slightly different. For instance, in the case of a clustering algorithm, the question to measure the performance could be:

"Is the MLA able to provide a data clustering in such a way that objects in the same group are more similar to each other than to those in other groups?"

Whereas for a classification algorithm, the corresponding question would be:

"Given a training dataset, is the ML library able to provide predictions of the class of each data, according to some related categorical variable?"

In both cases, once the corresponding metric is defined (accuracy, average distance to closest centroid, etc.), the Goal is achieved if the metric obtained with MMLL is within a 5% of variation with respect to the metric obtained with the reference library (scikit-learn, Kheras, Tensorflow, etc.)

In the assessment Sections 5 and 6 we will provide detailed (quantitative) answers for the questions defined in D6.1, such that the Goals are adequately covered. For operative reasons we will provide separate assessment results for POMs under the federated learning setting (Section 5) and those POMs that operate under the Hones but Curious hypothesis (Section 6) but we have followed the same GQM methodology in all cases.

3 Selected datasets and their characteristics

In this section we briefly describe the datasets selected to carry out the assessment experiments. To facilitate the replicability of the experiments by other researchers we will rely on publicly available datasets. We propose to use a collection of datasets with a wide range of characteristics to explore their behavior under different conditions: number of training patterns, number of features, type of target values (continuous for regression, discrete for classification, non-existent for clustering, etc.). For every experiment, one or more public datasets will be selected, to facilitate the replication of experiments by other researchers.

As already mentioned, the evaluation experiments described in this document focuses on the evaluation of the ML components from a general point of view, and therefore we will not use the datasets provided by the use cases. The datasets from the use case pilots will be used in WP7 for the final platform validation.

We have used the following datasets (listed in alphabetical order):

- **Abalone** (Predict the age of abalone from physical measurements): Dataset to predict the age or gender of abalone from eight physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope. Other measurements, which are easier to obtain, are used to predict the age. [Abalone]

- **Airfoil** (Airfoil Self-Noise Data Set): This NASA data set comprises different size airfoils at various wind tunnel speeds and angles of attack. The attributes include frequency, angle of attack, chord length, free-stream velocity and suction side displacement thickness, and the variable to predict is the scaled sound pressure level, in decibels. [Airfoil]
- **BlogFeedback** (BlogFeedback Dataset): This data originates from blog posts. The raw HTML-documents of the blog posts were crawled and processed. The prediction task associated with the data is the prediction of the number of comments in the upcoming 24 hours. In order to simulate this situation, we choose a basetime (in the past) and select the blog posts that were published at most 72 hours before the selected base date/time. Then, we calculate all the features of the selected blog posts from the information that was available at the basetime, therefore each instance corresponds to a blog post. The target is the number of comments that the blog post received in the next 24 hours relative to the basetime [BlogFeedback].
- **Boston (**Housing Dataset): contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. It was obtained from the StatLib archive [Boston]. It has two prototasks: nox (**Boston-nox**), in which the nitrous oxide level is to be predicted; and (**Boston-price**) price, in which the median value of a home is to be predicted. [Boston]
- **Cardio** (Cardiotocography Data Set): Fetal cardiotocograms (CTGs) plus respective diagnostic features are provided. The CTGs were also classified by three expert obstetricians and a consensus classification label assigned to each of them. Classification was both with respect to a morphologic pattern and to a fetal state. The features comprise fetal movements per second, percentage of time with abnormal short-term variability, histogram mean, beats per minute, among many others. The objective here is to identify a normal fetal state. [Cardio]
- **Diabetes** (Pima Indians): The dataset consists of several medical predictor (independent) variables and one target (dependent) variable. Independent variables include the number of pregnancies the patient has had, their BMI, insulin level, age, etc. The goal is to predict the onset of diabetes based on diagnostic measures [Diabetes].
- **Income (**Adult income): Prediction task is to determine whether a person makes over 50K dollars a year. The input variables are both numerical and categorical. [Adult]
- **Iris** (The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are

not linearly separable from each other. The attribute to predict is the class of iris plant.) [Iris]

- **Landsat** (Statlog (Landsat Satellite) Data Set): The database consists of the multi-spectral values of pixels in 3x3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. The aim is to predict this classification, given the multi-spectral values. The classification label of the central pixel. The number is a code for the following classes: red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble, mixture class and very damp grey soil. [Landsat]
- **MNIST**: (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is widely used for training and testing in the field of machine learning [MNIST]. The input data are pixels values of 28x28 images, and the targets are the '0-9' labels. A binary classification version is also used, where the objective is to differentiate between even and odd numbers.
- **Parkinson** (Parkinsons Telemonitoring Data Set): Created at the University of Oxford, in collaboration with 10 medical centers in the US and Intel Corporation who developed the telemonitoring device to record the speech signals. The original study used a range of linear and nonlinear regression methods to predict the clinician's Parkinson's disease symptom score on the UPDRS scale. The dataset is composed of a range of biomedical voice measurements from 42 people with early-stage Parkinson's disease recruited to a six-month trial of a telemonitoring device for remote symptom progression monitoring. The main aim of the data is to predict the motor and total UPDRS scores ('motor_UPDRS' and 'total_UPDRS') from the 16 voice measures. [Parkinson]
- **Redwine** (Red Wine Quality Index): contains 1,599 red wines with 11 variables on the chemical properties of the wine. At least 3 wine experts rated the quality of each wine, providing a rating between 0 (very bad) and 10 (very excellent) [Redwine].
- **Retinopathy** (Diabetic Retinopathy Debrecen): This dataset contains features extracted from the Messidor image set to predict whether an image contains signs of diabetic retinopathy or not. All features represent either a detected lesion, a descriptive feature of an anatomical part or an image-level descriptor. Attributes include quality assessment, pre-screening, Euclidean distance of the center of the macula and the center of the optic disc, diameter of the optic disc or the binary result of the AM/FM-based classification. The objective is to predict if the image contains signs of DR (Accumulative label for the Messidor classes 1, 2, 3) or not. [Retinopathy]

- **Segmentation** (Statlog (Image Segmentation) Data Set): The instances were drawn randomly from a database of 7 outdoor images. The images were manually segmented to create a classification for every pixel. Attributes are like the column of the center pixel of the region, contrast of horizontally adjacent pixels, measure of the excess red, etc. The objective is to classify images in the following groups: brick face, sky, foliage, cement, window, path, grass. [Segmentation]
- **Superconductivity** (Superconductivity Data Set): Dataset to predict the superconducting critical temperature based on the features extracted from the superconductor's chemical formula [Superconductivity].

3.1 Summary of datasets characteristics

In the following table (Table 1) we will summarize the characteristics of the used datasets. In the references section (Section 9) we also provide the link to their original download site. The characteristics indicated in the table correspond to the data partitioning used in the assessment experiments, mainly showing the number of patterns available for training, the size of the validation and test datasets, and the number of input features, as well as their type. The kind of tasks that a dataset can be used to are summarized in the last column.

Dataset Name	Number of Patterns[1] (train)	Number of Patterns (validation[2])	Number of Patterns (test[2])	Number of Features	Type of features	Targets	Tasks[3]
Abalone	4177			8	float	cat	MCC, CL
Airfoil	750	250	503	5	float	float	R
BlogFeedback	52397			280	float	float	R
Boston	51,630			14	float, int	float	R
Diabetes	500	100	168	8	Int, float	int	BC

[1] Training patterns or records.

[2] In some cases, no validation set was provided in the original partition, so a fraction of the training data was selected to serve as a validation set.

[3] BC: Binary Classification, MCC: Multi-Class Classification, CL: Clustering, R: Regression

Fashion-MNIST	50,000	10.000	10,000	784	Int	cat	MCC, BC*, CL
Income	26,049	6,512	16,281	107	float, cat	int	BC, R
Iris	100	25	25	4	float	cat	MCC, CL
Landsat	3,500	1,435	2000	36	float	cat	MCC
MNIST	50,000	10,000	10,000	784	Int	cat	MCC, BC*, CL
Parkinson	4,000	800	1075	18	float	float	R
Redwine	1,000	200	398	11	float	float	R
Retinopathy	900	100	151	18	float	bin	BC
Segmentation	1,900	200	210	19	float	cat	MCC
Superconductivity	21263			81	float	float	R

(*) If the partition is available in the original dataset.

(**) If the original dataset does not provide validation or test sets, we define them by splitting the training set.

Table 1. Summary of datasets characteristics

4 Experimental setup

We assume that, additionally to the training data from every user, local validation and test datasets are available. This assumption is only needed for the assessment purpose, in the normal operation of the MUSKETEER platform those datasets are not mandatory, and all of the algorithms are able to train a model without a validation set, although in some cases an improved convergence/speedup is obtained if such a validation dataset is provided. Anyhow, it is a common situation that the user in charge of aggregating the weights of a model has a test set, to evaluate the final performance achieved by the trained models.

Also, the local validation dataset may be used to adjust some of the parameters of the model in an experimental loop external to MMLL, i.e., the end user needs to run the MUSKETEER experiments to obtain those hyperparameters, since they are not obtained automatically within MUSKETEER. After the training is complete, the final performance is evaluated using the local test set.

The assessment experiments have been run for every available algorithm under the different POMs, such that conclusions on their behavior and characteristics can be extracted.

Every process in the experiments (aggregator, worker/participant, cryptonode) is run using a single core and all communications take place using the MUSKETEER Cloud communications facilities [Pycloudmessenger].

- The Aggregator is the central object or process that controls the execution of the training procedure.
- The Workers or participants run at the end user side as a part of the MUSKETEER client, and they have a direct access to the raw data provided by every user.
- Some POMs require the cooperation of a special node named as Crypto Node (CN), which is an object/process providing support for some of the cryptographic operations not supported by the

In every experiment, the datasets have been equally divided among the workers (e.g. when 5 workers are running, each one loads 20% of the data samples).

5 Assessment of POMs 1, 2 and 3.

5.1 Introduction

For this specific assessment we intended to measure above some performance metrics by running MUSKETEER federated framework in a set of AWS virtual machines.

We had 4 different setups to execute a full training cycle with the following number of worker processes: 1, 5, 10, 20.

Each machine will host up to 5 worker processes.

Each worker process will run inside a Docker container.

There will be a dedicated machine for each master process.

In the following picture a graphic representation of the setup is presented:

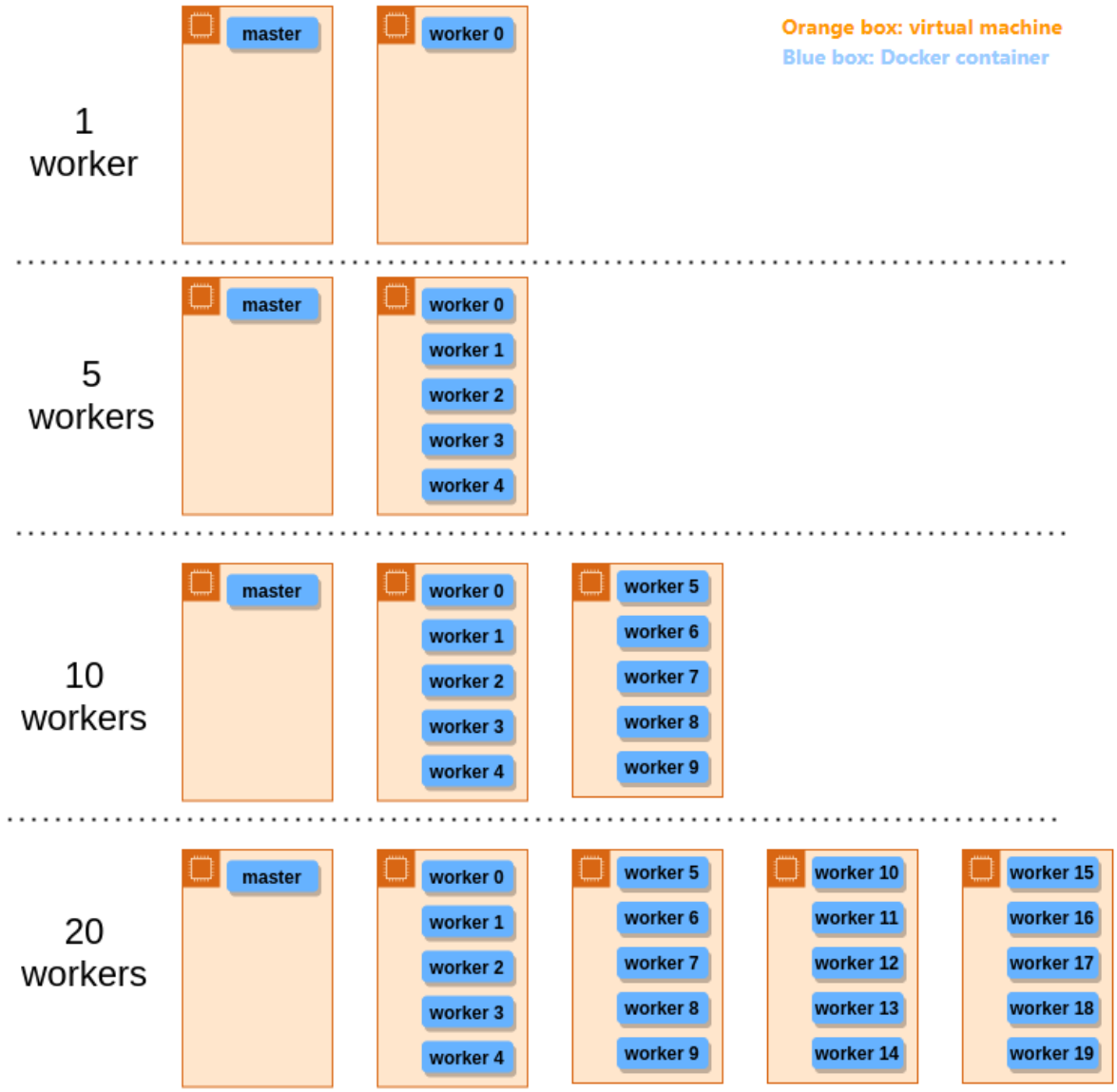


Figure 3. AWS Machines setup

Specifically, we have assessed the following variables:

- RAM consumption: We have measured the RAM consumption of every Docker container every two seconds. To obtain a single metric of RAM consumption in the master node and a worker node, we have averaged the samples of RAM.
- Training time: Total time for a model to complete the training process excluding the pre-processing and testing steps.
- Data transferred in bytes: bytes sent / bytes received by the master node.
- The performance of the ML model over the test set, according to the problem.

5.1.1 Clustering

5.1.1.1 K Means

Clustering aims at dividing the population or data into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar characteristics and assign them into clusters.

Algorithm implemented: K-Means

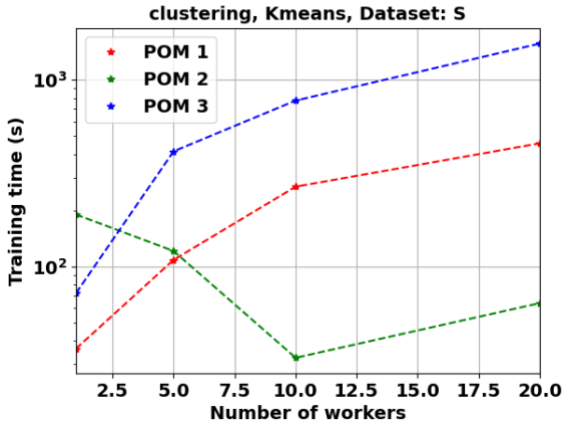
Goal: In order to analyse the scalability, we have executed the algorithms K-means for the three different federated learning POMs (1, 2 and 3) using different number of workers and dataset sizes (different number of samples and features) and we have measured different performance metrics to study how the POM, dataset and number of workers influence in these metrics.

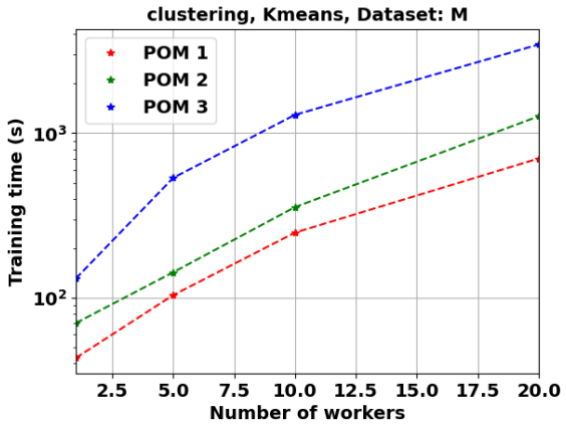
Performance metric: Average Euclidean distance to the closest centroid.

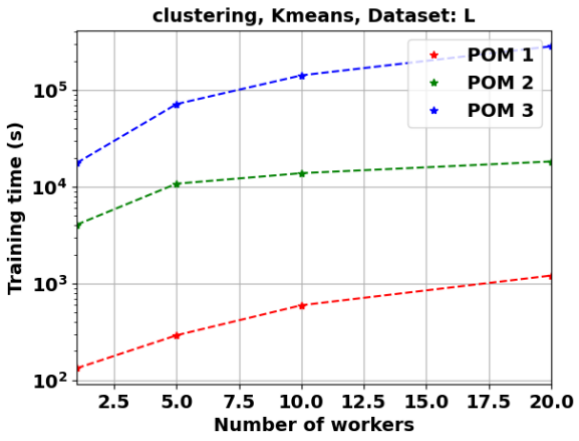
Dataset used:

- Iris (Small size)
- Abalone (Medium size)
- MNIST (Large size)

Training time:







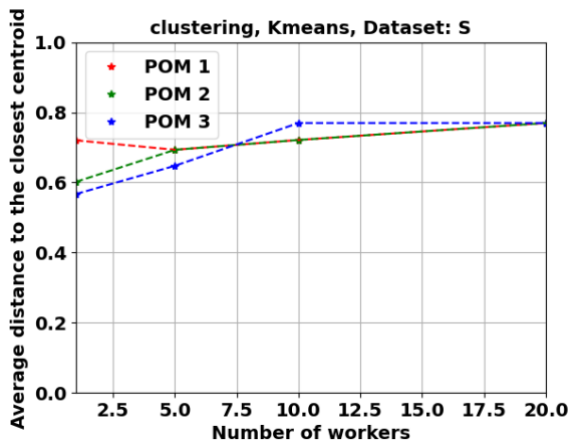
For medium and large size datasets, POM1 is the fastest one. It does not need encryption and the master node can broadcast the information to the worker nodes at the same time in every iteration.

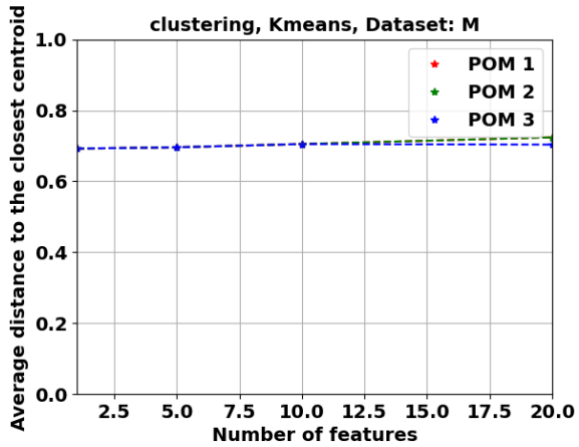
The behaviour of POM2 is similar to POM1 but it needs to encrypt and decrypt the information in every iteration, that is why the training time is higher than in POM1.

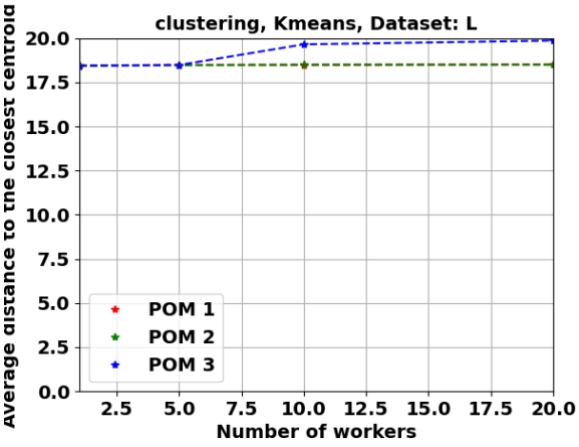
POM3 cannot make use of the broadcast command (and the communication is one of the bottlenecks in federated learning). That is why the runtime is higher than POM2.

Every iteration of the training process is fast for small datasets. That is why we can observe some strange behaviour in the dependencies (the state of the network influences more the training time than the process itself).

Performance metric:





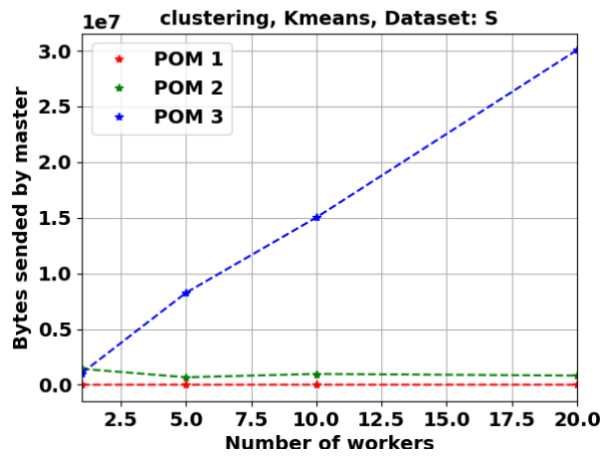


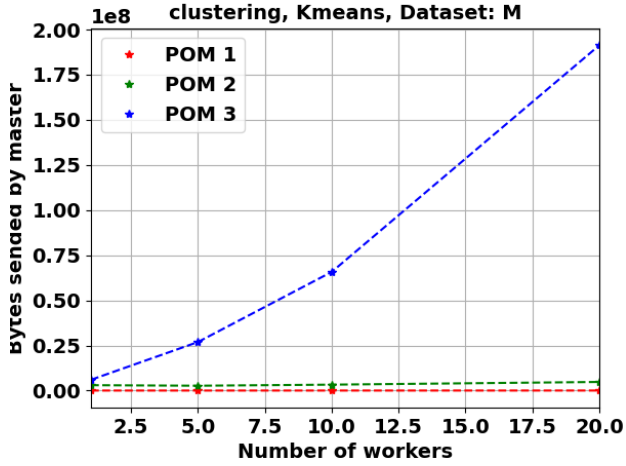
The performance metric of the clustering algorithm is not affected by the number of worker nodes or the POM used.

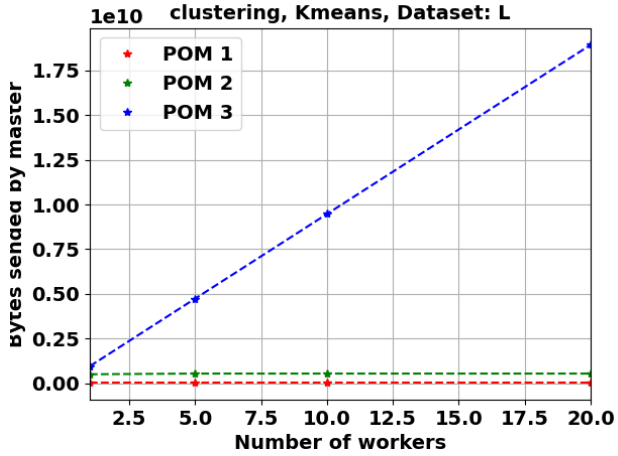
However, for small datasets, the random initialization of the algorithm and the reduced number of training samples can affect the result. That is why we can observe some differences when we run the algorithm several times.

As we increase the number of training patterns, we reduce the effect of this random initialization.

Bytes sent by the aggregator:







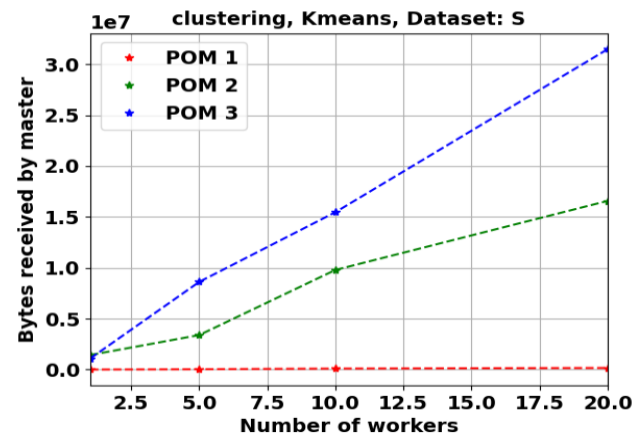
In every iteration of the training process, the aggregator send the K means centroids to every worker.

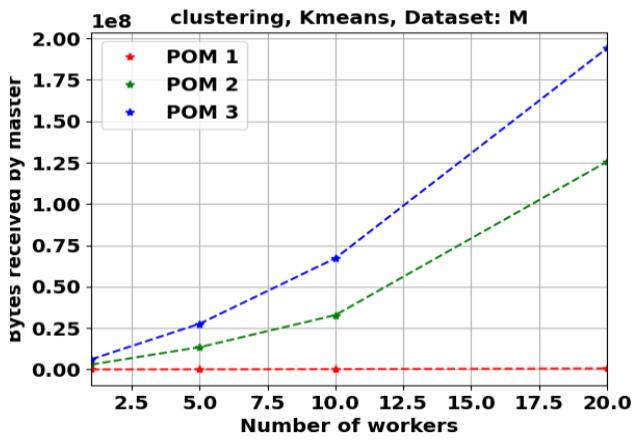
However, in POM1 and POM2, the master node broadcast the data, so it only needs to submit the data once to the pycloudmessenger library and the bytes sent by the master node does not depend on the number of workers and remains constant.

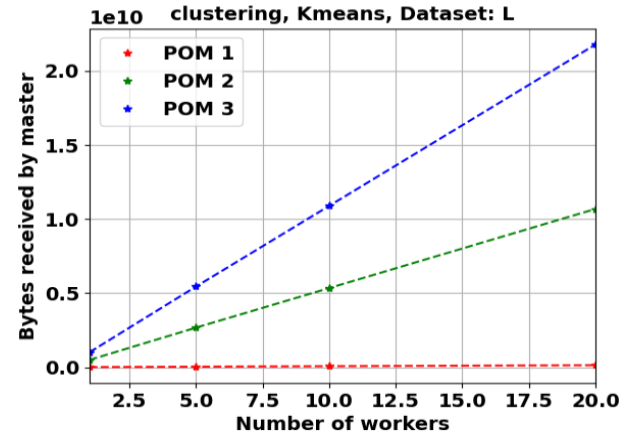
In the case of POM3, the training process is sequential with the number of workers and the centroids are sent to every worker one by one, for that reason we can observe a linear dependency of the information sent with the number of workers.

POM 2 and POM3 make use of homomorphic encryption (and encrypted centroids take up more memory), for that reason, in training processes with 1 worker, the information sent by the master is lower in POM1 than in POM2 and POM3 and is similar in both methods with encryption.

Bytes received by the aggregator:



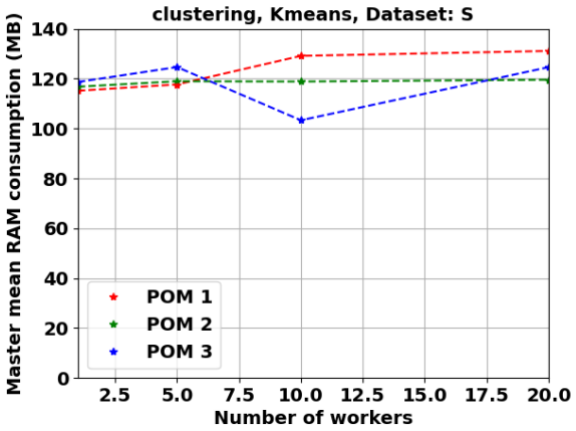


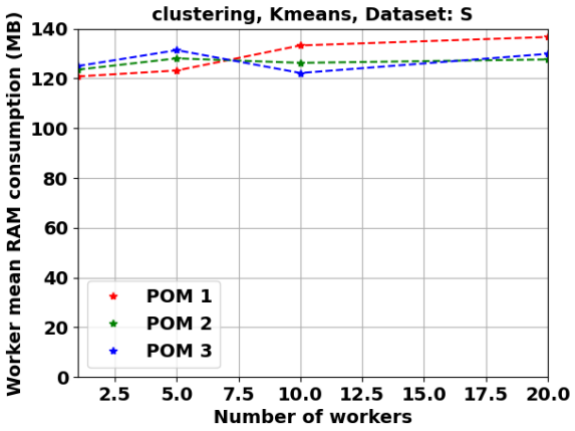


In all the POMs analysed here, in every training iteration, the worker nodes send a copy of their local centroids to the master node. For this reason, we can observe a linear dependency of the information received by the master node with the number of worker nodes.

POM 2 and POM3 make use of homomorphic encryption (and encrypted centroids take up more memory). For this reason, the slope of the linear dependency is higher in POM3 and POM2 than in POM1 and when a single worker is running, the information received by the master is lower in POM1 than in POM2 and POM3.

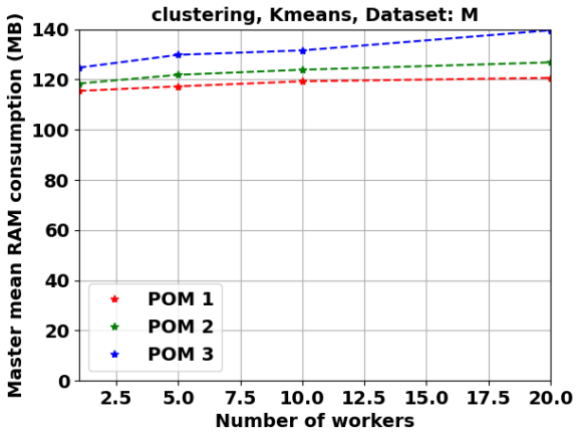
RAM (MB) consumption by master and workers:

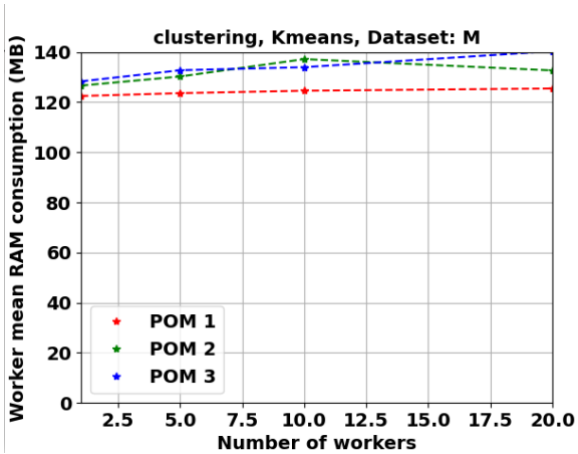




For small datasets, the RAM memory used by the processes is negligible than the RAM memory used by the complete container.

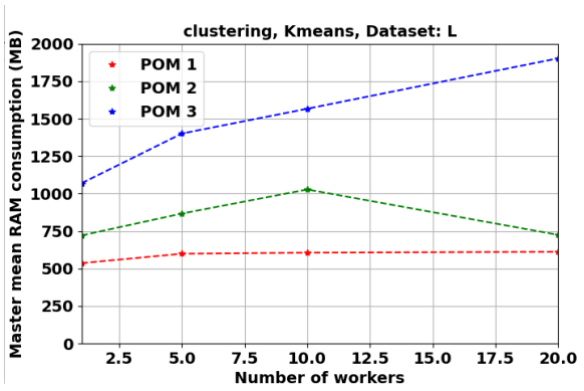
The influence of number of workers and the POM cannot be observed.

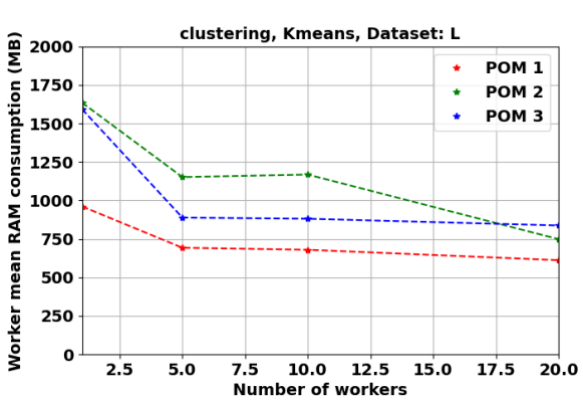




For medium size datasets, the RAM memory used by the processes is still more or less negligible than the RAM memory used by the complete container.

Since POM1 does not make use of encrypted information in memory, we can observe a lower memory consumption.





For large size datasets, we can observe the influence of the POM and dataset size in the memory consumption.

In POM1 we can observe a lower memory consumption (no encryption has been used).

Since we split the dataset among the different workers, the memory consumption in every worker decreases as we increase the number of workers.

The master node of POM3 receives the centroids of every worker, so the memory increases as we increase the number of worker nodes.

5.1.2 Binary classification

Binary classification is a process or task of classification, in which a given data is being classified into two classes. It's basically a kind of prediction about which of two groups the thing belongs to.

Algorithms implemented: Neural Networks, DSVM, FBSVM

Goal: To analyse the scalability, we have executed the algorithms K-means for the three different federated learning POMs (1, 2 and 3) using different number of workers and dataset sizes (different number of samples and features) and we have measure different performance metrics to study how the POM, dataset and number of workers influence in these metrics.

Performance metric: Accuracy.

Dataset used:

- Diabetes(Small size)
- Adult (Medium size)
- MNIST (Large size)

5.1.2.1 Neural Networks

We have used neural networks with an input layer that contains a number of neurons equal to the number of input features, two hidden layers and every hidden layer contains a number of neurons equal to two times the number of features and an output layer that contains a single neuron.

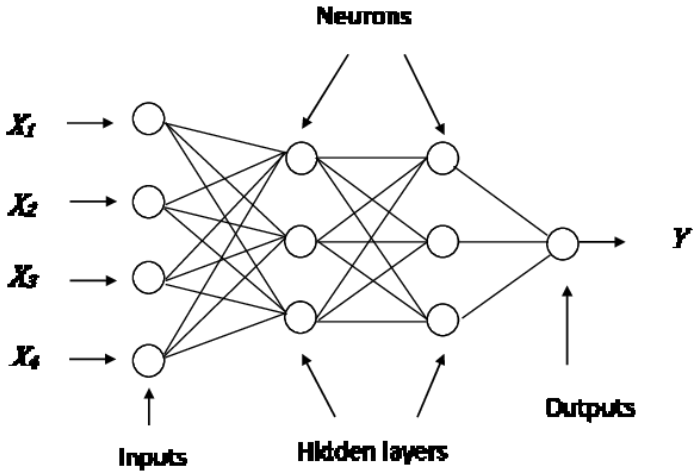


Figure 4. Neural networks (binary)

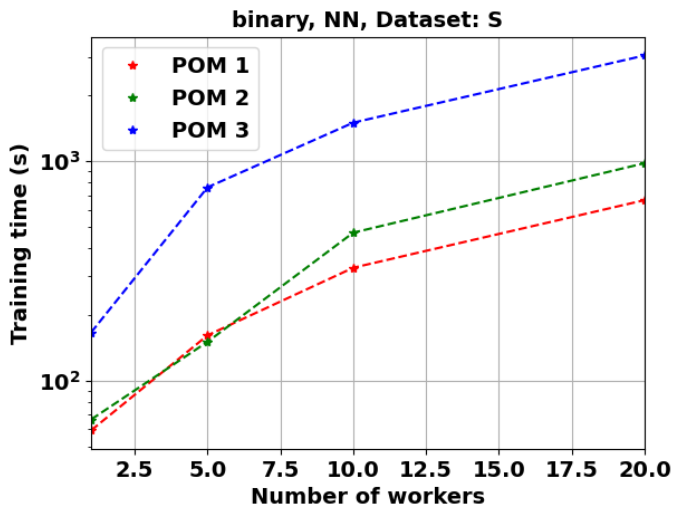
For this reason, the number of weights for every dataset is:

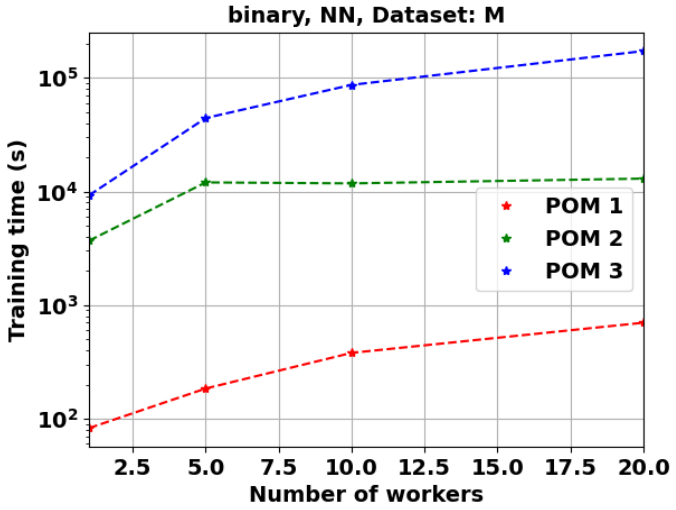
- Small dataset: 8x16 + 16x16 + 16x1 = 400 weights
- Medium dataset: 123x246 + 246x246 + 246x1 = 91020 weights
- Large dataset: 4x8 + 8x8 + 8x3 = 3708348 weights

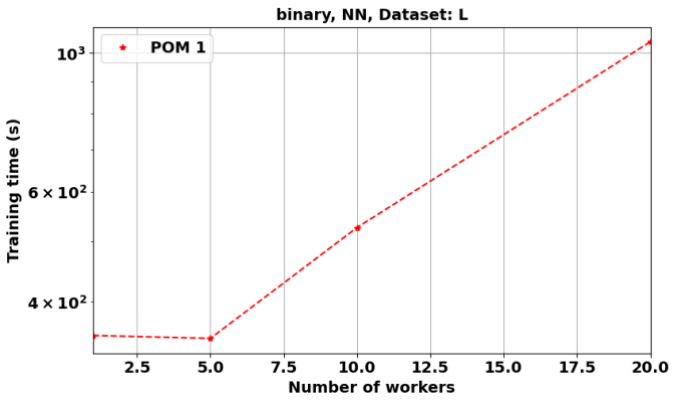
The federated training procedure has been model average, using an Adam optimizer in every worker. The step in the Adam optimizer and the number of training iterations have been obtained using validation.

For the large dataset, only POM1 could be used since the excessive training time in POM 2 and 3 associated to the encryption and decryption of millions of weights (the training procedure could take weeks).

Training time:





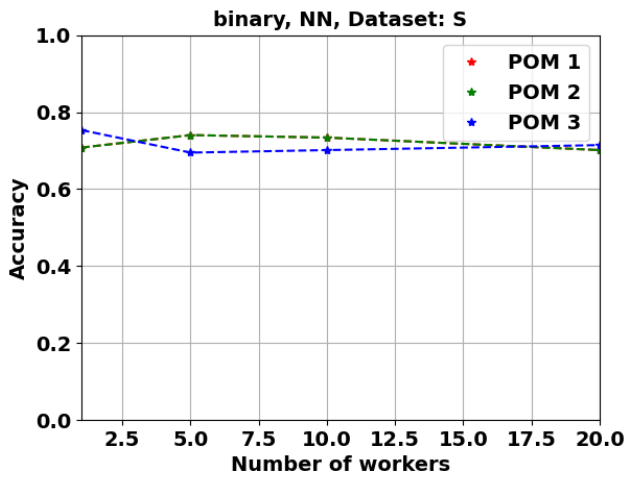


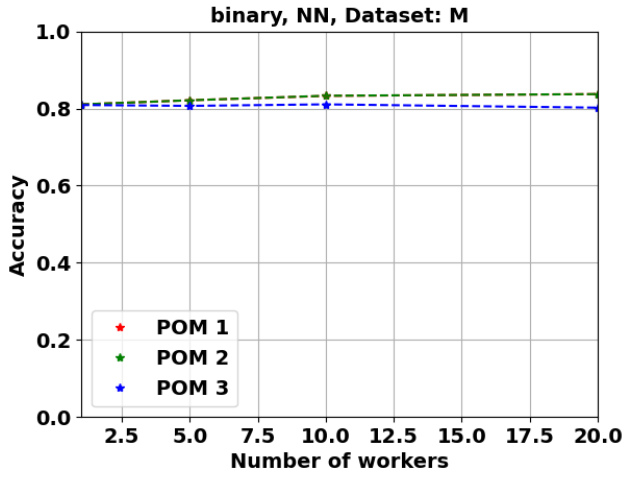
For small datasets, the training time of POM1 and POM2 is comparable.

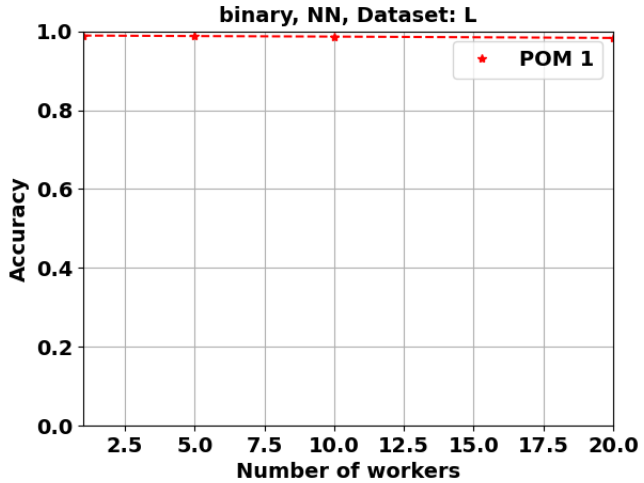
However, as we increase the dataset (and also the size of the neural network that contains two hidden layers that depends on the number of input features), the training time of POM2 grow up.

For large datasets, POM 1 is the only privacy operation mode capable to address the training process in a reasonable amount of time.

Performance metric:



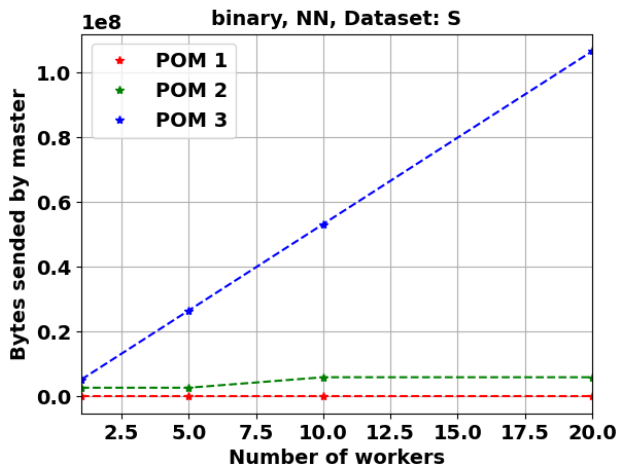


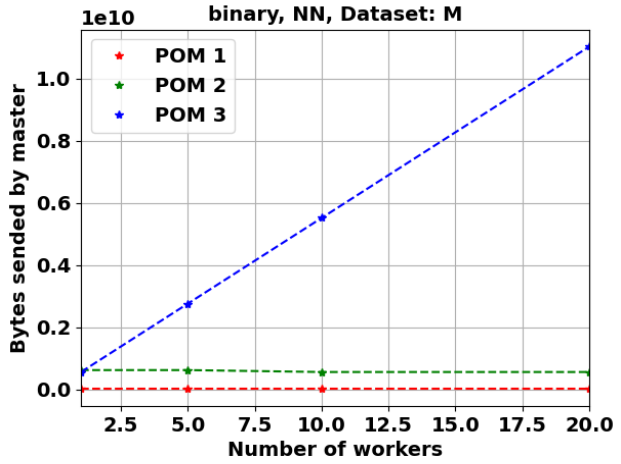


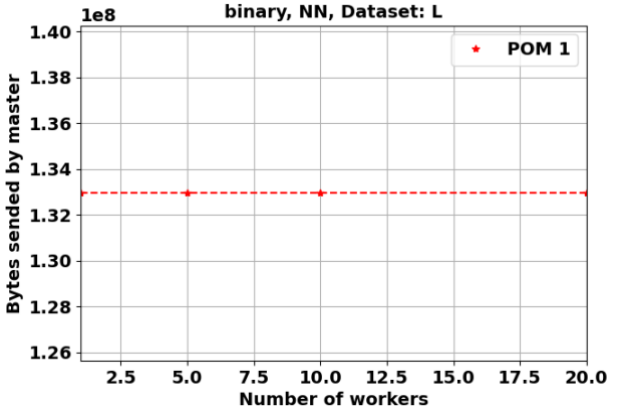
In a Neural Network, the random initialization of the weights and also the random shuffle of data along the training process can influence the final accuracy of the predictive model.

For small datasets, this effect can be appreciated, different executions obtain ML models with different accuracy. This effect is reduced as we increase the dataset size.

Bytes sent by the aggregator:





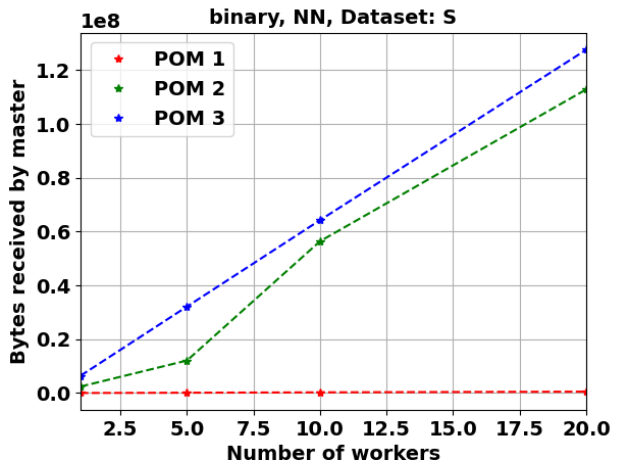


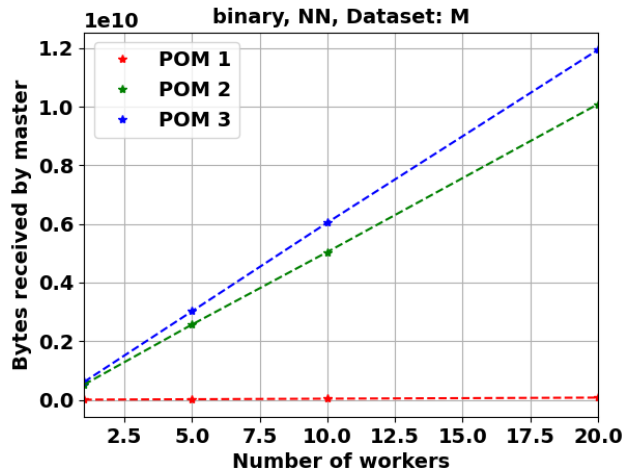
In POM1 and POM2 the information remains constant with the number of worker nodes. The master node broadcast the data, so it only needs to submit the data once to the pycloudmessenger library and the bytes sent by the master node does not depend on the number of workers and remains constant.

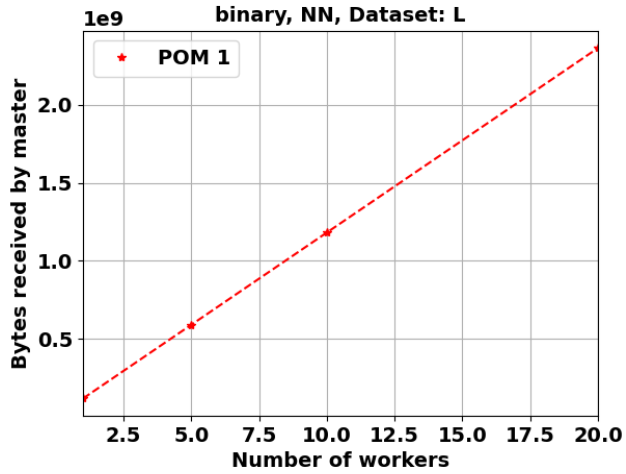
In the case of POM3, the training process is sequential with the number of workers and the centroids are sent to every worker one by one, for that reason we can observe a linear dependency of the information sent with the number of workers.

POM 2 and POM3 make use of homomorphic encryption (and encrypted data take up more memory), for that reason, in training processes with 1 worker, the information sent by the master is lower in POM1 than in POM2 and POM3 and is similar in both methods with encryption.

Bytes received by the aggregator:



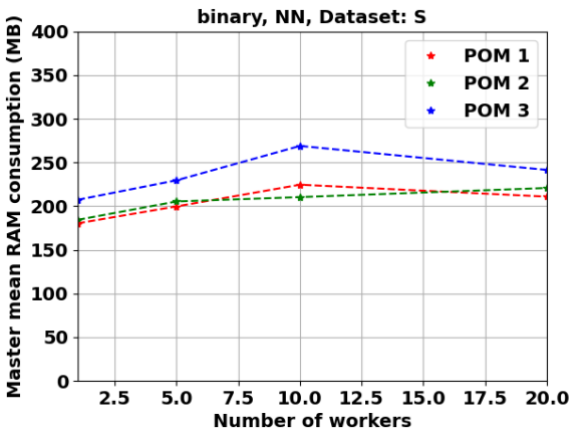


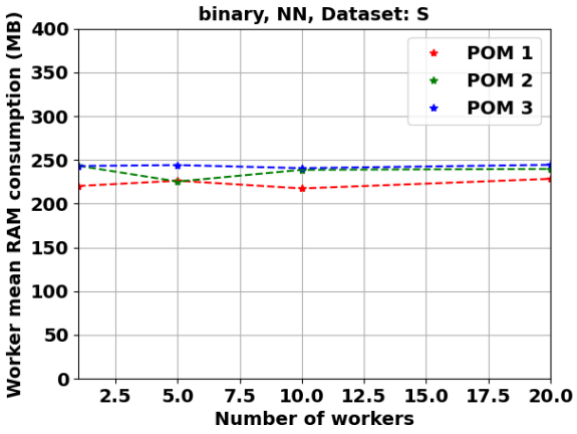


After every training iteration, the worker nodes send a copy of their local centroids to the master node. For this reason, we can observe a linear dependency of the information received by the master node with the number of worker nodes.

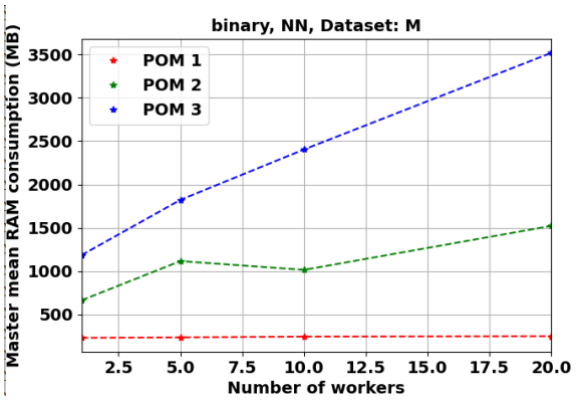
POM 2 and POM3 make use of homomorphic encryption (and encrypted data take up more memory). For this reason, the slope of the linear dependency is higher in POM3 and POM2 than in POM1 and when a single worker is running, the information received by the master is lower in POM1 than in POM2 and POM3.

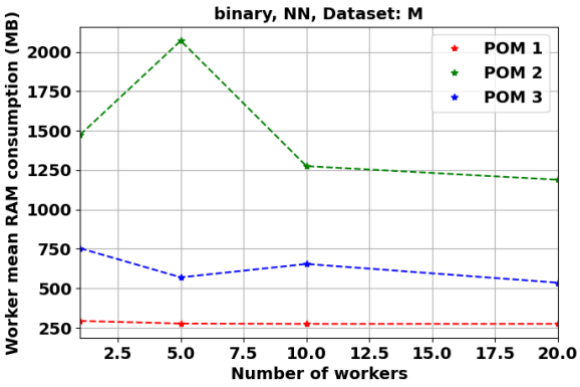
RAM (MB) consumption by master and workers:





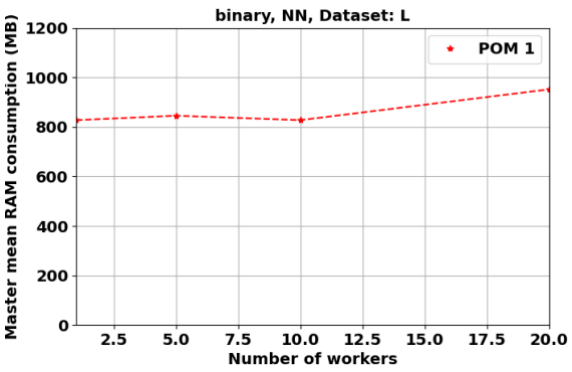
For small datasets, the influence of RAM consumption can not be observed because it is negligible in the RAM memory used by the complete Docker container.

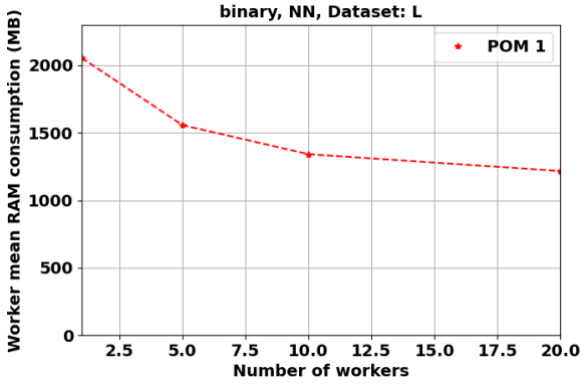




Since POM1 does not make use of encrypted information in memory, we can observe a lower memory consumption.

In POM3, the memory consumption in the master node is higher and increases with the number of workers because it receives a copy of the ML model from every worker in with different encryption keys.





Since we split the dataset among the different workers, the memory consumption in every worker decreases as we increase the number of workers.

5.1.2.2 **DSVM**

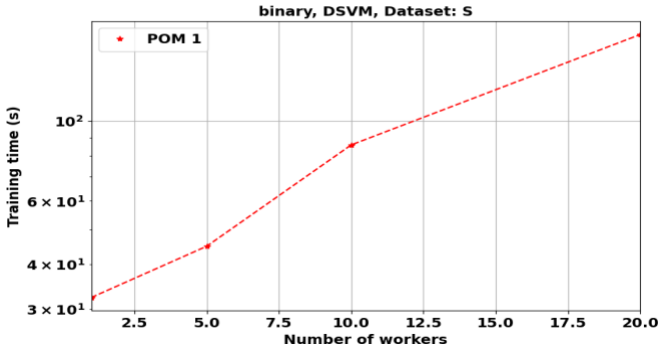
A gaussian kernel has been used. The kernel parameter and the number of centroids have been obtained using cross validation.

The number of centroids used is:

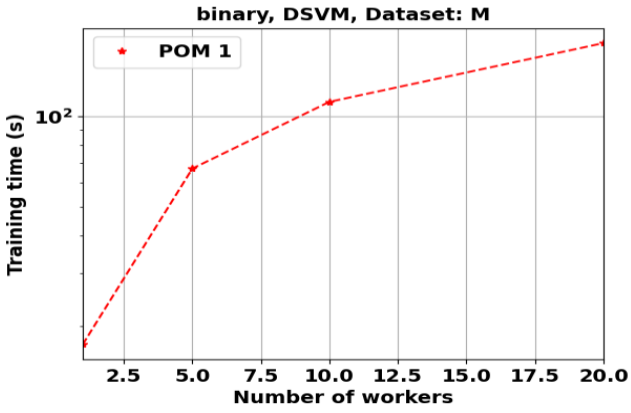
- Small dataset: 100 centroids
- Medium dataset: 100 centroids
- Large dataset: 1500 centroids

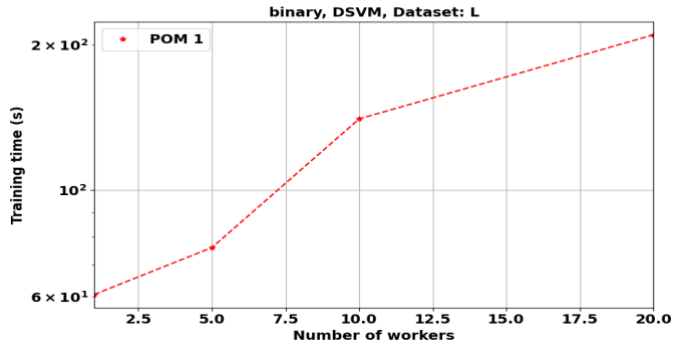
DSVM has been just implemented for POM1.

Training time:

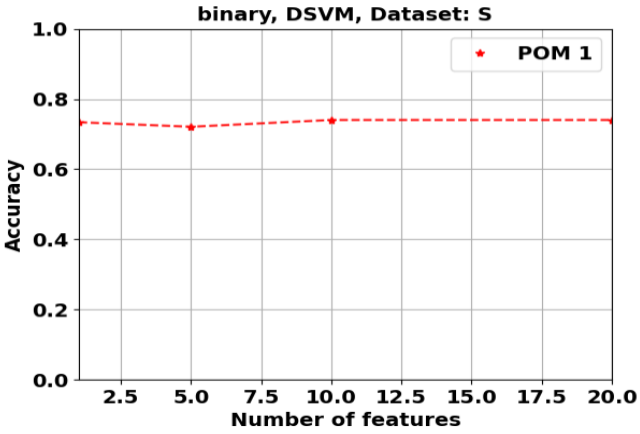


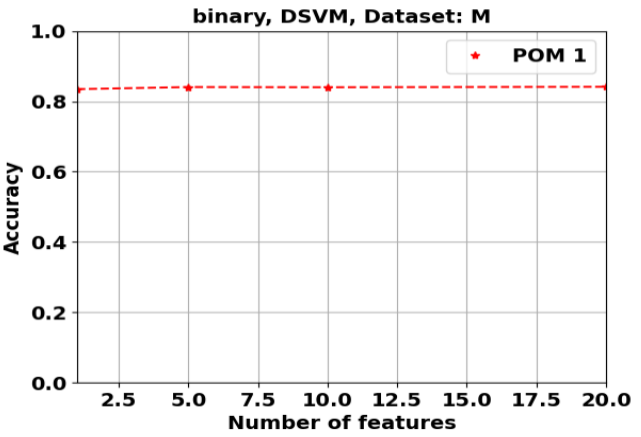
The training time increases with the number of workers due to the bottleneck associated to receive the information from every worker after every iteration.

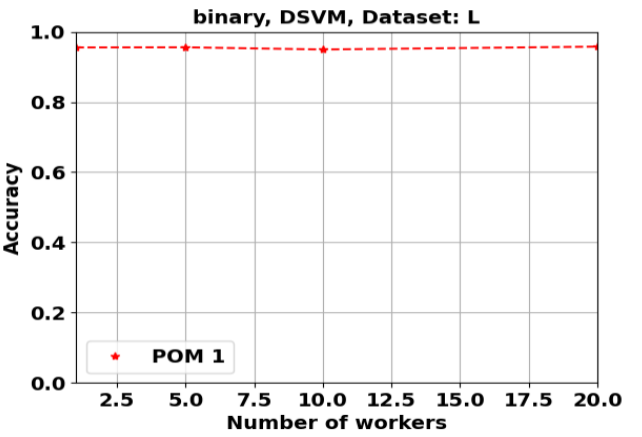

binary, DSVM, Dataset: M
POM 1
Training time (s)
10^2
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Number of workers


binary, DSVM, Dataset: L
POM 1
Training time (s)
2×10^2
10^2
6×10^1
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Number of workers

Performance metric:







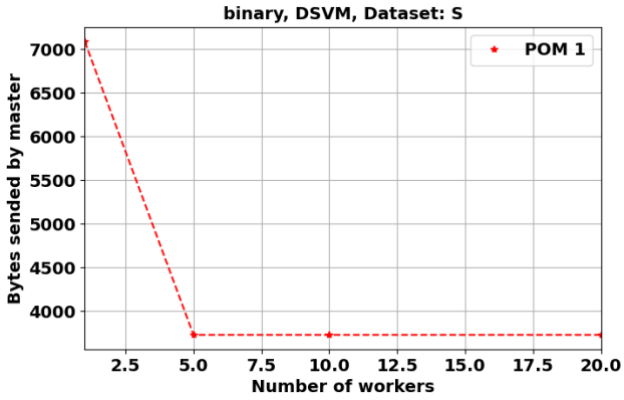
The accuracy obtained with DSVM is very stable. The number of workers does not affect the predictive model.

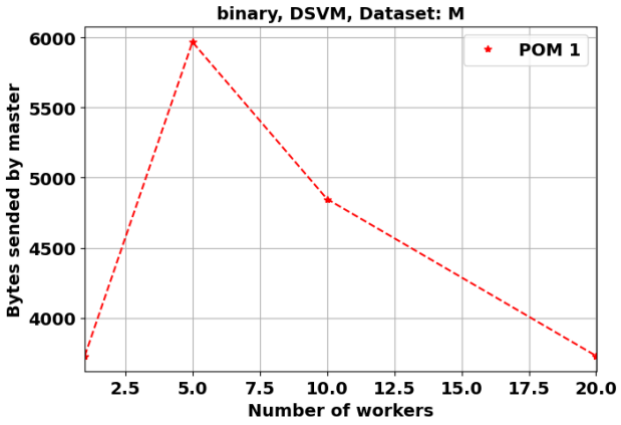
This is the main advantage of this training procedure. Other approaches such as model averaging in NN or FBSVM can be affected in non-iid problems.

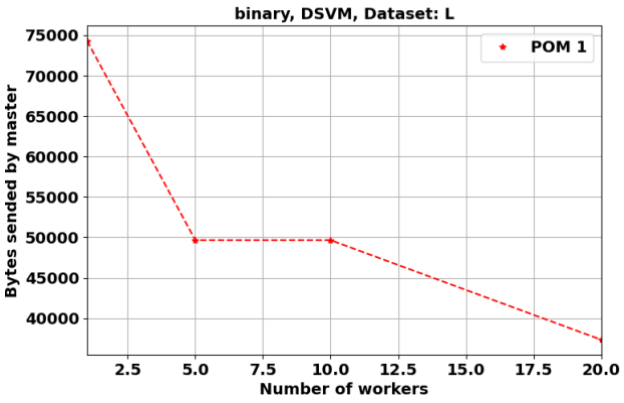
The accuracy obtained with DSVM has been higher than the obtained with FBSVM.

In the case of the large dataset, the accuracy of DSVM is lower than then deep neural network (that is the state of art in many large scale problems).

Bytes sent by the aggregator:



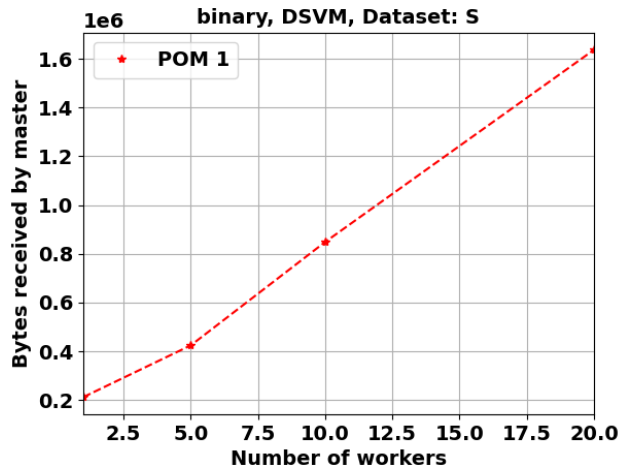


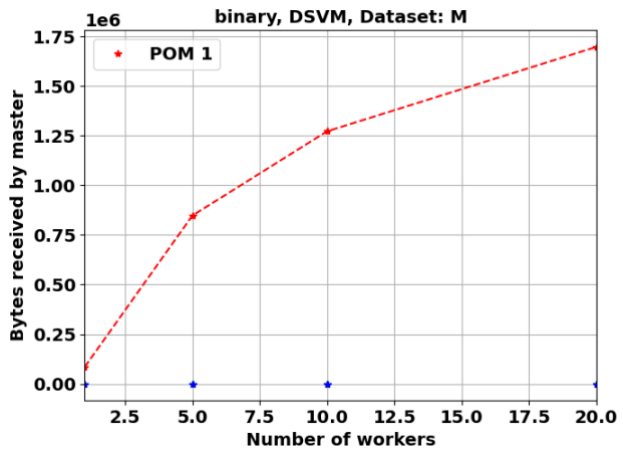


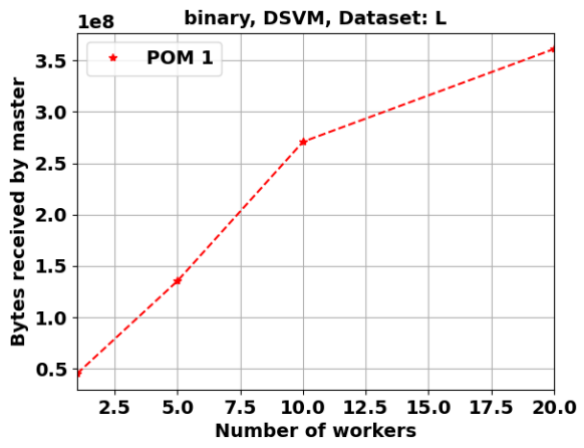
The number of iterations of DSVM is not fixed. Different executions could have different number of iterations.

That is why the information sent by the aggregator is not constant with the number of workers as in other algorithms.

Bytes received by the aggregator:



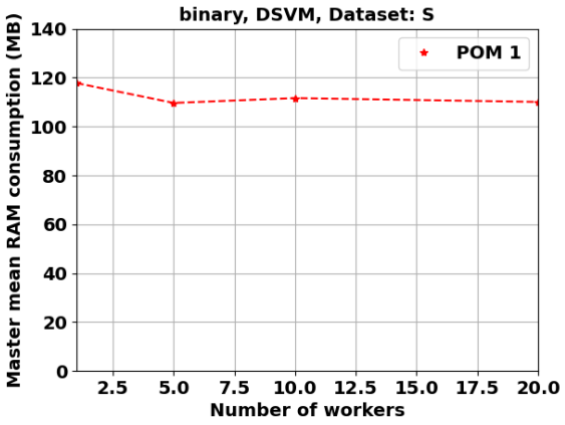


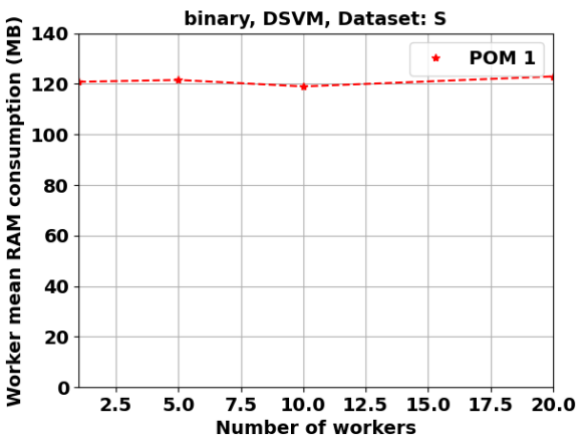


In every training iteration, the worker nodes send a copy of their local centroids to the master node. For this reason, we can observe a linear dependency of the information received by the master node with the number of worker nodes.

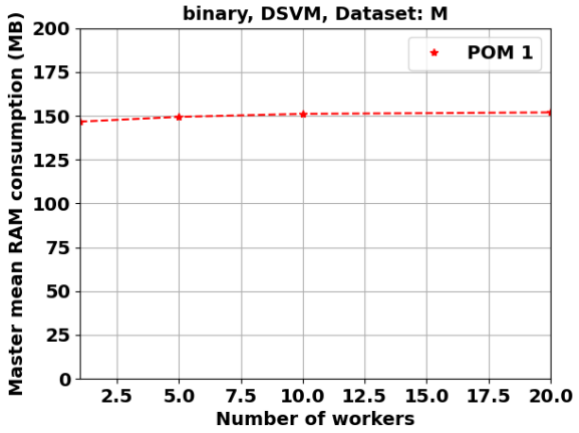
As we increase the number of worker nodes, we increase the information received by the master node.

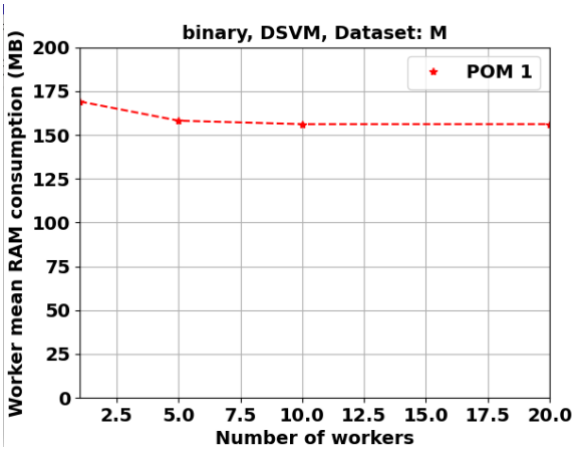
RAM (MB) consumption by master and workers:



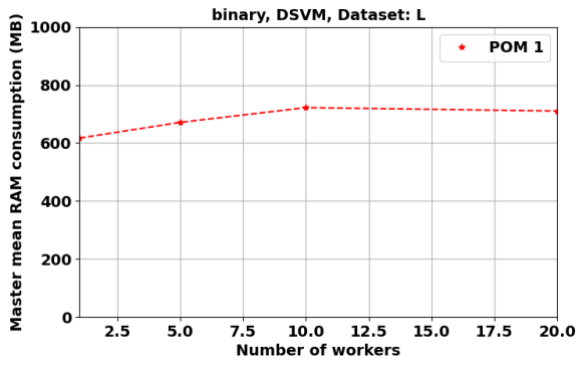


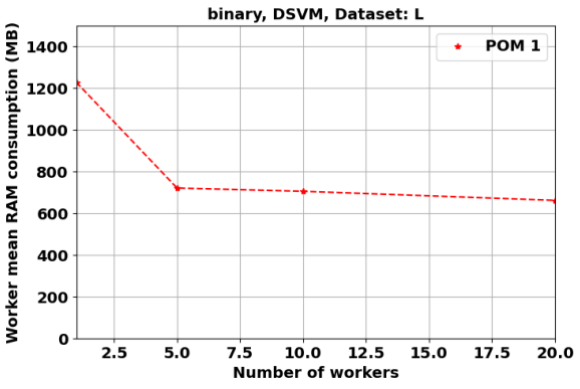
RAM memory used by the processes is negligible in the container. The influence of number of workers and the POM cannot be observed.





We can observe a reduction in the memory of the worker with the number of workers (because the dataset is split among the different workers and as we increase the number of workers, less data contains every worker)





The effect of the memory reduction in the worker is higher for large datasets.

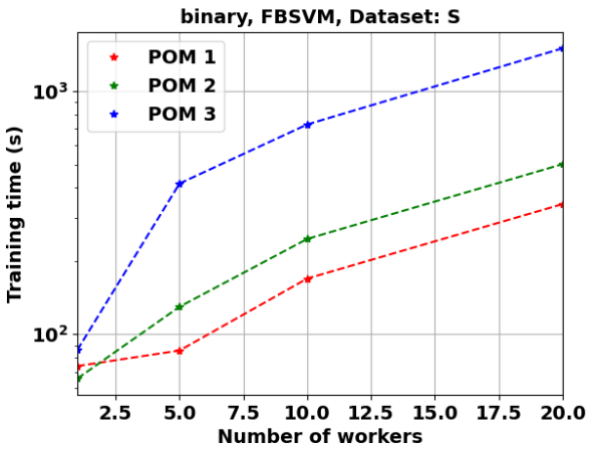
5.1.2.3 **FBSVM**

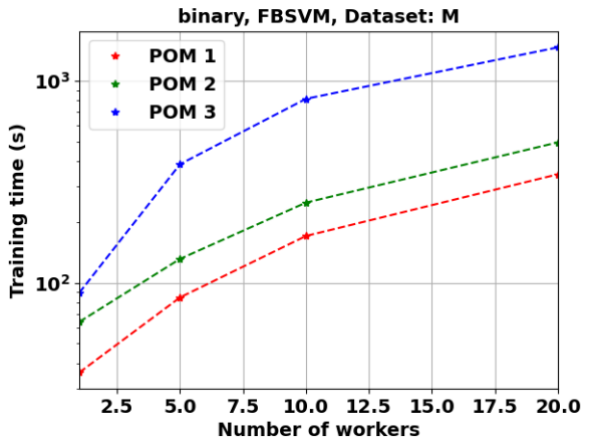
A gaussian kernel has been used. The kernel parameter and the number of centroids has been obtained using cross validation.

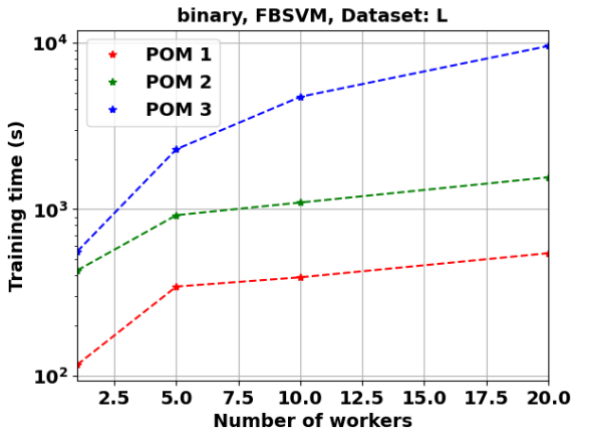
The number of centroids used is:

- Small dataset: 100 centroids
- Medium dataset: 100 centroids
- Large dataset: 1500 centroids

Training time:







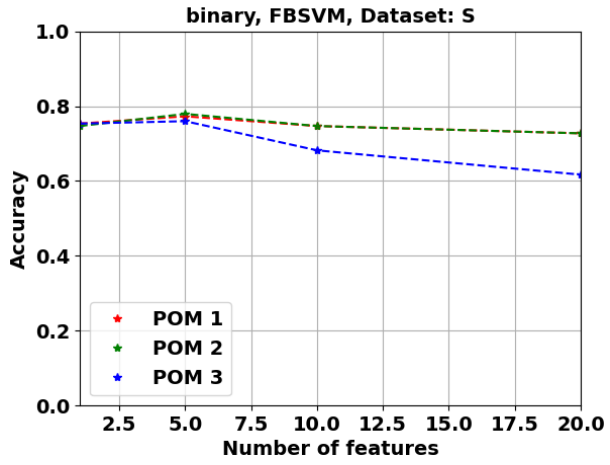
FBSVM is the fastest of the algorithm implemented for binary classification.

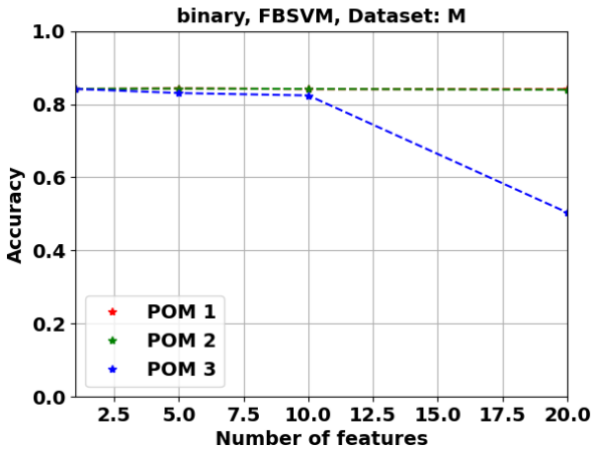
POM1 is the fastest one. It does not need encryption and the master node can broadcast the information to the worker nodes at the same time in every iteration.

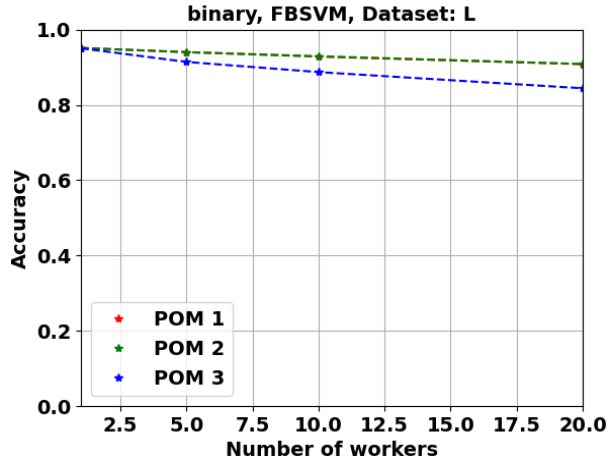
The behaviour of POM2 is similar to POM1 but it needs to encrypt and decrypt the information in every iteration, that is why the training time is higher than in POM1.

POM3 cannot make use of the broadcast command (and the communication is one of the bottlenecks in federated learning). That is why the runtime is higher than POM2.

Performance metric:





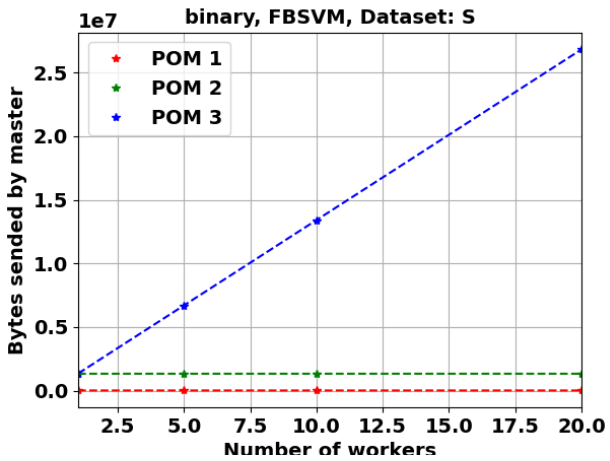


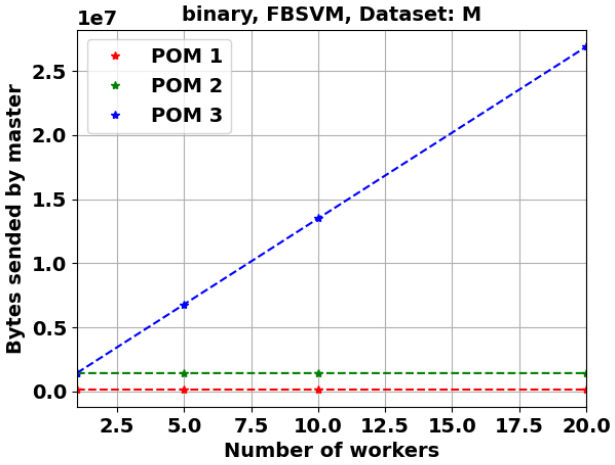
Although FBSVM is faster, the accuracy is not stable.

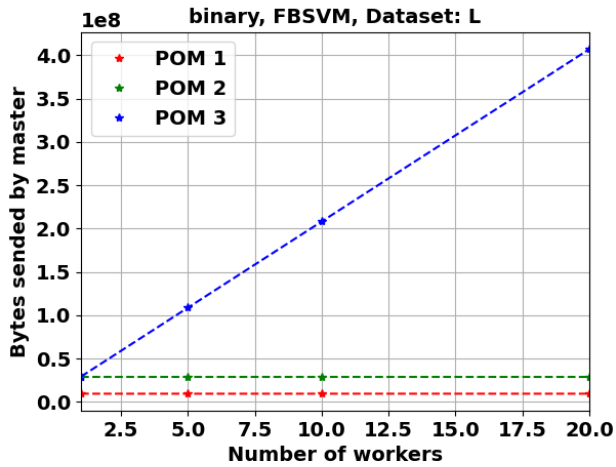
We can appreciate how the accuracy decreases slowly as we increase the number of workers.

This training procedure presents some problems in non-iid datasets.

Bytes sent by the aggregator:







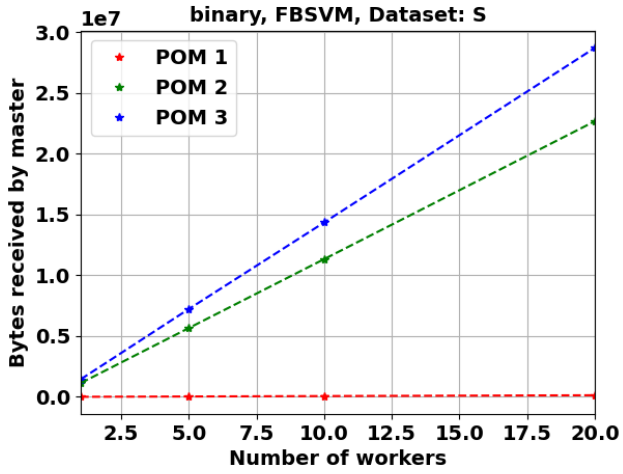
In every iteration of the training process, the aggregator sends the SVM weights to every worker.

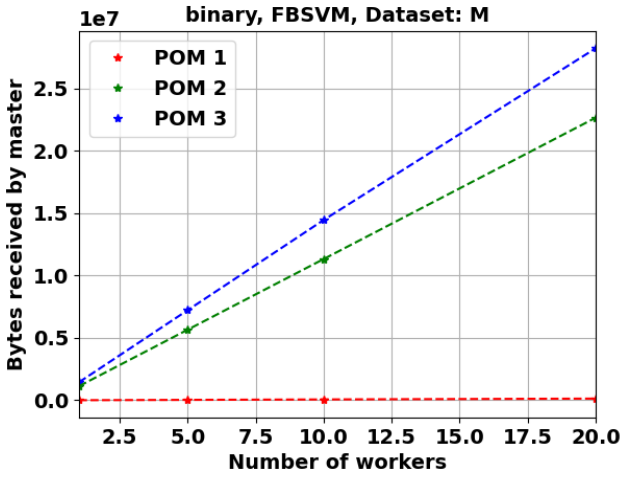
However, in POM1 and POM2, the master node broadcast the data, so it only needs to submit the data once to the pycloudmessenger library and the bytes sent by the master node does not depend on the number of workers and remains constant.

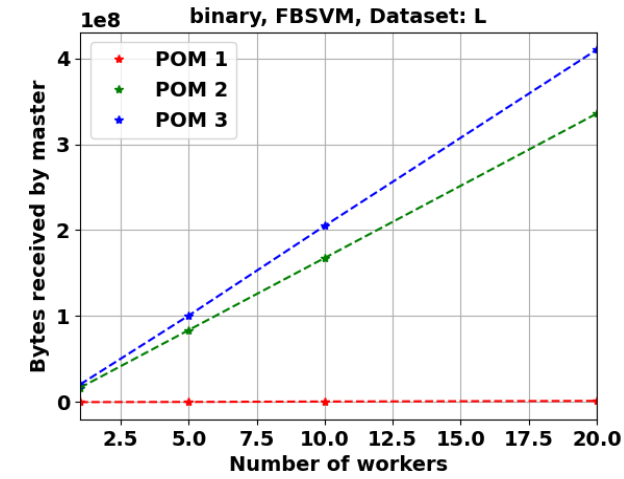
In the case of POM3, the training process is sequential with the number of workers and the SVM weights are sent to every worker one by one, for that reason we can observe a linear dependency of the information sent with the number of workers.

POM 2 and POM3 make use of homomorphic encryption (and encrypted SVM weights take up more memory), for that reason, in training processes with 1 worker, the information sent by the master is lower in POM1 than in POM2 and POM3 and is similar in both methods with encryption.

Bytes received by the aggregator:



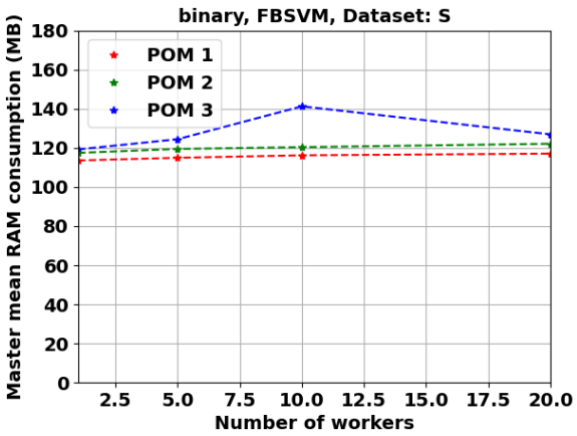


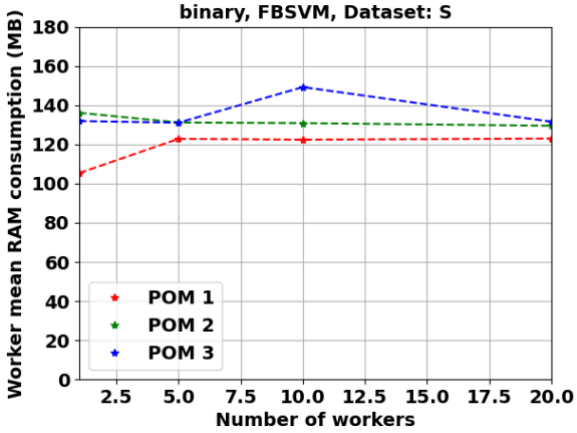


In every training iteration, the worker nodes send a copy of their local SVM weights to the master node. For this reason, we can observe a linear dependency of the information received by the master node with the number of worker nodes.

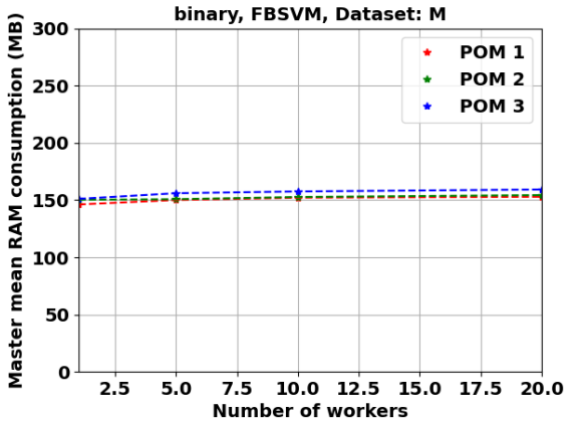
POM 2 and POM3 make use of homomorphic encryption (and encrypted weights take up more memory). For this reason, the slope of the linear dependency is higher in POM3 and POM2 than in POM1 and when a single worker is running, the information received by the master is lower in POM1 than in POM2 and POM3.

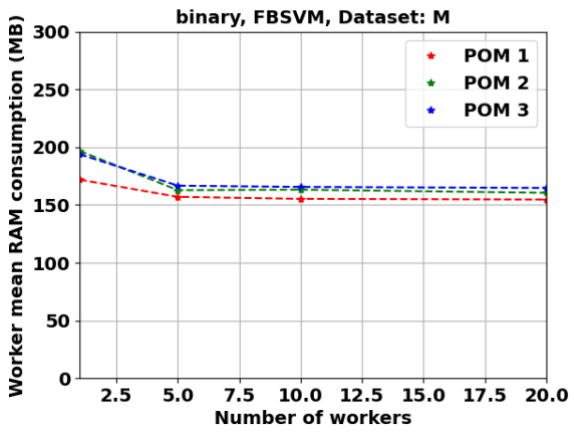
RAM (MB) consumption by master and workers:



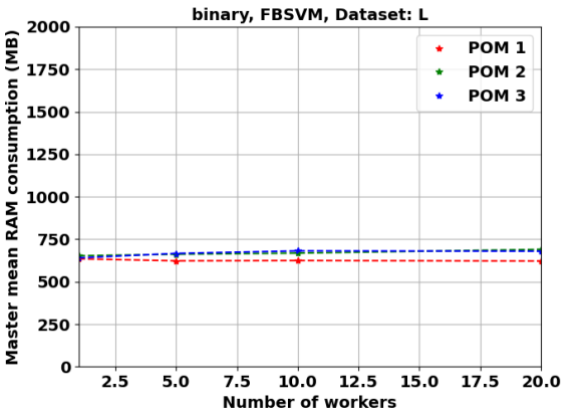


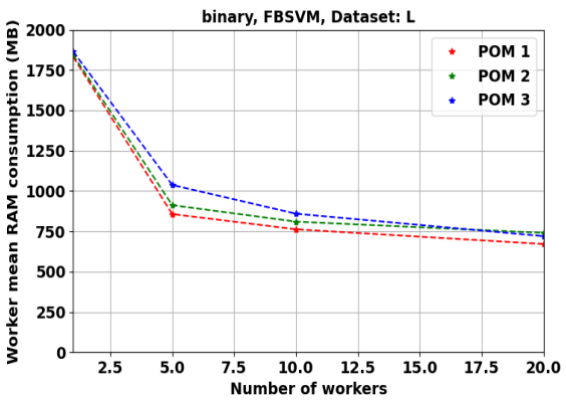
RAM memory used by the processes is negligible in the container. The influence of number of workers and the POM cannot be observed.





The information exchanged among the master and workers is smaller than the other ML models of the platform, that why the memory consumption is similar among the three different POMs (no large matrices are exchanged).





For large size datasets, we can observe the influence of the POM and dataset size in the memory consumption.

The memory consumption in the master node remains constant. However, since we split the dataset among the different workers, the memory consumption in every worker decreases as we increase the number of workers.

The information exchanged among the master and workers is an array containing the weights instead of data matrices, that why the memory consumption is similar among the three different POMs.

5.1.3 Multiclass classification

In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of three or more classes. Multiclass classification makes the assumption that each sample is assigned to one and only one label but not both at the same time.

Algorithms implemented: Neural Networks

Goal: To analyse the scalability, we have executed the algorithms K-means for the three different federated learning POMs (1, 2 and 3) using different number of workers and dataset sizes (different number of samples and features) and we have measure different performance metrics to study how the POM, dataset and number of workers influence in these metrics.

Performance metric: Accuracy.

Dataset used:

- Iris(Small size). 4 features
- Abalone (Medium size) 8 features
- MNIST (Large size) 784 features

5.1.3.1 Neural Networks

We have used neural networks with two hidden layers and every hidden layer contains a number of neurons equal to two times the number of features.

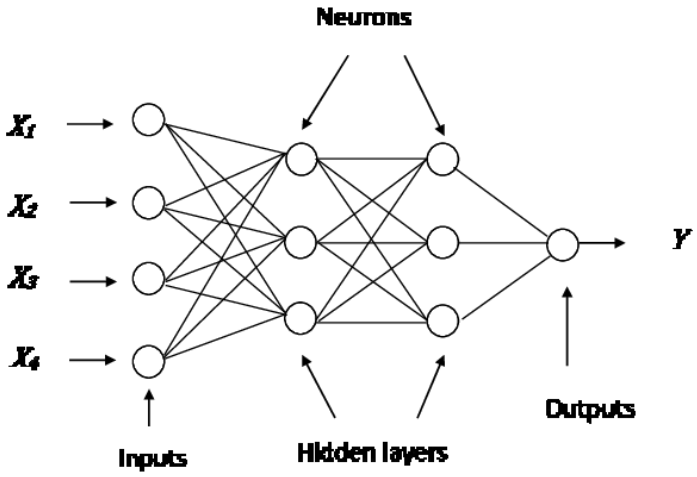


Figure 5. Neural networks (multilclass)

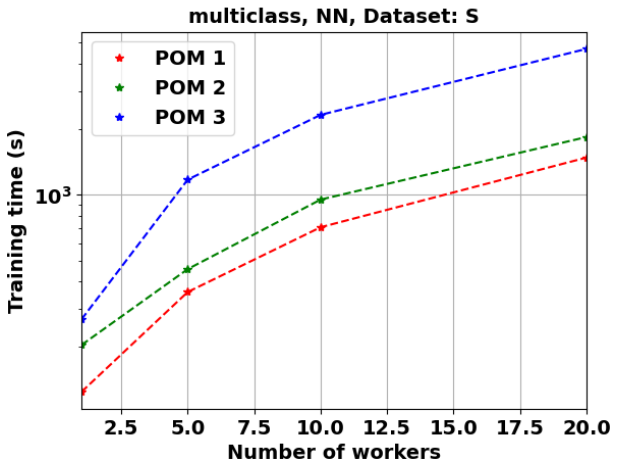
For this reason, the number of weights for every dataset is:

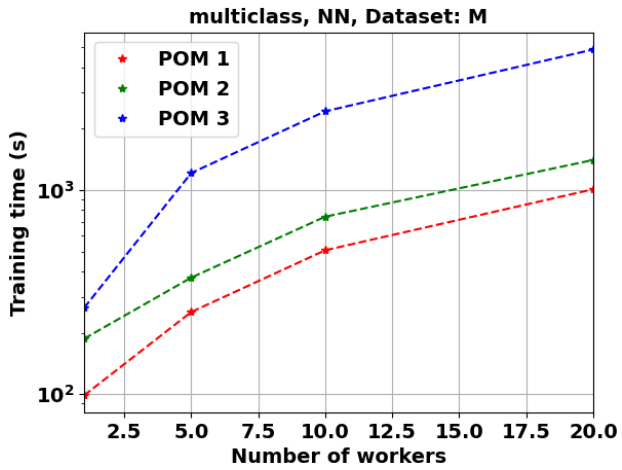
- Small dataset: 4x8 + 8x8 + 8x3 = 120 weights
- Medium dataset: 4x8 + 8x8 + 8x3 = 432 weights
- Large dataset: 4x8 + 8x8 + 8x3 = 3711492 weights

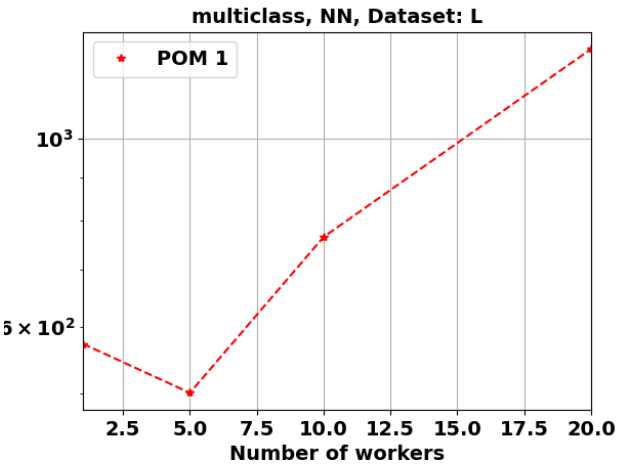
The federated training procedure has been model average, using an Adam optimizer in every worker. The step in the Adam optimizer and the number of training iterations have been obtained using validation.

For the large dataset, only POM1 could be used since the excessive training time in POM2 and POM3 associated to the encryption and decryption of millions of weights (the training procedure could take weeks).

Training time:







The training of POM2 and POM3 takes weeks for the large dataset.

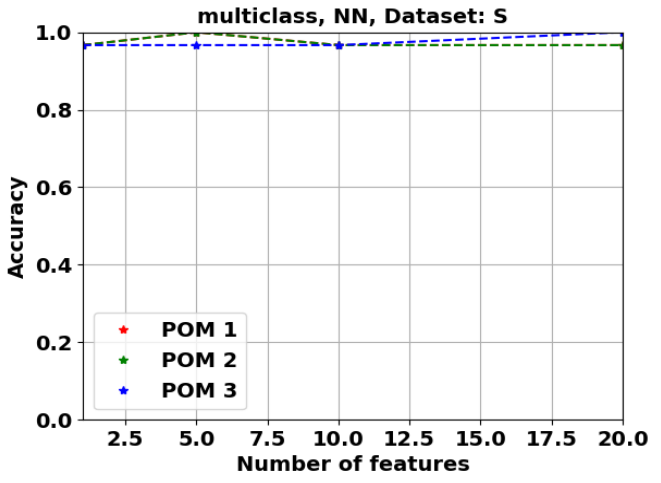
POM1 is obviously the fastest one. It doesn't need encryption and the master node can broadcast the information to the worker nodes at the same time in every iteration. The training time increases with the number of workers due to the bottleneck associated to receive the information from every worker after every iteration.

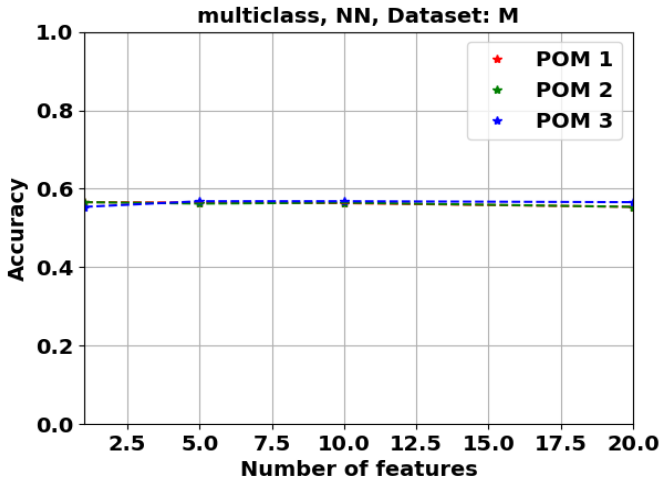
The behaviour of POM2 is similar to POM1 but it needs to encrypt and decrypt the information in every iteration, that is why the training time is higher than in POM1.

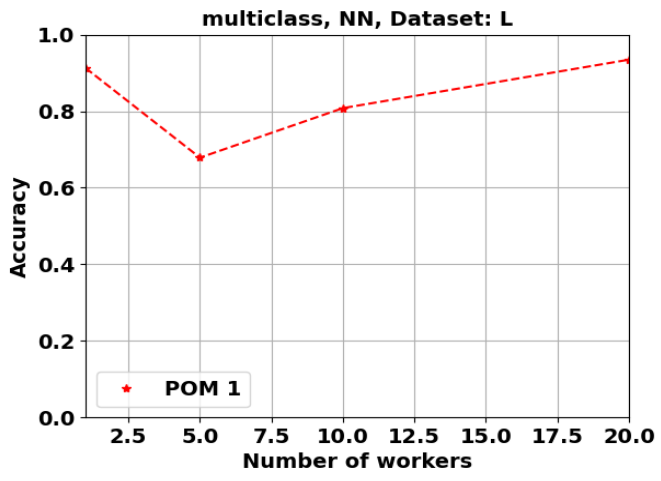
POM3 cannot make use of the broadcast command (and the communication is one of the bottlenecks in federated learning). That is why the runtime is higher than POM2.

Every iteration of the training process is very fast for small datasets. That is why we can observe some strange behaviour in the dependencies (the state of the network influences more the training time than the process itself).

Performance metric:



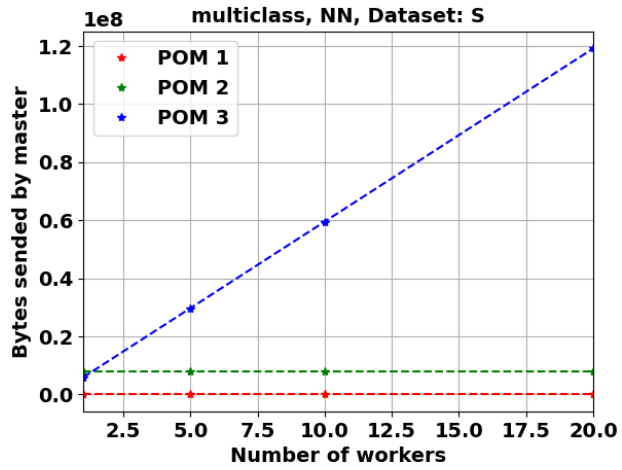


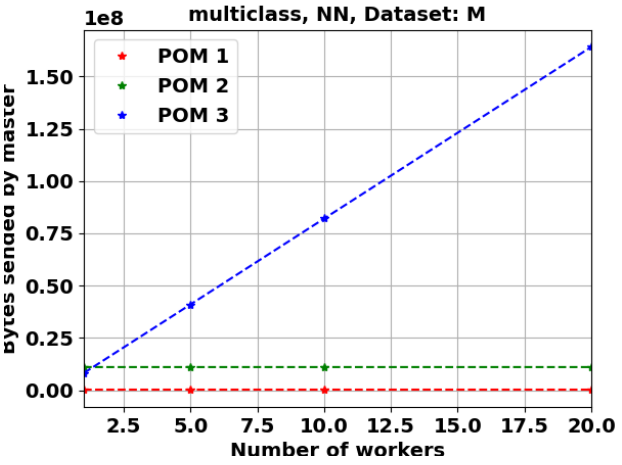


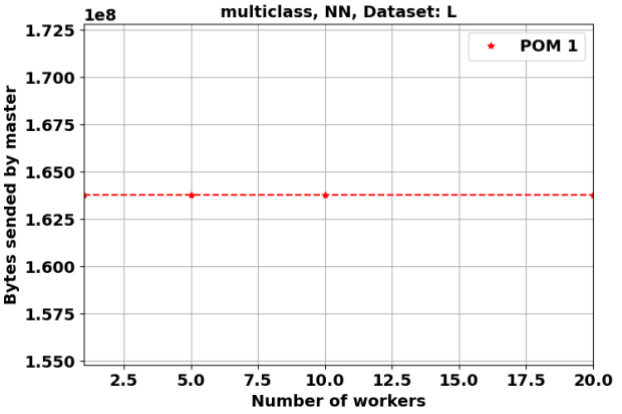
The accuracy obtained in the different POMs is very similar since the operations performed are exactly the same.

The number of training iterations in the federated schema was fixed using a previous validation step. However, sometimes the training procedure needs more training iterations and obtains a lower accuracy (as we can see in the large dataset for POM 1 that is not stable and obtains a low accuracy for 5 workers).

Bytes sent by the aggregator:







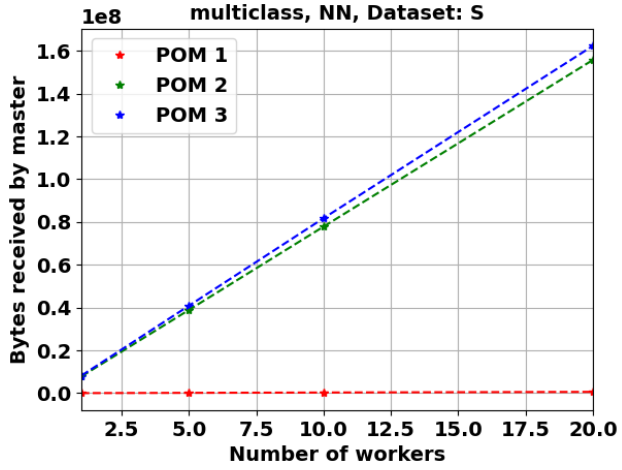
In every iteration of the training process, the aggregator sends the weights of the neural network.

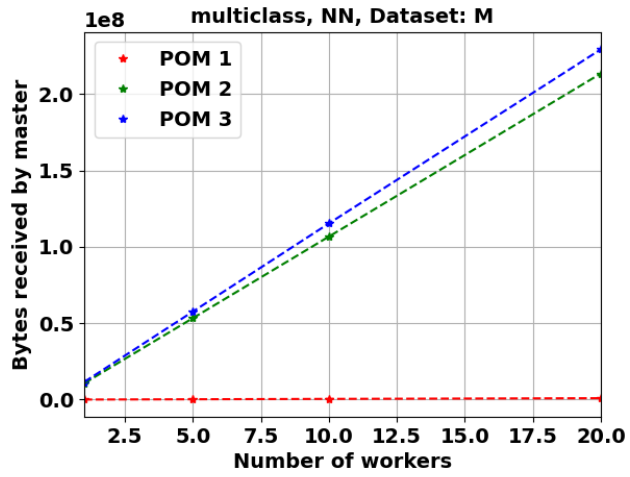
However, in POM1 and POM2, the master node broadcast the data, so it only needs to submit the data once to the pycloudmessenger library and the bytes sent by the master node does not depend on the number of workers and remains constant.

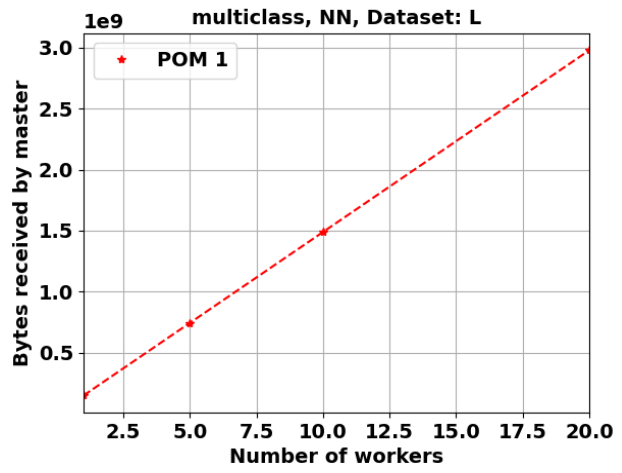
In the case of POM3, the training process is sequential with the number of workers and the weights are sent to every worker one by one, for that reason we can observe a linear dependency of the information sent with the number of workers.

POM 2 and POM3 make use of homomorphic encryption (and encrypted weights take up more memory), for that reason, in training processes with 1 worker, the information sent by the master is lower in POM1 than in POM2 and POM3 and is similar in both methods with encryption.

Bytes received by the aggregator:



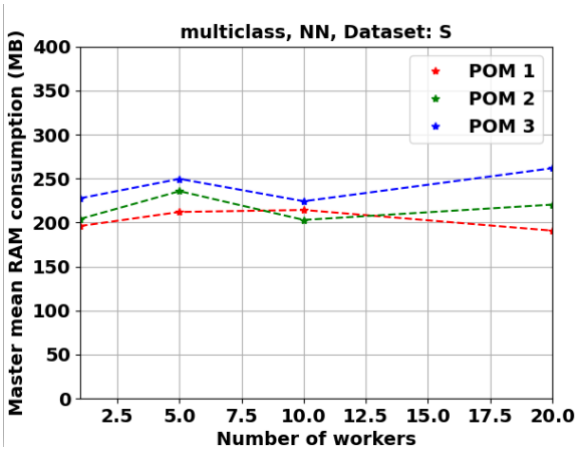


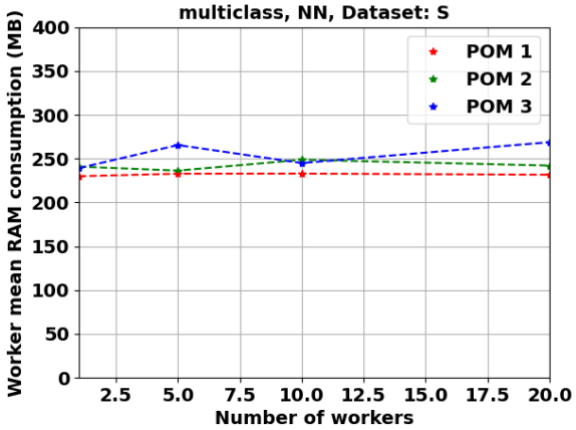


In all the POMs analysed here, in every training iteration, the worker nodes send a copy of their local weights to the master node. For this reason, we can observe a linear dependency of the information received by the master node with the number of worker nodes.

POM 2 and POM3 make use of homomorphic encryption (and encrypted weights take up more memory). For this reason, the slope of the linear dependency is higher in POM3 and POM2 than in POM1 and when a single worker is running, the information received by the master is lower in POM1 than in POM2 and POM3.

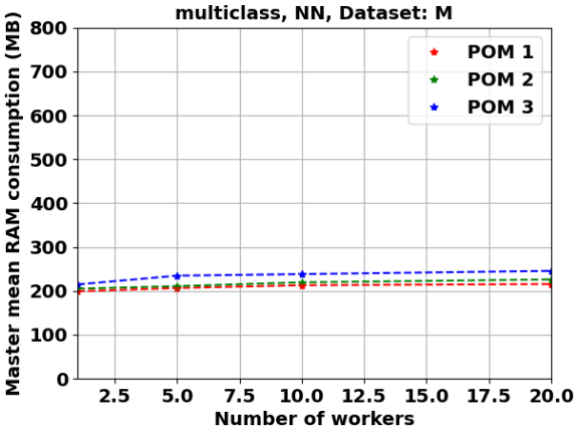
RAM (MB) consumption by master and workers:

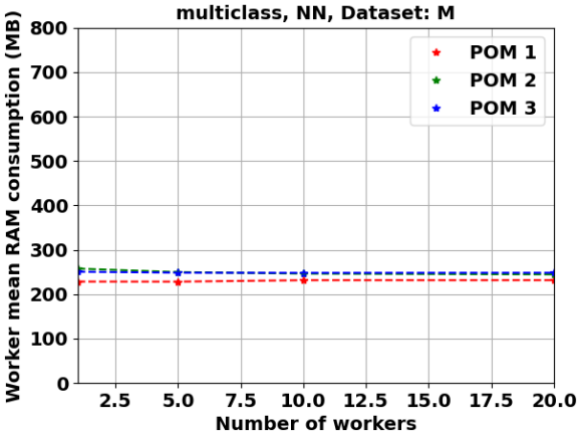




For small datasets, the RAM memory used by the processes is negligible than the RAM memory used by the complete container.

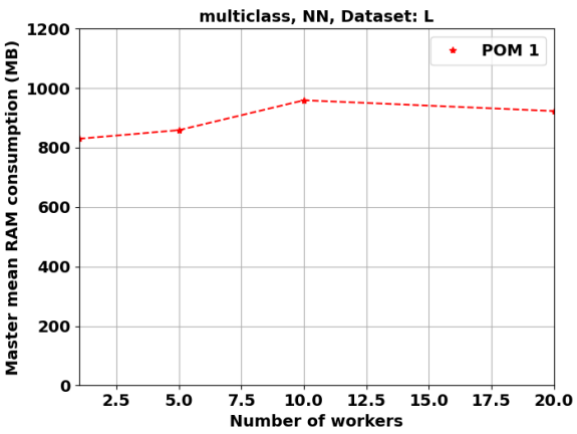
The influence of number of workers and the POM cannot be observed.

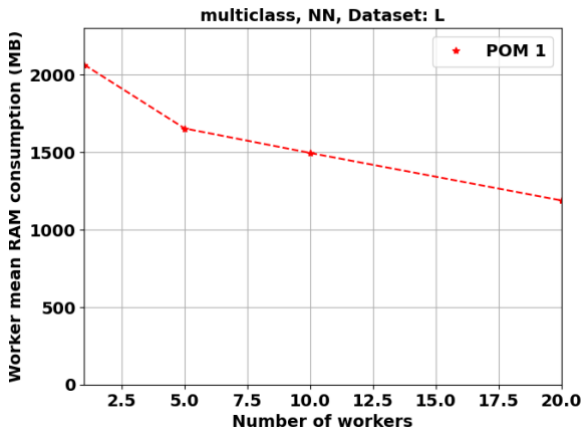




For medium size datasets, the RAM memory used by the processes is still more or less negligible than the RAM memory used by the complete container.

Since POM1 does not make use of encrypted information in memory, we can observe a lower memory consumption.





Since we split the dataset among the different workers, the memory consumption in every worker decreases as we increase the number of workers. However, the memory in the master node remains constant.

5.1.4 Regression

Regression predictive modelling is the task of approximating a mapping function (f) from input variables to a continuous output variable. A continuous output variable is a real-value, such as an integer or floating-point value. These are often quantities, such as amounts and sizes.

Algorithms implemented: Neural Networks

Goal: To analyse the scalability, we have executed the algorithms K-means for the three different federated learning POMs (1, 2 and 3) using different number of workers and dataset sizes (different number of samples and features) and we have measure different performance metrics to study how the POM, dataset and number of workers influence in these metrics.

Performance metric: R^2 score, that is the most extended performance metric for regression problem.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Best possible R^2 score is 1.0 and it can be negative. A constant model that always predicts the expected value of y, disregarding the input features, would get a score of 0.0.

Dataset used:

- Boston Housing (Small size). 13 features
- Superconductivity (Medium size) 81 features
- Blog Data (Large size) 280 features

5.1.4.1 Neural Networks

We have used neural networks with two hidden layers and every hidden layer contains a number of neurons equal to two times the number of features.

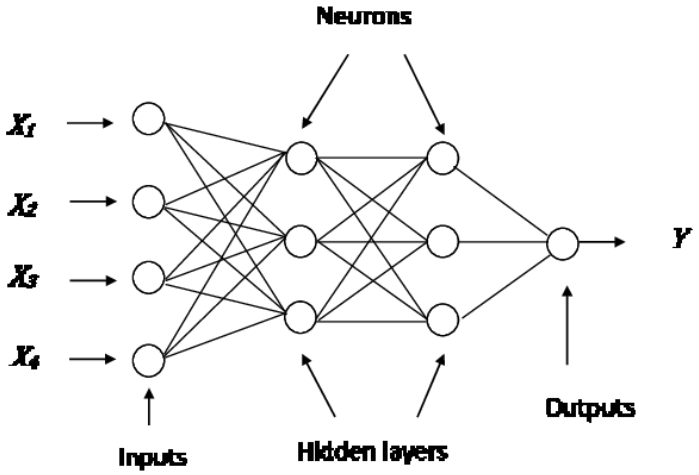


Figure 6. Neural networks (regression)

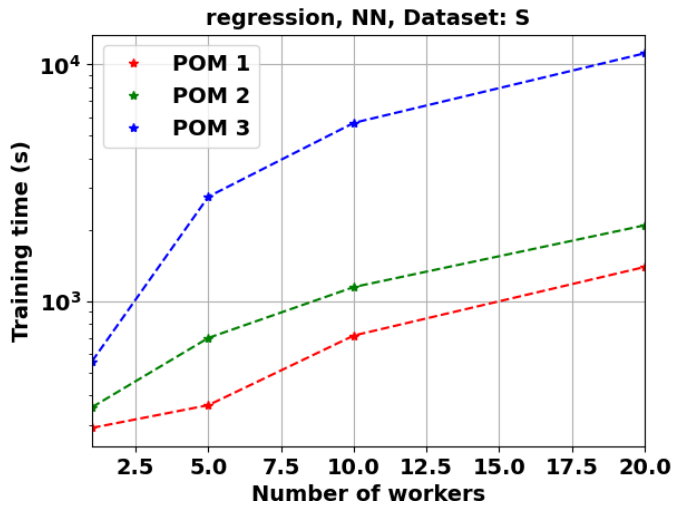
For this reason, the number of weights for every dataset is:

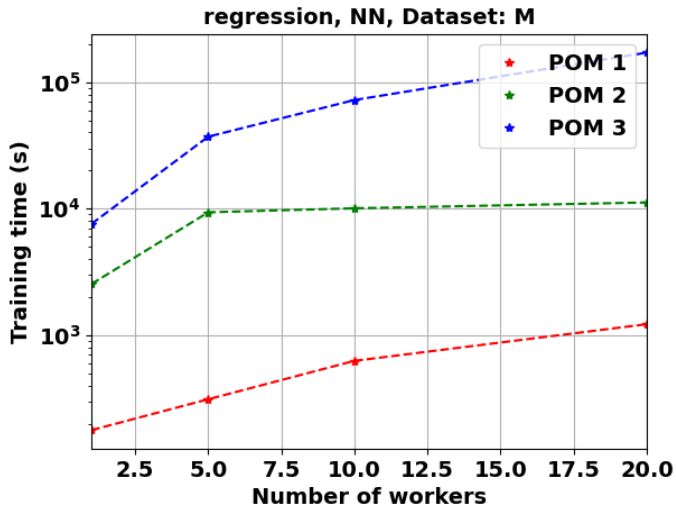
- Small dataset: 13x26 + 26x26 + 26x1 = 1040 weights
- Medium dataset: 81x162 + 162x162 + 162x1 = 39528 weights
- Large dataset: 280x560 + 560x560 + 560x1 = 470680 weights

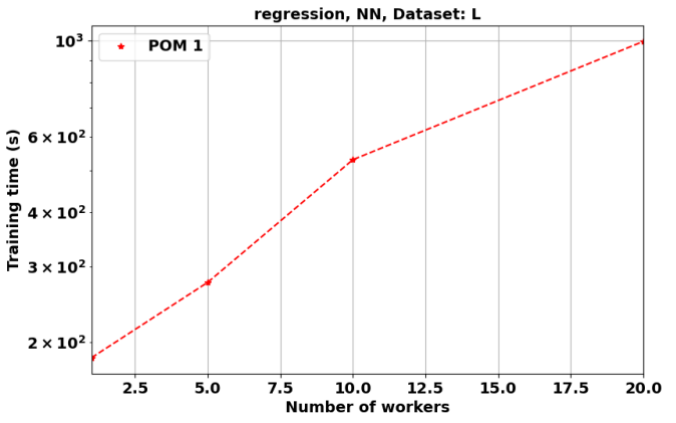
The federated training procedure has been model average, using an Adam optimizer in every worker. The step in the Adam optimizer and the number of training iterations have been obtained using validation.

For the large dataset, only POM1 could be used since the excessive training time in POM2 and POM3 associated to the encryption and decryption of millions of weights (the training procedure could take weeks).

Training time:





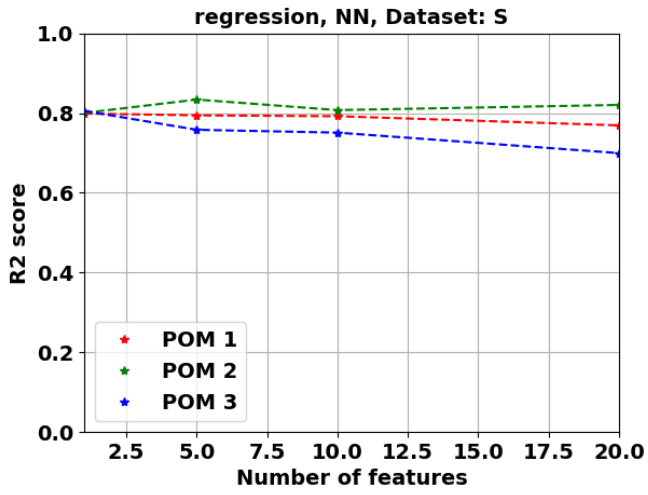


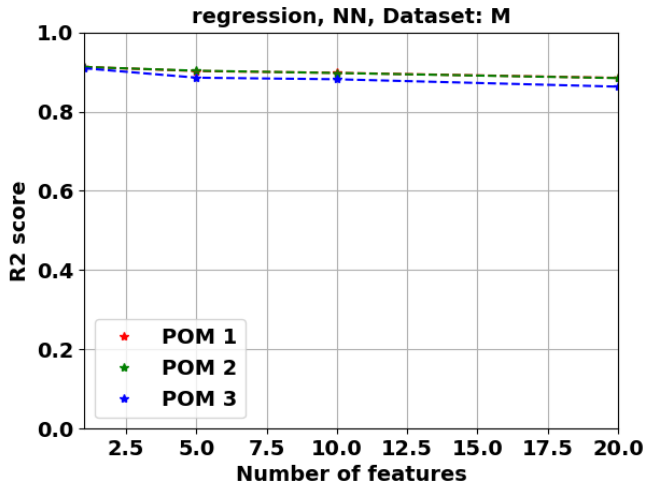
The training of POM2 and POM3 could take weeks for the large dataset. That why the experiments couldn't finish.

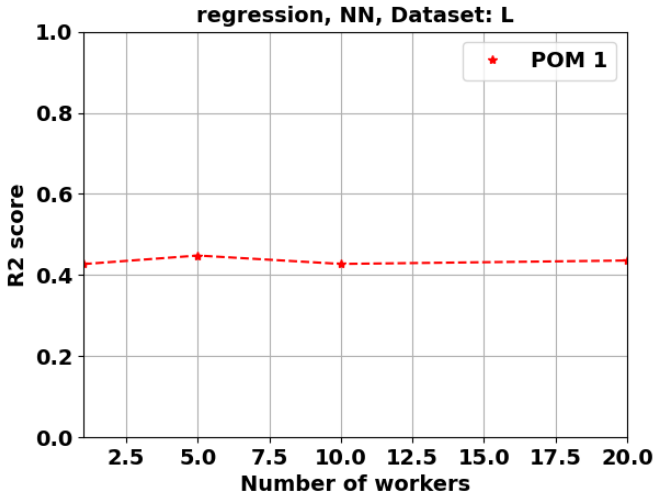
POM3 cannot make use of the broadcast command (and the communication is one of the bottlenecks in federated learning). The training process is sequential worker by worker and that is why the computational cost is higher than POM 1 and 2.

For small datasets, the training time of POM1 and POM2 are similar. As we increase the dataset size, the encryption and decryption of the ML model in POM2 increases the training time.

Performance metric:



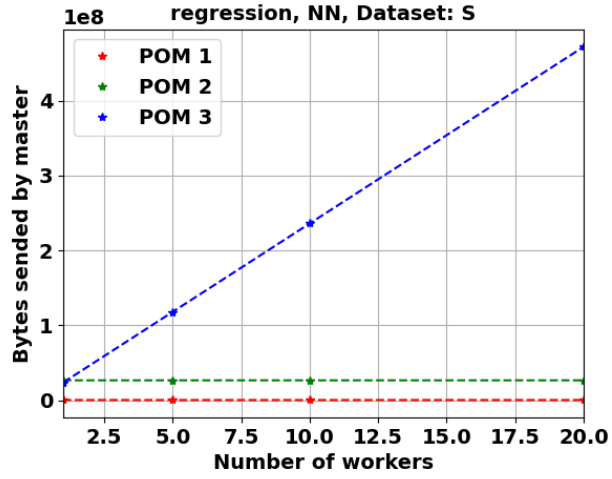


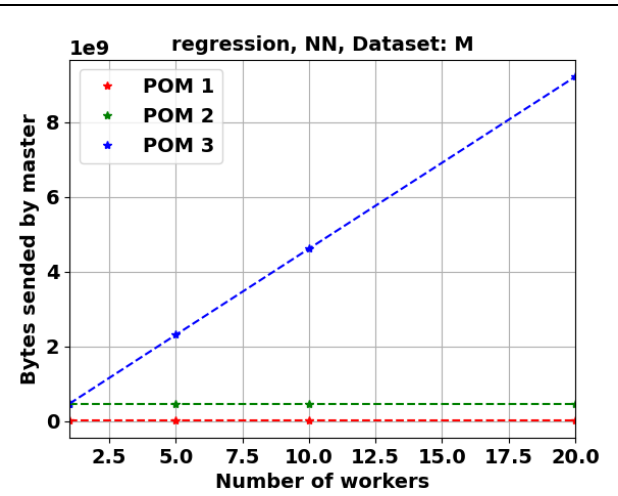


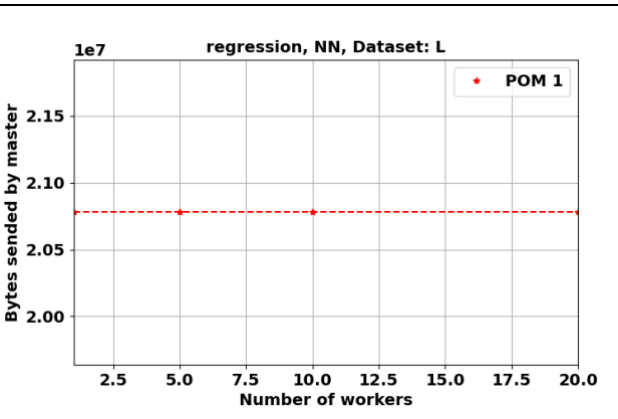
In a Neural Network, the random initialization of the weights and also the random shuffle of data along the training process can influence the final accuracy of the predictive model.

For small datasets, this effect can be appreciated, different executions obtain ML models with different accuracy. This effect is reduced as we increase the dataset size.

Bytes sent by the aggregator:







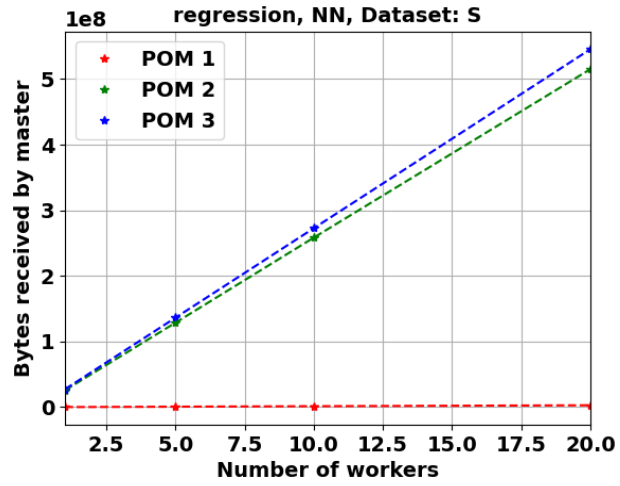
In every iteration of the training process, the aggregator sends the weights of the neural network.

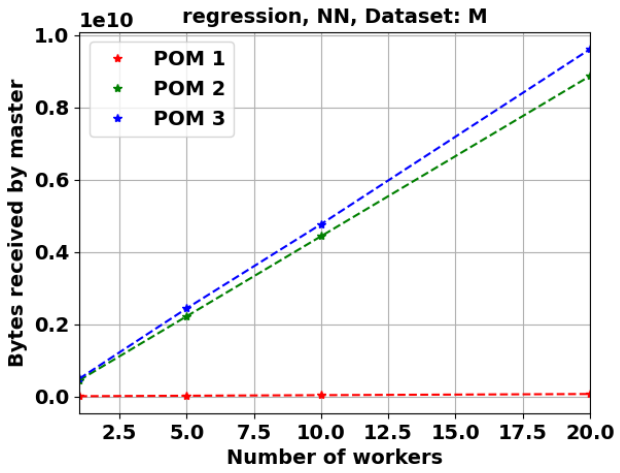
However, in POM1 and POM2, the master node broadcast the data, so it only needs to submit the data once to the pycloudmessenger library and the bytes sent by the master node does not depend on the number of workers and remains constant.

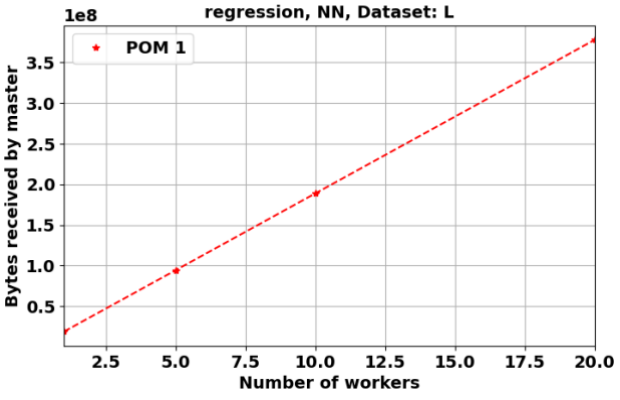
In the case of POM3, the training process is sequential with the number of workers and the weights are sent to every worker one by one, for that reason we can observe a linear dependency of the information sent with the number of workers.

POM 2 and POM3 make use of homomorphic encryption (and encrypted weights take up more memory), for that reason, in training processes with 1 worker, the information sent by the master is lower in POM1 than in POM2 and POM3 and is similar in both methods with encryption.

Bytes received by the aggregator:



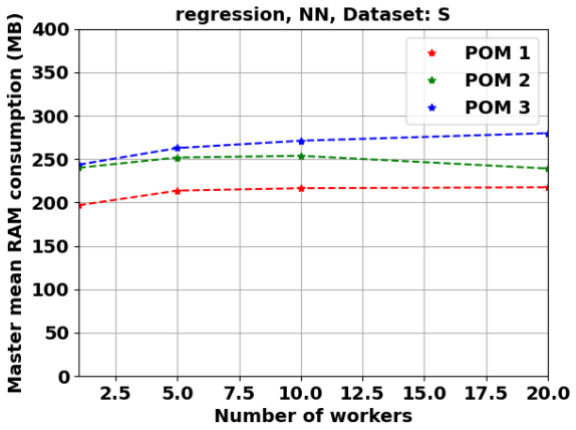


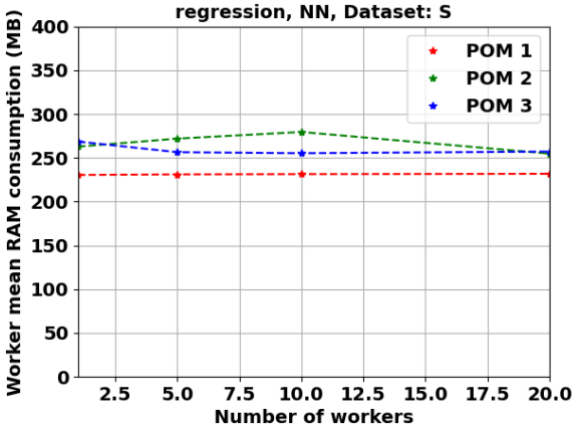


In every training iteration, the worker nodes send a copy of their local weights to the master node. For this reason, we can observe a linear dependency of the information received by the master node with the number of worker nodes.

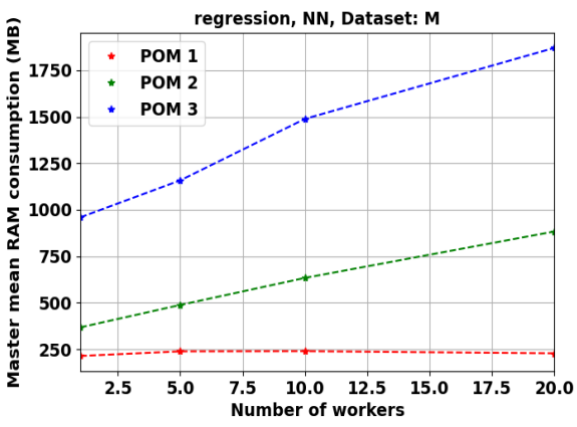
POM2 and POM3 make use of homomorphic encryption (and encrypted weights take up more memory). For this reason, the slope of the linear dependency is higher in POM3 and POM2 than in POM1 and when a single worker is running, the information received by the master is lower in POM1 than in POM2 and POM3.

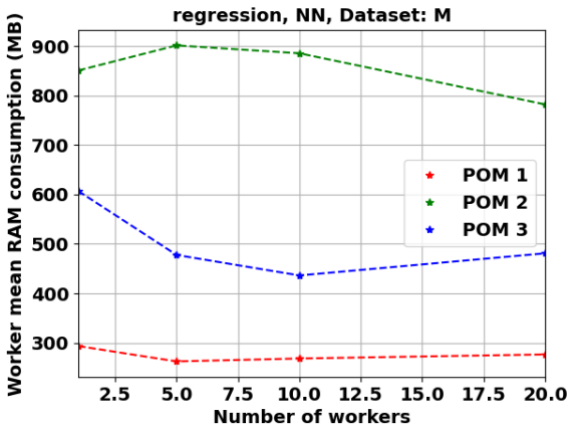
RAM (MB) consumption by master and workers:





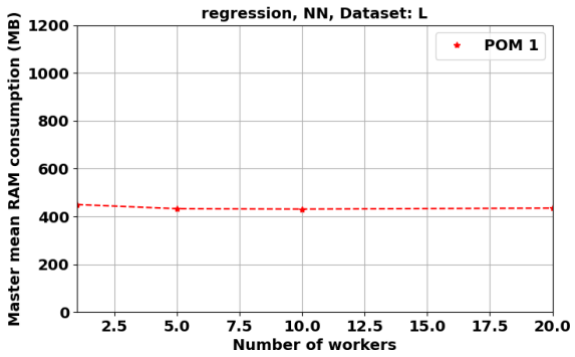
As in previous models, for small datasets, the influence of number of workers and the POM cannot be observed.

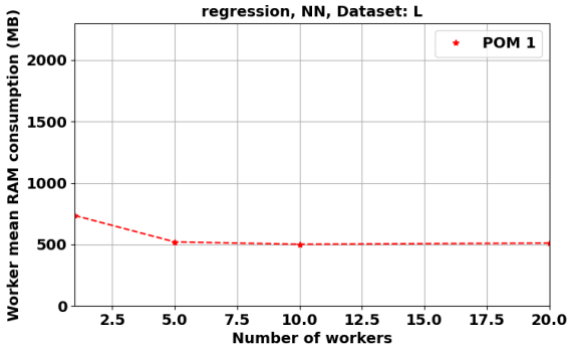




Since POM1 makes not use of encrypted information in memory, we can observe a lower memory consumption.

In POM3, the memory consumption in the master node is higher and increases with the number of workers because it receives a copy of the ML model from every worker in with different encryption keys.





The memory in the master node remains constant. Since we split the dataset among the different workers, the memory consumption in every worker decreases a little bit as we increase the number of workers.

5.2 GQM Goal 1 (G1): Assessing performance, scalability, and computational efficiency.

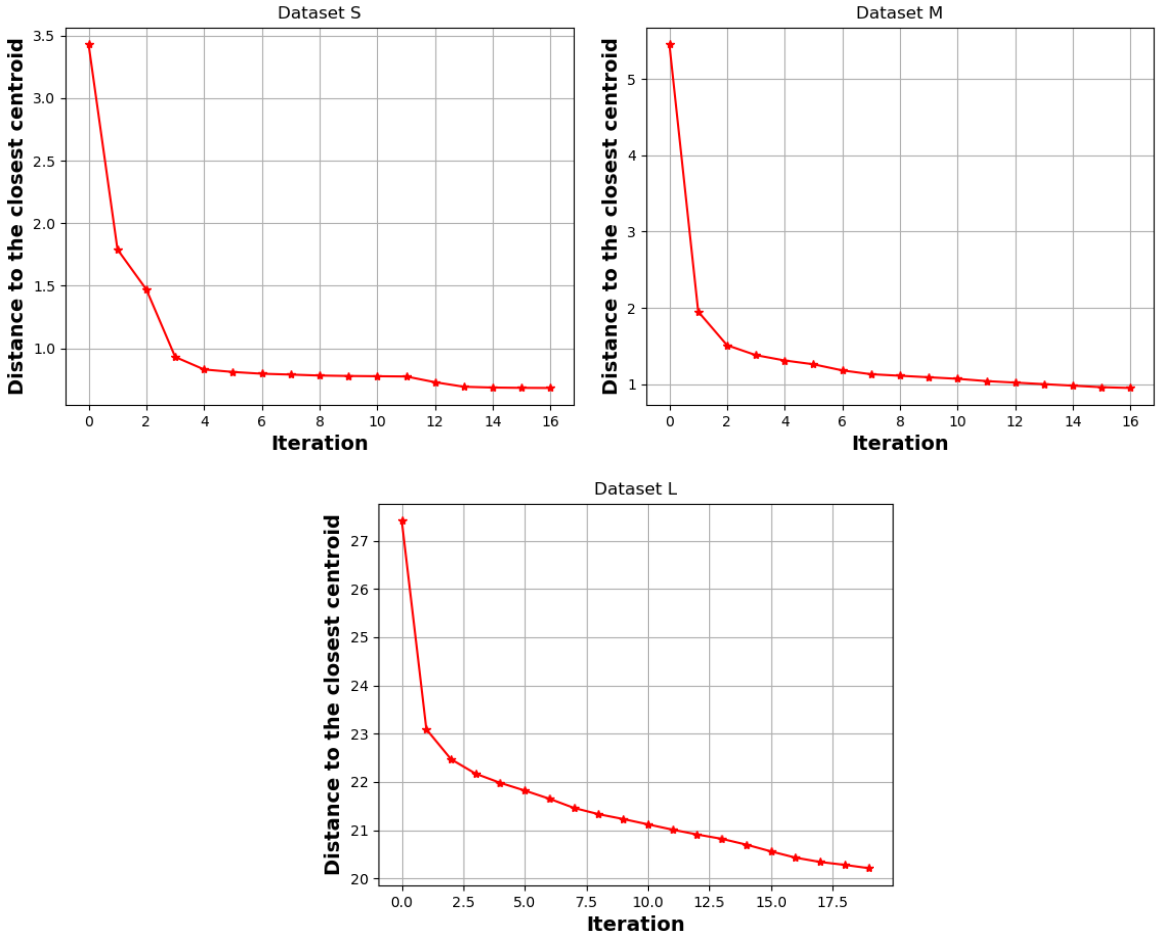
In this section we will provide an example of results analysis for some of the algorithms in POMs 1, 2, 3, following the GQM methodology described in D6.1.

5.2.1 G1.1 and G1.2: Assessing the performance and reliability of MLA

G1.1_Q1	Is the ML library able to provide a data clustering in such a way that objects in the same group are more similar to each other than to those in other groups?

MMLLK- counts with the K-means algorithm, that is the most popular unsupervised machine learning algorithm for clustering. Its objective is to minimize the average squared Euclidean distance of data from their centroids where a centroid is defined as the mean of data in a cluster.

If we take the Euclidean distance as the similarity metric, then the algorithm provides an effective way to cluster them. The next graphics show how the distance to the closest centroid is reduced after every iteration of the training process for the 3 datasets used in the assessment.



Section 5.1.1 contains more information about the performance of the K-means algorithm in the library.

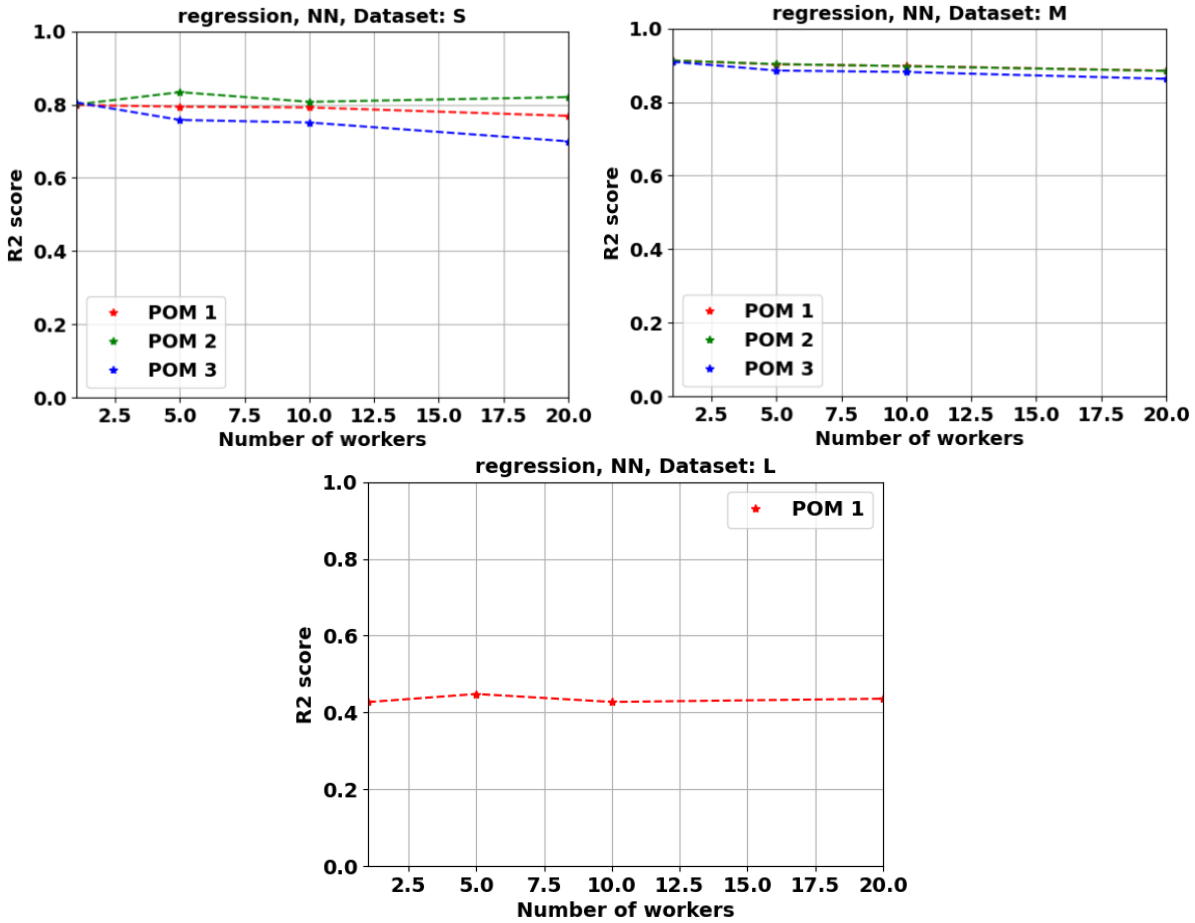
G1.1_Q2	Given a dataset, is the ML library able to provide predictions for unseen values of related non-categorical (real valued) variables?

POM1, POM2 and POM3 count with Neural Networks capable to minimize distinct functions based on the prediction error over real valued variables: Mean Squared Error, Mean Absolute Error, Mean Absolute Percentage Error, Mean Squared Logarithmic Error, ...

We have tested three different datasets (see section 5.1.4.1) and measured the R^2 score, that is the most extended performance metric for regression problem.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Best possible R^2 score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a score of 0.0.



Considering that a 1.0 R^2 score means a model with a perfect prediction and 0.0 means a model that always predicts the expected value of the variable, in every dataset we obtain a score over 0.0. This means, the prediction is always better than the expected value of the variable. The dataset S and M achieve an R^2 score around 0.8 and 0.9 respectively (1.0 is a perfect prediction), while dataset L obtain an R^2 score around 0.45.

G1.1_Q3	Given a training dataset, is the ML library able to provide predictions of the class of each data, according to some related categorical variable?

MMLL library counts with different classification algorithms in POM1, POM2 and POM3. We have tested these algorithms using different datasets.

For multiclass classification we have used the datasets Iris (Small Size, 3 classes), Abalone (Medium size, 3 classes) and MNIST (Large size, 10 classes). For binary classification we have

used Diabetes (Small), Adult (Medium) and MNIST odd vs even (Large size). The accuracy obtained is summarized below:

MULTICLASS PROBLEM	Dataset S	Dataset M	Dataset L
Neural Network (Multiclass)	0.97	0.57	0.91

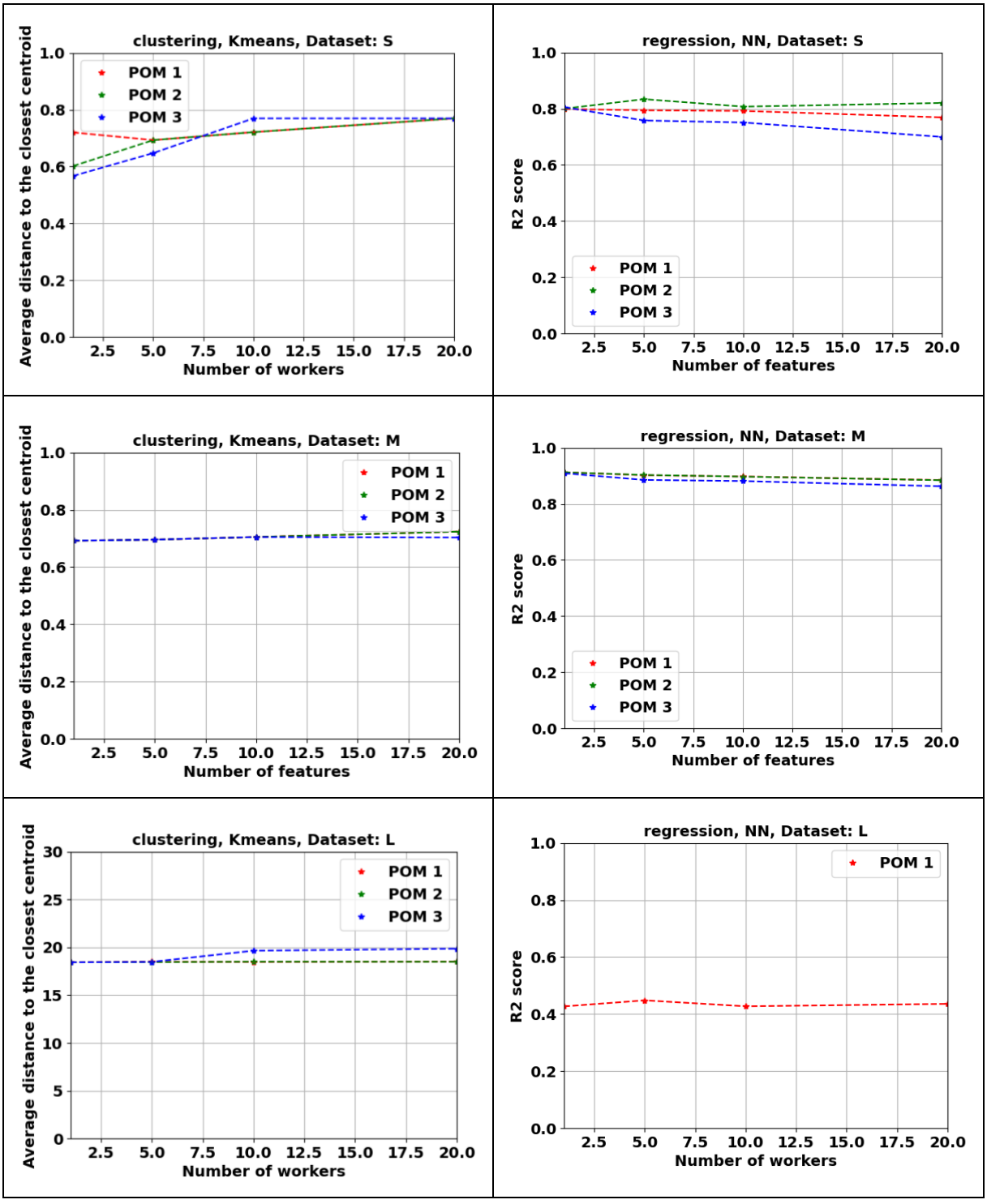
BINARY PROBLEM	Dataset S	Dataset M	Dataset L
Neural Network (Binary)	0.71	0.81	0.99
DSVM	0.73	0.83	0.96
FBSVM	0.75	0.84	0.95

G1.2_Q1	Does each ML algorithm give comparable output working on the same data and in the same conditions in different sessions (reliability)?

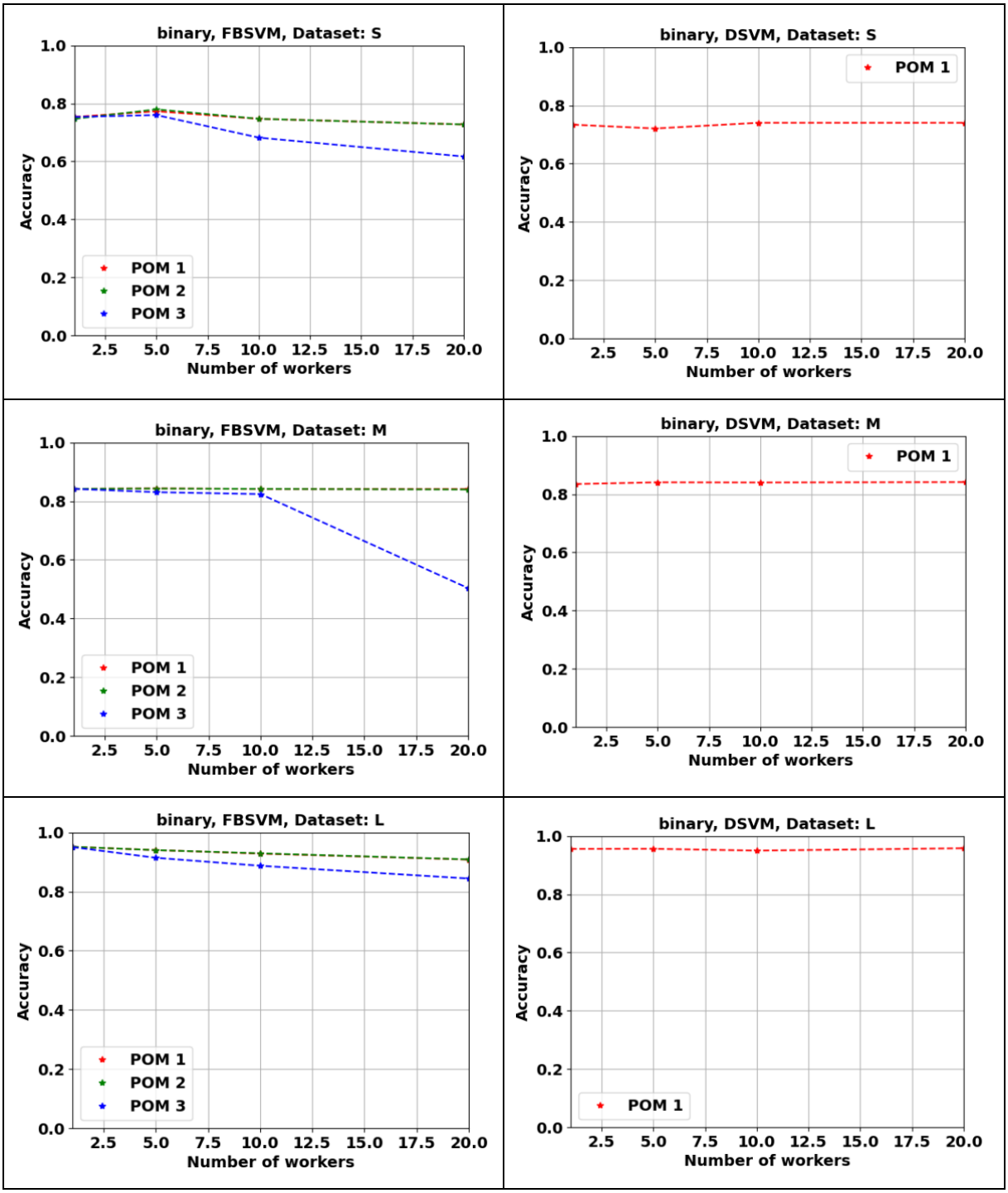
In POM 1, when one worker is running, the resulting predictive model is equivalent to a standard (centralized) ML library since the worker completes the training process with no encryption and the master node doesn't aggregate models from different workers.

Every algorithm has a random initialization. According to the algorithm and training conditions we can observe different behavior of those models to converge to the same solution.

In the case of Kmeans or Neural Networks, for small datasets, the random initialization influences the result and we can observe differences in the performance metric among different executions of the algorithm. However, for medium and large datasets, this effect is reduced and the results tend to be the same independently of the POM and number of workers used in the training process.



In the case of SVM implementations. In FBSVM we can appreciate how the accuracy decreases slowly as we increase the number of workers. This training procedure presents some problems in non-iid datasets. This problem is solved in DSVM, that tend to achieve the same solution independently of the number of workers.

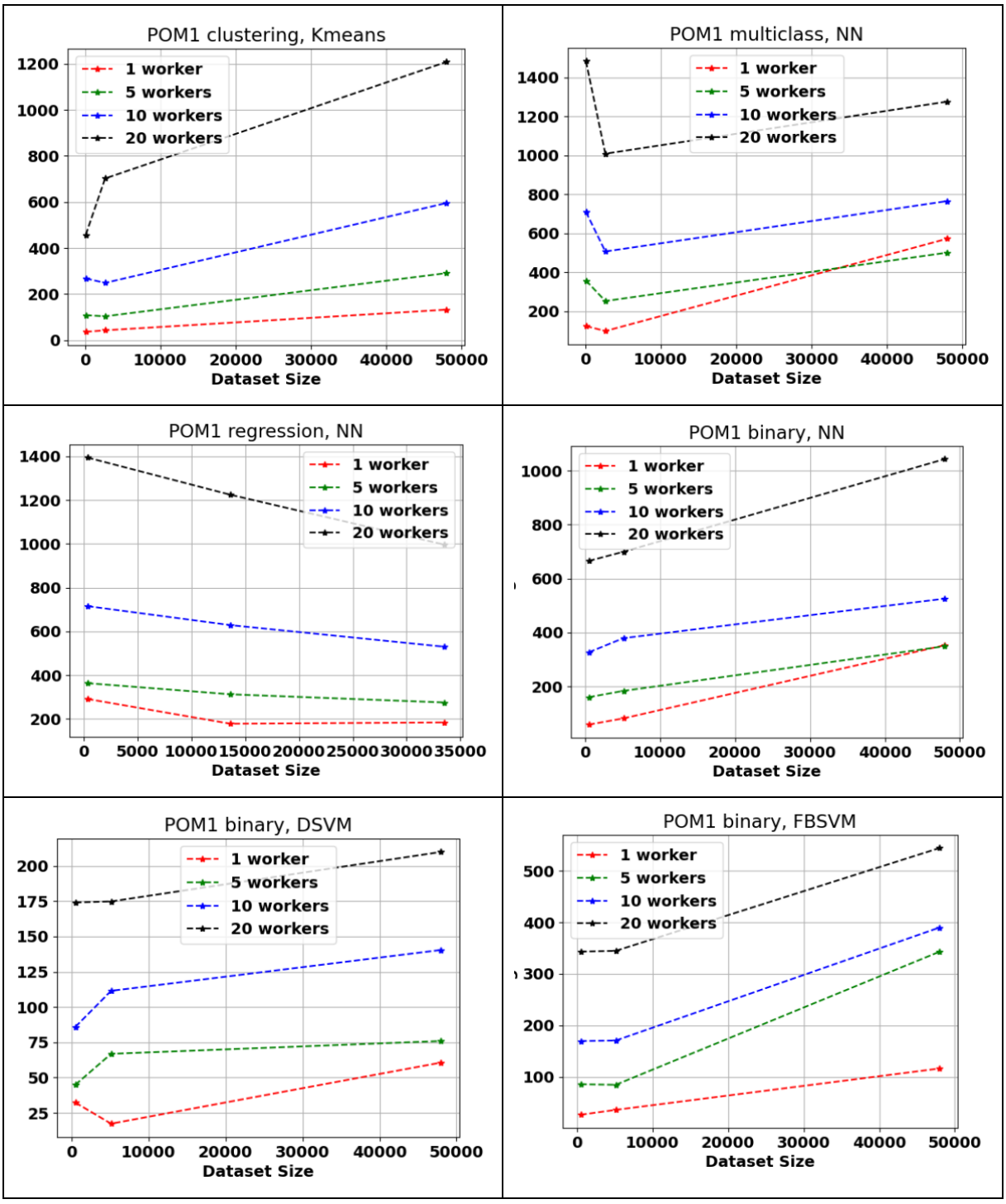


5.2.2 G1.3: Assessing the scalability of MLA

G1.3_Q1	Does the training algorithm scale up when the dimension of the application scenario grows in terms of the amount of data?

We have tested every algorithm using 3 different datasets that contains different number of training samples.

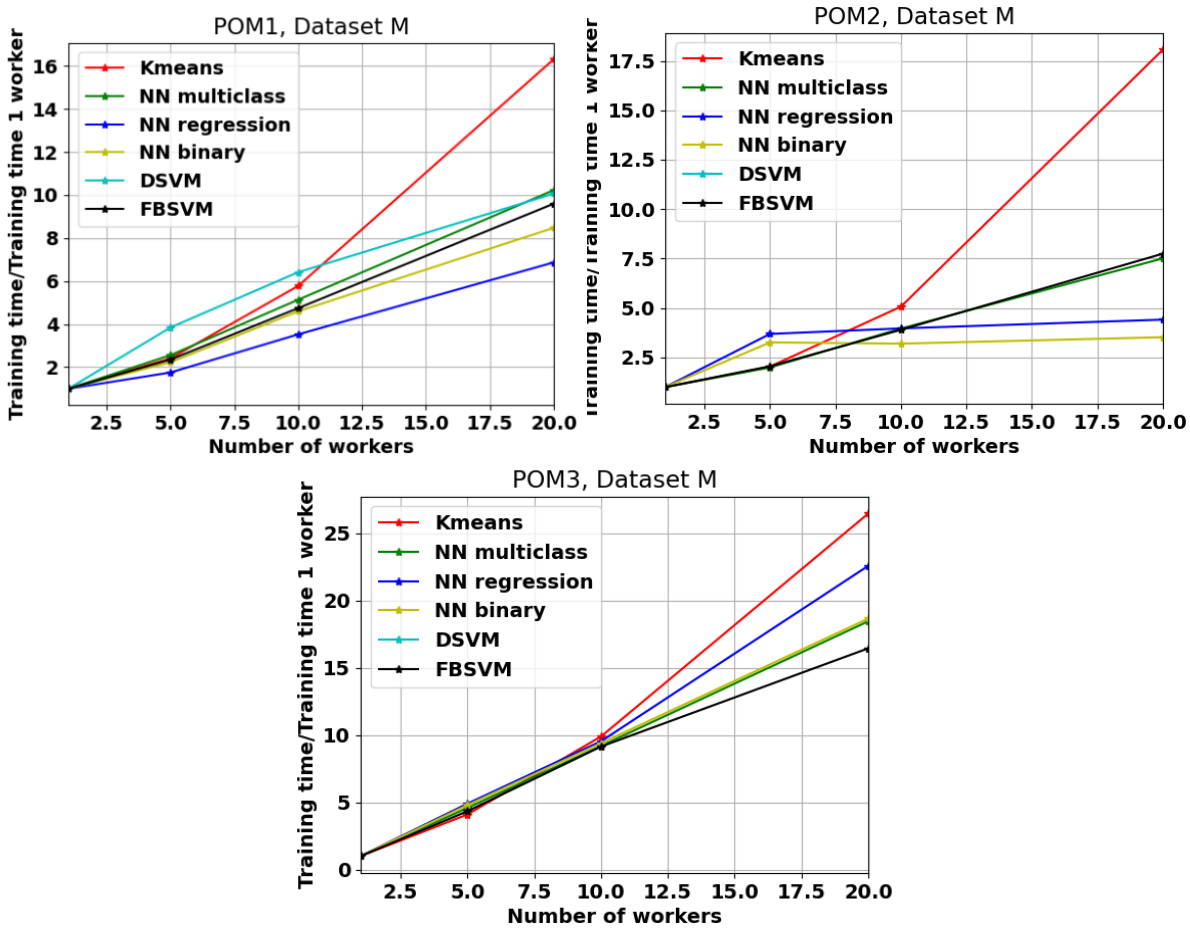
The next pictures represent the training time as a function of the number of training samples in POM 1. According to the number of features we can observe different behaviours in terms of scalability.



Typically, the runtime increases with the number of training samples. However, every dataset is different and has associated a different number of training steps until convergence. For this reason, in some cases such as the Neural Networks for regression, we can observe how the runtime decreases because an increase of input features has motivated a decrease in the training iterations.

G1.3_Q2	Does the training algorithm scale up when the dimension of the application scenario grows in terms of the number of users (data providers)?

The following pictures contain the training time when N workers (data providers) are running divided by the training time when there is only 1 data provider.



POM1 is the fastest one (see section 5.1). It doesn't need encryption and the master node can broadcast the information to the worker nodes at the same time in every iteration. We can see how the training time increases linearly with the number of workers due to the bottleneck associated to receive the information from every worker after every iteration.

The behaviour of POM2 is similar to POM1 but it needs to encrypt and decrypt the information in every iteration, that is why the training time is higher than in POM1.

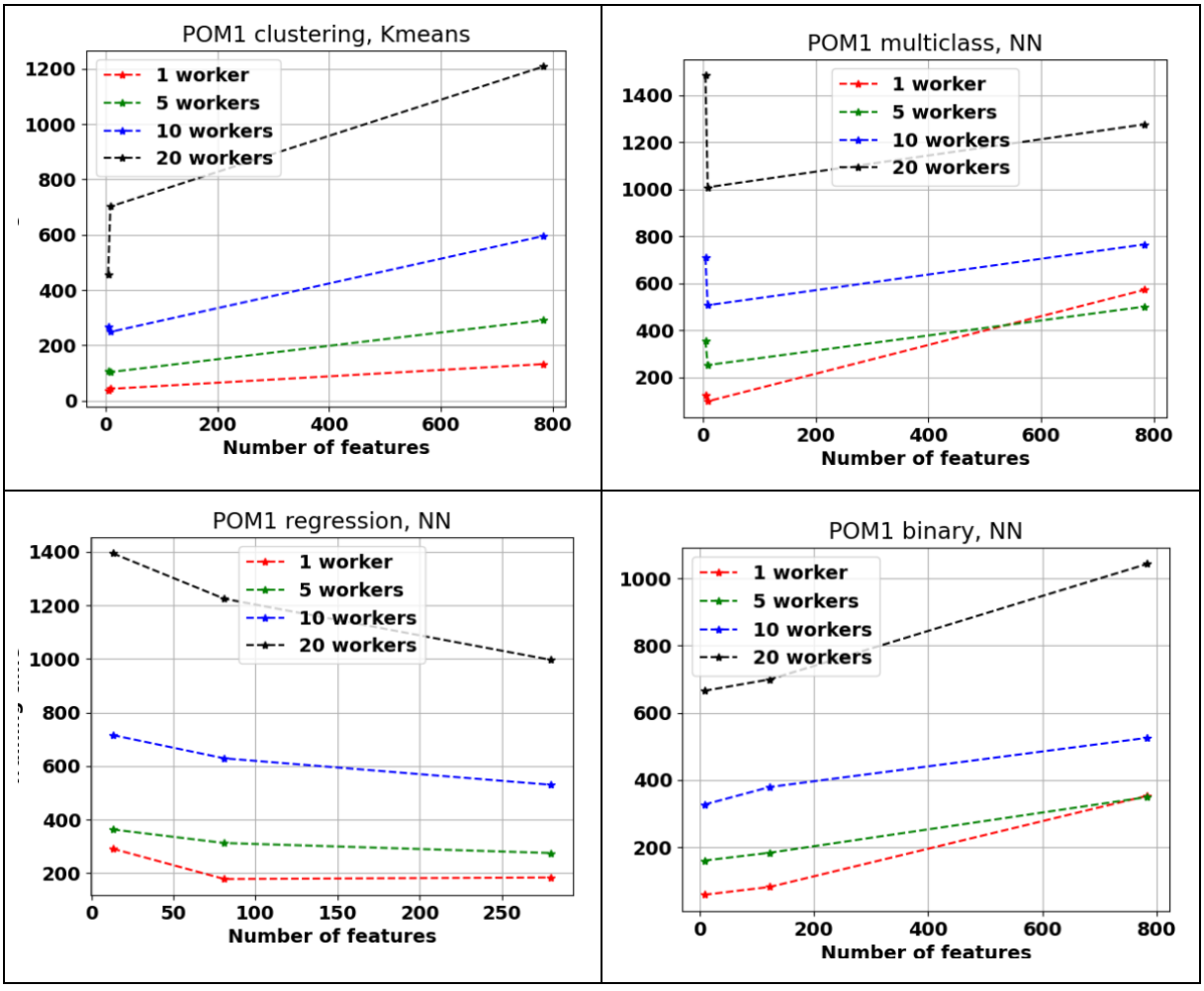
However, this encryption and decryption in the worker side can be performed in parallel, that why it doesn't affect to the scalability.

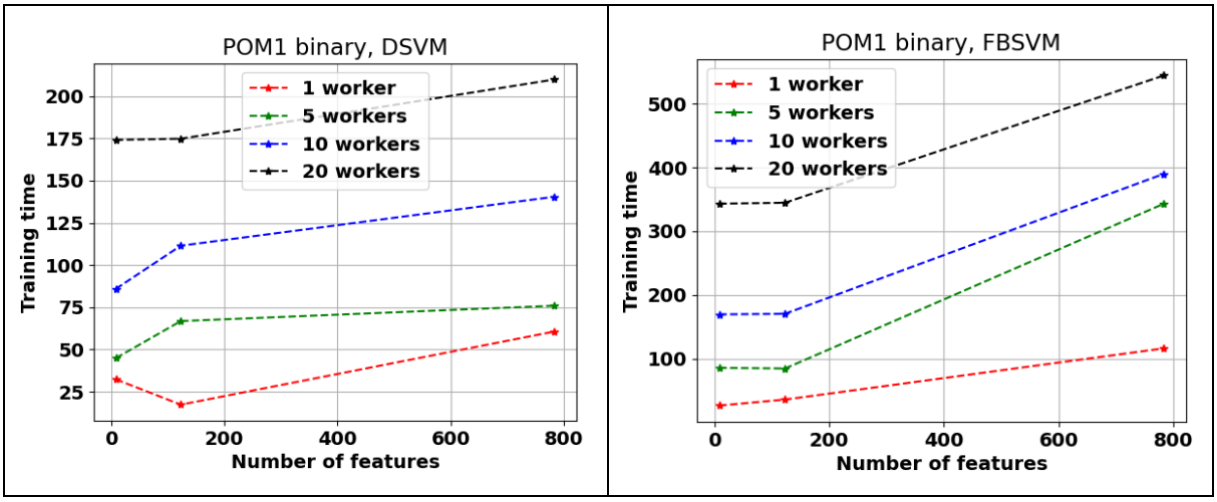
POM3 cannot make use of the broadcast command (and the communication is one of the bottlenecks in federated learning). That is why the runtime is higher and achieves poorer scalability than POM1 and POM2.

G1.3_Q3	Does the training algorithm scale up when the dimension of the application scenario grows in terms of the amount of input features?

We have tested every algorithm using 3 different datasets that contains different number of input features.

The next pictures represent the training time as a function of the number of input features in POM1. According to the number of features we can observe different behaviours in terms of scalability.





Typically, the runtime increases with the number of input features. However, we are working with different datasets and problems and in the case of Neural Networks for regression, we can observe how the runtime decreases because an increase of input features has motivated a decrease in the training iterations.

5.2.3 G1.4: Assessing the computational efficiency of MLA

G1.4_Q1	Are the MLAs faster than their counterparts in competing libraries?

We have identified a variety of privacy preserving ML systems, but for a fair comparison, these libraries should share same conditions than Musketeer MMLL.

We have identified several platforms for privacy preserving ML. The main characteristics of the identified platforms are described in section 6.4.2.

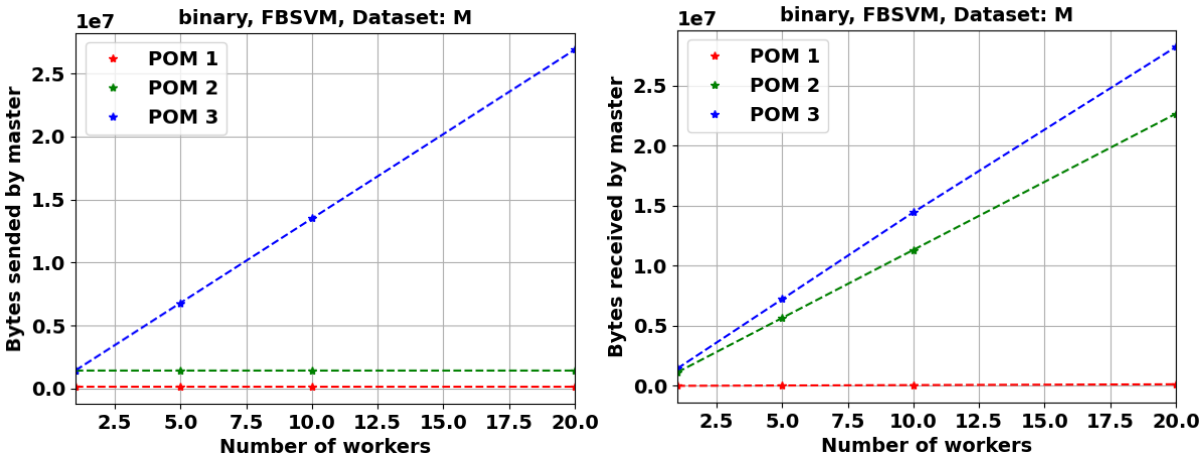
However, most of the systems are experimental with no accessible code. In other cases, the code is not ready to use for general purposes and it is just a demo just for a concrete dataset and algorithm with specific parameters and it is not capable to run a variety of datasets and models.

In other cases, the demos require to open port numbers and socket communications that are not secure. In MUSKETEER, all interactions take place through the MUSKETEER central platform [Pycloudmessenger]. In our platform, only the connection details for the broker are made available, with all other entities, protected from direct attack. In this sense, we will disregard all those competing platforms that do not provide a communication means among different processes or the same level of security as MUSKETEER, since direct connections are usually much more effective (and faster but also more insecure) and the comparison would be unfair.

G1.4_Q2	Are the transmission costs reasonable?

We have measured the bytes sent and received by the master node. Section 5.1 contains the complete list of graphics about data transmission in every algorithm.

The following pictures belongs to the FBSVM algorithm, but the behaviour is very similar in all of them:



In every iteration of the training process, the aggregator sends the ML model to every worker.

Information sent from master to workers:

However, in POM1 and POM2, the master node broadcast the data, so it only needs to submit the data once to the pycloudmessenger library and the bytes sent by the master node does not depend on the number of workers and remains constant.

In the case of POM3, the training process is sequential with the number of workers and the ML model is sent to every worker one by one, for that reason we can observe a linear dependency of the information sent with the number of workers.

POM2 and POM3 make use of homomorphic encryption (and encrypted SVM weights take up more memory), for that reason, in training processes with 1 worker, the information sent by the master is lower in POM1 than in POM2 and POM3 and is similar in both methods with encryption.

In every training iteration, the worker nodes send a copy of their local SVM weights to the master node. For this reason, we can observe a linear dependency of the information received by the master node with the number of worker nodes.

POM2 and POM3 make use of homomorphic encryption (and encrypted weights take up more memory). For this reason, the slope of the linear dependency is higher in POM3 and

POM2 than in POM1 and when a single worker is running, the information received by the master is lower in POM1 than in POM2 and POM3.

Information sent from workers to master:

In every training iteration, the worker nodes send a copy of their local SVM weights to the master node. For this reason, we can observe a linear dependency of the information received by the master node with the number of worker nodes.

POM2 and POM3 make use of homomorphic encryption (and encrypted weights take up more memory). For this reason, the slope of the linear dependency is higher in POM3 and POM2 than in POM1 and when a single worker is running, the information received by the master is lower in POM1 than in POM2 and POM3.

G1.4_Q3	Is the memory usage during training reasonable?

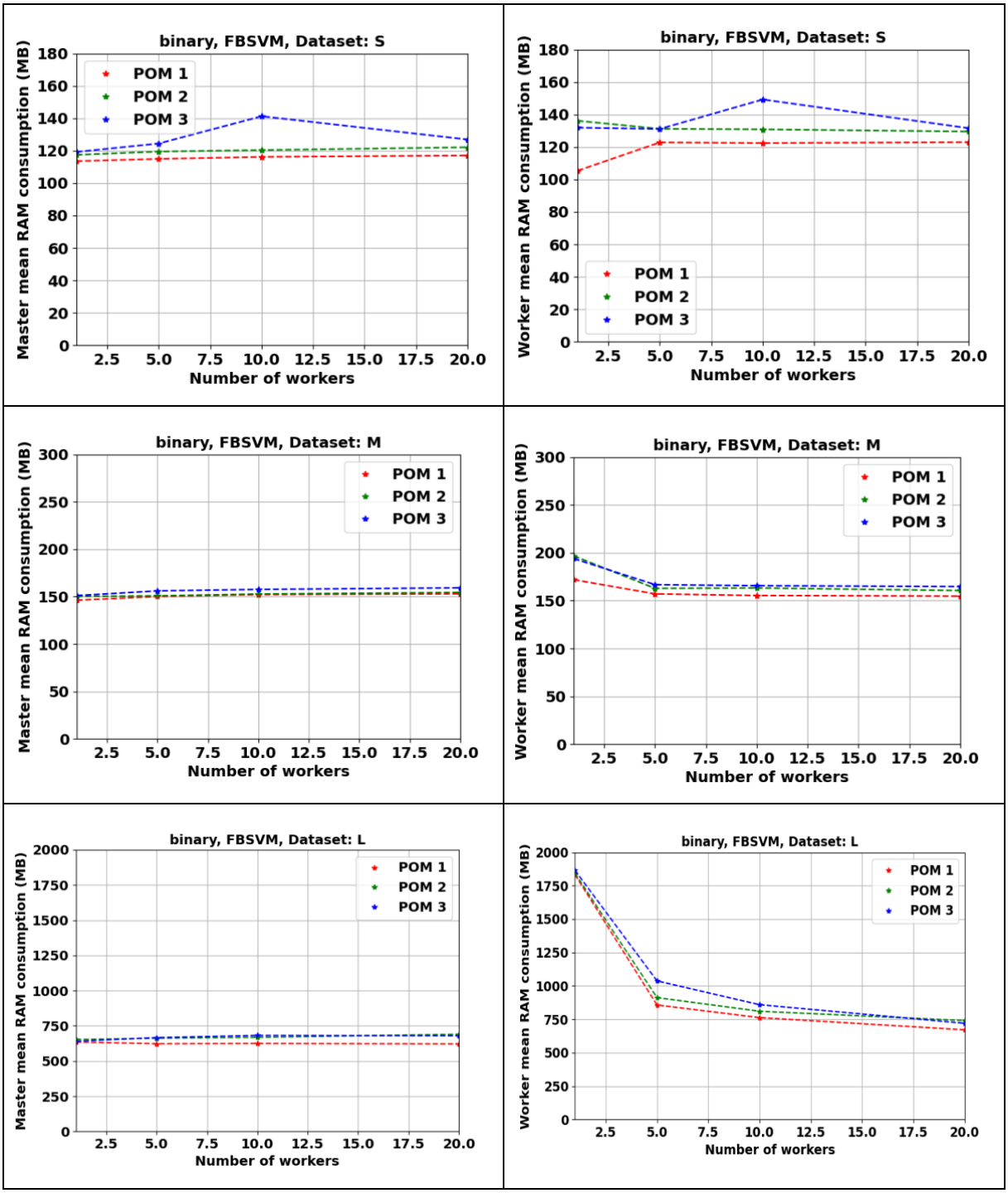
We have measured every two seconds the memory consumption of the Docker container that contains the master node and one of the Docker containers that contains a worker node. To obtain a single metric, we have to average the results.

The behaviour is very similar for every algorithm. For small datasets, the RAM memory used by the processes is negligible than the RAM memory used by the complete container. The influence of number of workers and the POM cannot be observed.

For medium size datasets, the RAM memory used by the processes is still more or less negligible than the RAM memory used by the complete container. Since POM1 makes not use of encrypted information in memory, we can observe a lower memory consumption.

For large size datasets, we can observe the influence of the POM and dataset size in the memory consumption. In POM1 we can observe a lower memory consumption (no encryption has been used). Since we split the dataset among the different workers, the memory consumption in every worker decreases as we increase the number of workers. The master node of POM3 receives the centroids of every worker, so the memory increases as we increase the number of worker nodes.

Here we can see an example for the algorithm FBSVM. The complete list of graphics for every algorithm can be seen in section 5.1.



5.3 Summary of results

We will summarize here the observed results for every algorithm and POM, in the form of a Table that indicates whether or not the GQM test is passed. We will use the results in this Table to draw the final conclusions about the performance of the implemented MMLL library and also to define some recommendations for the end users.

5.3.1 Assessment of algorithms under POM1

Table 1. Summary of GQM tests results for algorithms in POM1

		Kmeans	Neural Networks	DSVM	FBSVM
Performance	G1.1_Q1	OK	-	-	-
	G1.1_Q2	-	OK	-	-
	G1.1_Q3	-	OK	OK	OK
	G1.2_Q1	OK	OK	OK	NO (*)
Scalability	G1.3_Q1	L	L	L	L
	G1.3_Q2	L	L	L	L
	G1.3_Q3	L	L	L	L
Efficiency	G1.4_Q1	-	-	-	-
	G1.4_Q2	OK-	OK	OK	OK
	G1.4_Q3	OK	OK	OK	OK

(*) We could appreciate that the accuracy decreases as we increase the number of workers.
(**) For transmissions over the internet.

POM1 doesn't need encryption mechanisms and the master node broadcast information to the workers, that can perform the computations in parallel. With this advantage, this POM provides higher scalability and efficiency than other POMs based on federated learning.

As a weakness, the master node receives the ML model with no encryption, so this predictive model is not private for the orchestrator of the platform. This is a potential leakage risk if the orchestrator is an external third party provider.

The only test not passed in this POM is G1.2_Q1 in the case of FBSVM. We have observed a decrease in the accuracy as we increase the number of workers.

5.3.2 Assessment of algorithms under POM2

Table 2. Summary of GQM tests results for algorithms in POM2

		Kmeans	Neural Networks	FBSVM

Performance	G1.1_Q1	OK	-	-
	G1.1_Q2	-	OK	-
	G1.1_Q3	-	OK	OK
	G1.2_Q1	OK	OK	NO (*)
Scalability	G1.3_Q1	L	L	L
	G1.3_Q2	L	L	L
	G1.3_Q3	L	L- (**)	L
Efficiency	G1.4_Q1	-	-	-
	G1.4_Q2	OK-	OK-	OK-
	G1.4_Q3	OK	OK	OK

(*) We could appreciate that the accuracy decreases as we increase the number of workers.
(**) Neural Networks scale correctly with the number of input features, but we have used a higher number of neurons in the hidden layers as we increase the number of input features. The scalability as we increase the number of neurons in this way is low.

In POM 2, the model is encrypted using homomorphic cryptosystem. For this reason, in G1.4_Q2 the transmission cost increases respect to POM 1 and we have marked the test with an OK- instead of OK.

In addition, the computational cost of the encryption is high. We have used neural networks with a higher number of neurons in the hidden layers for datasets with higher number of input features. This reduces the scalability as a function of the input features in G1.3_Q3.

5.3.3 Assessment of algorithms under POM3

Table 3. Summary of GQM tests results for algorithms in POM3

		Kmeans	Neural Networks	FBSVM
Performance	G1.1_Q1	OK	-	-
	G1.1_Q2	-	OK	-
	G1.1_Q3	-	OK	OK
	G1.2_Q1	OK	OK	NO (*)

Scalability	G1.3_Q1	L	L	L
	G1.3_Q2	L -- (**)	L -- (**)	L -- (**)
	G1.3_Q3	L - (***)	L- (***)	L
Efficiency	G1.4_Q1	-	-	-
	G1.4_Q2	NO (****)	NO (****)	OK
	G1.4_Q3	OK	OK	OK

(*) We could appreciate that the accuracy decreases as we increase the number of workers.
(**) The training procedure in this pom works in a sequential way with the number of data users. The scalability is lower than in previous POMs.

(***) Neural Networks scale correctly with the number of input features, but we have used a higher number of neurons in the hidden layers as we increase the number of input features. The scalability as we increase the number of neurons in this way is low.

(****) This algorithm sends matrices not in a broadcast way. It is a sequential training, so the transmission cost increases with low efficiency.

In POM 3, the models are encrypted with homomorphic encryption and the training process is sequential with the number of workers. For that reason, the test of efficiency in transmission cost were not passed and the scalability as a function of the number of workers and input features is lower than in previous POMs.

6 Assessment of POMs 4, 5 and 6.

In this chapter we will analyse the experimental results from different perspectives, to extract qualitative measurements about every algorithm/POM with the goal of better understanding the characteristics of every approach, such that the end users will have a reference guide to select the best method for a given scenario. It is also important to understand the associated complexity of every approach, since the computational and communications resources needed by every POM may be different. Before presenting the detailed results of the assessment of MMLL algorithms developed under POMs 4, 5 and 6, we will first carry out some general experiments to obtain a global perspective of the complexity of the involved protocols and their requirements. Secondly, we will in depth analyse the result of the GQM experiments for models in POMS 4, 5 and 6, as described in D6.1, trying to assert the defined KPIs in the project. Finally, in Section 7, we will present a brief Q&A guide to help the end users to select the POM that best fits their needs.

6.1 Preliminary computational measurements

Before analysing the results of the GQM assessment we will briefly discuss some of the general aspects concerning the main protocols and techniques used to implement the algorithms under POMs 4, 5 and 6. In the implemented algorithms, two fundamental confidentiality preserving mechanisms have been used. The first one is the Homomorphic Encryption (HE) used to encrypt the data or the model such that the operations take place in the encrypted domain. This technique has been used in the context of POMs 4 (data is encrypted) and 5 (model is encrypted). Another fundamental procedure used to build the algorithms is the Two-Party Computation protocol used to compute a Secure Dot Product (SDP), as described in [Zhu_2015]. In Figure 7 below, we show, for every dataset used in the assessment of POMs 4, 5 and 6, the computation time of a relatively simple operation: computing the dot product between the input training patterns and a linear model represented by a weight vector **w**.

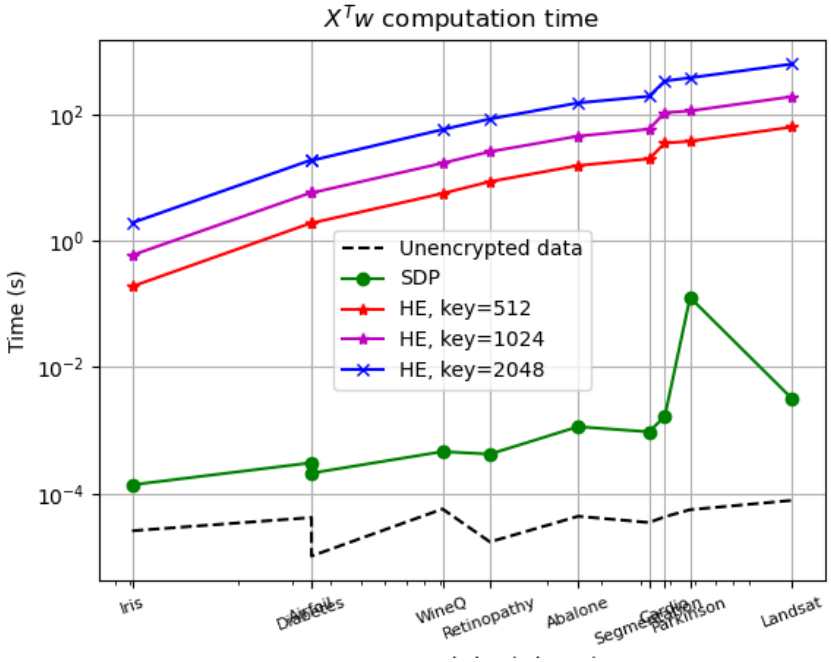


Figure 7. Computational comparison of different confidentiality preserving protocols.

We observe how the fastest computation takes place when data is locally available and it is non encrypted (dashed line marked as “Unencrypted data”). This could also be the computation time when the data is distributed and we send the model to the participants in plain text (e.g. POM1), such that the operations take place in the unencrypted domain (clear text). When we try to protect the model from the participants, the dot product operation is not so straightforward, and secure protocols are needed. The Two-party computation Secure Dot Product described in [Zhu_2015] has been used in the context of POM6, and it provides

a distributed computation which preserves the model confidentiality, but at a computational cost of roughly 2 orders of magnitude with respect to the plain text computation. For the datasets indicated in Figure 7, this operation is completed in less than 1 second.

If a Secure Multiparty Computing protocol based on Homomorphic Encryption is used to solve that operation (as in POMs 4 and 5), the computation times increase by several (up to 4 extra) orders of magnitude, requiring 100 seconds or more to complete this specific computation for some datasets. Furthermore, for an increased security, the key length in the HE schemes may also need to be increased, such that every time we double the key length, the computation time is increased -on average, multiplied by 3.3-. These computation times have been evaluated on a single core with the same characteristics in all cases. A very important speedup could be obtained by using specific hardware, such as Graphical Processing Units (GPUs) or any other parallel computation architecture that could speed up local computations. The good news here is that the encryption library used in POMs 4 and 5 is external to MUSKETEER, such that if a specific speedup can be achieved by a particular combination of HE encryption software and specific hardware, that speedup could directly be applied to the training algorithms in the MUSKETEER platform.

Therefore, trying to obtain a trained model only available to the aggregator while hiding it to the participants is very costly, so this confidentiality option must be used only when it is really necessary. The most economic approach is that implemented using the Secure Dot Product and other clear-text operations used in POM 6, although this approach needs to reveal some partial statistics about the training data or some intermediate results, such as model outputs. The approaches that operate in the encrypted domain imply a much larger computational cost but they reveal less information about the training data.

The actual computational requirements and training times will be further detailed when the assessment results are presented in the next sections.

6.2 Data storage: memory and transmission

In POM 6 data is not encrypted, son it basically uses the same (order of magnitude) storage space as in the centralized situation. However, the POMs relying on data encryption (POMs 4 and 5), require an extra storage and transmission capability. To illustrate the scale of needed storage/transmission, we have computed the size in Mbytes of every one of the used datasets, and compared the plain size with the encrypted one, as shown in Figure 8 below.

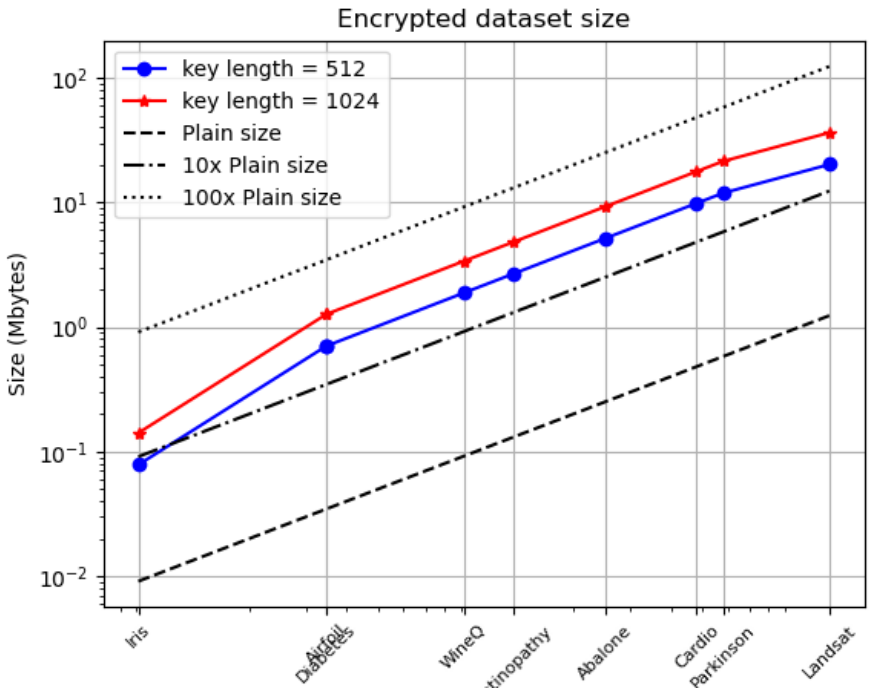


Figure 8. Size of the assessment datasets, plain vs. encrypted.

We observe that the size of the encrypted datasets is between 10-100 times larger than the plain data. This is also a factor that affects the total training time, when encrypted models or patterns need to be transmitted. We have estimated that, in our particular experimental conditions, the transmission rate is about 2.7 Mbytes per second on a steady regime. Once again, the communication means used by MMLL could be replaced by faster ones (dedicated transport networks) if needed for a specific application, but when the Internet is used as the communication platform, rates like this one are expected.

6.3 GQM Goal 1 (G1): Assessing performance, scalability and computational efficiency.

In this section we will analyse the obtained results, following the GQM methodology described in D6.1. We have selected here some representative results, but the exhaustive and complete set of figures of merit have been included in Section 10 (Appendix I), and at the end of this section we have included a Table summarizing the observed results for every model/POM.

6.3.1 G1.1 and G1.2: Assessing the performance and reliability of MLA

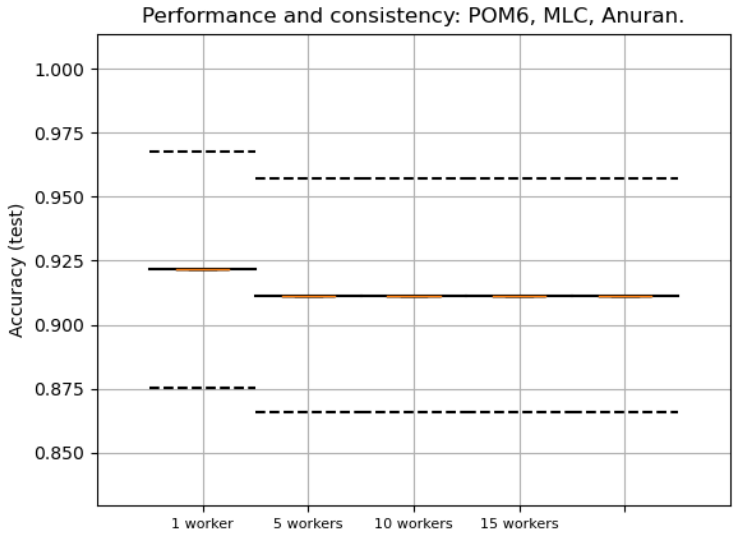
We cover in this section these two questions because they are answered/measured in the same graphics. The main questions to be answered here are:

G1.1_Q1	Are results comparable, within a 5% margin, to those obtained using a standard (centralized) ML library (e.g. scikit learn)?
G1.1_Q2	Is the variation in different training experiments less than 5%?

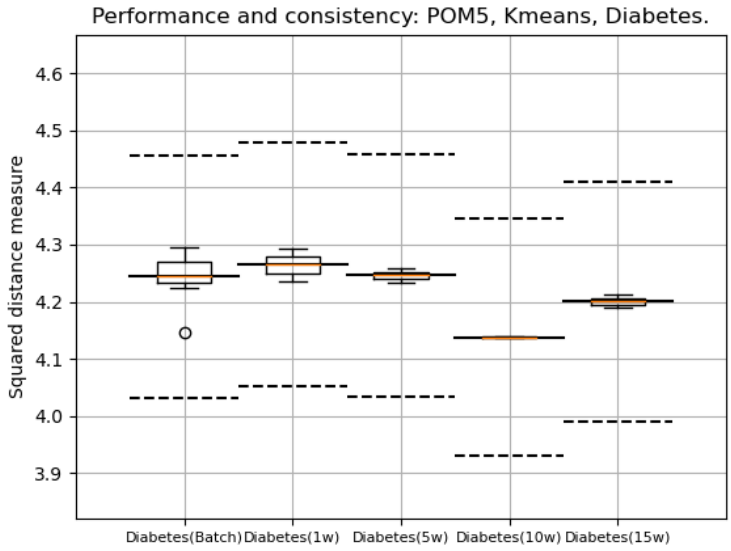
G1.1_Q1: "Are results comparable, within a 5% margin, to those obtained using a standard (centralized) ML library (e.g. scikit learn)?"

G1.1_Q2: "Is the variation in different training experiments less than 5%?"

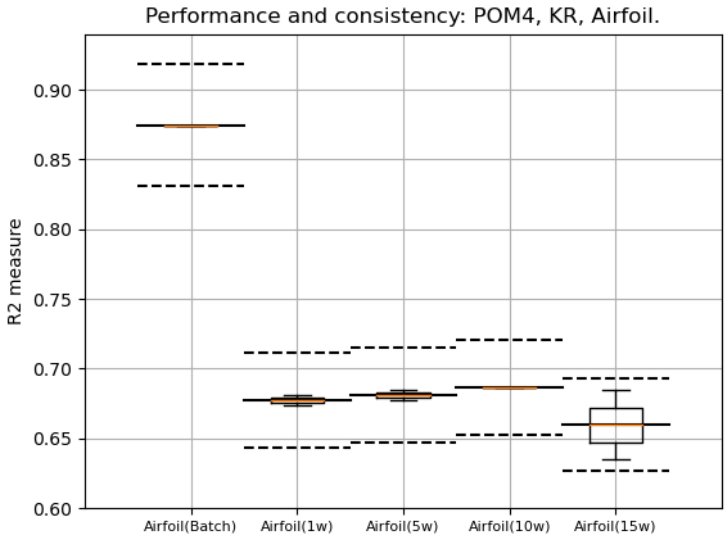
To address these questions, we have run several training experiments with every algorithm and dataset, for every POM and with different number of workers (250 different trainings, each one repeated several times), and we represent the variation in performance of the collection of resulting performances for both the reference result (batch scikit-learn [scikit]) and MUSKETEER. The results are represented as "box plots" which represent the distribution of the obtained performances (the specific used metric depends on the task at hand). In the next Figure, we depict several examples of those experiments, the rest being included in Section 10 (Appendix I):



(a)



(b)



(c)

Figure 9. Examples of observed performance and consistency. Metric distribution and 5% variation limits (dashed lines).

We observe in Figure (a) the typical result for a model with little variance, i.e., those models that converge to almost the same solution irrespectively of the random initialization (e.g., LR, LC, MLC, etc.). We observe that all of the results, for an increasing number of workers, are within the 5% margin (marked with dashed lines in the figures), and are also very close to the centralized solution (first plot on the left). In all cases, there is a little variance from one experiment to another. We observe on (b) an example of those methods that are more sensitive to the initialization conditions or that rely on some random parameterization (e.g.,

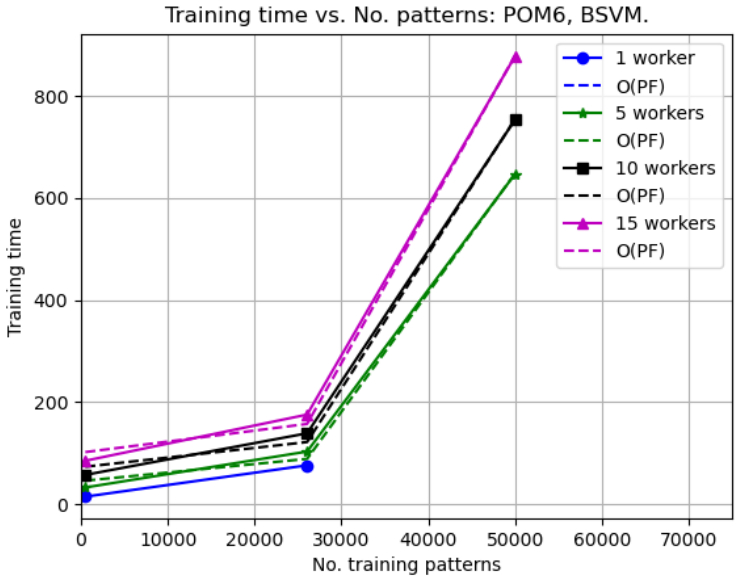
Kmeans, BSVM, etc.). They show a higher deviation among different runs, even in the centralized case solved with scikit-learn. Anyhow, the results are within the limits defined for the corresponding metric. Whenever the obtained performances are within these limits, we will represent these results in the summary Table 2 as "OK". Finally, there are some cases where the limited operations available in the POM have also imposed a limit in the complexity of the optimization algorithm (solver) and, for some of the models, a poor convergence to the minimum is observed. This is the case mainly of POM4, where in the most complex models (e.g., MLC, KR, BSVM), the simple gradient descent solver has presented some convergence problems for some of the datasets.

6.3.2 G1.3: Assessing the scalability of MLA

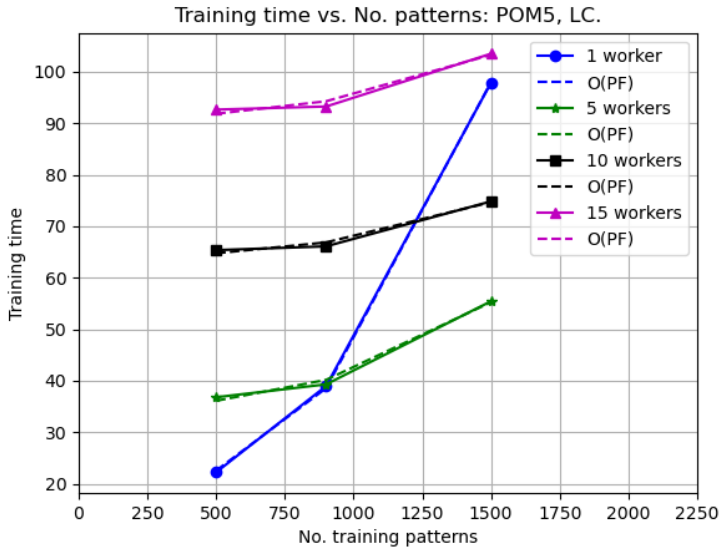
To evaluate this aspect of MMLL we will have to answer the following questions:

G1.3_Q1	Is the training time growing less than quadratic w.r.t. No. patterns?
G1.3_Q2	Is the training time growing less than quadratic w.r.t. No. workers?
G1.3_Q3	Is the training time growing less than quadratic w.r.t. No. features?

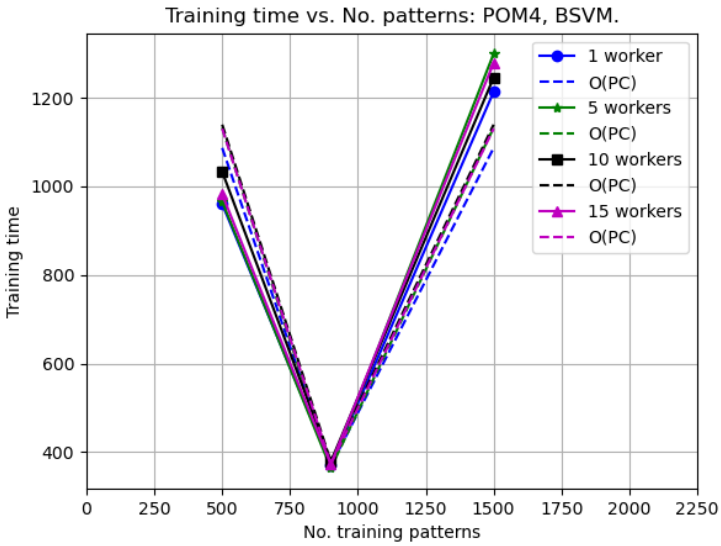
To answer the first question Q1.3_Q1, we have measured the training time in datasets with different number of training patterns, as indicated in the next Figure:



(a)



(b)



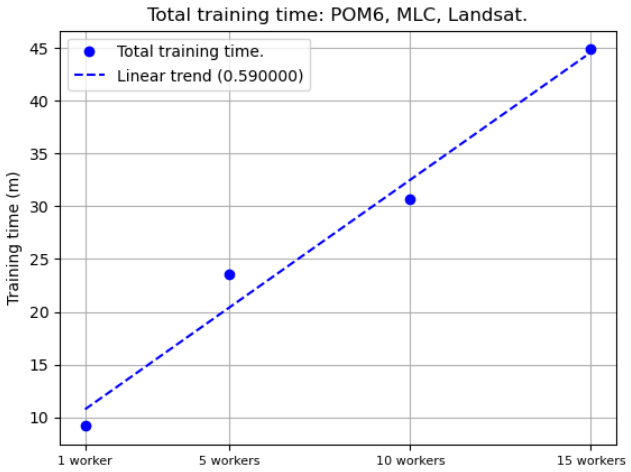
(c)

Figure 10. Examples of training time as a function of the No. of training patterns.

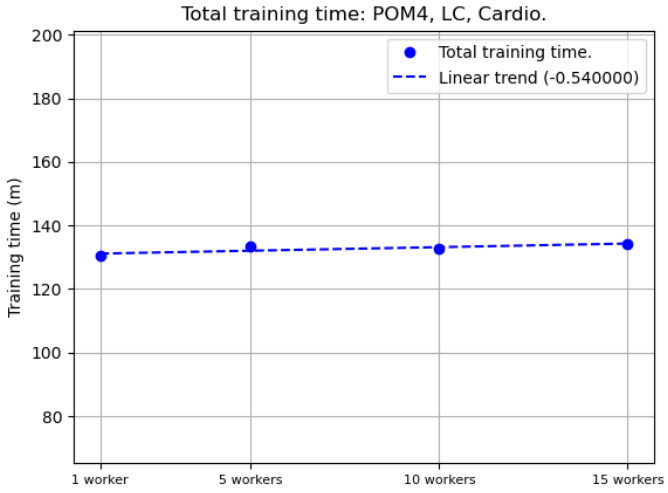
Although it may seem at first glance that a quadratic growing may be observed, actually the best fit to these measurements is obtained using a polynomial curve which is a linear function of both the number of training patterns (P) and the number of input features, i.e.(F), as in Figure 10 (a) and (b). In these cases the complexity is O(PF) and therefore the behaviour is linear with respect to the number of training patterns. For instance, the observed large growth in Figure 10 (a) from 26.000 patterns (Income) to 50.000 (MNIST) is mainly due to the fact that the number of features grows from 107 to 784. In other cases, the complexity has a better fit with respect to the model size (number of centroids, C), as in

the case depicted in Figure 10 (c), where be observe a good fit with a complexity estimation of O(PC). Analogous reasoning can be applied to the corresponding figures for most of the algorithms, all of them depicted in Section 10 (Appendix I). Whenever a complexity growth less than quadratic is observed, we will annotate the result as "OK" in Table 2.

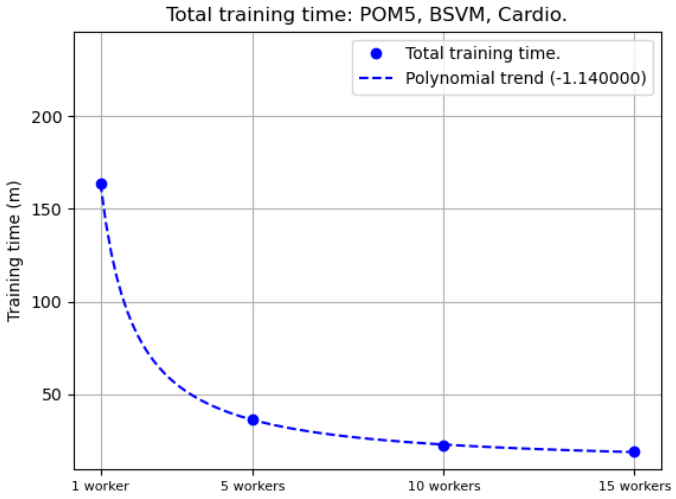
The second question Q1.3_Q2 can be answered by plotting the training time of every dataset when an increasing number of workers are used. The machine were we have run the experiment is able to run 20 separate threads/cores, so we have tested the cases with 1, 5, 10 and 15 workers, some of the remaining threads/cores being used for the aggregator (and cryptonode, in POM 4). Some typical observed evolution of training times is depicted in Figure 11 below:



(a)



(b)



(c)

Figure 11. Typical examples of training time as a function of the No. of workers: POM6 in (a), POM4 in (b), POM5 in (c).

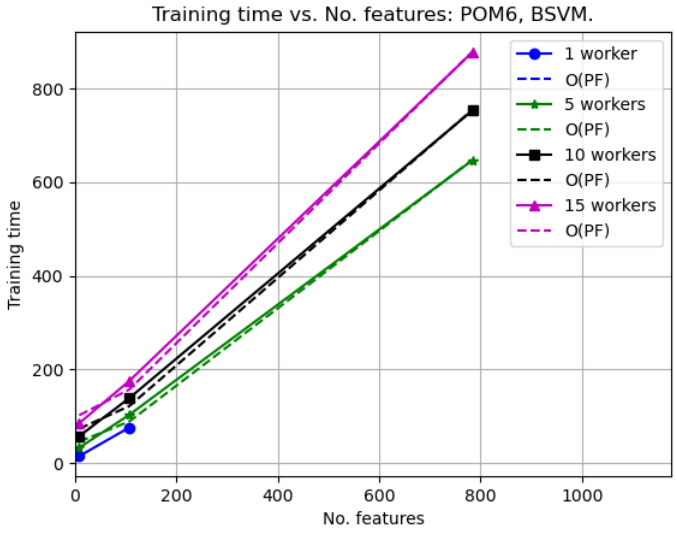
We observe in Figure 11(a) that, in the case of POM6, the training time grows linearly with the number of workers because the used SDP needs the cooperation between aggregator and the workers to compute the results, and more workers means more serial interactions between aggregator and workers, plus the needed communications. No quadratic dependence is observed with respect to the number of workers, though, so the test is positive in these cases.

In the POM4 case (b), we observe that the training time is almost independent of the number of workers, since the encrypted data is first transmitted to the aggregator and the training interactions mainly take place between the aggregator and the cryptonode, irrespectively of the number of contributing users. In any case, no quadratic dependence is observed with respect to the number of workers, so the test is also positive.

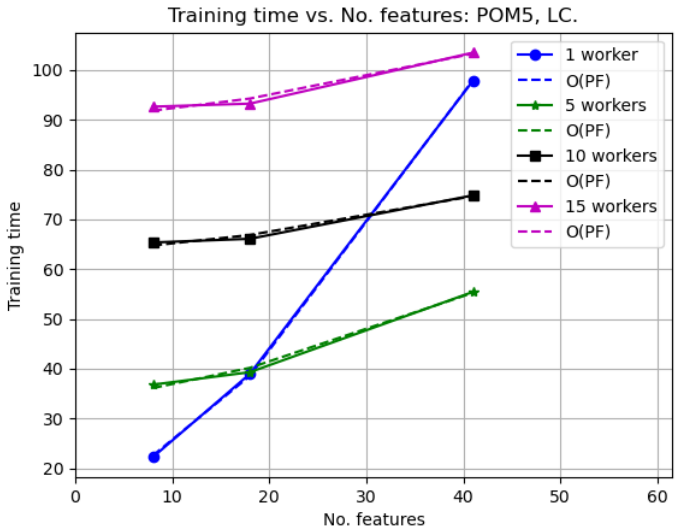
In the POM5 case (c), we observe that the training time decreases with the number of workers. This is due to the fact that computations take mainly place in the workers, since they operate their local data with the received encrypted model, and more workers means less data in every one of them[4], and therefore the total computation time is reduced, since the needed operations are run in parallel. No quadratic growth is observed with respect to the number of workers, so the test is also positive.

[4] Remember that in all experiments, the same amount of data has been used, so if 2 workers are used, each worker has half the data, for 10 workers, each has 1/10 of the data and so on.

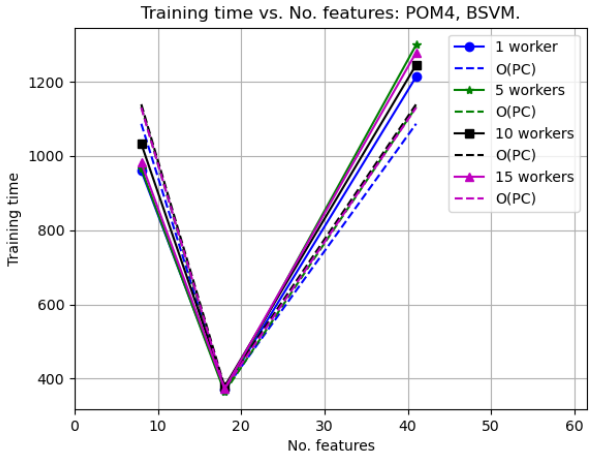
To answer the third question Q1.3_Q3, we have represented the training time in datasets with different number of input features, as indicated in Figure 12:



(a)



(b)

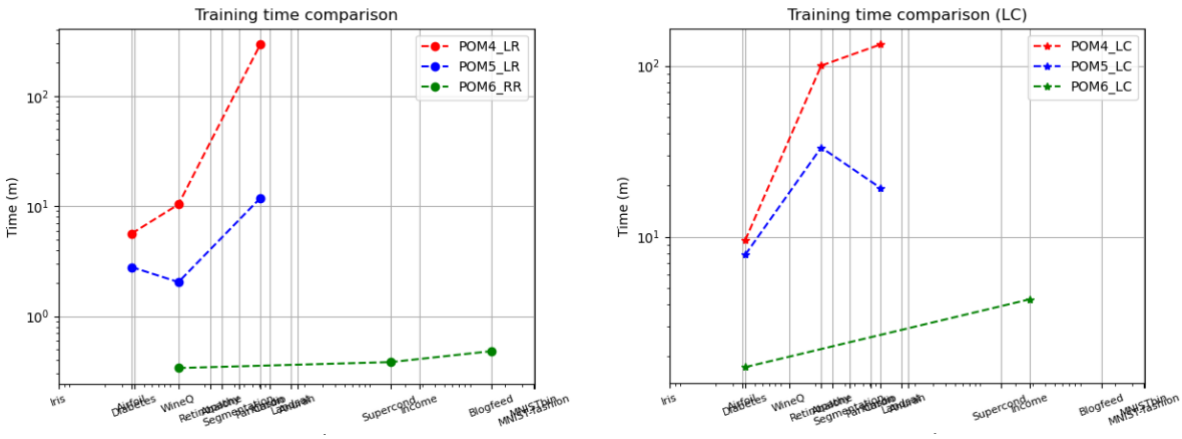


(c)

Figure 12. Examples of training time as a function of the No. of input features.

Again, as already discussed in relationship with question Q1.3_Q1, we observe a linear relationship with both the number of training patterns (P) and the number of input features (F), i.e., the complexity is O(PF) in cases (a) and (b), and complexity O(PC) in case (c). Therefore the behaviour is also linear with respect to the number of input features, the test being positive in all cases.

Finally, we will compare the relative computational costs among POMs 4, 5 and 6. We have depicted in Figure 13 the comparison among POMs for the different common models.



(a) (b)

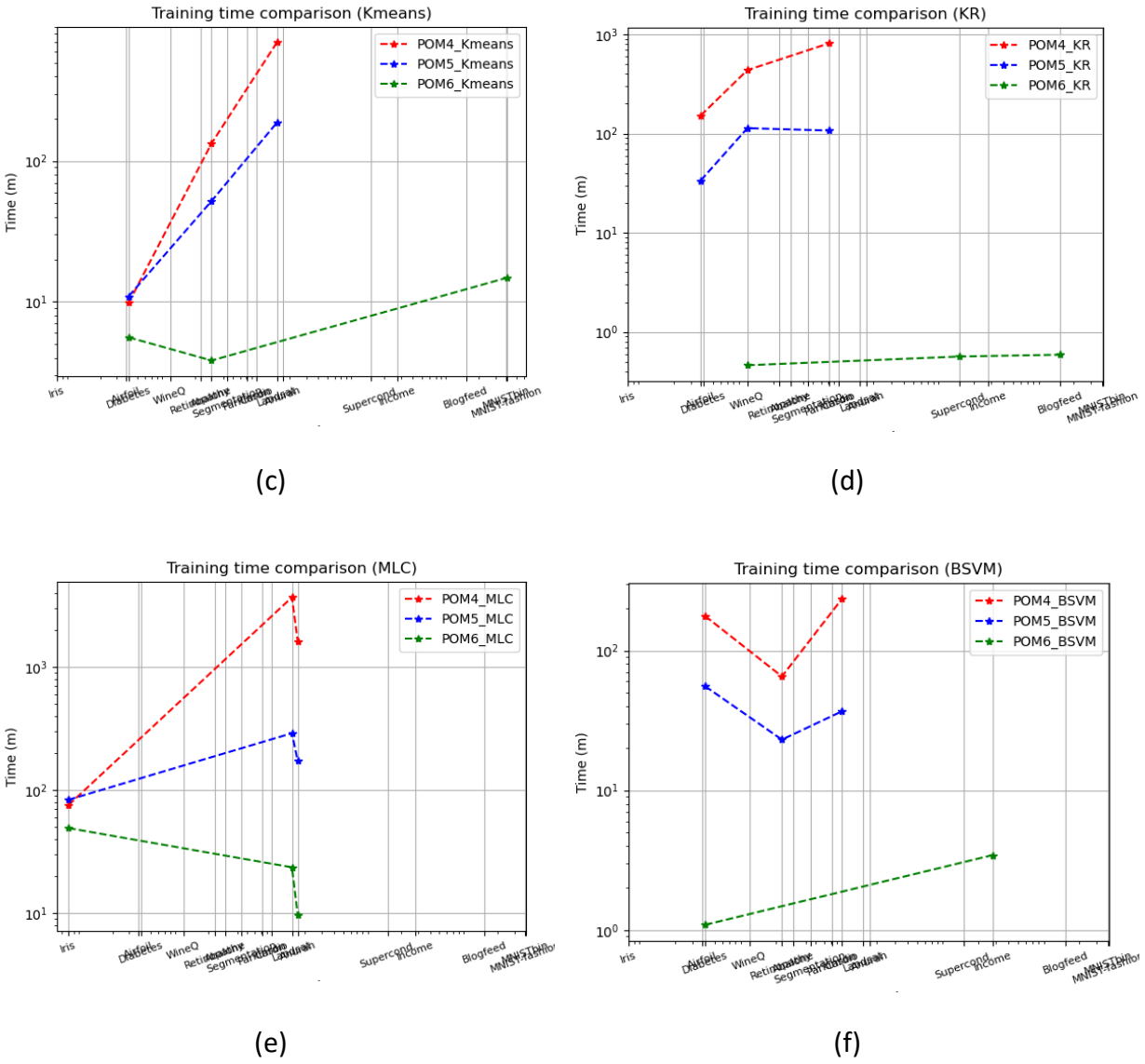


Figure 13. Total training time comparison for the different models under every POM. RR/LR in (a), LC in (b), Kmeans in (c), KR in (d), MLC in (e), BSVM in (f), MBSVM in (g).

We clearly observe, as already anticipated in Figure 7, that algorithms in POM6 are the most lightweight ones, and those in POM4 are the most computationally demanding, often requiring 2 orders of magnitude more time to complete the training process with respect to POM6 ones.

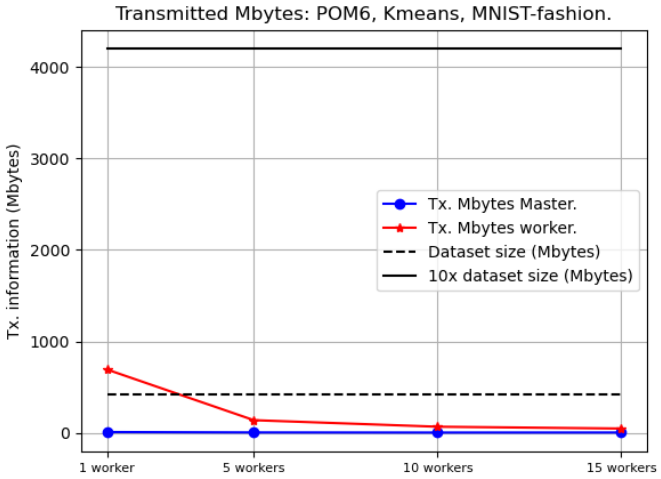
6.3.3 G1.3: Assessing the computational efficiency of MLA

This goal has three associated questions:

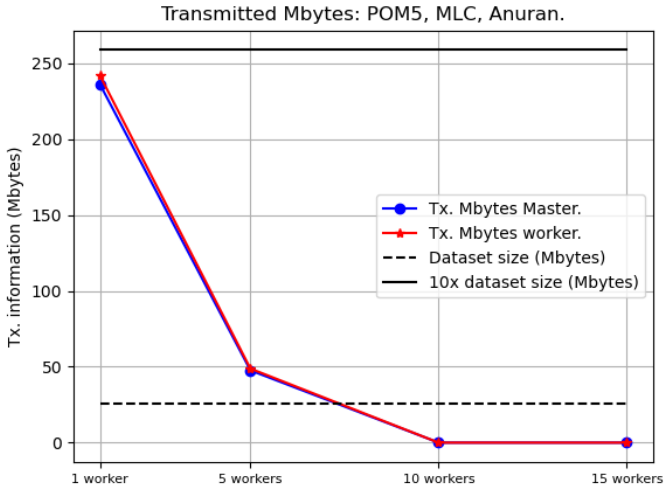
G1.4_Q1	Is MUSKETEER faster than competing platforms?

G1.4_Q2	Do we have a reasonable transmission cost?
G1.4_Q3	Is there a reasonable memory usage?

Concerning the first question, it comes out that we have found it difficult to carry out that comparison since the concept of "competing platform" is very dependent on the underlying security model and/or on the required security in the communications. We will postpone until Section 6.4 the analysis/discussion of this Question, and we concentrate by now in answering questions No. 2 and 3. To evaluate G1.4_Q2 we have measured the total amount of transmitted information and the total processing or transmission times, as depicted in the following Figures:



(a)

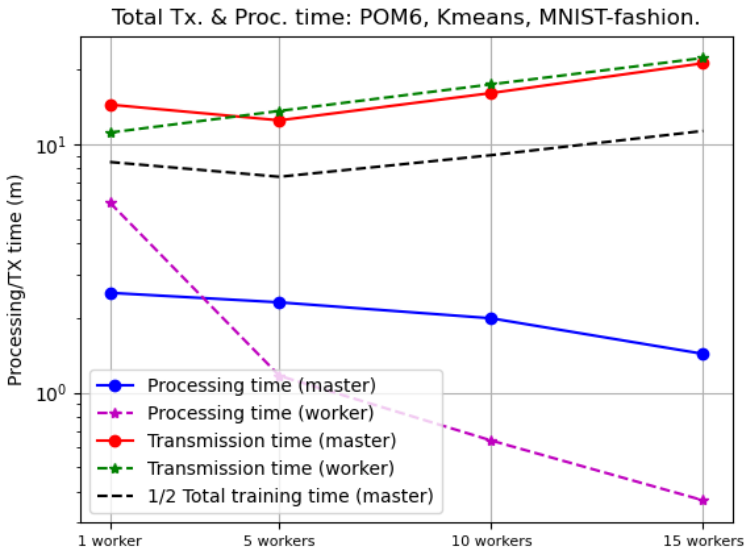


(b)

Figure 14. Examples of transmitted information, processing and transmission times as a function of the No. of workers.

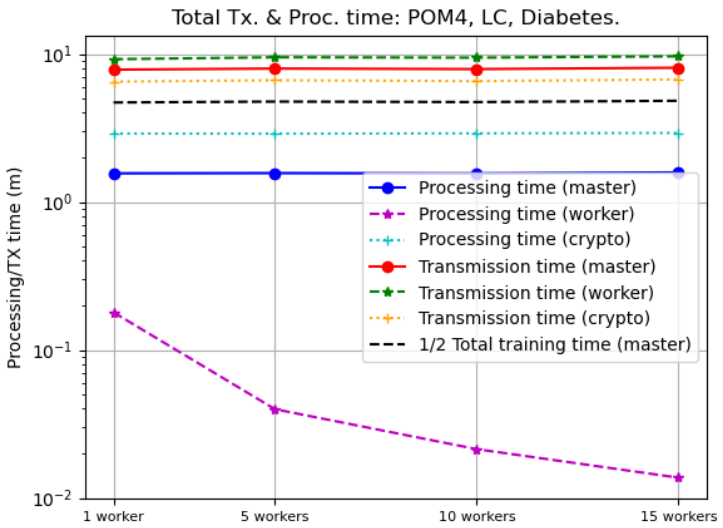
We observe in Figure 14 (a) that for this particular POM and algorithm, the amount of transmitted information by both the master and workers is below the reference thresholds (10x the dataset size[5]). In other cases, we have observed that, when the dataset is very small, the overhead transmission costs dominate and the ratio is higher, but for average or large sized dataset, the test is passed since the amount of transmitted information is mainly lower than 10 times the dataset size.

Concerning the fraction of time dedicated to processing vs. transmission, we have represented in Figure 15 below the corresponding curves.



(a)

[5] In POMs 4 and 5, that operate in the encrypted domain, the reference dataset size is defined using the encrypted version of the dataset, using the corresponding key length.



(b)

Figure 15. Examples of transmitted information as a function of the No. of workers.

We observe in Figure 15 that the transmission time is superior to the processing time in almost all cases. This is mainly caused by the slow rate of transmission over the internet and the specific network characteristics where the experiments have been run. It is always possible to balance this ratio by providing faster communication conditions to the platform. Anyhow, we will mark this test as NOT PASSED given the current assessment scenario.

6.4 Comparison with other competing platforms

When trying to complete the assessment for question G1.4_Q1: "Is MUSKETEER faster than competing platforms?" in the context of POMs 4, 5 and 6, our first problem was to properly identify those systems that could be considered as "competing platforms" under equal execution conditions. First of all, it is obvious that they should serve to the same purpose as the methods described in POMs 4, 5 and 6, i.e., training ML models while preserving the confidentiality of both training data and the resulting model. But not only that, to be fair in the comparison, those potential competitors must be based on the same boundary assumptions, otherwise the comparison would be unfounded.

We have identified a variety of systems that claim to serve for the same purpose, but many of them do not share the same conditions. In what follows (Subsection 6.4.1), we will firstly review the requirements to consider a system as a "fair" competitor and in Subsection 6.4.2 we will describe the main characteristics of the identified platforms for the Honest But Curious assumption (POMs 4, 5 and 6).

6.4.1 Assumptions to select potential competitors

Many of the systems described in the literature are proprietary or experimental systems and their code is not made publicly accessible, sometimes only briefly described in a published paper without access to any working code. Obviously, it is impossible to benchmark against those systems. In other cases, although some code is available, it is not ready to be used, or it is simply a demo illustrating some specific usage in a particular case. So a potential competing platform must be available/capable to run a variety of ML models.

Another important factor is the underlying security model, since it is not fair to compare systems that require different hypothesis to operate. For instance, a very popular trend is to use the Secure Multiparty Computing approach under the (arithmetic) Secret Sharing principles. This design claims to be highly efficient, since no encryption is needed to protect the data, and operations can be implemented on clear data, conveniently distributed in shares stored in several non-colluding servers. It is clear that the security model used under these approaches is very different to those in MUSKETEER, since the needed non-colluding multiple servers may not be available in many scenarios.

Another key aspect is that of communications. In many of the analysed potential competitors the means of communication between remote modules is a direct transfer, essentially presenting an interface that allows other modules to connect. This usually requires publishing details of an IP address and port number. The actual implementation of the communications can take different forms: direct socket communications, RESTful (REpresentational State Transfer, REST) APIs, gRPC remote procedure call, etc. These procedures that allow direct connections from the outside world may represent a potential security risk/threat (malicious attackers, man-in-the-middle attacks), ultimately compromising the entire system or even revealing sensitive raw data. In the MUSKETEER architecture, there are no direct connections between parties, and all interactions take place through the MUSKETEER central platform [Pycloudmessenger], which acts as a service broker, orchestrating and routing information between the different participants. In this way, only the connection details for the broker are made available, with all other entities, protected from direct attack. In this sense, we will disregard all those competing platforms that do not provide a communication means among different processes or the same level of security as MUSKETEER, since direct connections are usually much more effective (but also more insecure) and the comparison would be unfair.

Summarizing, these are the factors that we will take into consideration:

- Code must be publicly accessible and usable.

- The system must be ready to be used, with flexibility to choose different models and training datasets
- The underlying security model must be the same
- The implemented communications must be secure

6.4.2 Analysis of competing platforms and their characteristics

FRESCO is a FRamework for Efficient and Secure COmputation, written in Java and licensed under the open source MIT license [Fresco]. It is not a Machine Learning platform, it aims at supporting the development of both new applications using secure computation, and the development of new secure computation techniques.

HElib is a software library that provides low-level routines for homomorphic encryption (HE) [HElib]. Currently available is an implementation of the Brakerski-Gentry-Vaikuntanathan (BGV) scheme, along with many optimizations to make homomorphic evaluation runs faster, focusing mostly on effective use of the Smart-Vercauteren ciphertext packing techniques and the Gentry-Halevi-Smart optimizations. This library is written in C++ and uses the NTL mathematical library. It is not a Machine Learning suite, though.

HUSKY is a distributed framework to develop ML algorithms, with improved characteristics with respect to Spark, for instance. Husky offers a finer grain access than the synchronous map-reduce coarse grain approach used by Spark, and with a better asynchronous access. It is not a Machine Learning suite and it does not provide model confidentiality. [HUSKY] [Yang_2015]

NuCypher is a fully homomorphic encryption (FHE) library implemented in Python [NuCypher] and running on GPUs. It provides basic encryption/decryption functionalities, but not Machine Learning models.

PAPAYA (PlAtform for PrivAcY preserving data Analytics) develops a specific component named Privacy Engine (PE), with mechanisms to manage the privacy preferences, but also exercise the rights derivative from the GDPR (e.g. the right to erasure any personal data) [PAPAYA]. The main aim of the PAPAYA project is to make use of advanced cryptographic tools such as homomorphic encryption, secure two-party computation, differential privacy and/or functional encryption, to design and develop three main big data analytics operations. One application is Privacy preserving Neural Networks (PP-NN): this makes use of two-party computation and homomorphic encryption to enable a third-party server to perform neural network classification over encrypted data. It provides cryptographic primitives that can be exploited by future new data analytics modules. Mainly oriented towards data analytics over encrypted outsourced data. PAPAYA uses the advanced

cryptographic tools once the original neural network is modified in order to make it compatible with the actual cryptographic tool (for example, complex operations are approximated to low degree polynomials). In other cases, privacy-preserving collaborative training based on differential privacy is used. The code is not available.

Pyfhel: PYthon For Homomorphic Encryption Libraries, implements functionalities of multiple Homomorphic Encryption libraries such as addition, multiplication, exponentiation or scalar product in Python. Useful both for simple Homomorphic Encryption Demos as well as for complex problems such as Machine Learning algorithms, but the latter are not implemented. [Pyfhel]

PySyft is a generic framework for privacy preserving deep learning [Pysyft][Ryffel_2019]. Pysyft expands pytorch tensors to support DP, SMC (SPDZ protocol, secret sharing) and FL mechanisms. Incorporates federated learning, and with this framework you can either mitigate private data leakage with a trusted aggregator or use secure computation to keep the model updates encrypted until after they've been combined. They provide a standardized protocol to communicate tensors between workers using "pointers". Some previous versions of the library used to offer "VirtualWorkers" to operate locally, but this option is no longer available. It also accepts plain network sockets and web sockets. They rely on a security model that requires at least three non-colluding servers that hold the data shares. They report extremely good and fast results –possibly unbeatable-, under the SMC assumptions (secret sharing over several non-colluding servers). Pysyft currently relies on peer-to-peer communications instantiated by "the Duet" service (communications that do not seem to be end-to-end encrypted). We have not been able to initiate that communication service due to firewall restrictions in our institution, since the implemented (socket-based) communication means are essentially non-secure. The data owner receives individual access requests from Data Scientists and every one of them has to be manually approve or define an exhaustive list of permitted/denied operations.

TF-Encrypted is also a generic framework for privacy preserving Deep Learning [TFencrypted], that incorporates federated learning. The TFE github is unmaintained for two years, and currently it is not possible to execute the code due to incompatibilities with the most recent Tensorflow library.

SHAREMIND is a framework for Fast Privacy-Preserving Computations. [Bogdanov_2008] [SHAREMIND] A virtual machine for privacy-preserving data processing that relies on SMC techniques. Information-theoretically secure in the honest-but-curious model with three computing participants. All computations are done by dedicated miner parties, less susceptible for external corruption. Uses secret sharing and share computing techniques for privacy-preserving data aggregation. It is possible to try out the privacy-preserving

programming paradigm and estimate the running time of a given application in a fully encrypted environment, compatible with the Sharemind Application Server. The user has to implement the desired privacy preserving application using SecreC 2, a specific programming language. No ML algorithms are available.

CodedPrivateML is a fast and Privacy-Preserving Framework for Distributed Machine Learning [So_2019]. The software is not available.

CrypTen is a framework for Privacy Preserving Machine Learning built on PyTorch. It relies on SMC via arithmetic secret sharing, and it also requires the non-colluding property among the shareholders.

Tensorflow Privacy. Python library that includes implementations of TensorFlow optimizers for training machine learning models with differential privacy. [TFPrivacy]

6.5 Summary of results

Since the number of experiments is very large, as depicted in Figures collected in Section 10 (Appendix I), we will summarize here the observed results for every algorithm and POM, in the form of a Table that indicates whether or not the GQM test is passed. We will use the results in this Table to draw the final conclusions about the performance of the implemented MMLL library and also to define some recommendations for the end users. Before going into detail about the results obtained with every POM, let us recall that the architecture of the models used across the different POMs is essentially the same, the only differences are in the way some computations take place, or the solver method that has been implemented in every POM given the operational limitations imposed by every approach.

Although a detailed description of Goals, Subgoals and Metrics is included in D6.1, we briefly summarize here the main concepts:

G1.1: Assessing the performance of MLA: Are the results comparable (within 5%) with scikit learn?

G1.2: Assessing the reliability of MLA: Is the variation in different runs less than 5%?

G1.3: Assessing the scalability of MLA

- G1.3_Q1: Is the training time less than quadratic w.r.t. no patterns?
- G1.3_Q2: Is the training time less than quadratic w.r.t. no users?
- G1.3_Q3: Training time less than quadratic w.r.t. no features?

G1.4: Assessing the computational efficiency of MLA

- G1.4_Q1: Faster than competing platforms (if any is available)?
- G1.4_Q2: Is the transmission cost reasonable?

Is the transmission ratio less than 10?

Is the time dedicated to transmission less than half the total training time?

G1.4_Q3: Is the memory usage reasonable (storage ratio is less than 10) w.r.t. number of o workers.

6.5.1 Assessment of algorithms under POM4

Table 4. Summary of GQM tests results for algorithms in POM4

		LR	LC	Kmeans	KR	MLC	BSVM
Performance	G1.1	OK	OK	OK	NO	OK-(*)	OK-(*)
	G1.2	OK	OK	OK	OK	OK	OK
Scalability	G1.3_Q1	L	L	L	L	L	L
	G1.3_Q2	L	L	L	L	L	L
	G1.3_Q3	L	L	L	L	L	L
Efficiency	G1.4_Q1	-	-	-	-	-	-
	G1.4_Q2	NO-(**)	NO-(**)	NO-(**)	NO-(**)	NO-(**)	NO-(**)
	G1.4_Q3	OK	OK	OK	OK	OK	OK

(*) In this case gradient descent was used and two out of three datasets gave poor performance, hence the NO+.
(**) For transmissions over the internet.

As already discussed in the general section POM4 is the most computationally demanding approach, since the whole dataset is encrypted and both computations and communications are increased accordingly. The main advantage (but also disadvantage) of this POM is that the training data is outsourced to the aggregator and the workers do not need to contribute/participate/operate during the training process. This may represent a potential leakage risk but as a positive characteristic the workers do not need to spend computational power during training, so they can be more lightweight. The cooperation of a non-colluding process (cryptonode) is needed in this POM. Again, this can be seen as a potential –leakage-weakness or as an opportunity to speed up computations if the aggregator and cryptonode are able to count with especially powerful computing resources that can speed up operations in the encrypted domain by several orders of magnitude.

Taking all this into account, you may observe in Table above that most GQM tests are positive for the simpler models, but we observe convergence problems in the more complex

ones. One fundamental restriction in POM4 is that we only have access to simple solvers (mainly those relying on gradient descent and its variants such as "momentum"), since second order statistics computations are prohibitive. In some complex models, the gradient descent has to operate in a very large –possibly non-convex and ill-conditioned- space, and the optimization mechanism may have serious problems to achieve a competitive solution (global minimum of the cost function).

The test that presents more problems is that associated to G1.4_Q1 (Is transmission time less that computation time?). As discussed in Subsection 6.3.3, the transmission times at aggregator and cryptonode are not lower than the processing times, so we have marked that test as "not passed". As also discussed, improved communication speed in an alternative scenario could facilitate to positively pass this test.

6.5.2 Assessment of algorithms under POM5

Table 5. Summary of GQM tests results for algorithms in POM5

		LR	LC	Kmeans	KR	MLC	BSVM	MBSVM
Performance	G1.1	OK	OK	OK	NO+(*)	OK	OK	OK
	G1.2	OK	OK	OK	OK	OK	OK	OK
Scalability	G1.3_Q1 (np)	L	L	L	L-	L	L	L
	G1.3_Q2 (nw)	L	L	L-	L-	L	L	L
	G1.3_Q3 (nf)	L	L	L	L-	L	L	L
Efficiency	G1.4_Q1	-	-	-	-	-	-	-
	G1.4_Q2	NO(**)	NO(**)	NO(**)	NO(**)	NO(**)	NO(**)	NO-(**)
	G1.4_Q3	OK	OK	OK	OK	OK	OK	OK

(*) In this case gradient descent was used and two out of three datasets gave poor performance, hence the NO+.

(**) For transmissions over the internet.

In POM 5, only the model is encrypted and sent to the workers to operate with it, the aggregator acting as a sort of cryptonode, giving cooperation to the workers to solve the unsupported operations. In this case there is no need for non-colluding third parties. As

already discussed in Subsection 6.1, this POM is not as computationally demanding as POM4, but the operation is still costly, since it takes place in the encrypted domain. However, in this case, the workers are able to compute second order statistics (e.g., correlation/covariance matrices), and the aggregator can implement improved solver methods with respect to vanilla gradient descent. As a result, performance assessment is positive in all cases but one (the Kernel Regression case), where gradient descent was used. We also observe that, apart from minor deviations in some of the experiments, the rest of GQM tests are also passed positively. Also in this case, the efficiency criterion metric (communication time less than half the processing time) is not passed, hence the "NO" in the G1.4_Q2 part (the other metric is positively passed).

6.5.3 Assessment of algorithms under POM6

Table 6. Summary of GQM tests results for algorithms in POM6

		RR	LC	Kmeans	KR	MLC	BSVM	MBSVM
Performance	G1.1	OK	OK	OK	OK	OK	OK	OK
	G1.2	OK	OK	OK	OK	OK	OK	OK
Scalability	G1.3_Q1 (np)	L	L	L	L	L	L	L
	G1.3_Q2 (nw)	L	L	L	L	L	L	L
	G1.3_Q3 (nf)	L	L	L	L	L	L	L
Efficiency	G1.4_Q1	-	-	-	-		-	-
	G1.4_Q2	NO(*)	NO(*)	NO(*)	NO(*)	NO(*)	NO(*)	NO(*)
	G1.4_Q3	OK	OK	OK	OK	OK	OK	OK-

(*) For transmissions over the internet.

We observe in POM6 that the performance tests are always positive, since the used solvers are more efficient than gradient descent and they are able to provide solutions comparable to scikit-learn. Also, scalability is very good. As in the previous cases, the high transmission times force that in this case the metric "communication time less than half the processing time" is not passed either, since the computation times are much reduced with respect to the previous POMs, but communications overheads do not reduce in the same proportion.

7 Simple guidelines for selecting the most appropriate POM.

In what follows, by answering some simple questions, the interested user may decide which POM is the best for a particular task/MUSKETEER deployment. We will use here the following terms:

- **Master or aggregator**: the process that centralizes/manages the training process.
- **Worker or participant**: every process that participates in the training mainly providing a portion of the training data.
- **Public model**: the final trained model is known to all, aggregator and participants.
- **Secret model**: a selective access to the final trained model is needed: only the aggregator or the participants get the model.

Q1: Do I need to protect the model?

- No, I want that the final trained model is **Public** to all participants:

  Use **POM1,** which corresponds to a standard Federated Learning implementation. Computations are fast (no encryption is needed), and a potentially large number of models with unlimited complexity is available (e.g., those provided by Tensorflow/Kheras libraries). Partial information from the participants is shared, mainly average gradients, that can be further be protected using Differential Privacy.

- **Yes**, I want that the final trained model is **Private**:

  Q2: Who gets the trained model)?

  - Only the **workers get the model**:

    - **Q3: Do workers trust each other?**

      - **Yes**: Use **POM2,** the model is hidden to the aggregator, encryption is implemented with a single private key in every worker.

      - **No**: Use **POM3,** the model is hidden to the aggregator, encryption is implemented with different private key different for the different data owners.

  - Only the **aggregator gets the model**:

- **Q4: can some partial information/statistics be exposed?**

  - **Yes**:

    Use **POM6,** since **i**t is fast (no encryption is used, a Secure Dot Product Two-Party protocol is used to compute Secure Dot Products, and Random Matrix Disguise is used to obtain averaged statistics), but not all models are feasible, and partial aggregated information (averaged gradient, covariance matrices) may be revealed.

  - **No**:

    Use **POM4** or **POM5**:

    Q5: can I run a process (cryptoprocessor) on a separate entity non-colluding with the aggregator?

    - **Yes**:

      Use **POM 4**: all training process takes place on encrypted values, only the trained model is decrypted by the aggregator for further use outside MUSKETEER. Workers do not actively cooperate in the computations during the training process, they just encrypt and send their encrypted data to the aggregator.

    - **No**:

      Use **POM5**: data does not leave the participants, but they need to collaborate in the training process with some operations on the encrypted data. The participants only receive encrypted information from the aggregator.

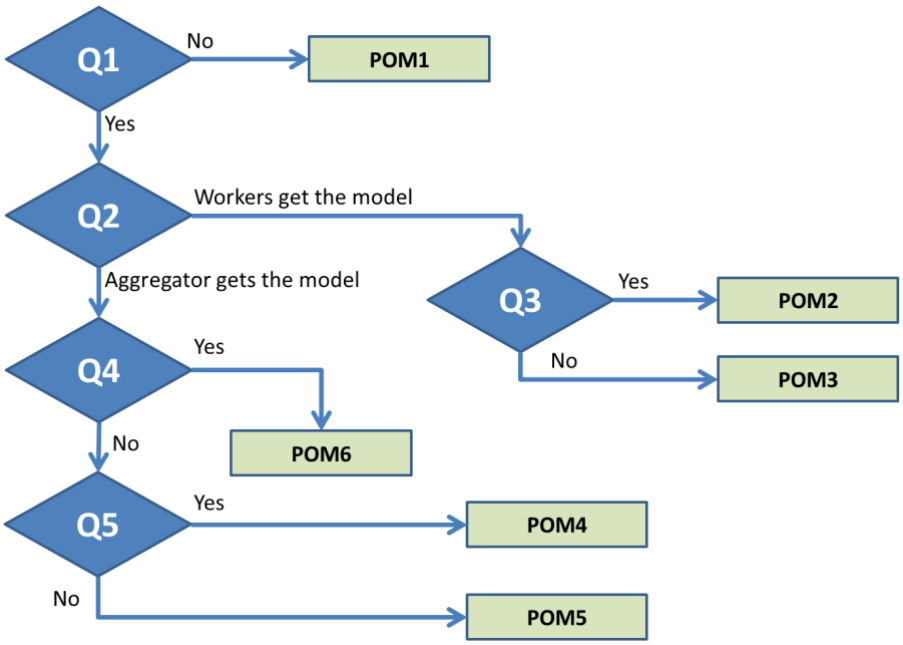


Figure 16. MUSKETEER's graphical guide to select the appropriate POM

8 Conclusions

In this document we have presented the results of the assessment phase corresponding to Task 6.2: performance and scalability of the Machine Learning algorithms implemented under the different POMs. We provide the experimental results and metrics that allow us to answer the questions described in D6.1. As a conclusion, we have found that:

- We have implemented and benchmarked a variety of Machine Learning Models to solve tasks such as prediction, classification or clustering, under the different Privacy Operation Modes (POMs), as described in the GA/DOW. The implemented models and training procedures comprise the MUSKETEER Machine Learning Library (MMLL).
- We have benchmarked here the implemented MMLL methods from the performance and scalability perspective. All experiments have been run in real world situations, i.e., every participant runs in a single process, and all communications take place through the Cloud Communication service [Pycloudmessenger], an experimental design that mimics the future operation of the MUSKETEER platform in the real world.
- There are two main groups of POMs: those that rely on Federated Learning assumptions (e.g., POMs 1, 2 and 3), and those that are not Federated and rely on the Honest but Curious assumption (POMs 4, 5 and 6), and besides protecting the

data confidentiality, also hide the model from the end users. In what follows we will describe the main conclusions observed about every one of these POMs:

- **POM1**: Under this paradigm, a shared global model is trained under the coordination of a central node, from a federation of participating devices. FML enables different devices to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to perform machine learning from the need to store the data in the cloud. At the end of the training, the master and every worker node have a copy of the model. Since there is not encryption, this is the fastest of the FL POMs, making possible to work with large scale datasets in a reasonable amount of time.
- **POM2**: In POM1 data information may be leaked to an honest-but-curious server since the server has access to the predictive model. In some use cases, data owners belong to the same company (e.g. different factories of the same company) and the server that orchestrates the training is in the cloud. POM2 fixes that problem with two properties.
  - No information is leaked to the server: POM2 leaks no information of participants to the honest-but-curious cloud server.
  - The accuracy is kept intact compared to POM1: Achieves identical accuracy to a corresponding system trained using stochastic gradient descent.

  For small datasets it provides similar scalability than POM1. However, for large ML models, the computational cost associated to the encryption and decryption increases the training time, making it impractical for large scale problems.
- **POM3**: In POM2, every data owner trusts each other and they can share the private key of the homomorphic encryption (e.g. different servers with data that belongs to the same owner). Using the same key, every data owner uses the same encrypted domain. In many situations it is not possible to transfer the private key in a safe way. POM3 is an extension of POM2 that makes use of a proxy re-encryption protocol to allow that every data owner can handle her/his own private key.

  Due to this re-encryption mechanism, the training procedure is sequential worker by worker instead on parallel. That why this POM does not scale correctly with the number of workers.

- **POM4**: In this case the training data is outsourced (after encryption) and the main advantage is that the end users (data providers) do not need to cooperate/participate during the training process; all computations take place with the interaction of the aggregator and cryptonode, that must not collude. If the aggregator and cryptonode are able to provide especially powerful computational means, the training time can be proportionally reduced, the computational power of the other participants (data providers) not being so relevant. On the bad side, all operations take place in the encrypted domain (Homomorphic Encryption), with the corresponding computational and transmission overheads. Also, the range of available operations is limited, so the solver methods that can be implemented are mainly restricted to gradient descent and some simple variants (e.g. momentum). We have observed some convergence problems in complex models with large parameters space and non-convex (possibly ill-conditioned) error function surfaces.
- **POM5**: Under this approach, the data is not encrypted and never leaves the data provider facilities; it is the model parameters that are encrypted, to protect the final trained model from the participants (only the aggregator gets that model). So, the computational burden is somehow reduced with respect to POM4, and it is also affordable to compute some other second order statistics that facilitate improved solver methods. Anyhow, those statistics are also exchanged in encrypted form, so the transmission costs are still high. As a result, we obtain improved performance results with respect to POM4 at a lower computational cost. However, in this case, the participants need to cooperate with the aggregator during the training process, so they must have as much computational capabilities as possible. Partial information such as model outputs on the training data or aggregated statistics (e.g., gradients, covariance matrices), are shared with the aggregator.
- **POM6**: This is the most lightweight approach among those that hide the model from the participants, since no encryption is used. Instead, some two-party protocols to compute a Secure Dot Product are used, as well as other information hiding approaches such as Random Matrix Disguise (RMD) approaches. The exchanged information is not encrypted, so the communication requirements are much reduced, and since the computations take place in clear text form, the speedup with respect to POMs 4 and 5 is also remarkable.

As a final summary of results, we have provided in Section 7 a simple guide that may help the end user to select the POM that is more appropriate to the task at hand, mainly taking

into account which information is being protected (and from whom), as well as the computational and communication requirements needed in every case.

9 References

[Abalone] https://archive.ics.uci.edu/ml/datasets/abalone

[Airfoil] https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise

[BlogFeedback] https://archive.ics.uci.edu/ml/datasets/BlogFeedback

[Bogdanov_2008] Bogdanov D., Laur S., Willemson J. (2008). Sharemind: A Framework for Fast Privacy-Preserving Computations. IACR Cryptology ePrint Archive.

[Boston] http://lib.stat.cmu.edu/datasets/boston

[Cardio] https://archive.ics.uci.edu/ml/datasets/Cardiotocography

[CrypTen] https://github.com/facebookresearch/CrypTen

[Diabetes] https://github.com/LamaHamadeh/Pima-Indians-Diabetes-DataSet-UCI

[Fashion-MNIST] https://github.com/zalandoresearch/fashion-mnist

[Fresco] http://fresco.readthedocs.org

[Grant Agreement] Grant Agreement number: 824988 — MUSKETEER — H2020-ICT-2018-2020/H2020-ICT-2018-2

[HElib] https://homenc.github.io/HElib/

[HUSKY] https://github.com/husky-team/husky

[Iris] https://archive.ics.uci.edu/ml/datasets/iris

[Landsat] https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)

[MMLL_2021] https://github.com/Musketeer-H2020/MMLL

[MNIST] http://yann.lecun.com/exdb/mnist/

[NuCypher] https://github.com/nucypher/nufhe

[PAPAYA] https://www.papaya-project.eu

[Parkinson] https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring

[Pycloudmessenger] https://github.com/IBM/pycloudmessenger/

[Pyfhel] https://pypi.org/project/Pyfhel/

[Retinopathy] https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set

[Ryffel_2019] Ryffel, Theo & Trask, Andrew & Dahl, Morten & Wagner, Bobby & Mancuso, Jason & Rueckert, Daniel & Passerat-Palmbach, Jonathan. (2018). A generic framework for privacy preserving deep learning.

[scikit] https://scikit-learn.org/

[Segmentation] https://archive.ics.uci.edu/ml/datasets/Statlog+%28Image+Segmentation%29

[SHAREMIND] https://sharemind-sdk.github.io/

[So_2019] So, Jinhyun & Guler, Basak & Avestimehr, A. (2021). CodedPrivateML: A Fast and Privacy-Preserving Framework for Distributed Machine Learning. IEEE Journal on Selected Areas in Information Theory. PP. 1-1.

[SODA] https://www.soda-project.eu/

[Solingen_1999] R. van Solingen, E. Berghout (1999). The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development, McGraw-Hill

[Superconductivity] https://archive.ics.uci.edu/ml/datasets/Superconductivty+Data

[TFencrypted] https://github.com/tf-encrypted/tf-encrypted

[TFPrivacy] https://github.com/tensorflow/privacy

[Redwine] https://archive.ics.uci.edu/ml/datasets/wine+quality

[Yang_2015] Fan Yang, Jinfeng Li, James Cheng (2015). Husky: Towards a More Efficient and Expressive Distributed Computing Framework. PVLDB Volume 9 Issue 5.

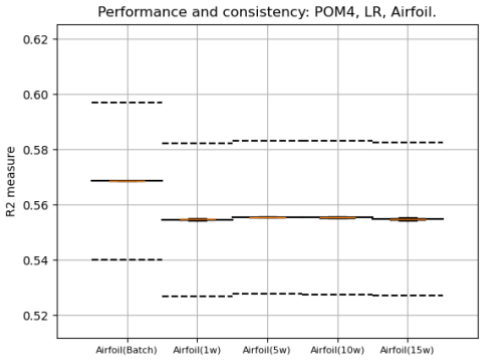
[Zhu_2015] Y. Zhu, T. Takagi, (2015) Efficient scalar product protocol and its privacy-preserving application, International Journal of Electronic Security and Digital Forensics 7, pp. 1–19.

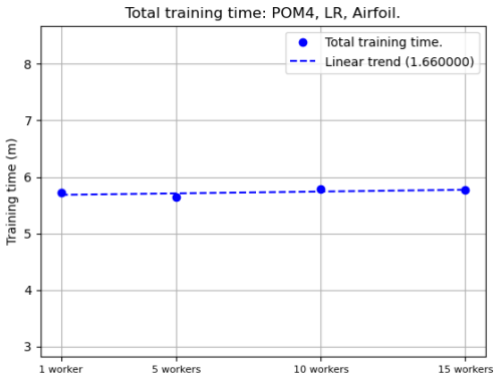
10 Appendix I: Full set of figures corresponding to POMs 4, 5 and 6

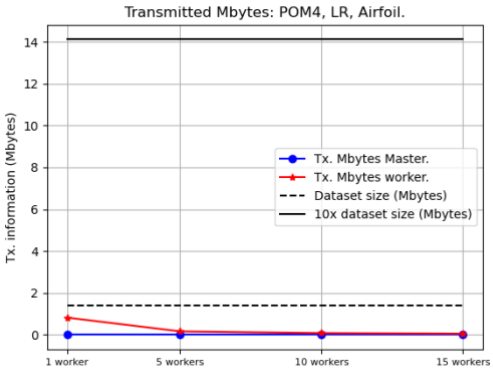
10.1 POM 4

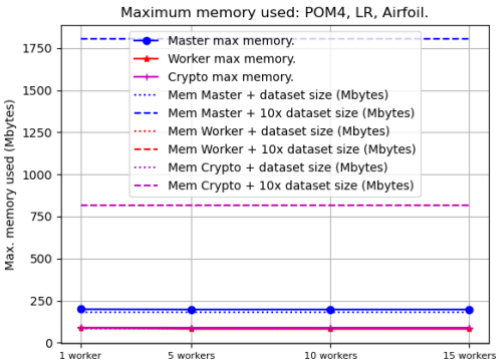
10.1.1 Linear Regression (LR)

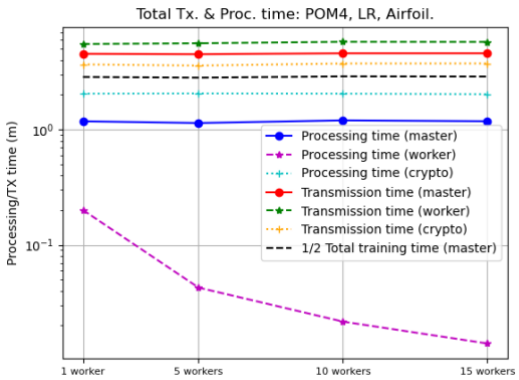
Airfoil



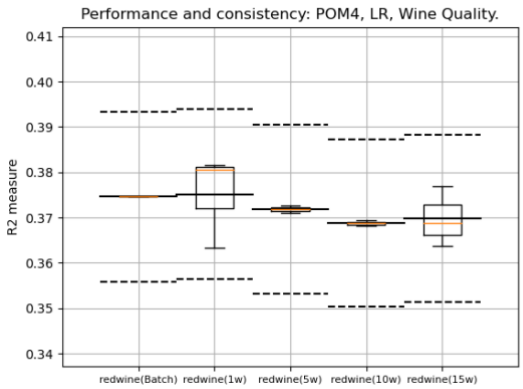


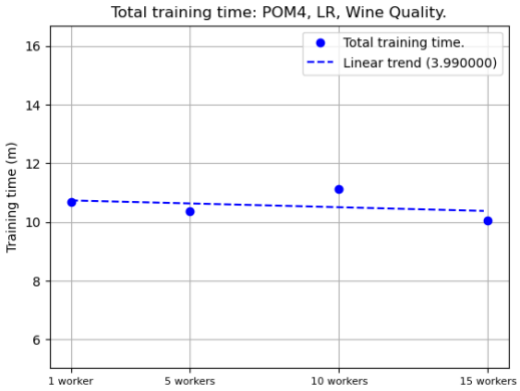


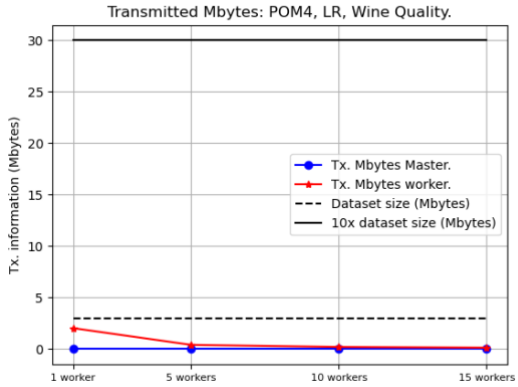


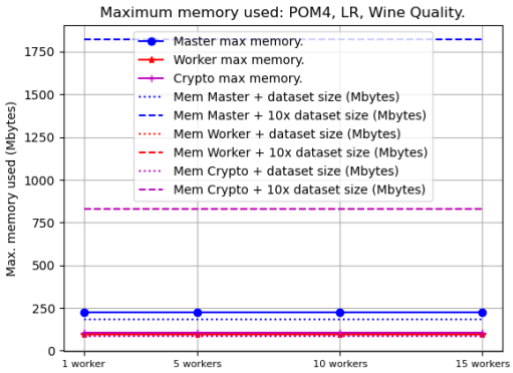


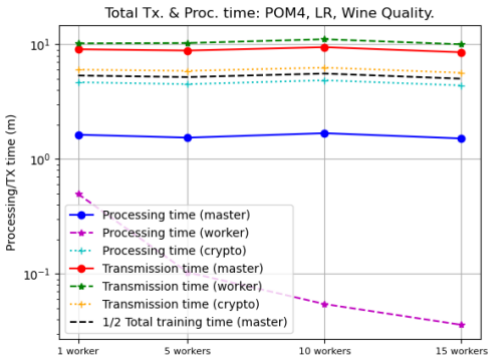
Wine quality



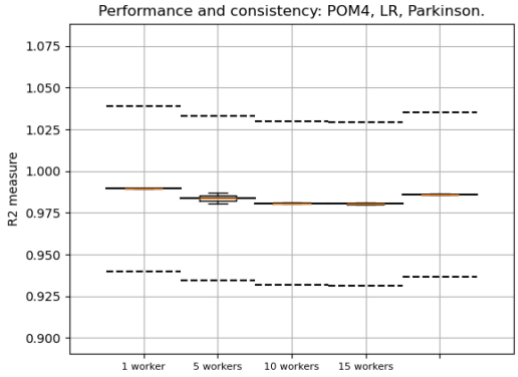


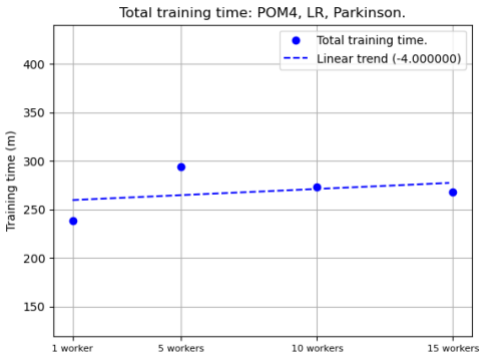


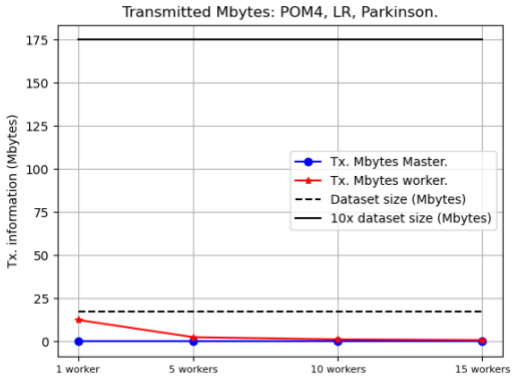


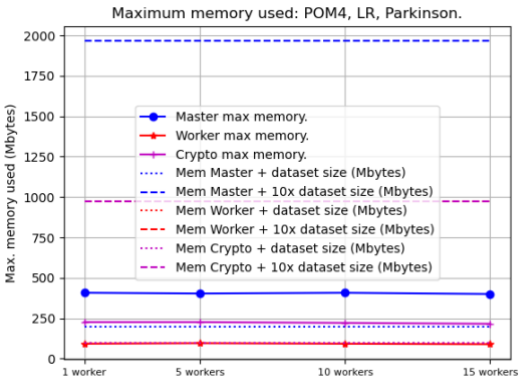


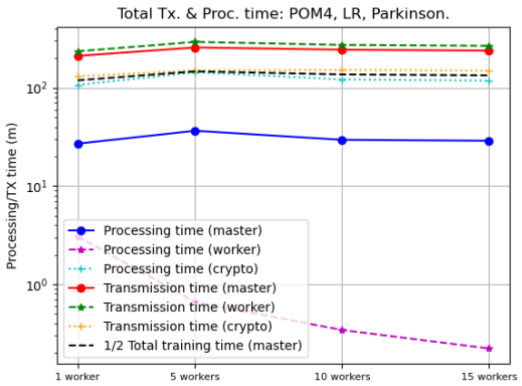
Parkinson


Performance and consistency: POM4, LR, Parkinson.
R2 measure
1.075
1.050
1.025
1.000
0.975
0.950
0.925
0.900
1 worker
5 workers
10 workers
15 workers

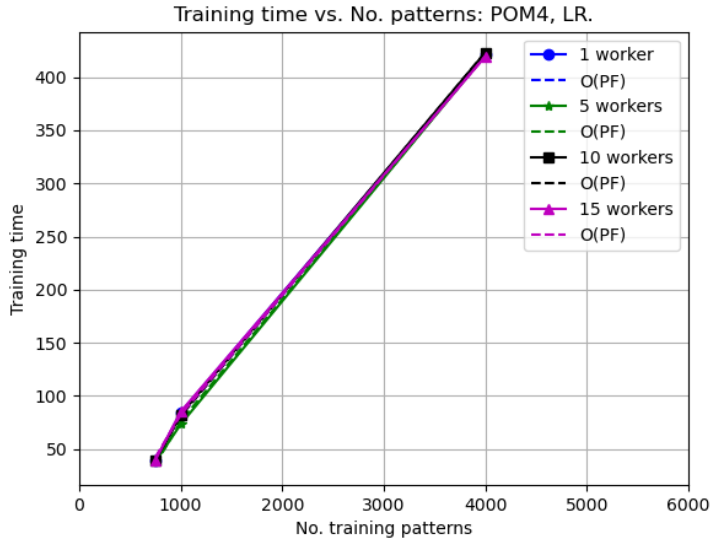

Total training time: POM4, LR, Parkinson.
Total training time.
Linear trend (-4.000000)
Training time (m)
400
350
300
250
200
150
1 worker
5 workers
10 workers
15 workers

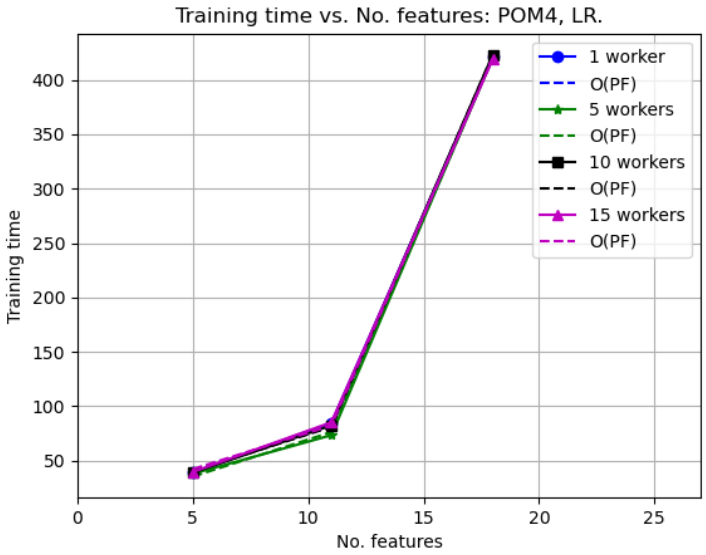

Transmitted Mbytes: POM4, LR, Parkinson.
Tx. information (Mbytes)
175
150
125
100
75
50
25
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Maximum memory used: POM4, LR, Parkinson.
Max. memory used (Mbytes)
2000
1750
1500
1250
1000
750
500
250
0
Master max memory.
Worker max memory.
Crypto max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Mem Crypto + dataset size (Mbytes)
Mem Crypto + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM4, LR, Parkinson.
Processing/TX time (m)
10^2
10^1
10^0
Processing time (master)
Processing time (worker)
Processing time (crypto)
Transmission time (master)
Transmission time (worker)
Transmission time (crypto)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

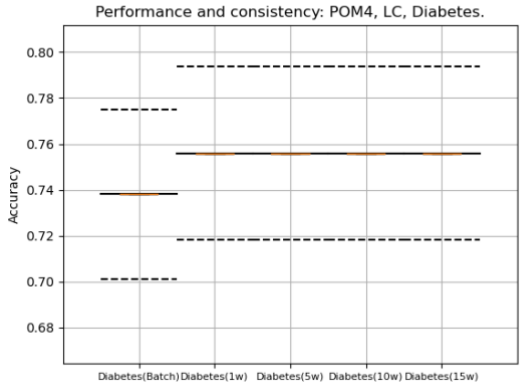
Scalability with respect to No. training patterns, number of features/model size.

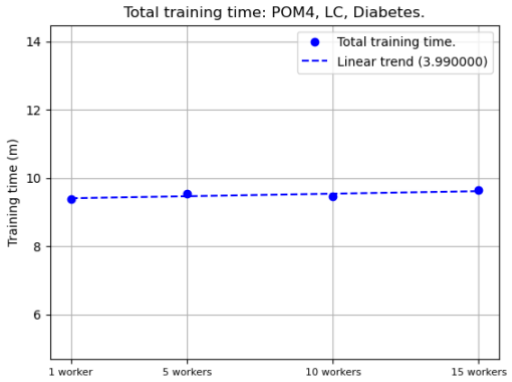


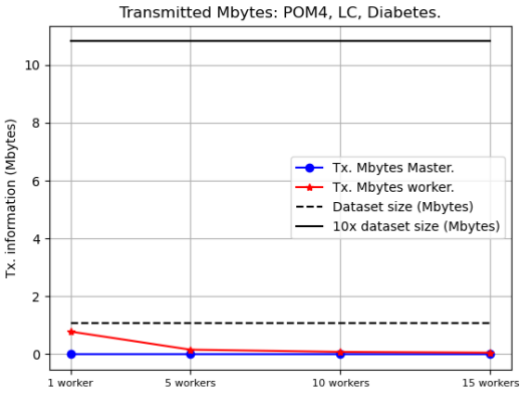


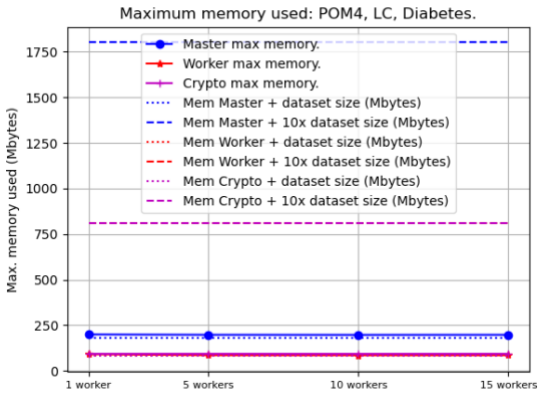
10.1.2 Logistic Classifier (LC)

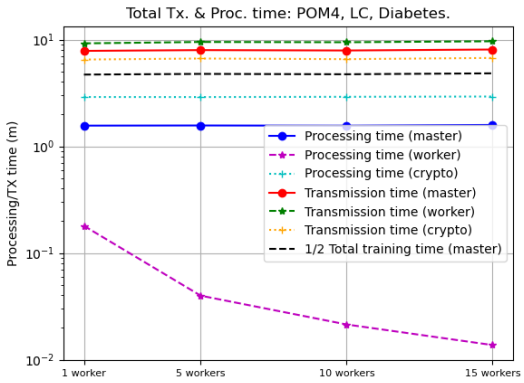
Diabetes



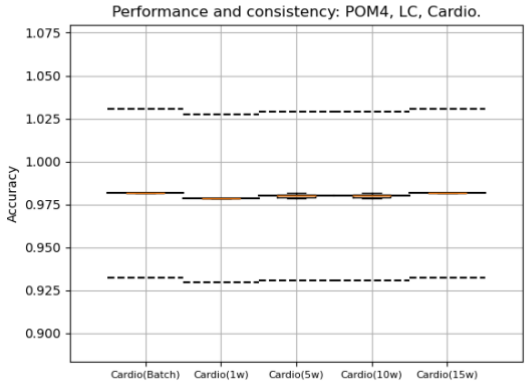


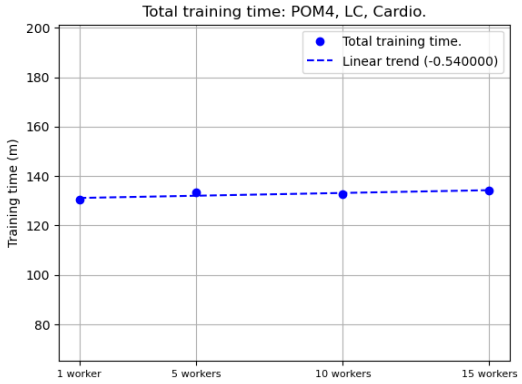


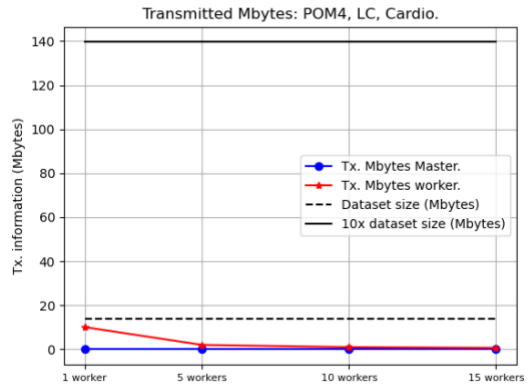


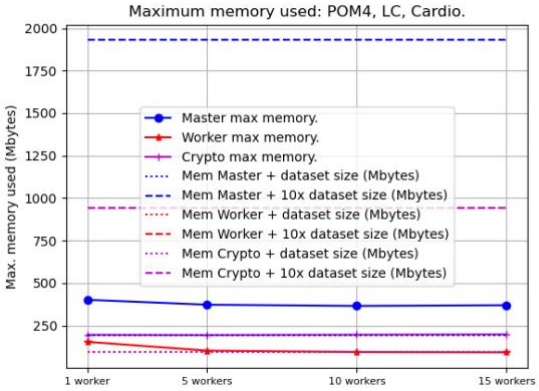


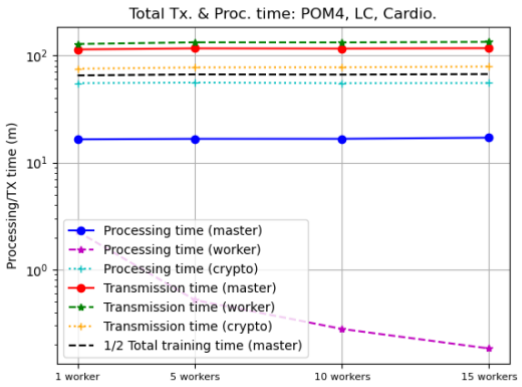
Cardiopathy



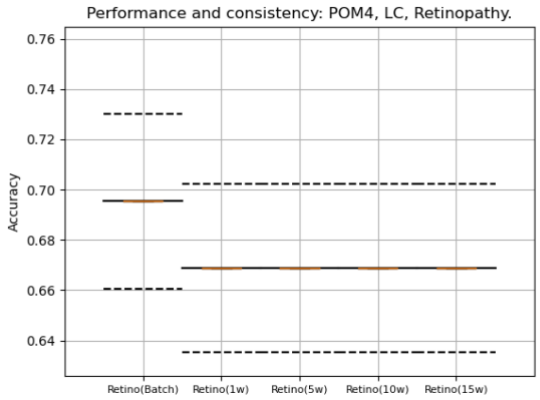


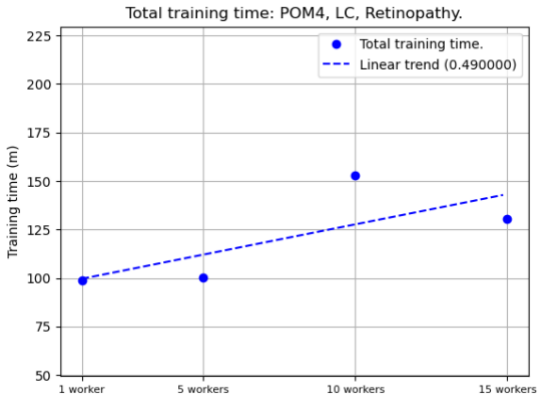


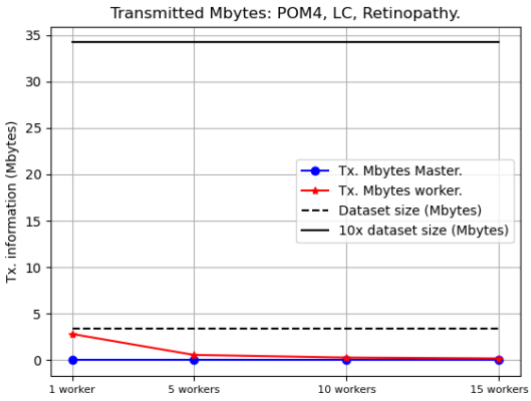


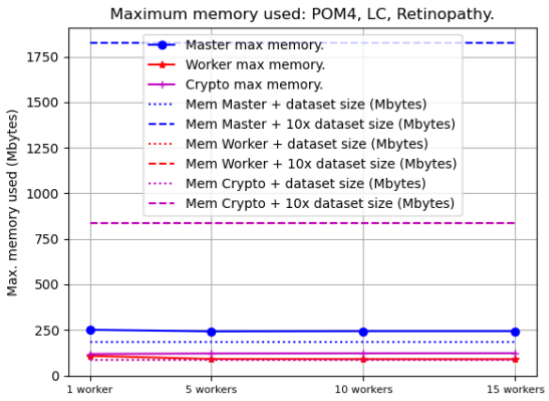


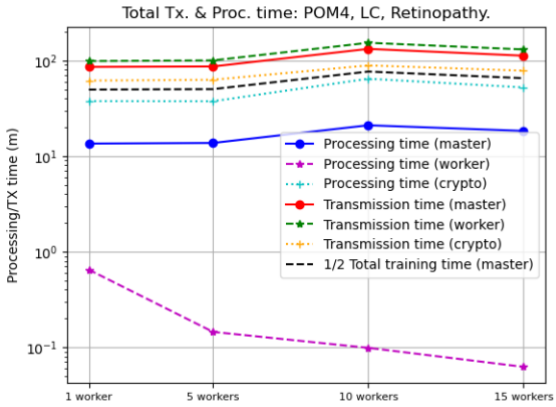
Retinopathy



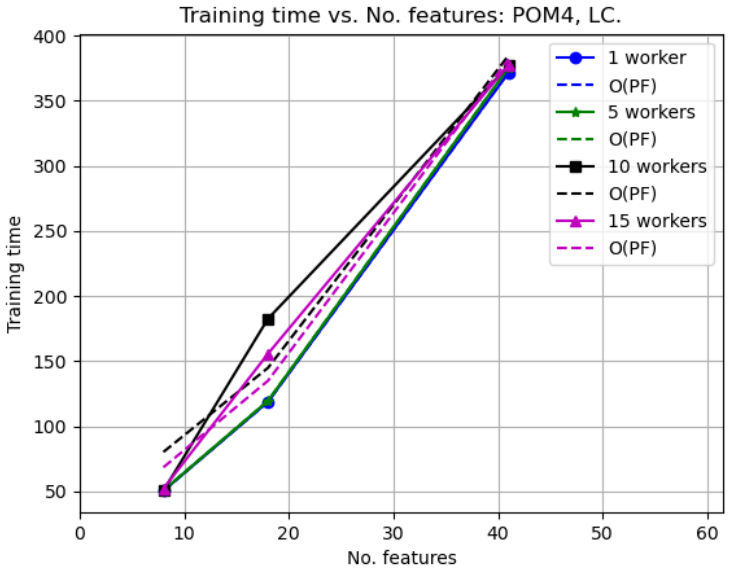


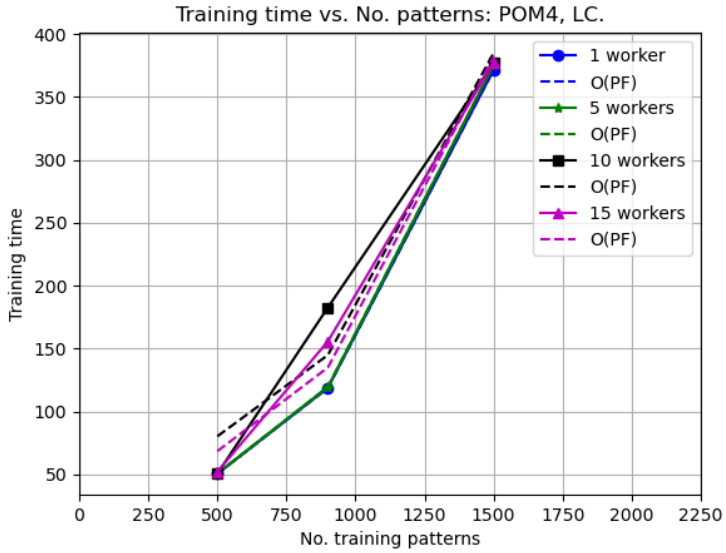






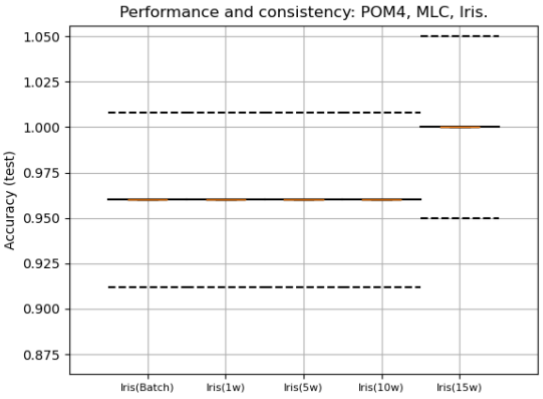
Scalability with respect to No. training patterns, number of features/model size.

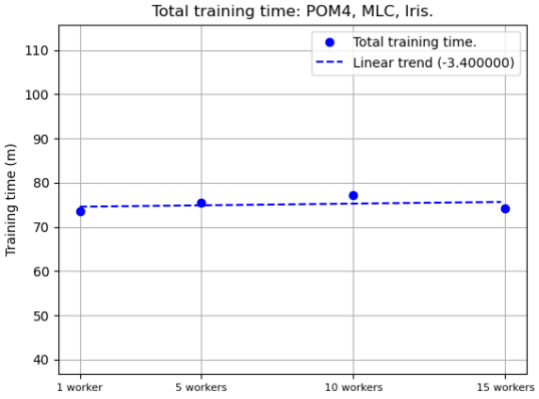


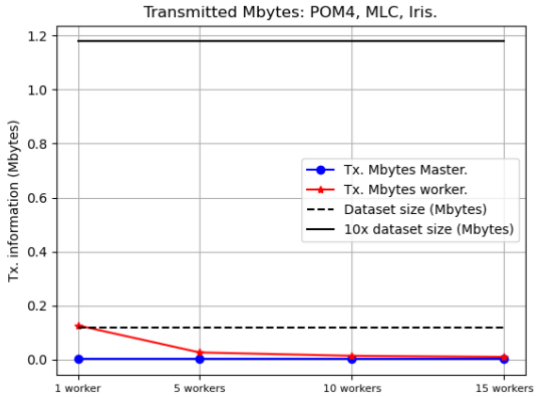


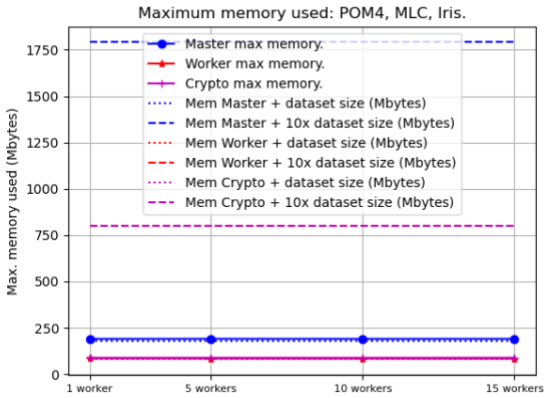
10.1.3 Multiclass Logistic Classifier (MLC)

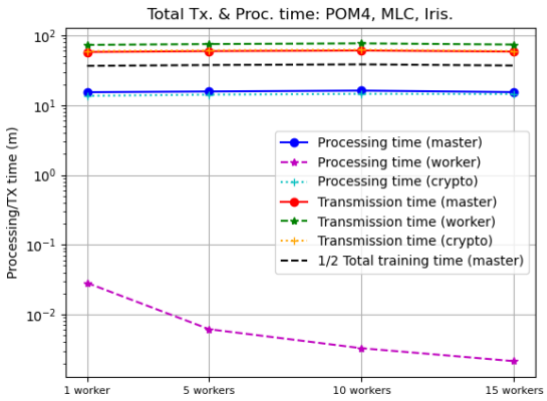
Iris



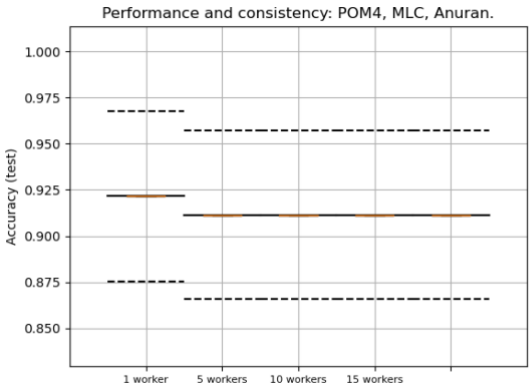


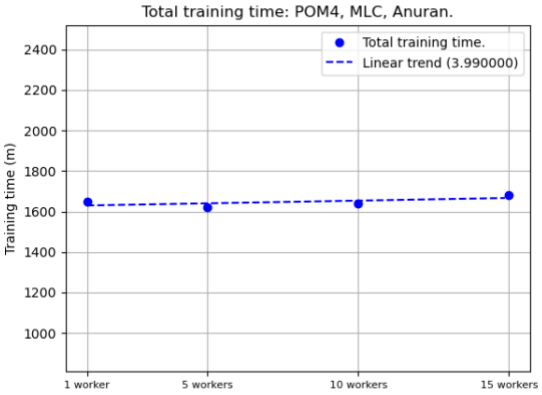


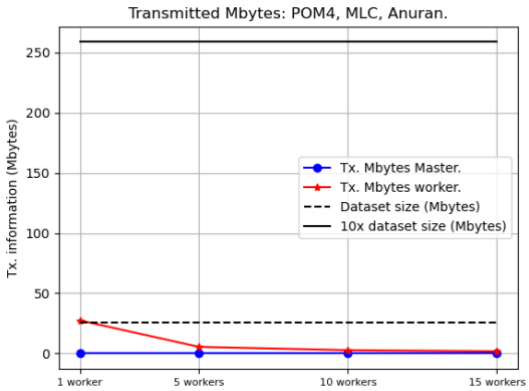


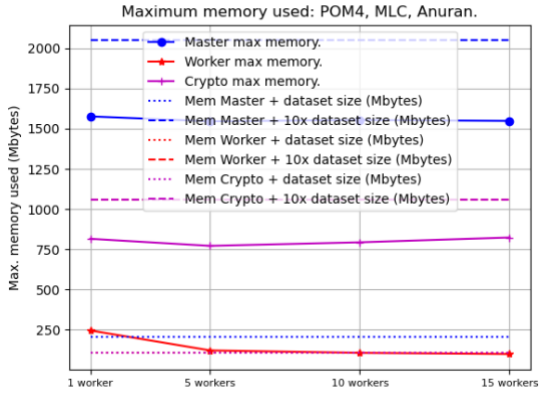


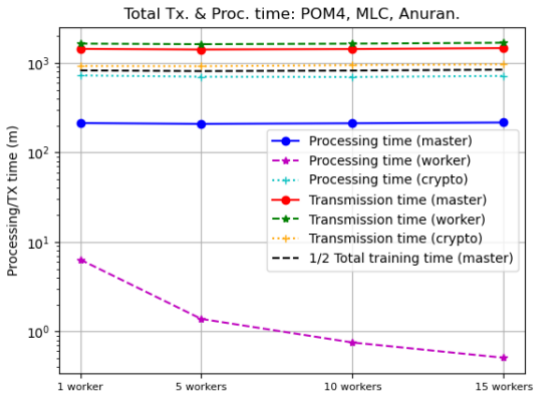
Anuran


Performance and consistency: POM4, MLC, Anuran.
Accuracy (test)
1.000
0.975
0.950
0.925
0.900
0.875
0.850
1 worker
5 workers
10 workers
15 workers

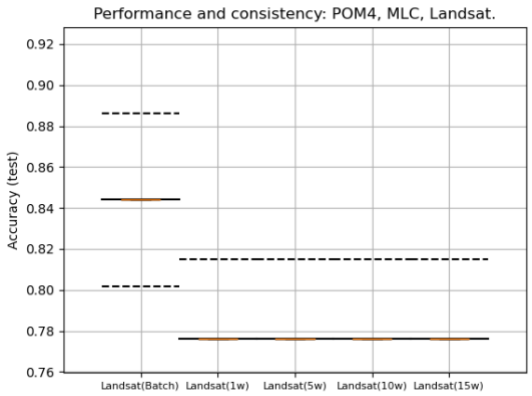

Total training time: POM4, MLC, Anuran.
Total training time.
Linear trend (3.990000)
Training time (m)
2400
2200
2000
1800
1600
1400
1200
1000
1 worker
5 workers
10 workers
15 workers

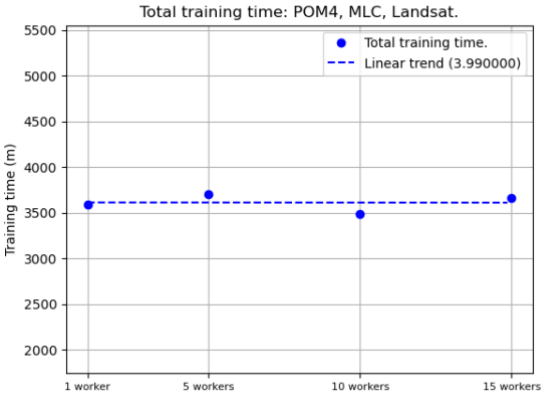

Transmitted Mbytes: POM4, MLC, Anuran.
Tx. information (Mbytes)
250
200
150
100
50
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

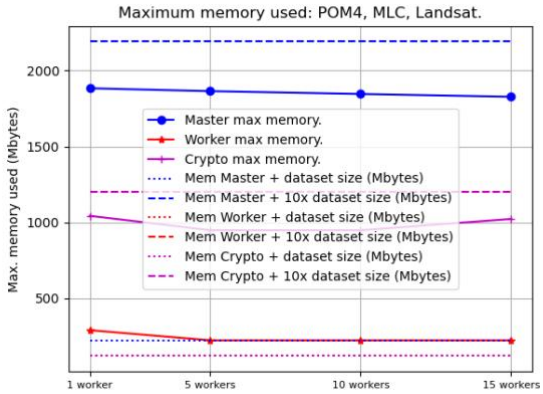

Maximum memory used: POM4, MLC, Anuran.
Max. memory used (Mbytes)
2000
1750
1500
1250
1000
750
500
250
Master max memory.
Worker max memory.
Crypto max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Mem Crypto + dataset size (Mbytes)
Mem Crypto + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

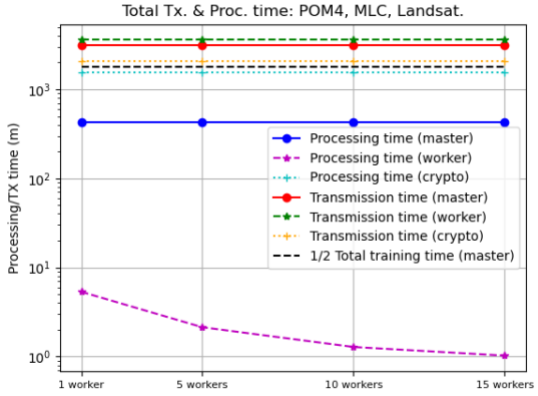

Total Tx. & Proc. time: POM4, MLC, Anuran.
Processing/TX time (m)
10^3
10^2
10^1
10^0
Processing time (master)
Processing time (worker)
Processing time (crypto)
Transmission time (master)
Transmission time (worker)
Transmission time (crypto)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

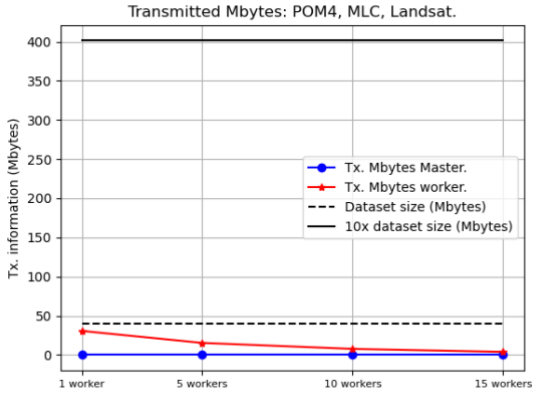
Landsat


Performance and consistency: POM4, MLC, Landsat.
Accuracy (test)
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.76
Landsat(Batch)
Landsat(1w)
Landsat(5w)
Landsat(10w)
Landsat(15w)

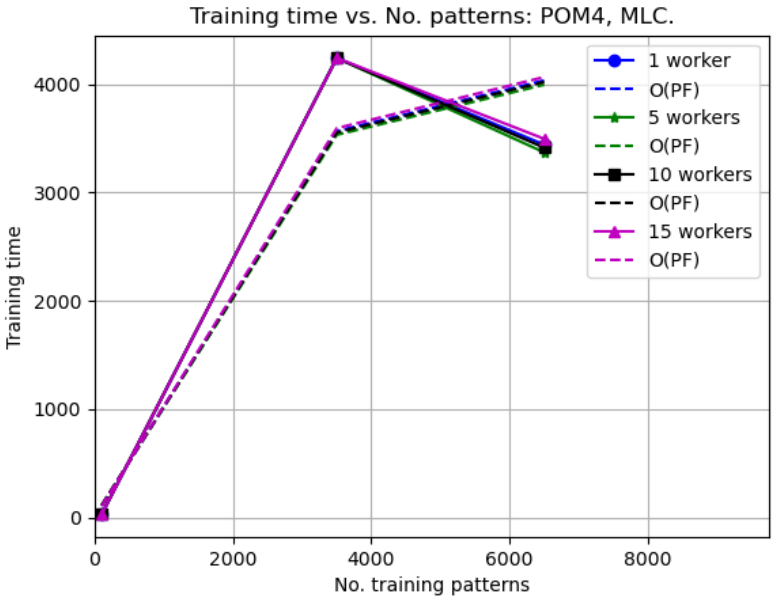

Total training time: POM4, MLC, Landsat.
Total training time.
Linear trend (3.990000)
Training time (m)
5500
5000
4500
4000
3500
3000
2500
2000
1 worker
5 workers
10 workers
15 workers

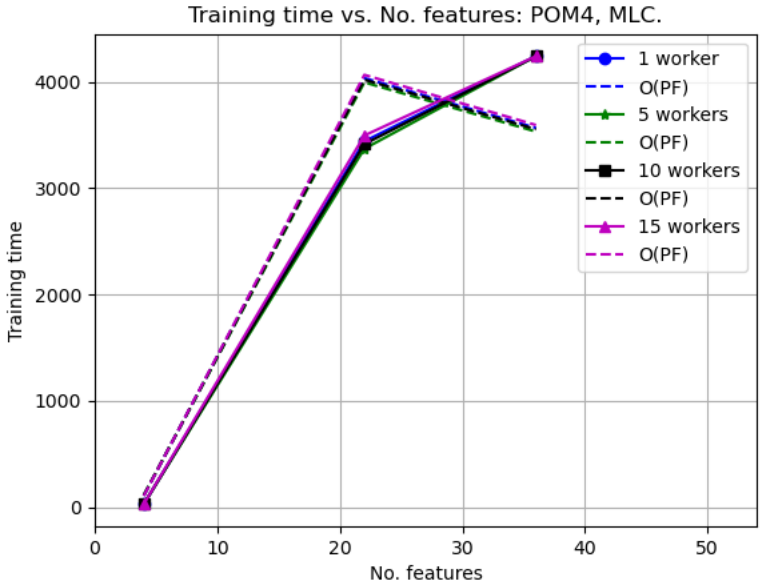

Maximum memory used: POM4, MLC, Landsat.
Max. memory used (Mbytes)
2000
1500
1000
500
Master max memory.
Worker max memory.
Crypto max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Mem Crypto + dataset size (Mbytes)
Mem Crypto + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM4, MLC, Landsat.
Processing/TX time (m)
10^3
10^2
10^1
10^0
Processing time (master)
Processing time (worker)
Processing time (crypto)
Transmission time (master)
Transmission time (worker)
Transmission time (crypto)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers


Transmitted Mbytes: POM4, MLC, Landsat.
Tx. information (Mbytes)
400
350
300
250
200
150
100
50
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

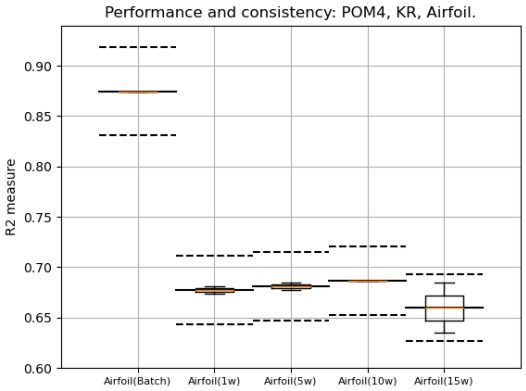
Scalability with respect to No. training patterns, number of features/model size.

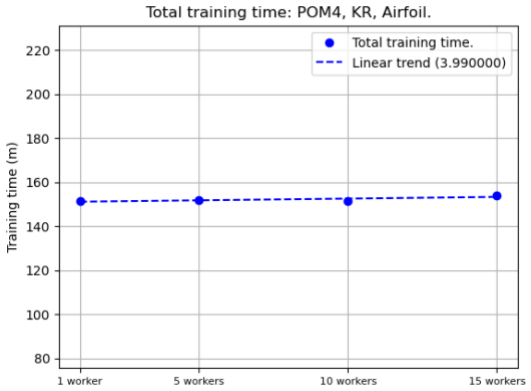


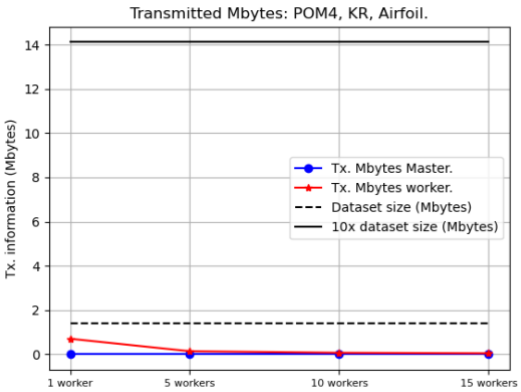


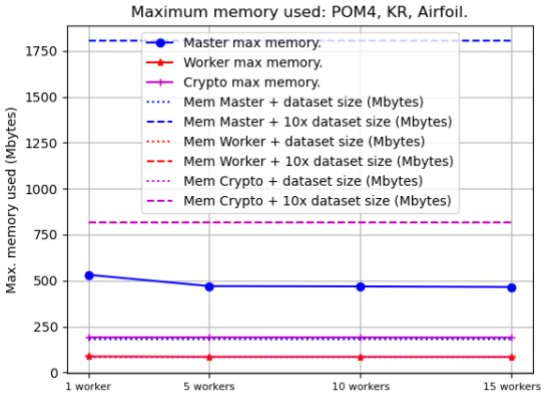
10.1.4 Kernel Regression (KR)

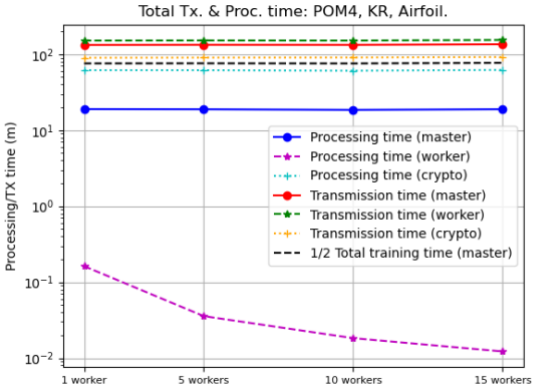
Airfoil



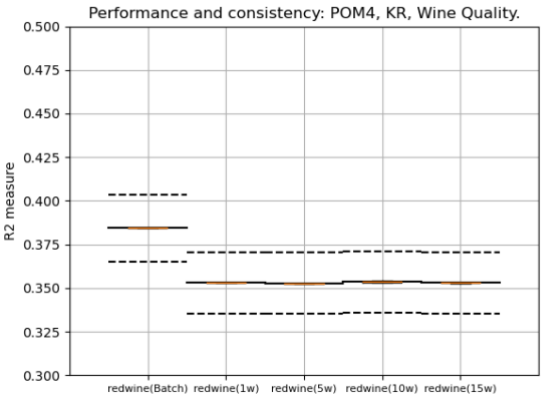


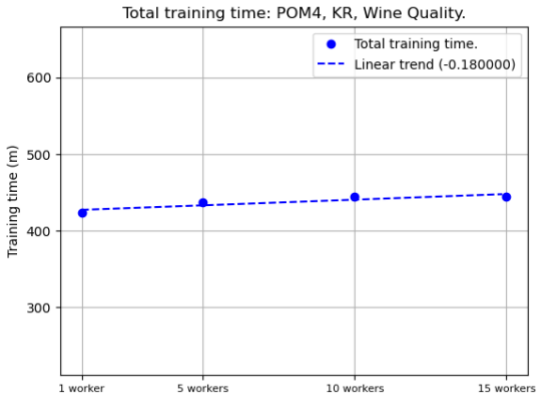


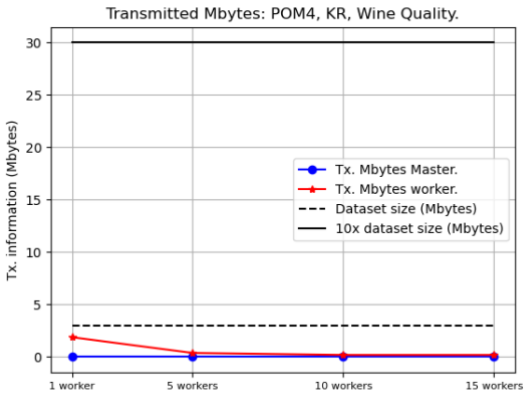


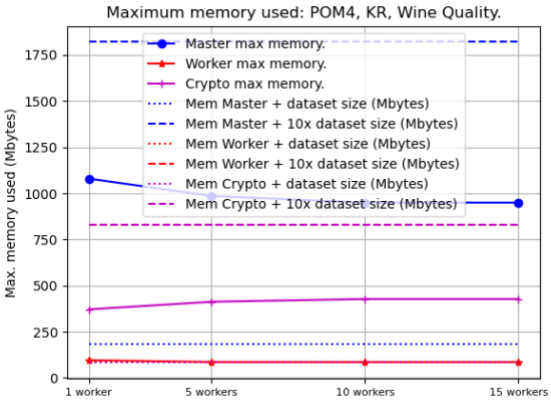


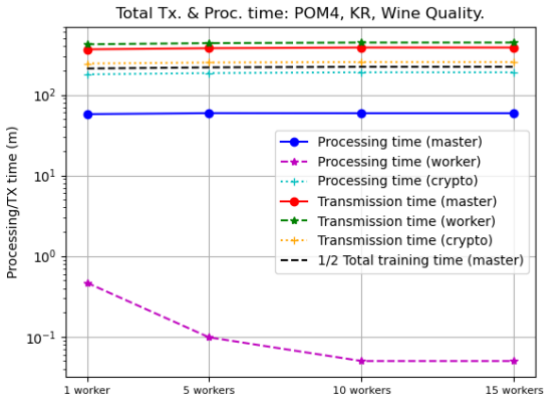
Wine quality



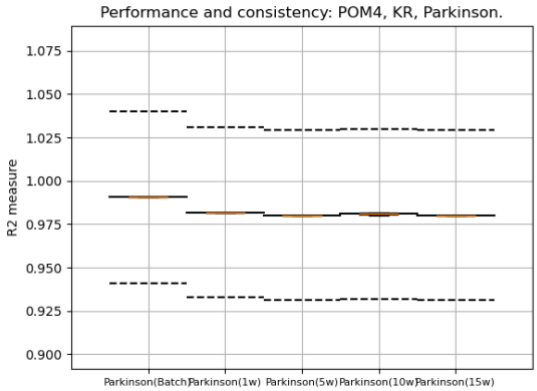


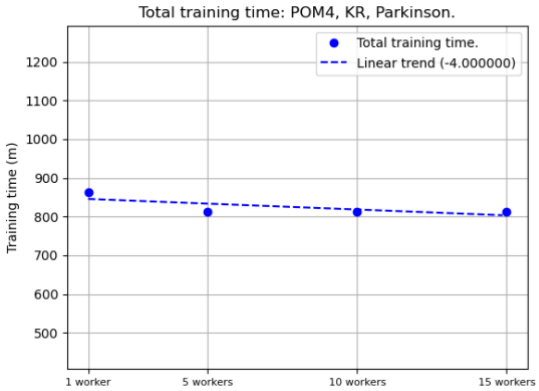


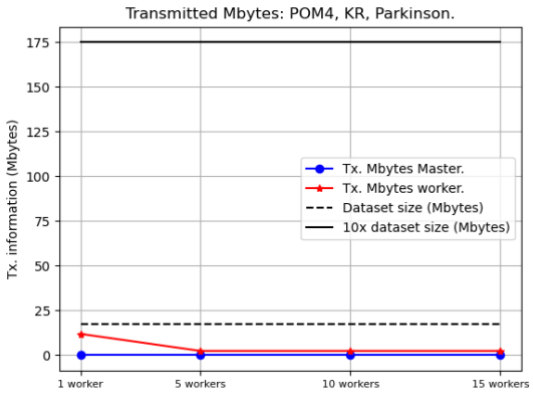


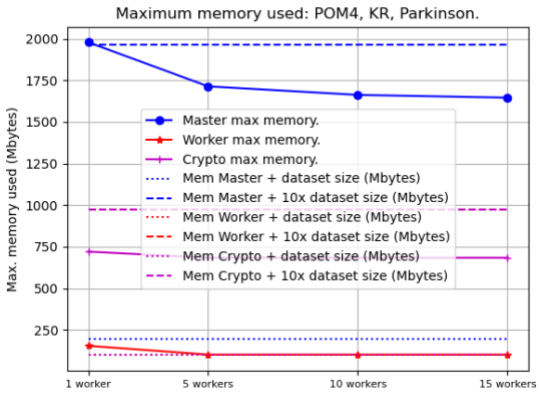


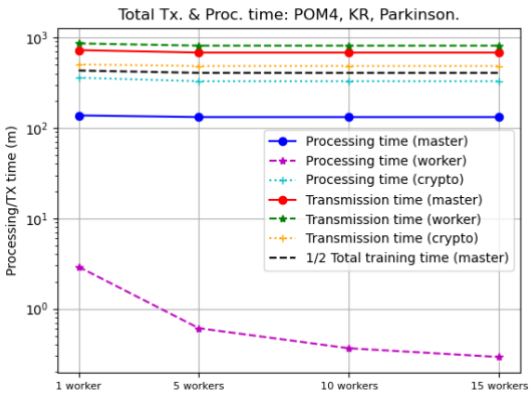
Parkinson


Performance and consistency: POM4, KR, Parkinson.
R2 measure
1.075
1.050
1.025
1.000
0.975
0.950
0.925
0.900
Parkinson(Batch)
Parkinson(1w)
Parkinson(5w)
Parkinson(10w)
Parkinson(15w)

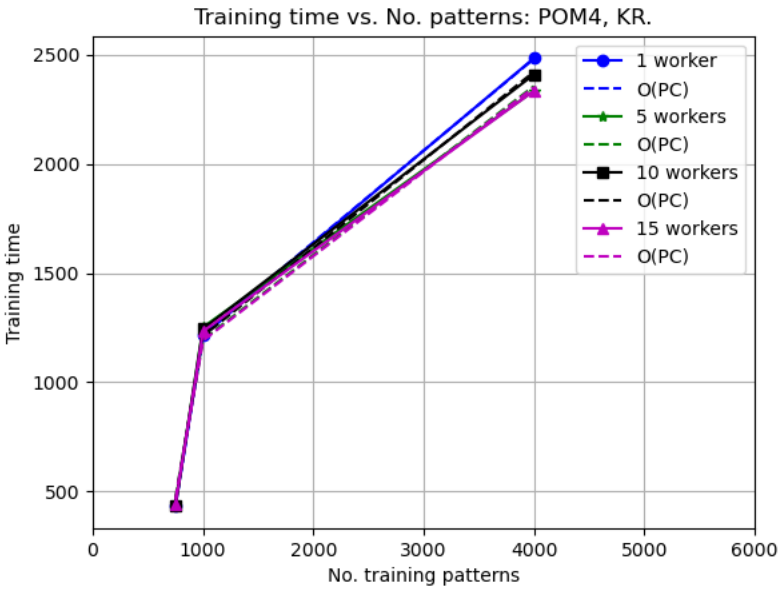

Total training time: POM4, KR, Parkinson.
Total training time.
Linear trend (-4.000000)
Training time (m)
1200
1100
1000
900
800
700
600
500
1 worker
5 workers
10 workers
15 workers

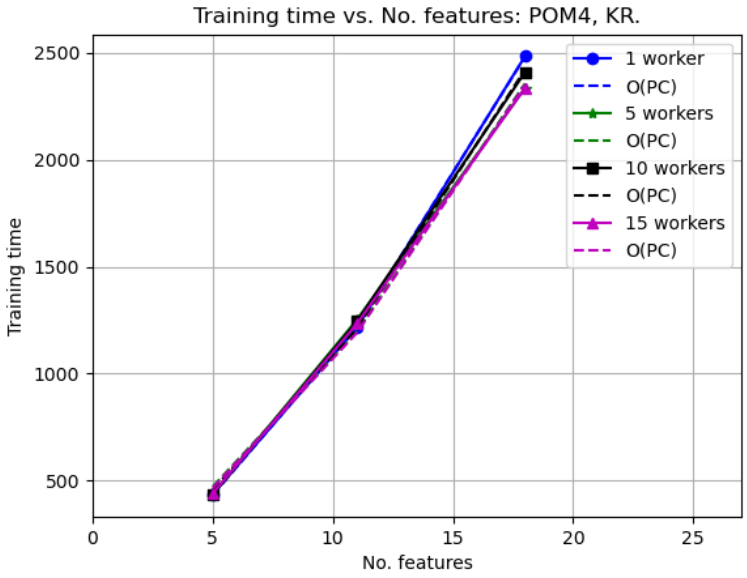

Transmitted Mbytes: POM4, KR, Parkinson.
Tx. information (Mbytes)
175
150
125
100
75
50
25
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Maximum memory used: POM4, KR, Parkinson.
Max. memory used (Mbytes)
2000
1750
1500
1250
1000
750
500
250
Master max memory.
Worker max memory.
Crypto max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Mem Crypto + dataset size (Mbytes)
Mem Crypto + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM4, KR, Parkinson.
Processing/TX time (m)
10^3
10^2
10^1
10^0
Processing time (master)
Processing time (worker)
Processing time (crypto)
Transmission time (master)
Transmission time (worker)
Transmission time (crypto)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

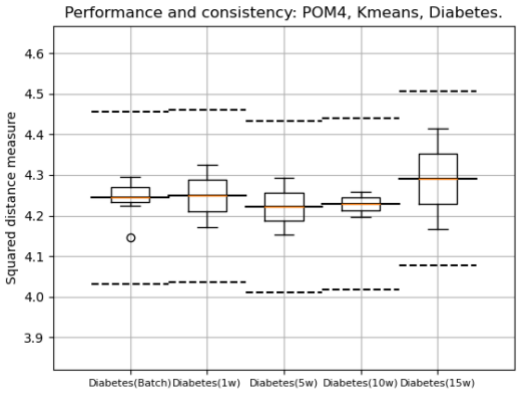
Scalability with respect to No. training patterns, number of features/model size.

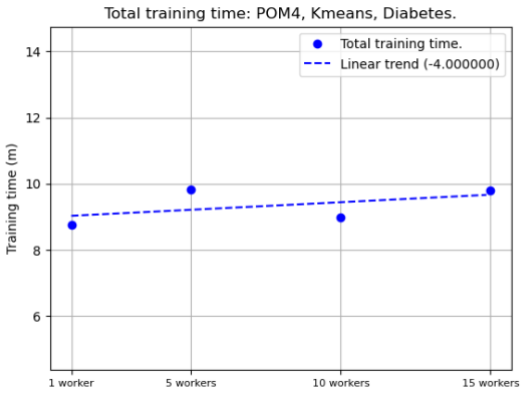


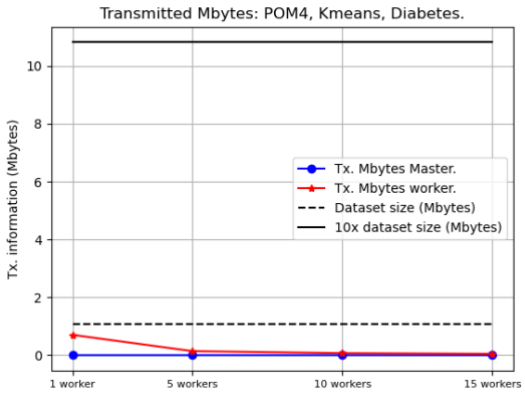


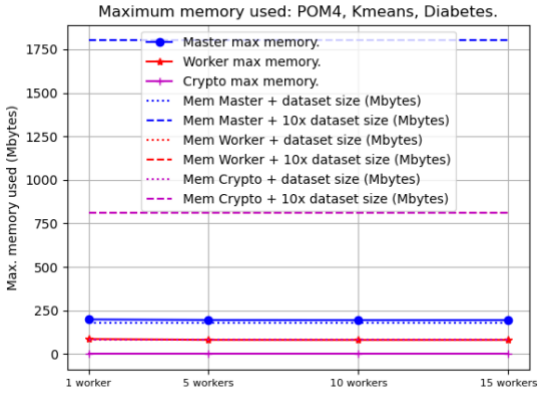
10.1.5 Clustering K-means (Kmeans)

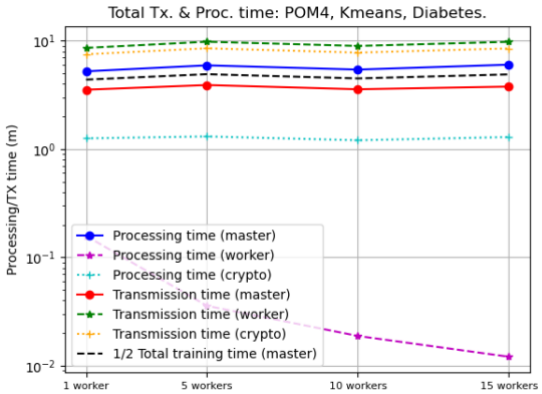
Diabetes



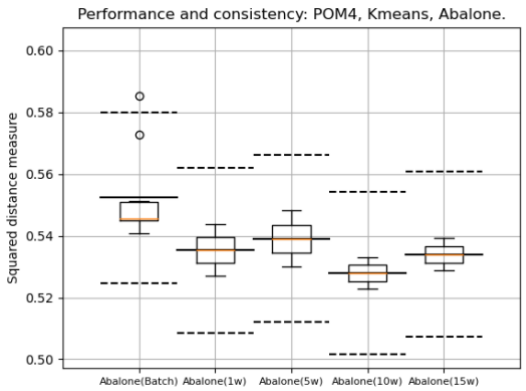


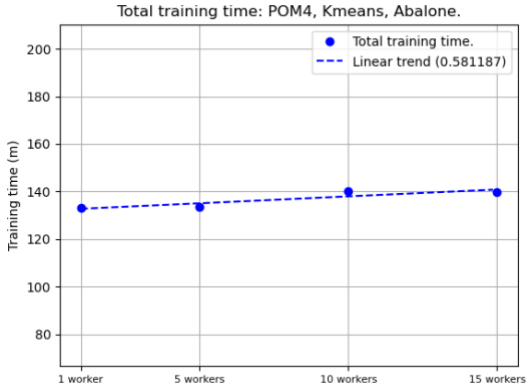


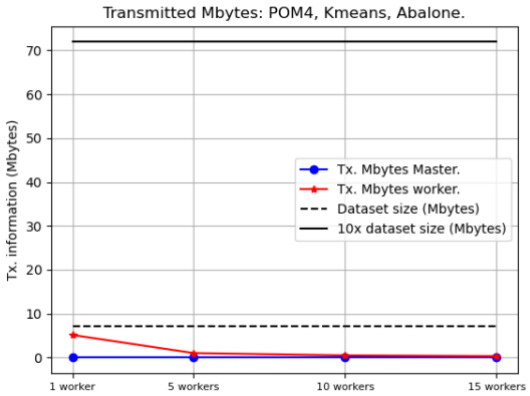


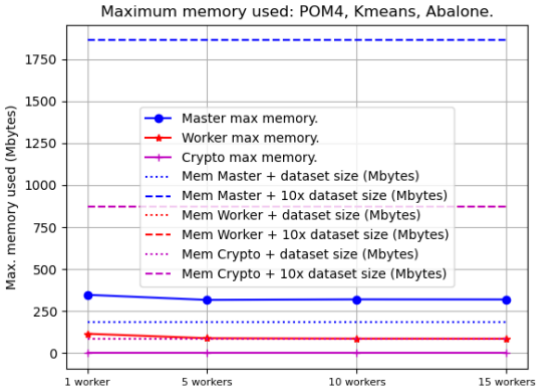


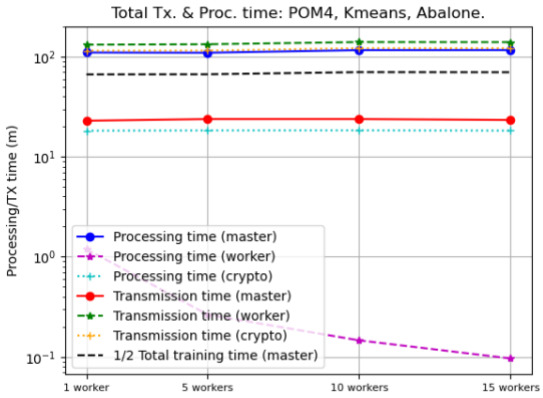
Abalone


Performance and consistency: POM4, Kmeans, Abalone.
Squared distance measure
0.60
0.58
0.56
0.54
0.52
0.50
Abalone(Batch)
Abalone(1w)
Abalone(5w)
Abalone(10w)
Abalone(15w)

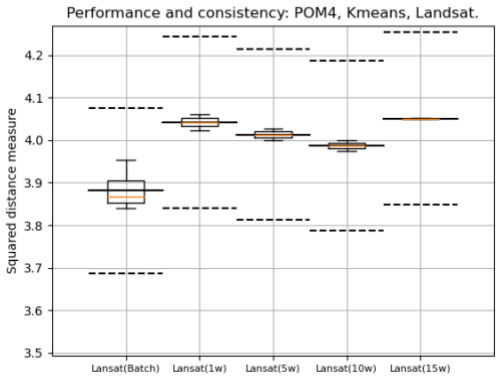

Total training time: POM4, Kmeans, Abalone.
Total training time.
Linear trend (0.581187)
Training time (m)
200
180
160
140
120
100
80
1 worker
5 workers
10 workers
15 workers

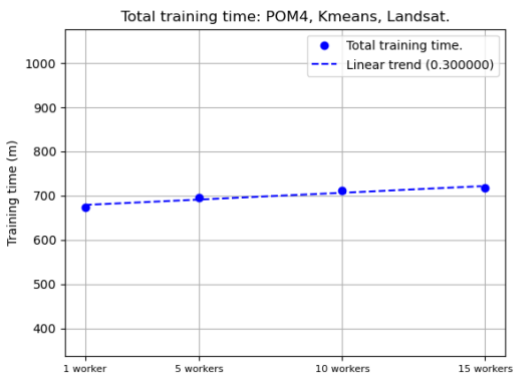

Transmitted Mbytes: POM4, Kmeans, Abalone.
Tx. information (Mbytes)
70
60
50
40
30
20
10
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

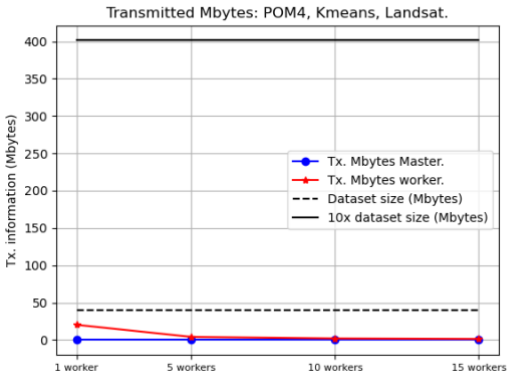

Maximum memory used: POM4, Kmeans, Abalone.
Max. memory used (Mbytes)
1750
1500
1250
1000
750
500
250
0
Master max memory.
Worker max memory.
Crypto max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Mem Crypto + dataset size (Mbytes)
Mem Crypto + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

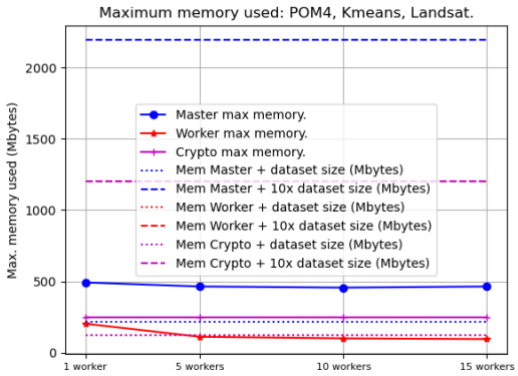

Total Tx. & Proc. time: POM4, Kmeans, Abalone.
Processing/TX time (m)
10^2
10^1
10^0
10^{-1}
Processing time (master)
Processing time (worker)
Processing time (crypto)
Transmission time (master)
Transmission time (worker)
Transmission time (crypto)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

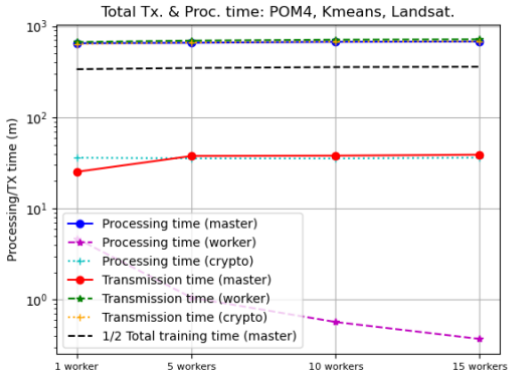
Landsat


Performance and consistency: POM4, Kmeans, Landsat.
Squared distance measure
Lansat(Batch)
Lansat(1w)
Lansat(5w)
Lansat(10w)
Lansat(15w)

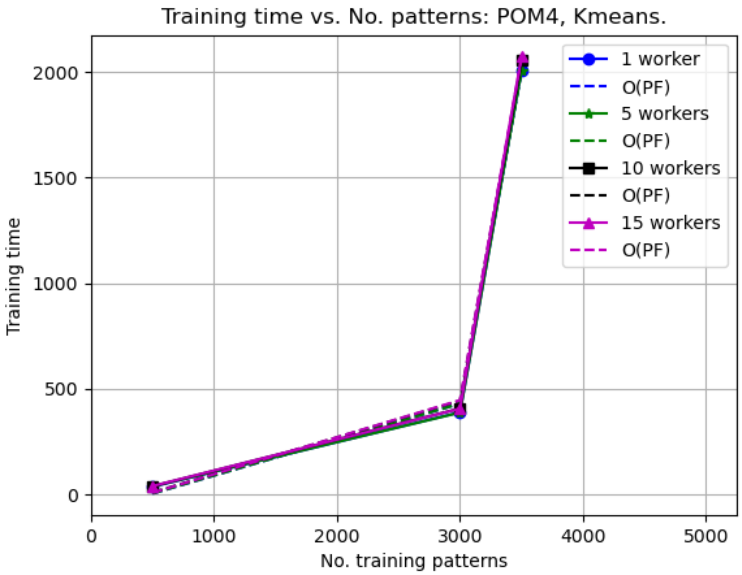

Total training time: POM4, Kmeans, Landsat.
Total training time.
Linear trend (0.300000)
Training time (m)
1 worker
5 workers
10 workers
15 workers

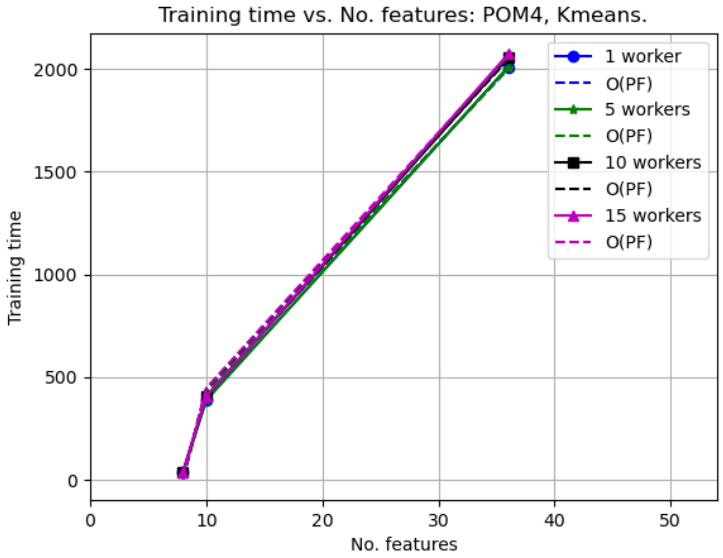

Transmitted Mbytes: POM4, Kmeans, Landsat.
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
Tx. information (Mbytes)
1 worker
5 workers
10 workers
15 workers


Maximum memory used: POM4, Kmeans, Landsat.
Master max memory.
Worker max memory.
Crypto max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Mem Crypto + dataset size (Mbytes)
Mem Crypto + 10x dataset size (Mbytes)
Max. memory used (Mbytes)
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM4, Kmeans, Landsat.
Processing time (master)
Processing time (worker)
Processing time (crypto)
Transmission time (master)
Transmission time (worker)
Transmission time (crypto)
1/2 Total training time (master)
Processing/TX time (m)
1 worker
5 workers
10 workers
15 workers

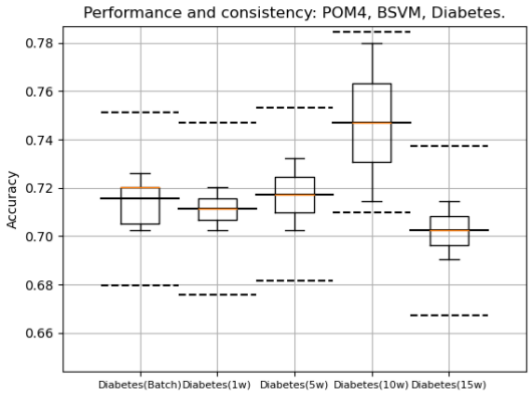
Scalability with respect to No. training patterns, number of features/model size.

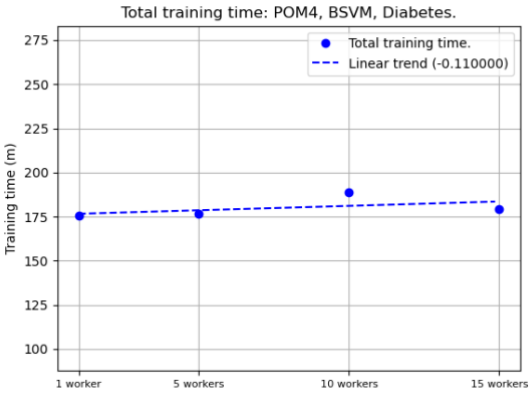


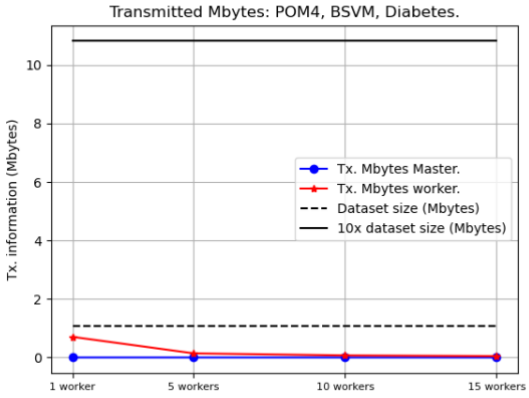


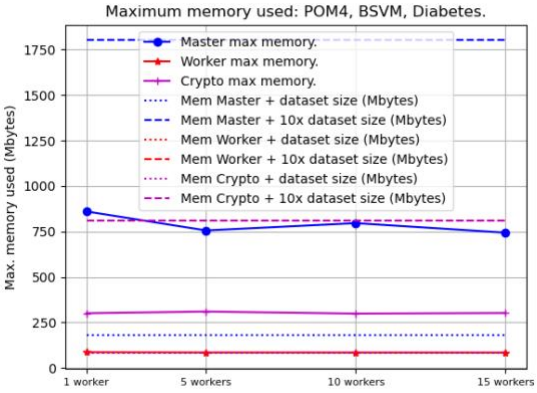
10.1.6 Budget Support Vector Machine (BSVM)

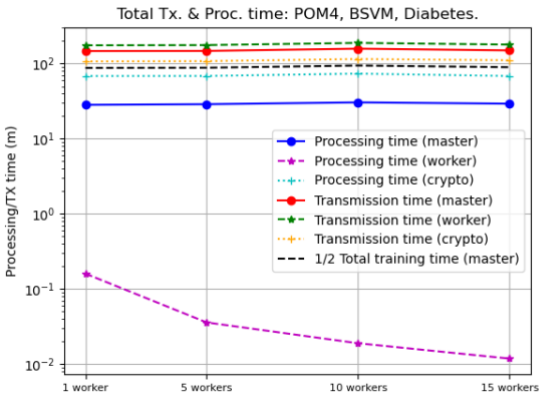
Diabetes



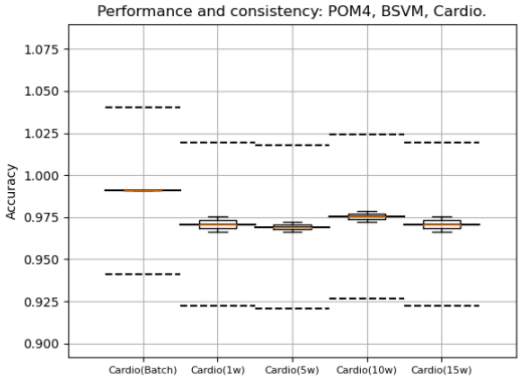


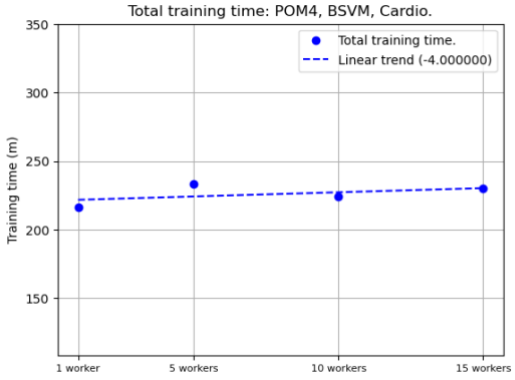


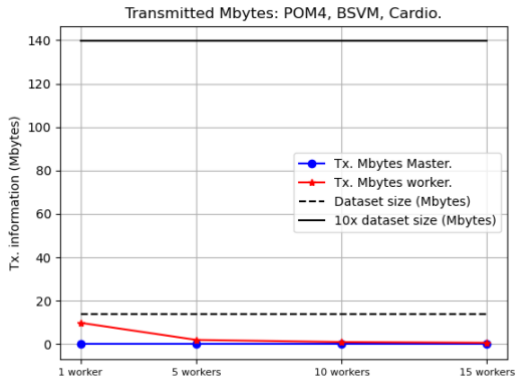


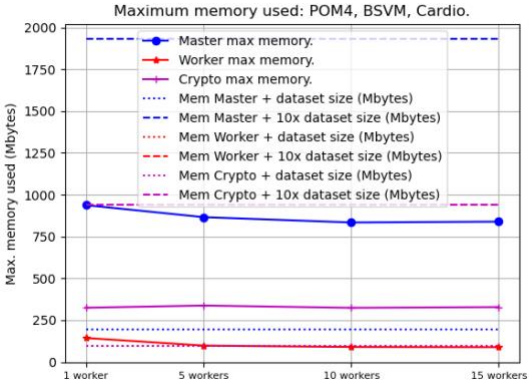


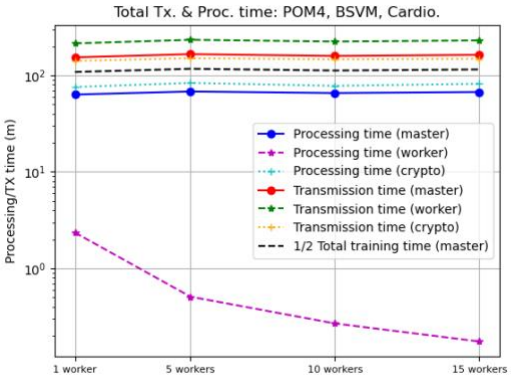
Cardiopathy



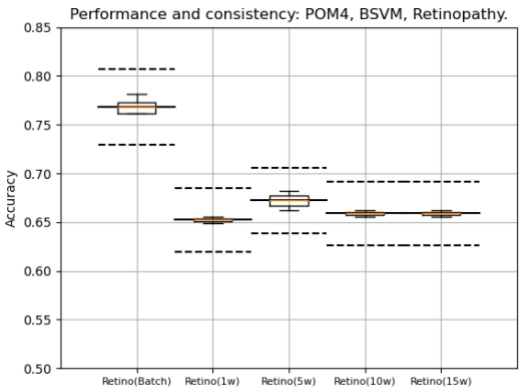


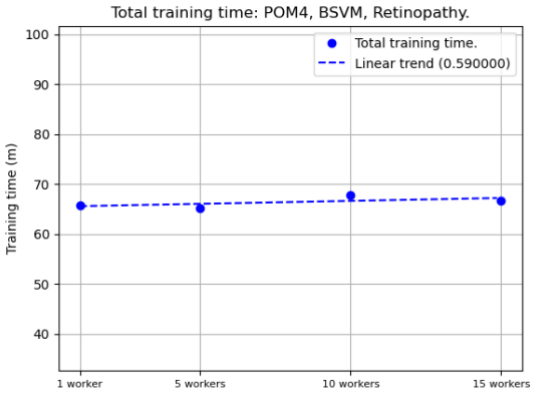


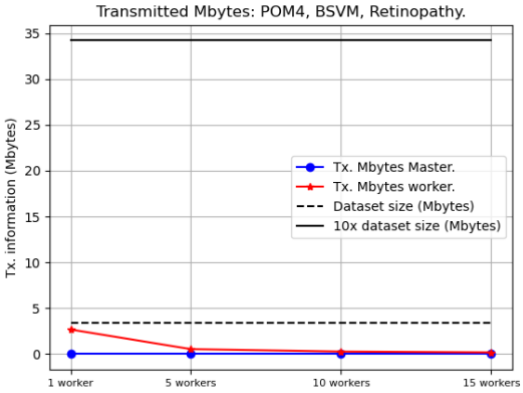


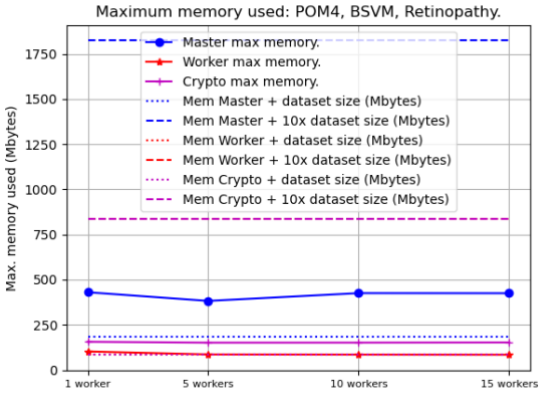


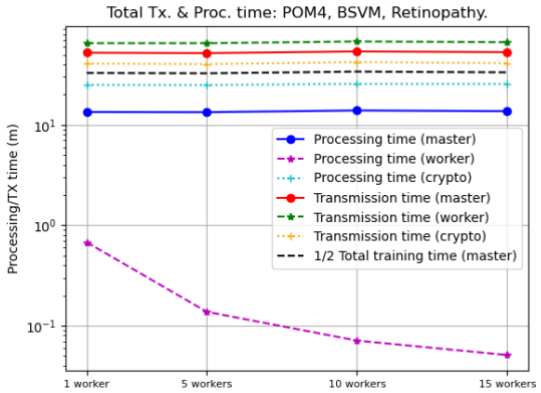
Retinopathy



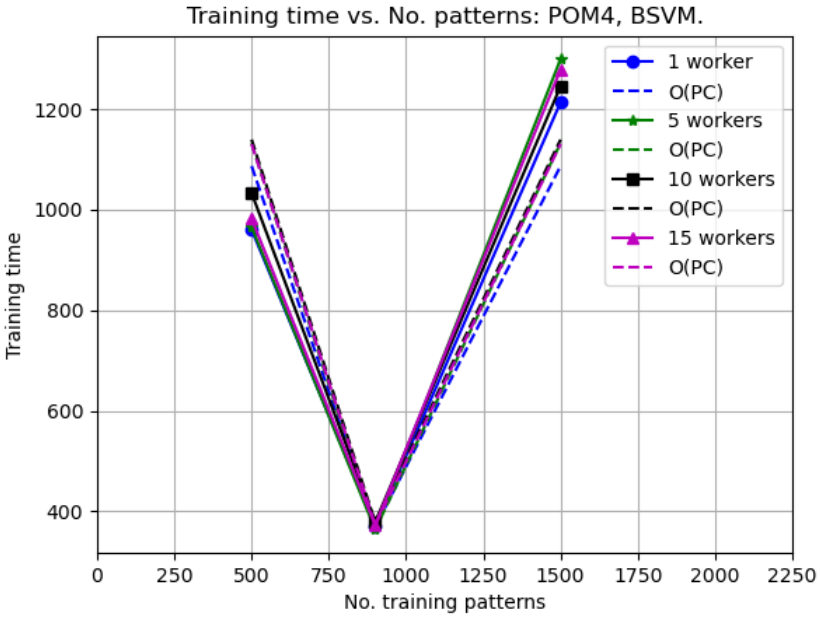


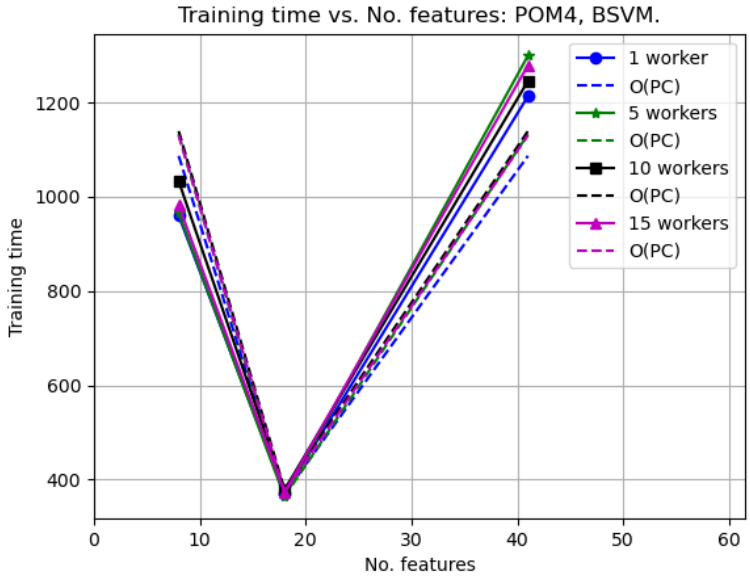






Scalability with respect to No. training patterns, number of features/model size.

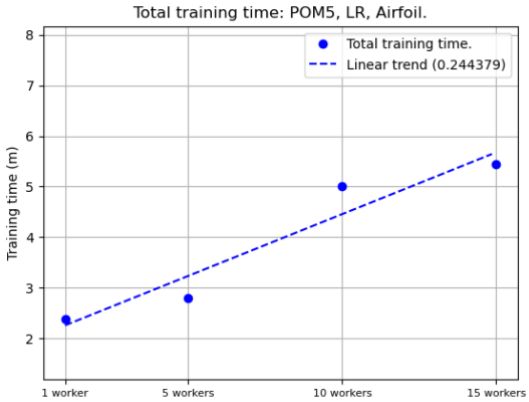


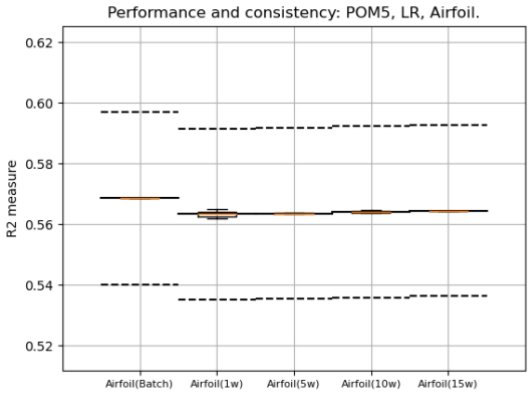


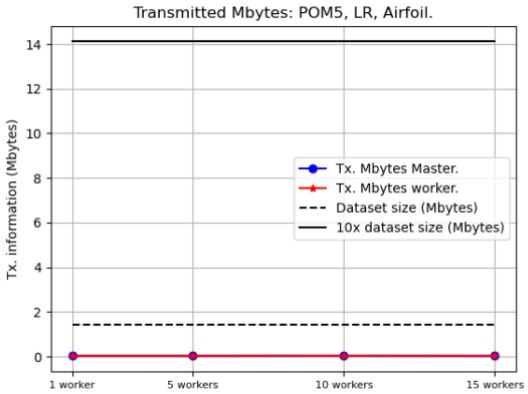
10.2 POM 5

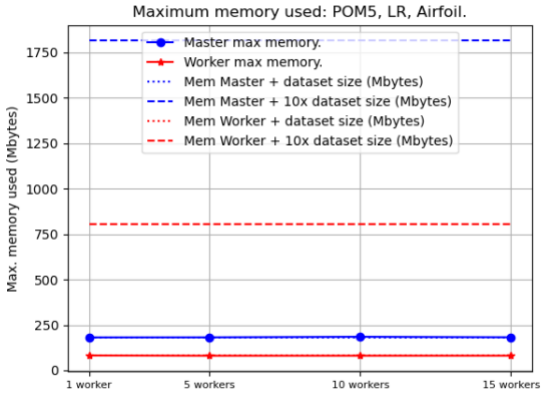
10.2.1 Linear Regression (LR)

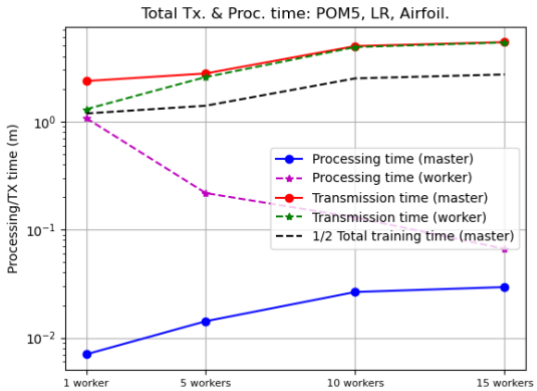
Airfoil



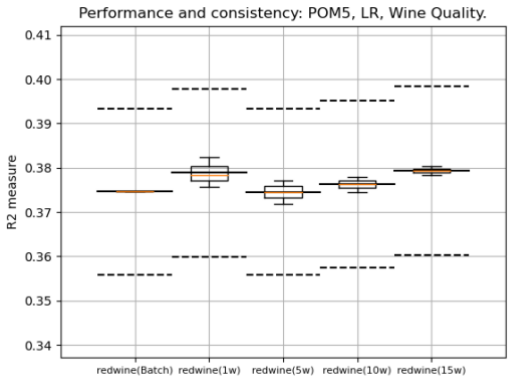


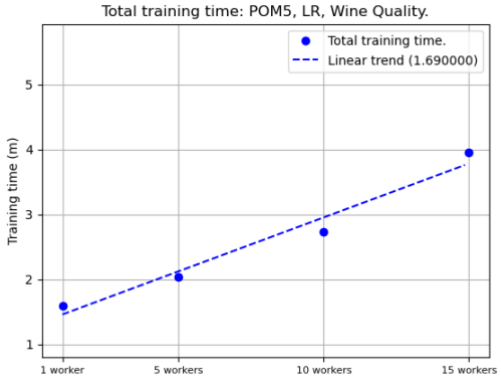


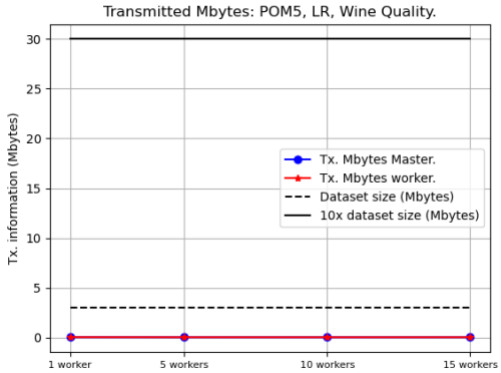


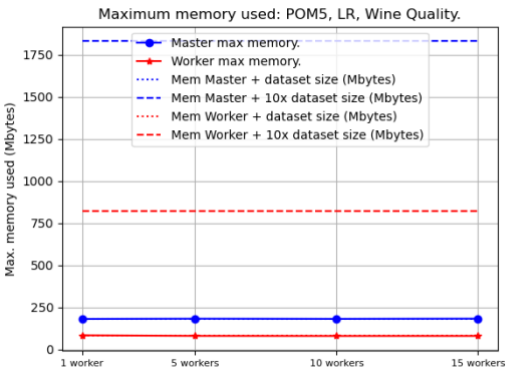


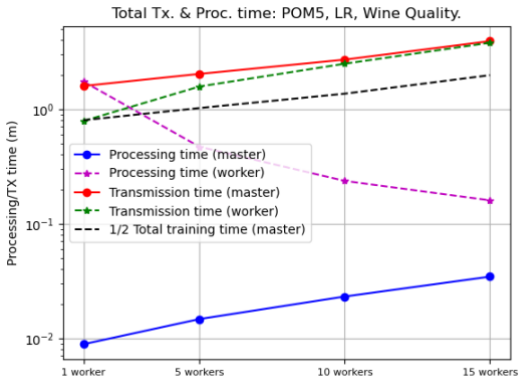
Wine quality



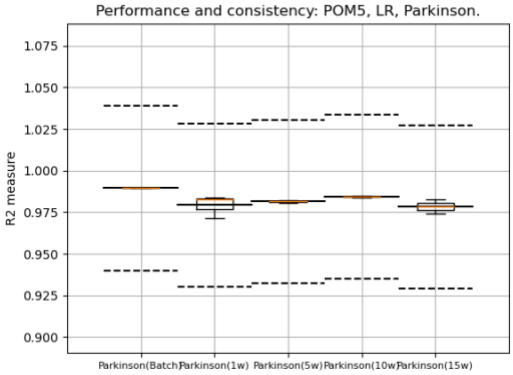


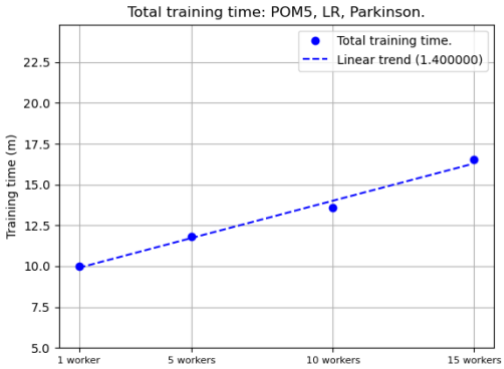


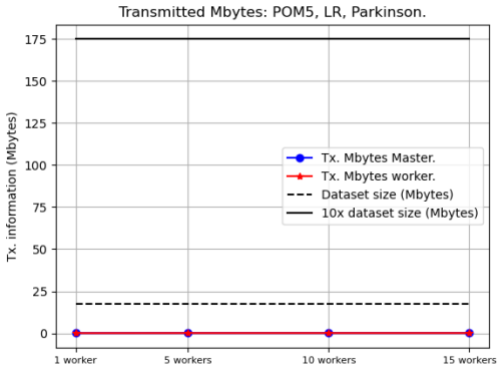


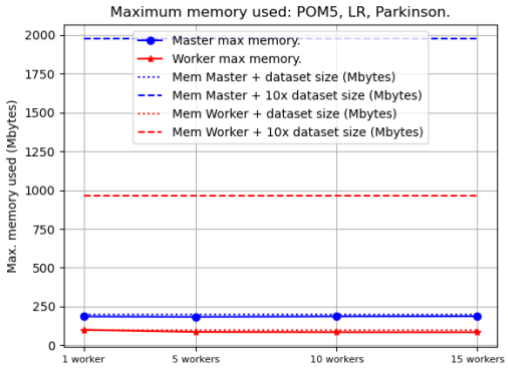


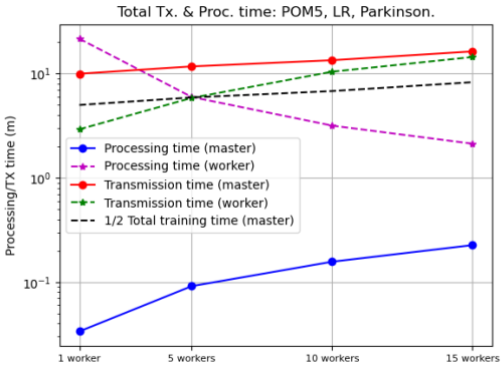
Parkinson


Performance and consistency: POM5, LR, Parkinson.
R2 measure
1.075
1.050
1.025
1.000
0.975
0.950
0.925
0.900
Parkinson(Batch)
Parkinson(1w)
Parkinson(5w)
Parkinson(10w)
Parkinson(15w)

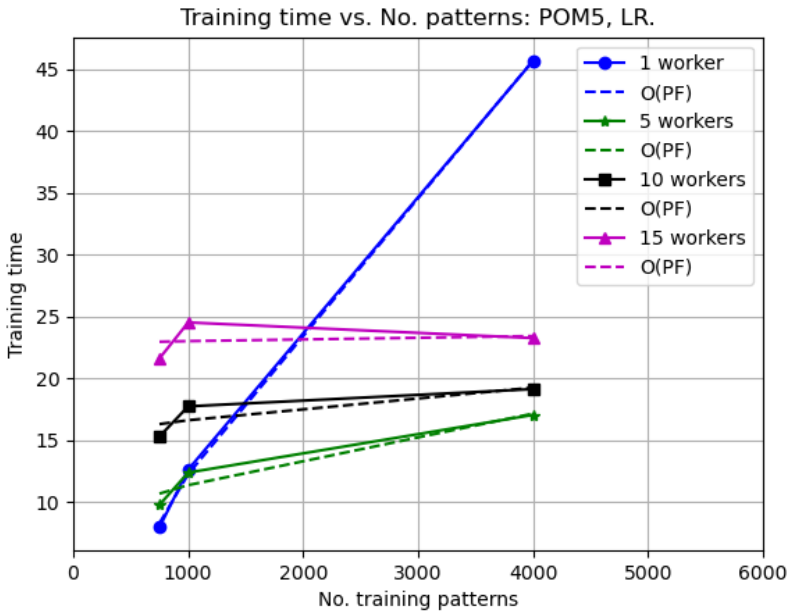

Total training time: POM5, LR, Parkinson.
Total training time.
Linear trend (1.400000)
Training time (m)
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
1 worker
5 workers
10 workers
15 workers

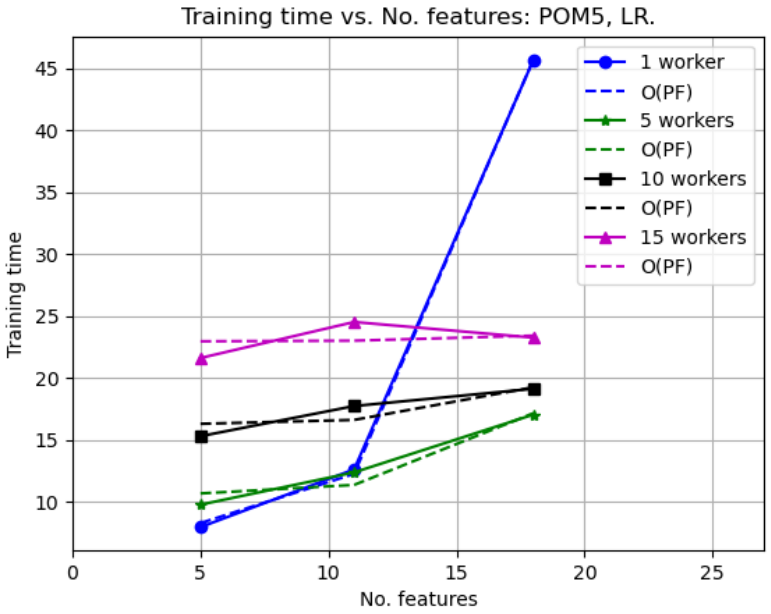

Transmitted Mbytes: POM5, LR, Parkinson.
Tx. information (Mbytes)
175
150
125
100
75
50
25
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Maximum memory used: POM5, LR, Parkinson.
Max. memory used (Mbytes)
2000
1750
1500
1250
1000
750
500
250
0
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM5, LR, Parkinson.
Processing/TX time (m)
10^1
10^0
10^{-1}
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

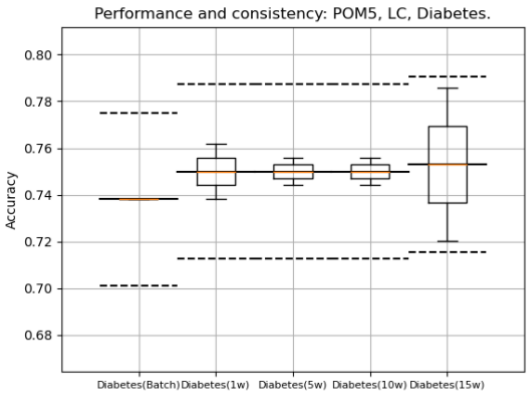
Scalability with respect to No. training patterns, number of features/model size.

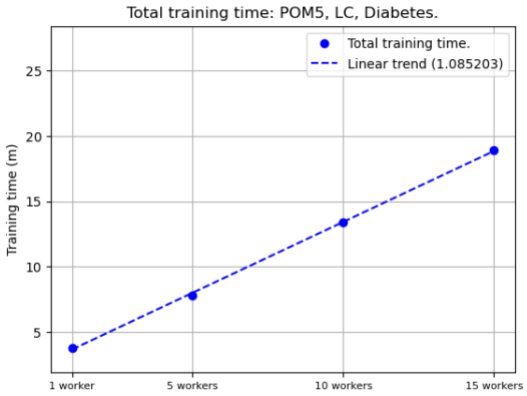


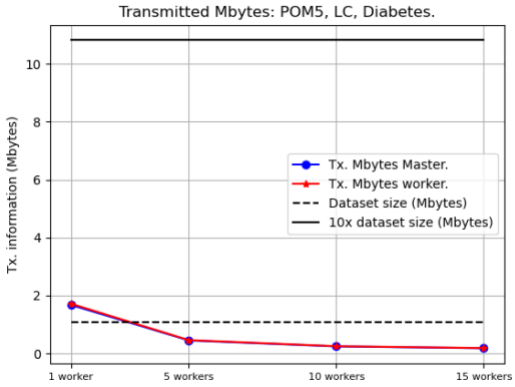


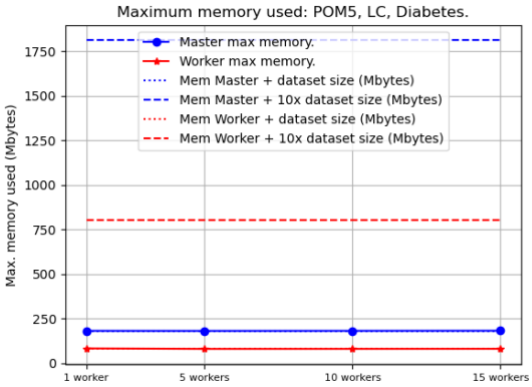
10.2.2 Logistic Classifier (LC)

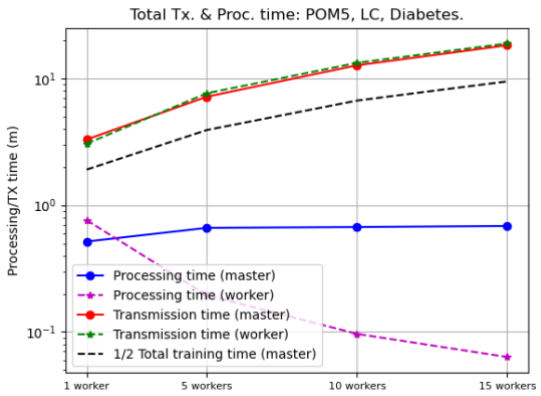
Diabetes



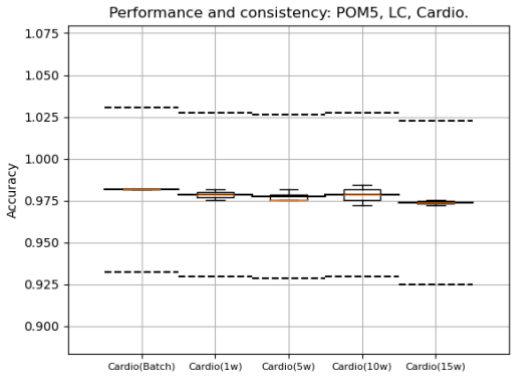


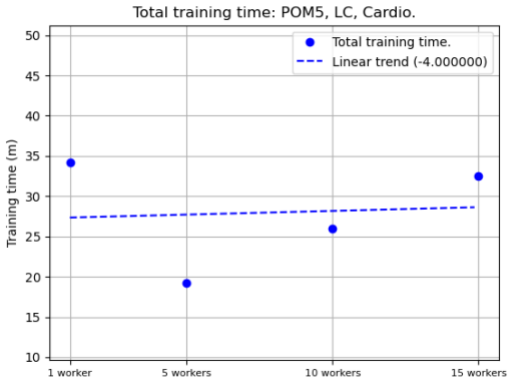


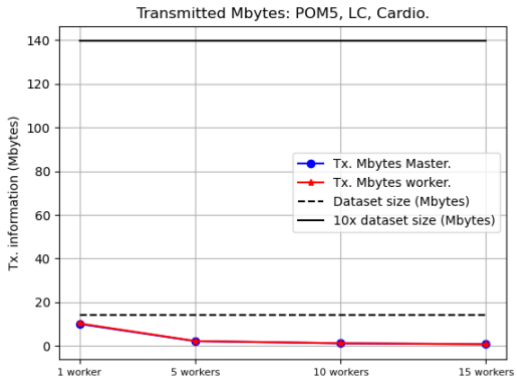


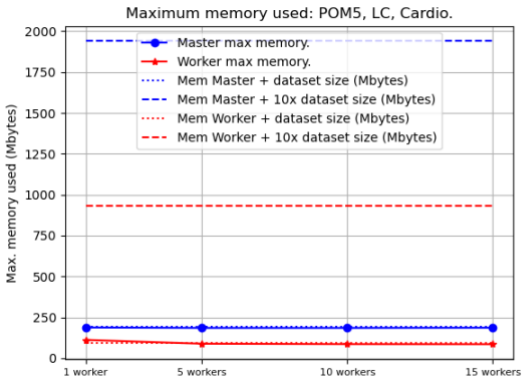


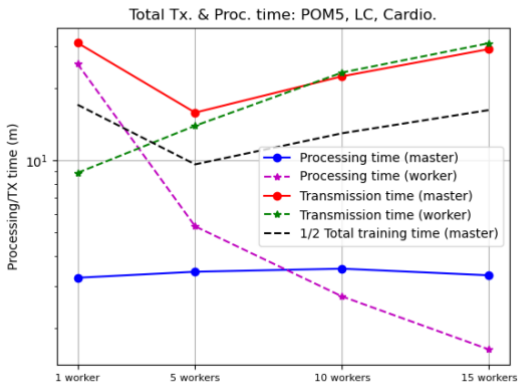
Cardiopathy



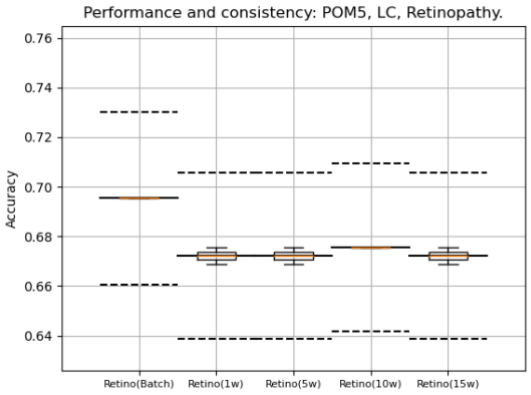


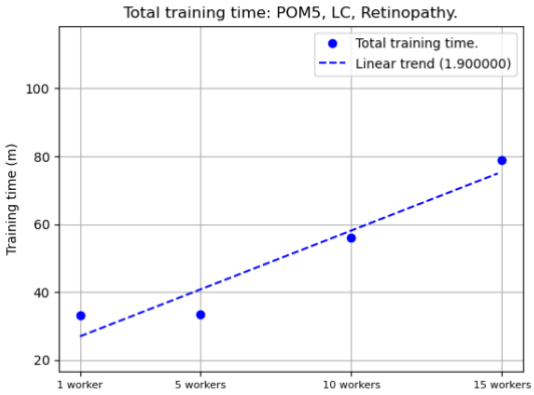


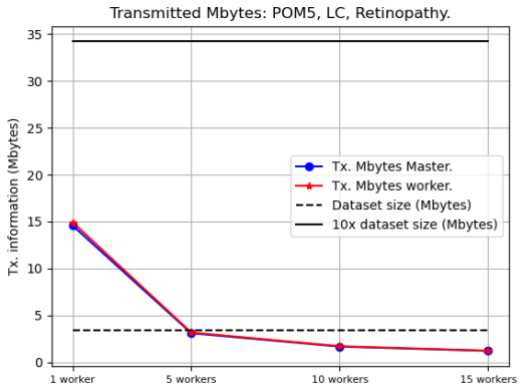


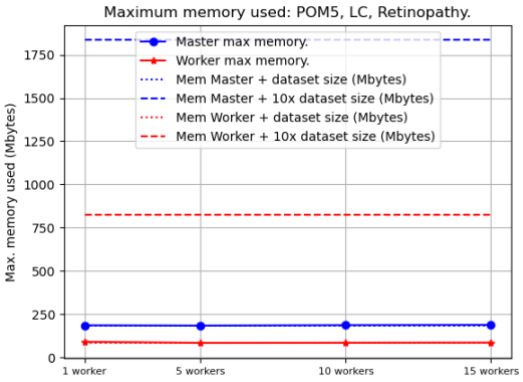


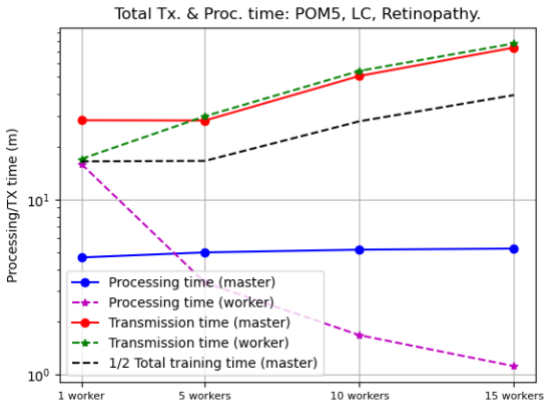
Retinopathy



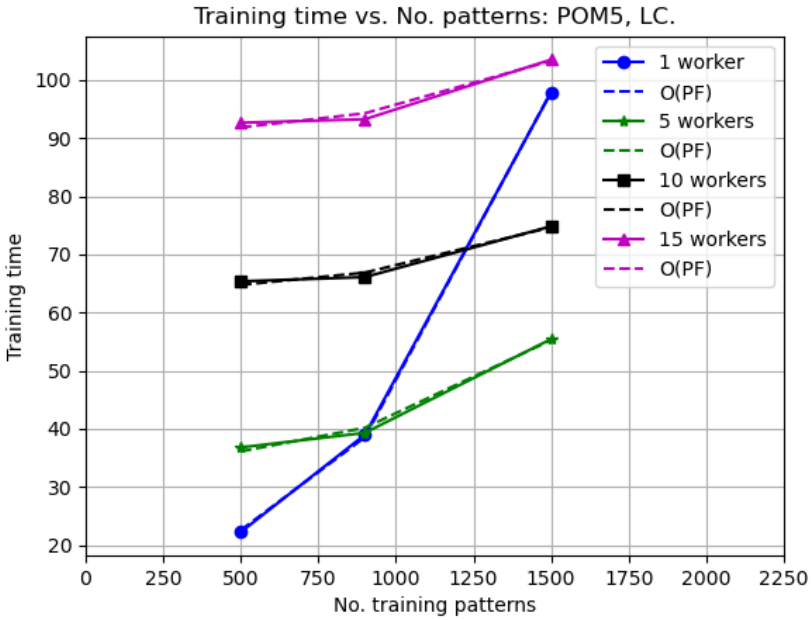


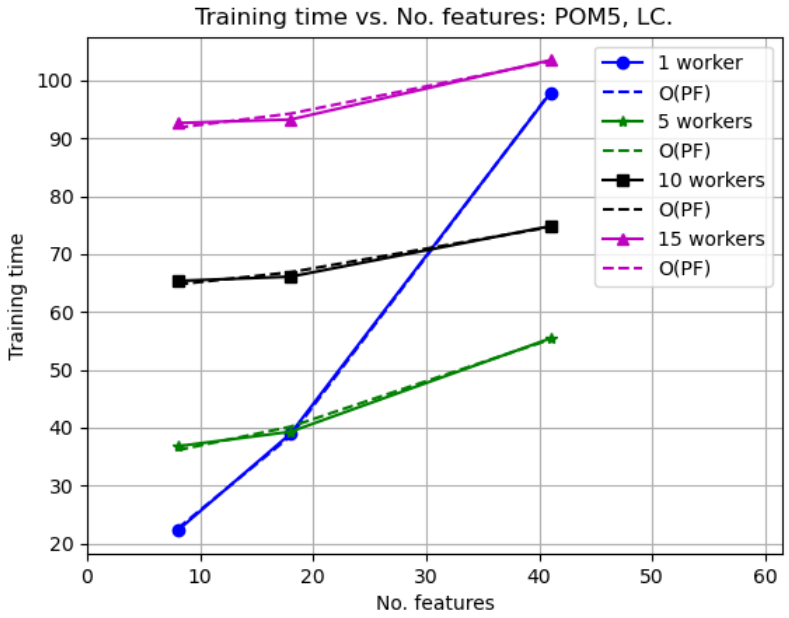






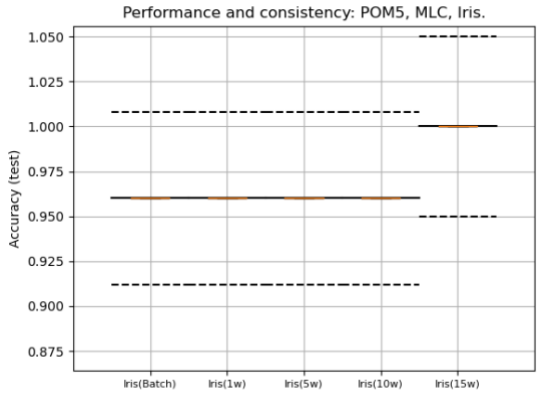
Scalability with respect to No. training patterns, number of features/model size.

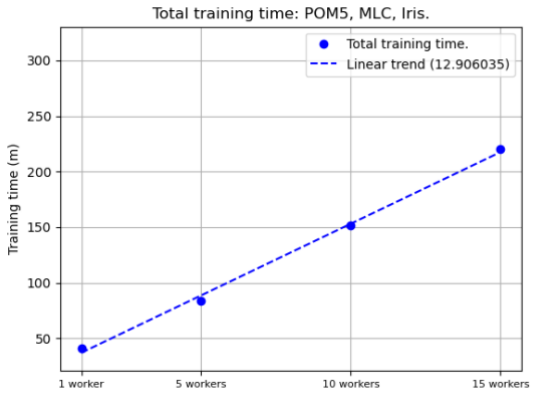


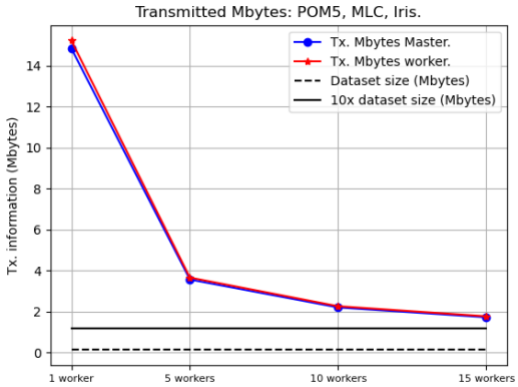


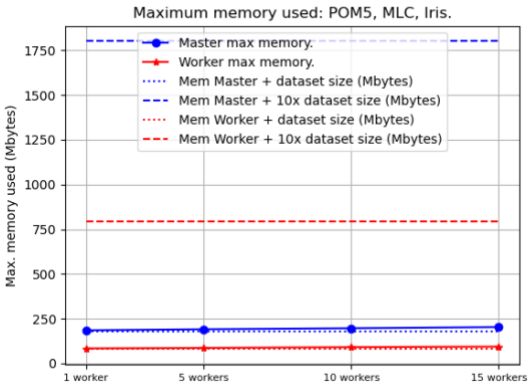
10.2.3 Multiclass Logistic Classifier (MLC)

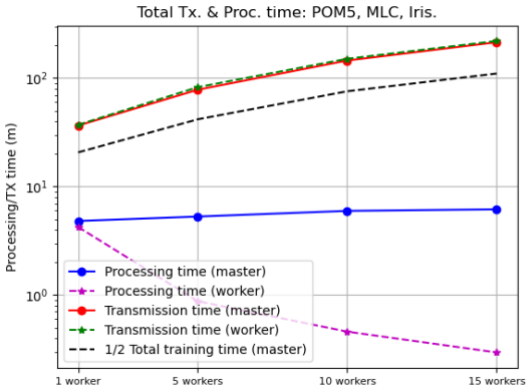
Iris



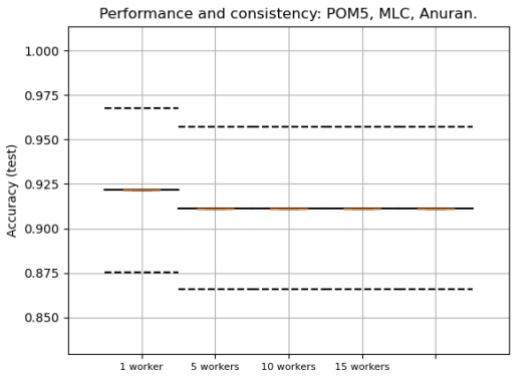


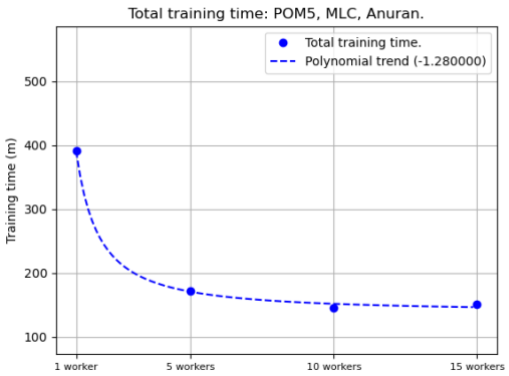


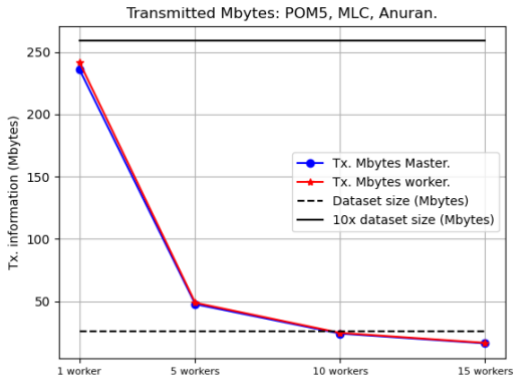


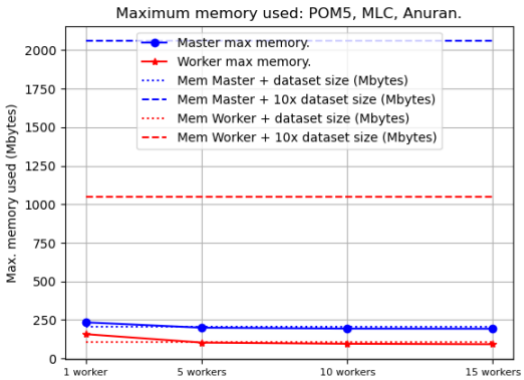


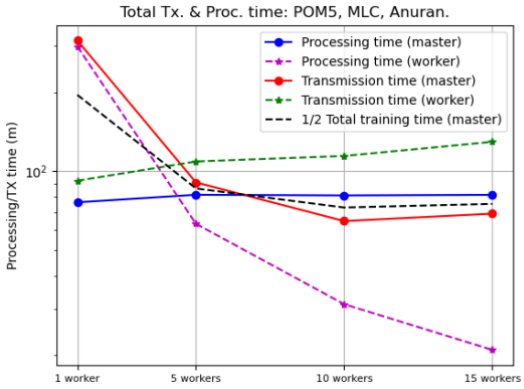
Anuran


Performance and consistency: POM5, MLC, Anuran.
Accuracy (test)
1.000
0.975
0.950
0.925
0.900
0.875
0.850
1 worker
5 workers
10 workers
15 workers

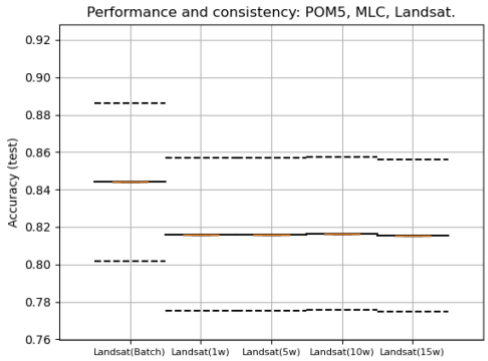

Total training time: POM5, MLC, Anuran.
Total training time.
Polynomial trend (-1.280000)
Training time (m)
500
400
300
200
100
1 worker
5 workers
10 workers
15 workers

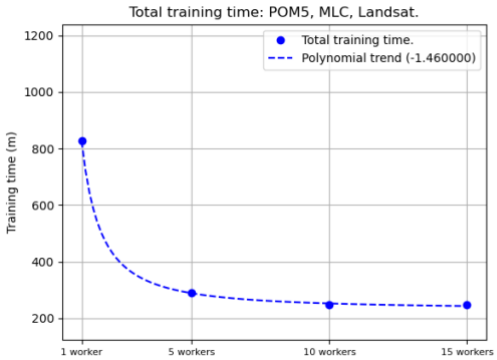

Transmitted Mbytes: POM5, MLC, Anuran.
Tx. information (Mbytes)
250
200
150
100
50
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

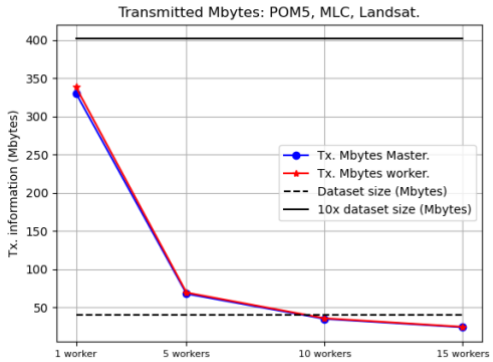

Maximum memory used: POM5, MLC, Anuran.
Max. memory used (Mbytes)
2000
1750
1500
1250
1000
750
500
250
0
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

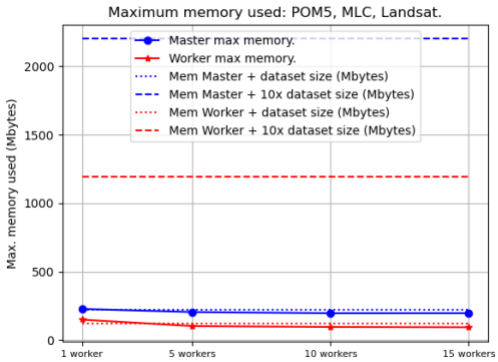

Total Tx. & Proc. time: POM5, MLC, Anuran.
Processing/TX time (m)
10^2
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

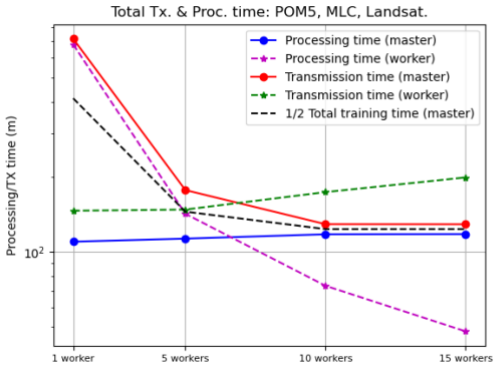
Landsat


Performance and consistency: POM5, MLC, Landsat.
Accuracy (test)
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.76
Landsat(Batch)
Landsat(1w)
Landsat(5w)
Landsat(10w)
Landsat(15w)

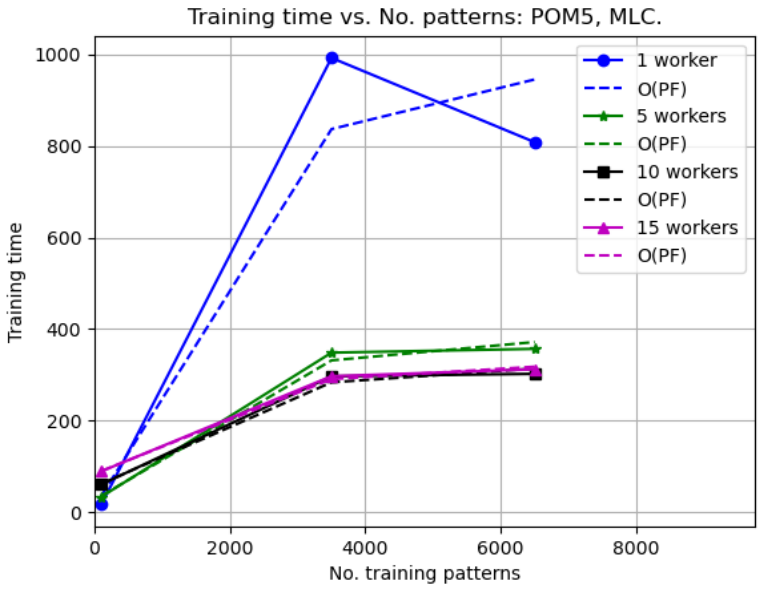

Total training time: POM5, MLC, Landsat.
Total training time.
Polynomial trend (-1.460000)
Training time (m)
1200
1000
800
600
400
200
1 worker
5 workers
10 workers
15 workers

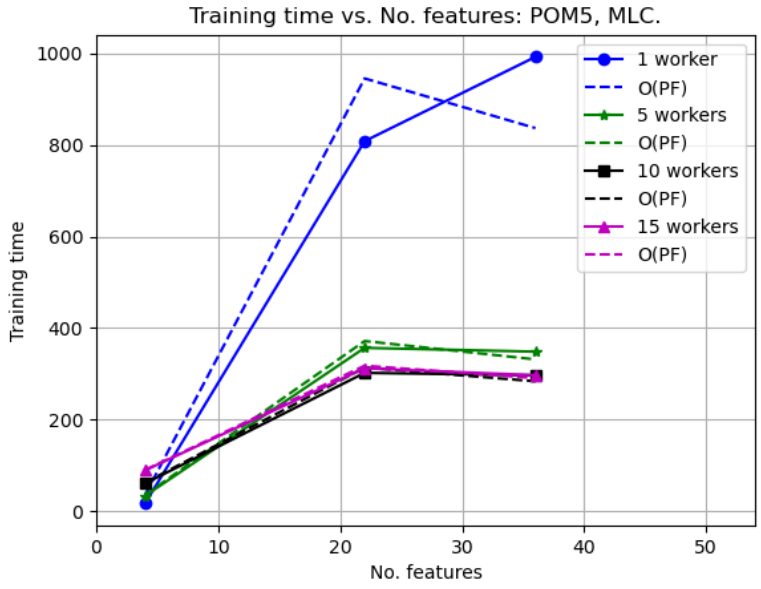

Transmitted Mbytes: POM5, MLC, Landsat.
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
Tx. information (Mbytes)
400
350
300
250
200
150
100
50
1 worker
5 workers
10 workers
15 workers


Maximum memory used: POM5, MLC, Landsat.
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Max. memory used (Mbytes)
2000
1500
1000
500
0
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM5, MLC, Landsat.
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
Processing/TX time (m)
10^2
1 worker
5 workers
10 workers
15 workers

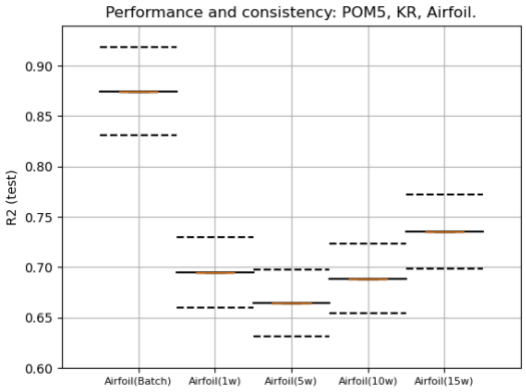
Scalability with respect to No. training patterns, number of features/model size.

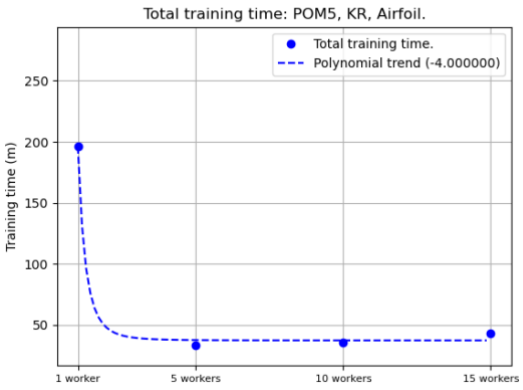


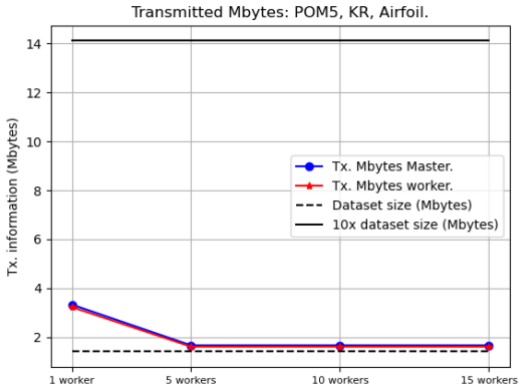


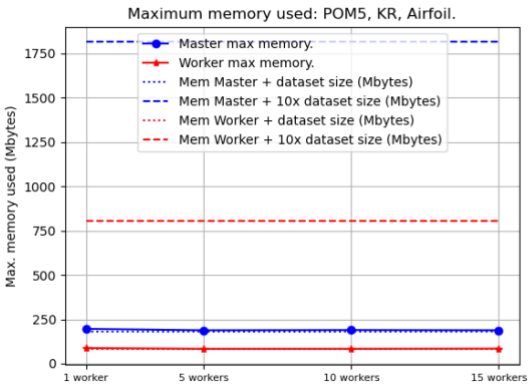
10.2.4 Kernel Regression (KR)

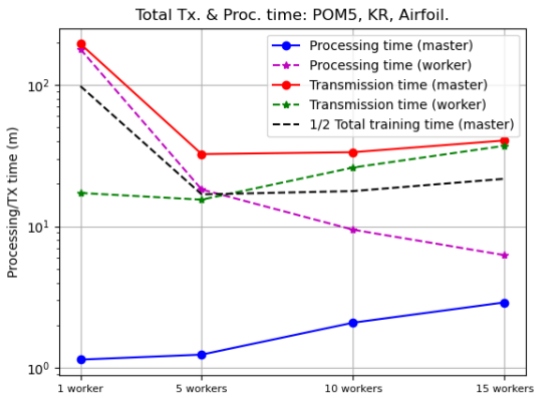
Airfoil



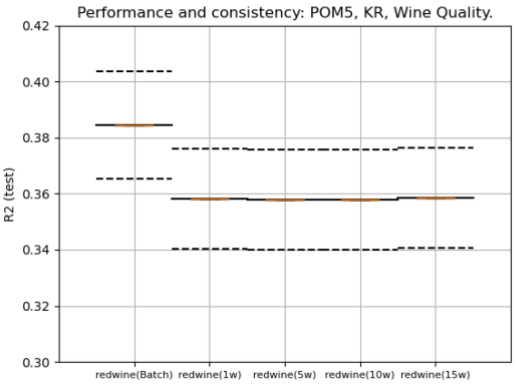


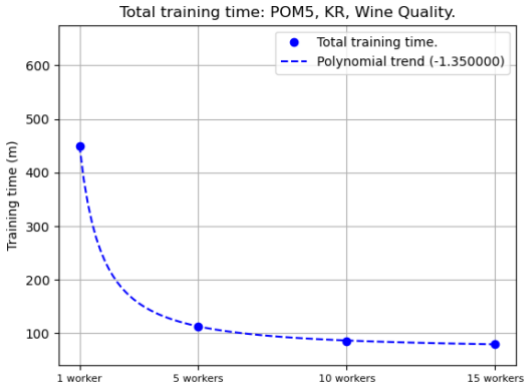


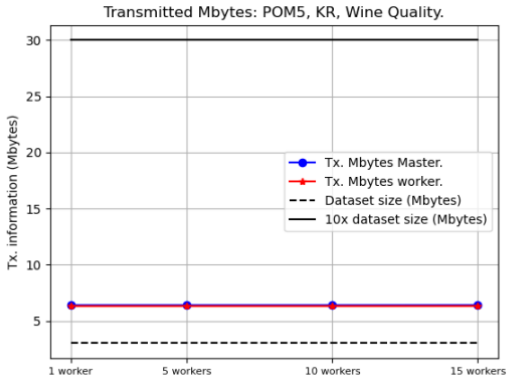


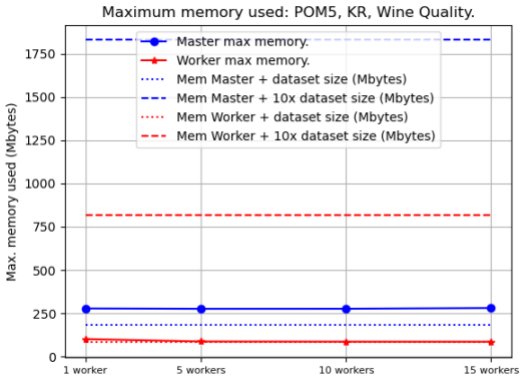


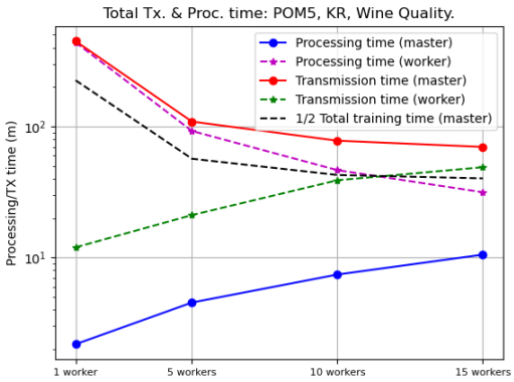
Wine quality



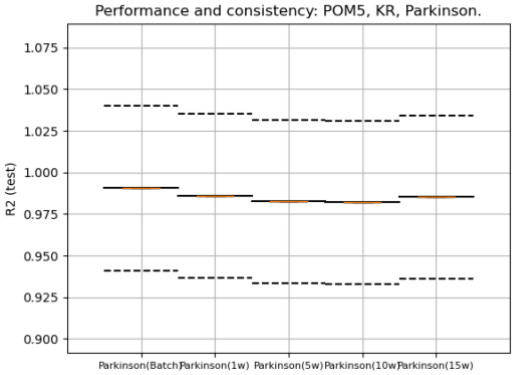


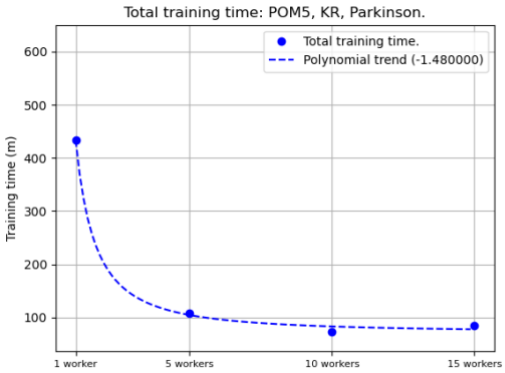


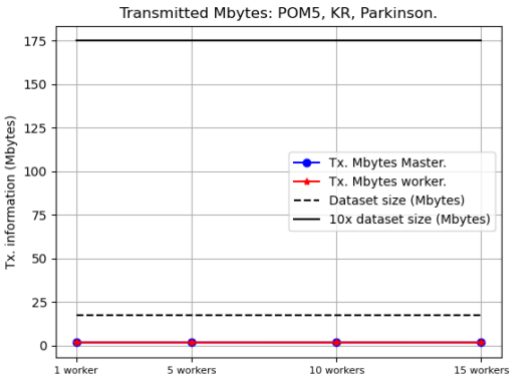


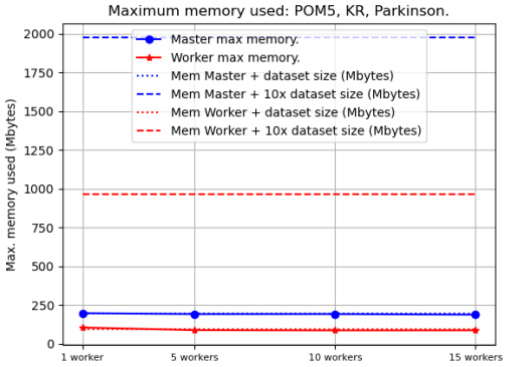


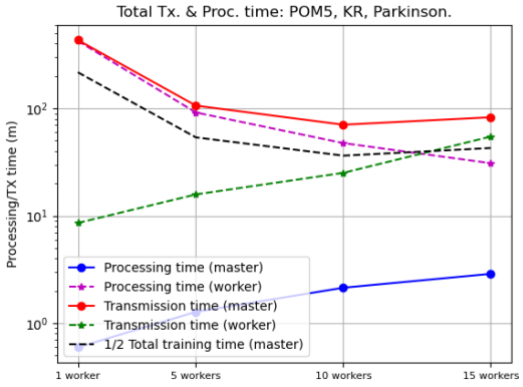
Parkinson


Performance and consistency: POM5, KR, Parkinson.
R2 (test)
1.075
1.050
1.025
1.000
0.975
0.950
0.925
0.900
Parkinson(Batch)
Parkinson(1w)
Parkinson(5w)
Parkinson(10w)
Parkinson(15w)

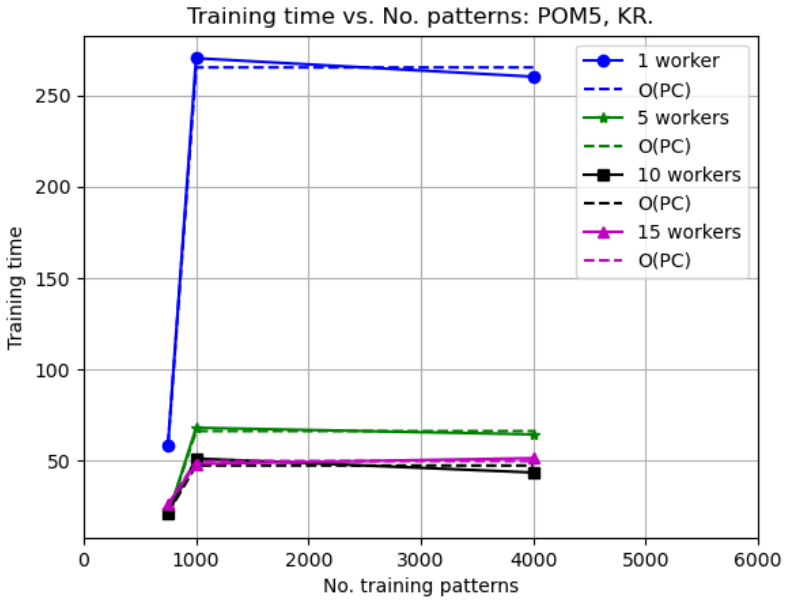

Total training time: POM5, KR, Parkinson.
Total training time.
Polynomial trend (-1.480000)
Training time (m)
600
500
400
300
200
100
1 worker
5 workers
10 workers
15 workers

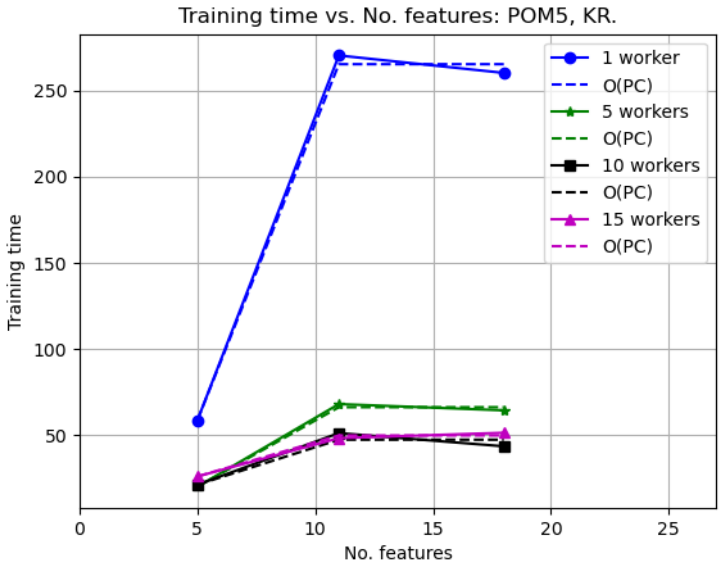

Transmitted Mbytes: POM5, KR, Parkinson.
Tx. information (Mbytes)
175
150
125
100
75
50
25
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Maximum memory used: POM5, KR, Parkinson.
Max. memory used (Mbytes)
2000
1750
1500
1250
1000
750
500
250
0
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM5, KR, Parkinson.
Processing/TX time (m)
10^2
10^1
10^0
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

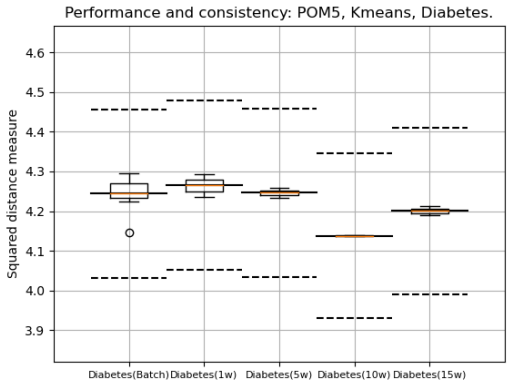
Scalability with respect to No. training patterns, number of features/model size.

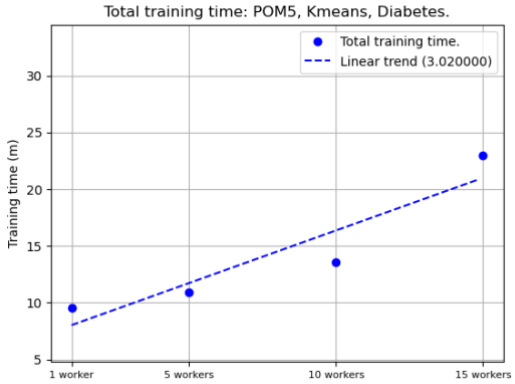


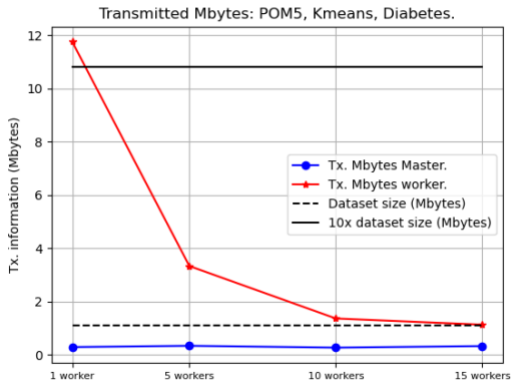


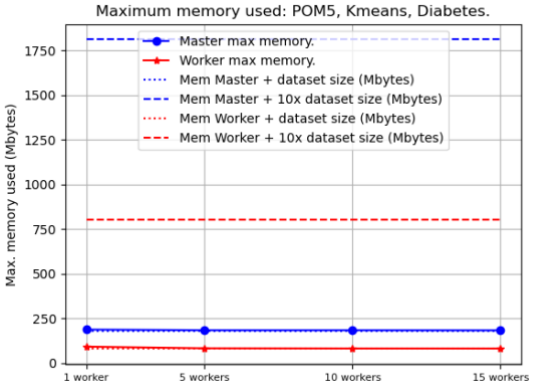
10.2.5 Clustering K-means (Kmeans)

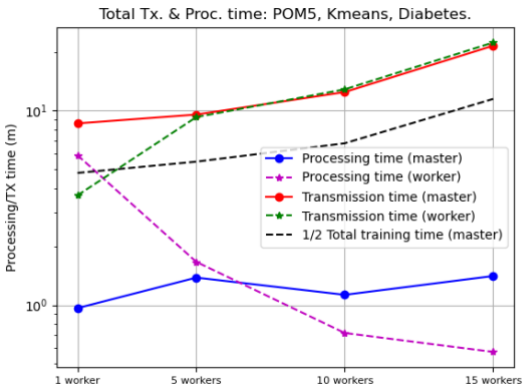
Diabetes



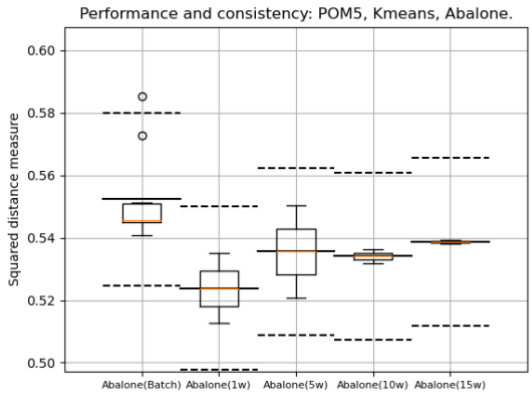


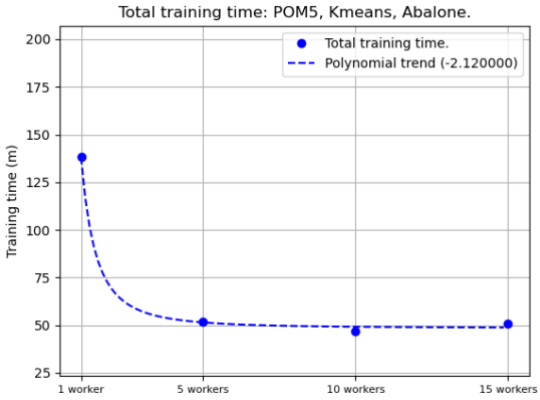


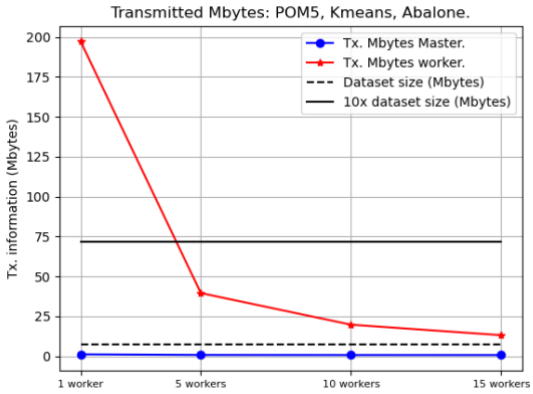


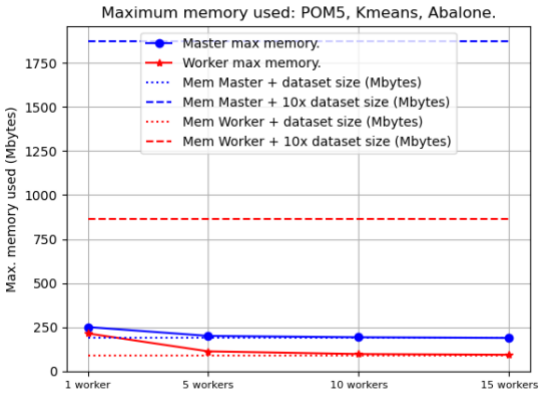


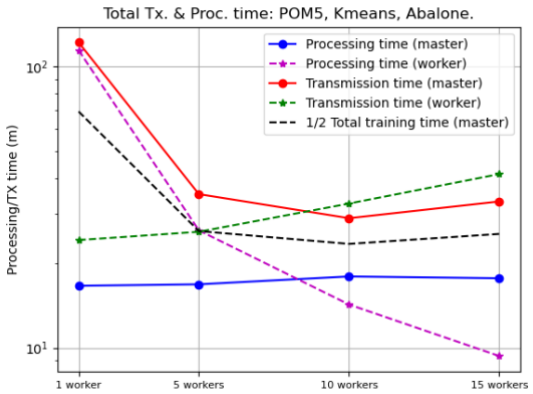
Abalone


Performance and consistency: POM5, Kmeans, Abalone.
Squared distance measure
0.60
0.58
0.56
0.54
0.52
0.50
Abalone(Batch)
Abalone(1w)
Abalone(5w)
Abalone(10w)
Abalone(15w)

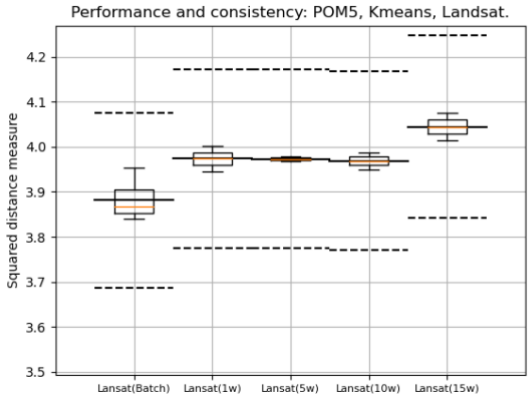

Total training time: POM5, Kmeans, Abalone.
Total training time.
Polynomial trend (-2.120000)
Training time (m)
200
175
150
125
100
75
50
25
1 worker
5 workers
10 workers
15 workers

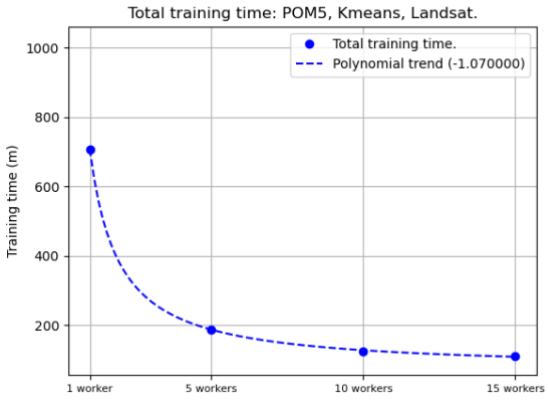

Transmitted Mbytes: POM5, Kmeans, Abalone.
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
Tx. information (Mbytes)
200
175
150
125
100
75
50
25
0
1 worker
5 workers
10 workers
15 workers

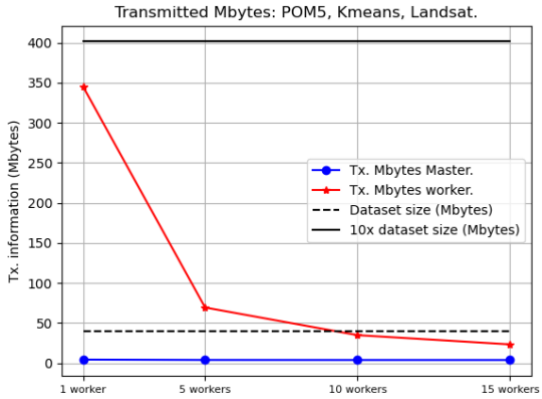

Maximum memory used: POM5, Kmeans, Abalone.
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Max. memory used (Mbytes)
1750
1500
1250
1000
750
500
250
0
1 worker
5 workers
10 workers
15 workers

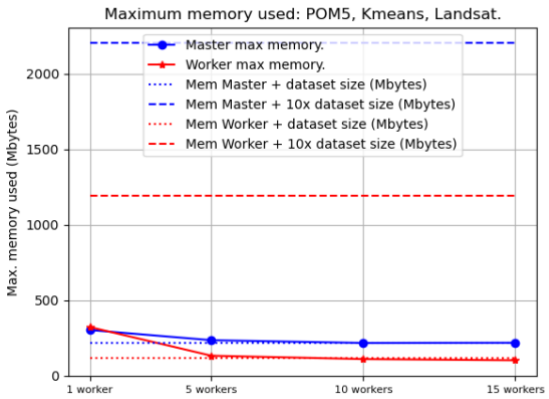

Total Tx. & Proc. time: POM5, Kmeans, Abalone.
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
Processing/TX time (m)
10^2
10^1
1 worker
5 workers
10 workers
15 workers

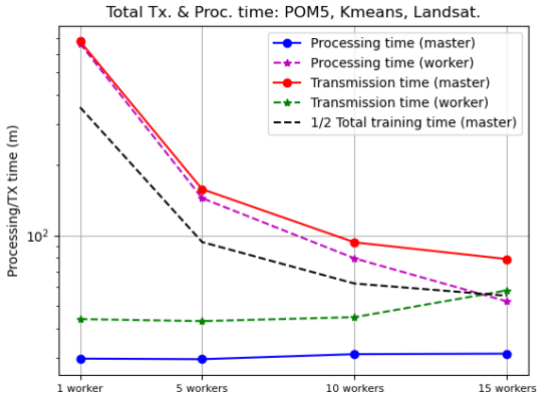
Landsat


Performance and consistency: POM5, Kmeans, Landsat.
Squared distance measure
4.2
4.1
4.0
3.9
3.8
3.7
3.6
3.5
Lansat(Batch)
Lansat(1w)
Lansat(5w)
Lansat(10w)
Lansat(15w)

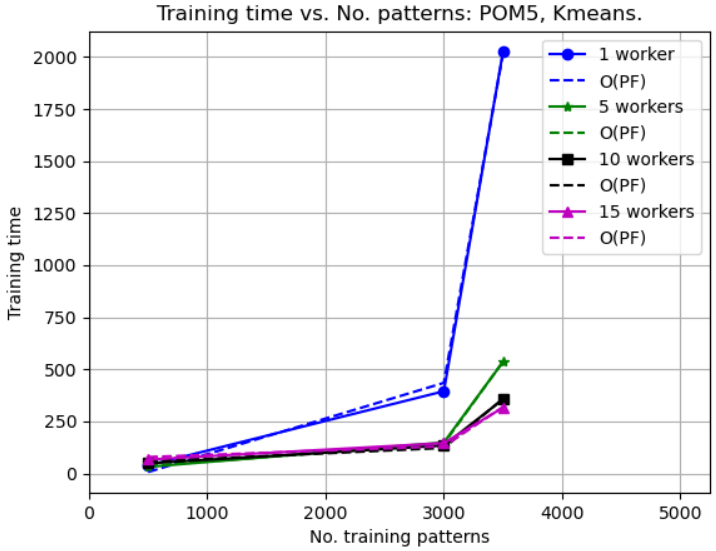

Total training time: POM5, Kmeans, Landsat.
Total training time.
Polynomial trend (-1.070000)
Training time (m)
1000
800
600
400
200
1 worker
5 workers
10 workers
15 workers

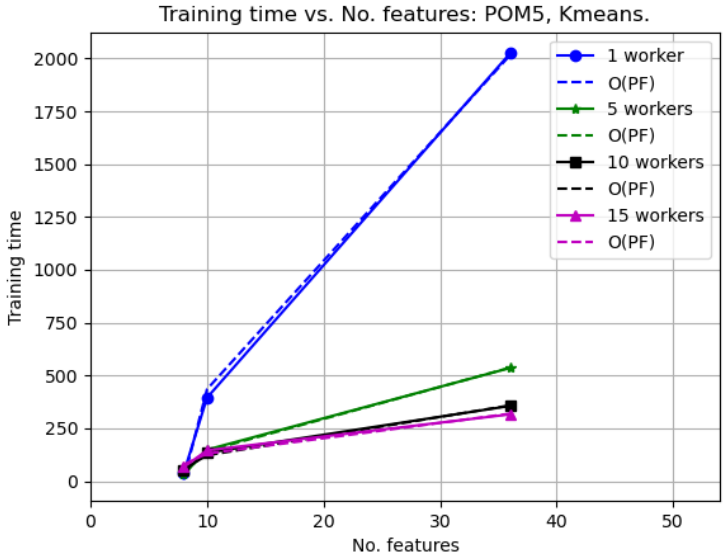

Transmitted Mbytes: POM5, Kmeans, Landsat.
Tx. information (Mbytes)
400
350
300
250
200
150
100
50
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Maximum memory used: POM5, Kmeans, Landsat.
Max. memory used (Mbytes)
2000
1500
1000
500
0
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM5, Kmeans, Landsat.
Processing/TX time (m)
10^2
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

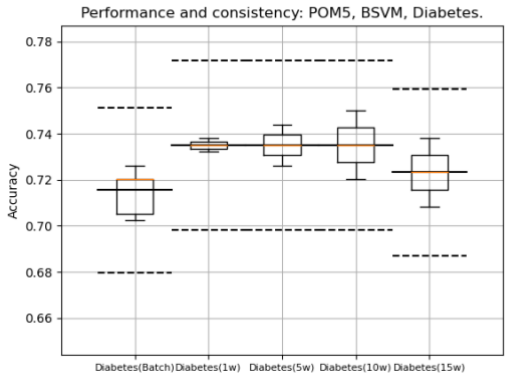
Scalability with respect to No. training patterns, number of features/model size.

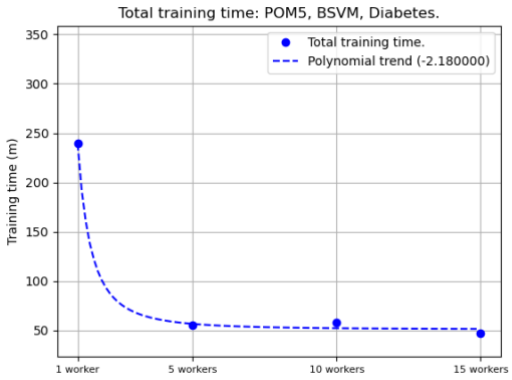


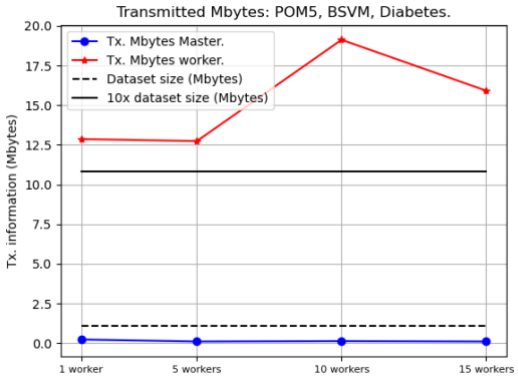


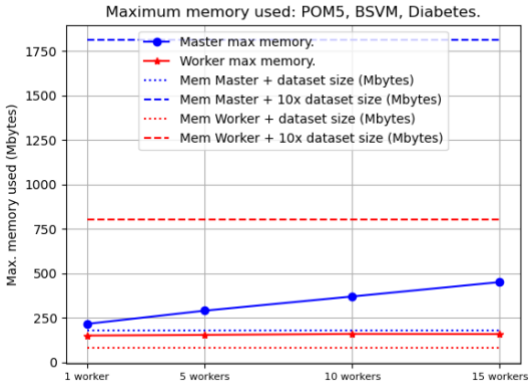
10.2.6 Budget Support Vector Machine (BSVM)

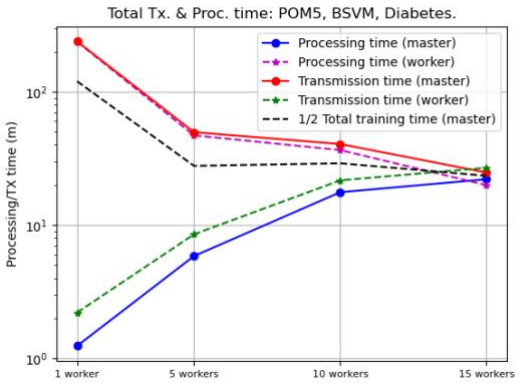
Diabetes



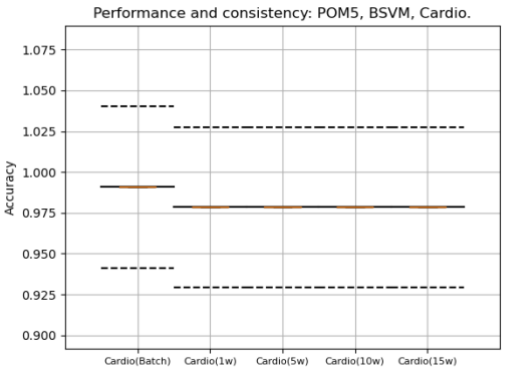


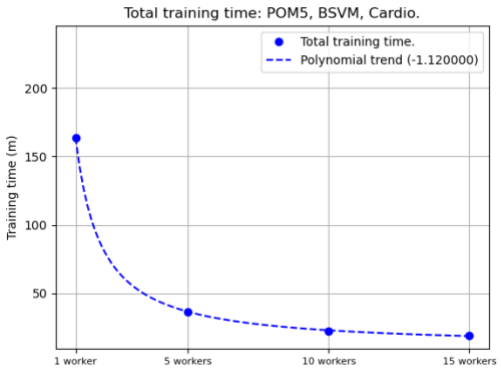


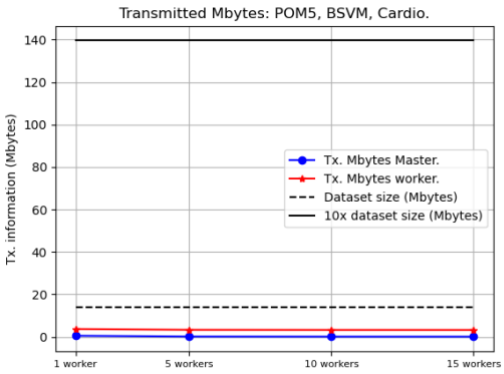


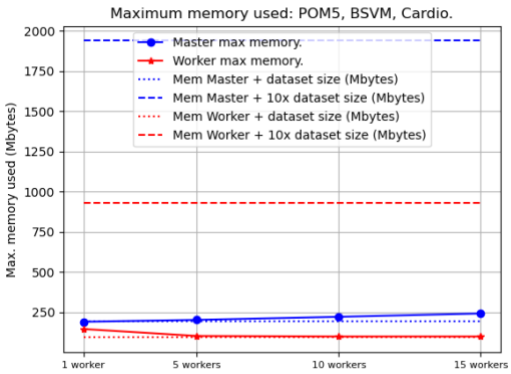


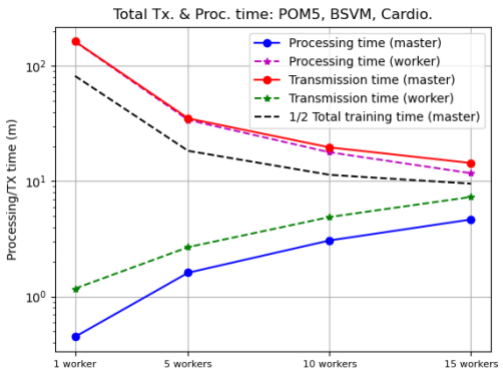
Cardiopathy



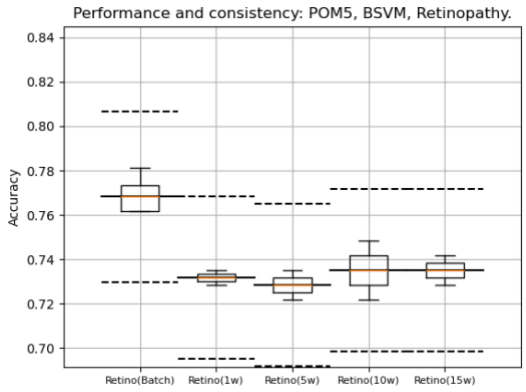


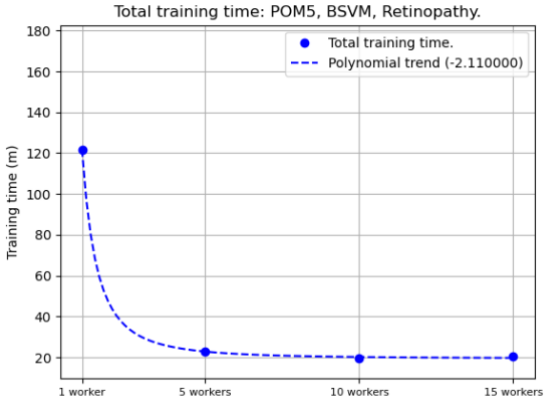


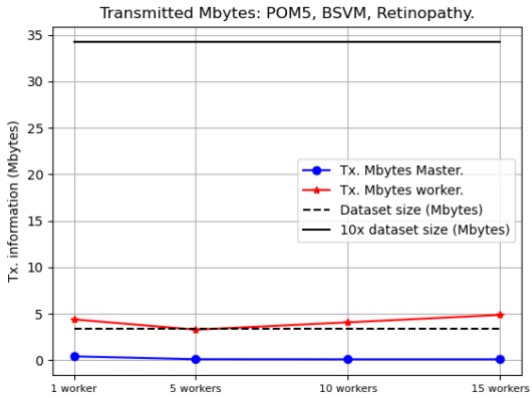


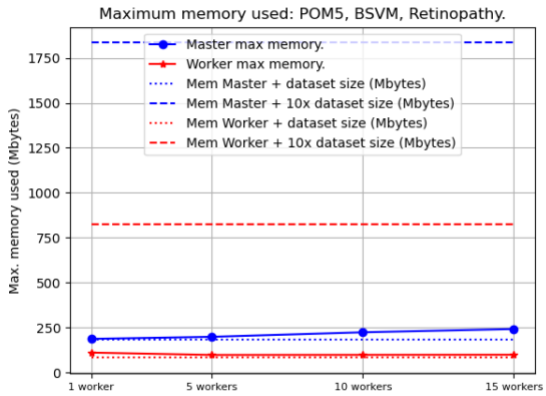


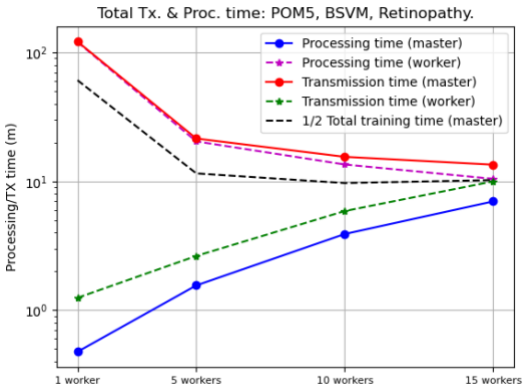
Retinopathy



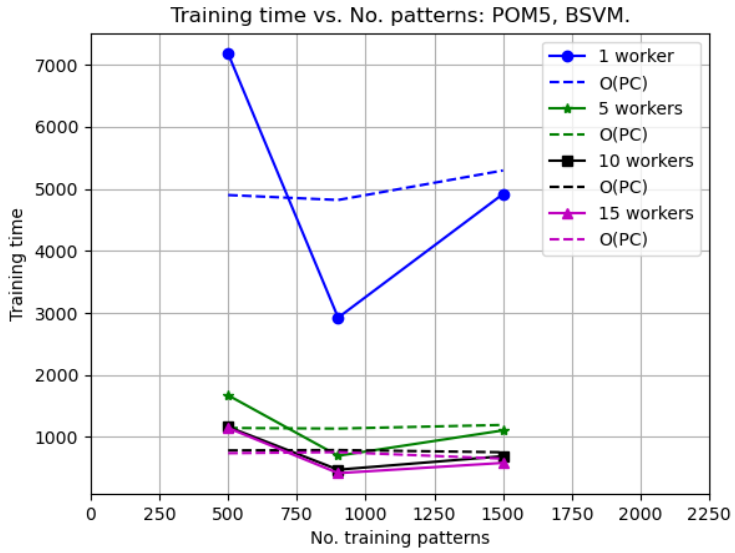


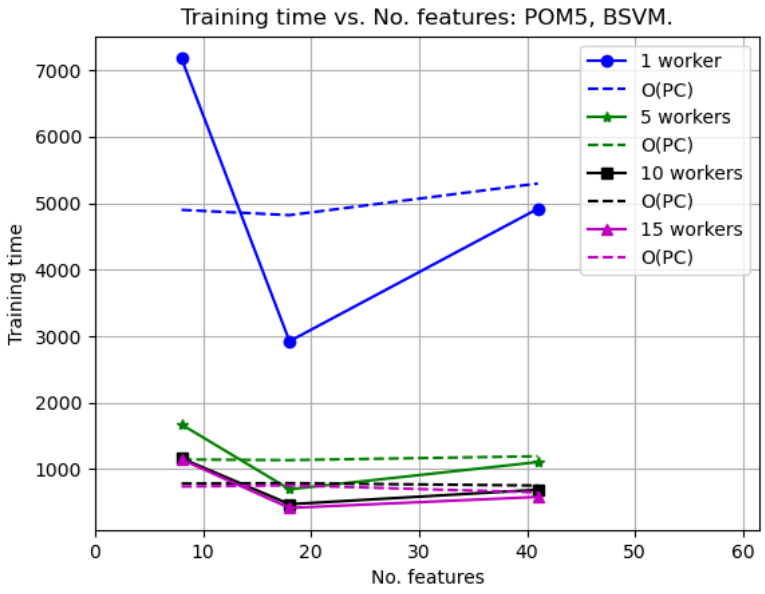






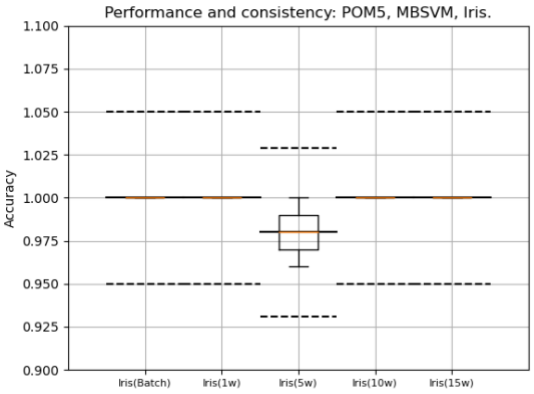
Scalability with respect to No. training patterns, number of features/model size.

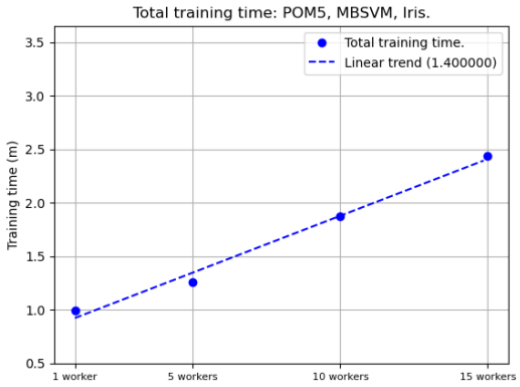


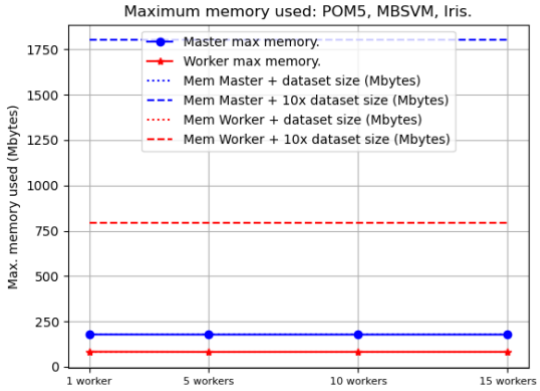


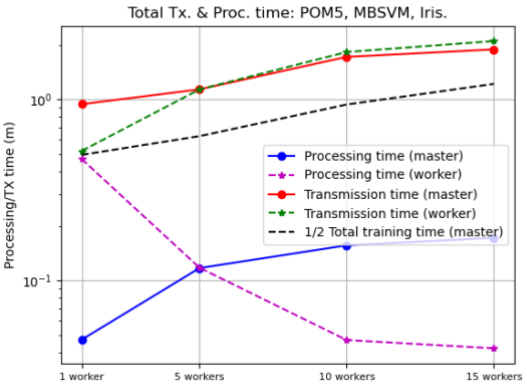
10.2.7 Multiclass Budget Support Vector Machine (MBSVM)

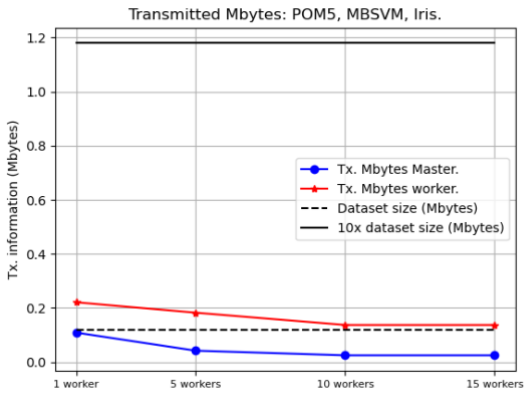
Iris







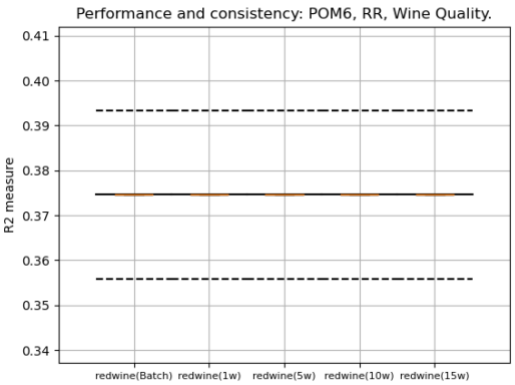


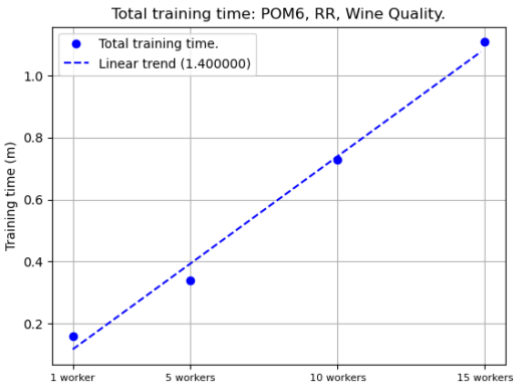


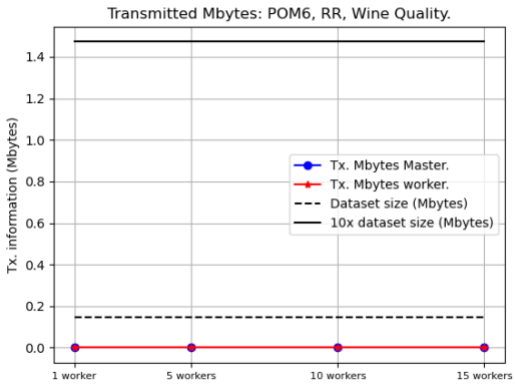
10.3 POM 6

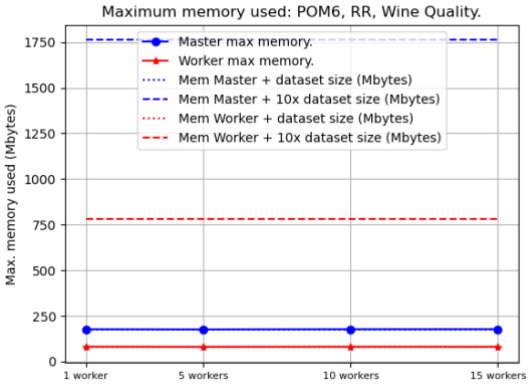
10.3.1 Ridge Regression (RR)

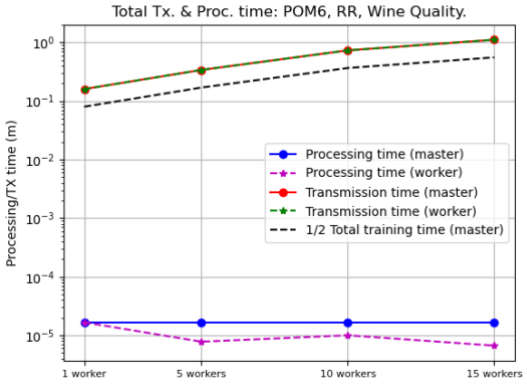
Wine quality



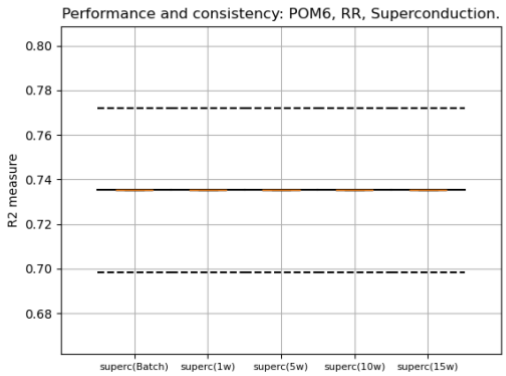


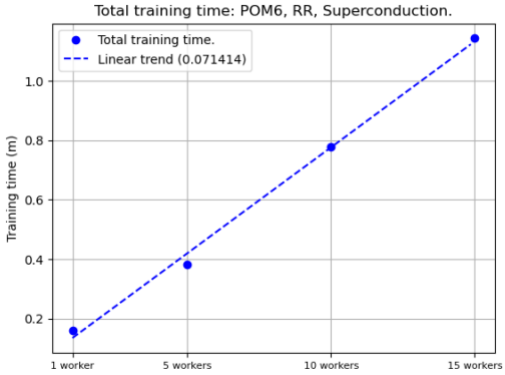


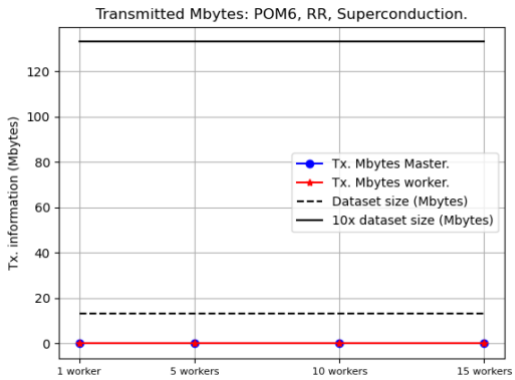


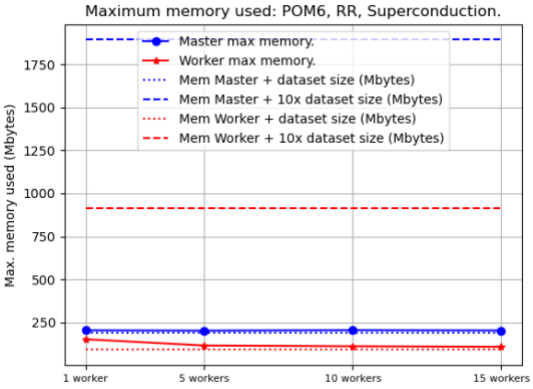


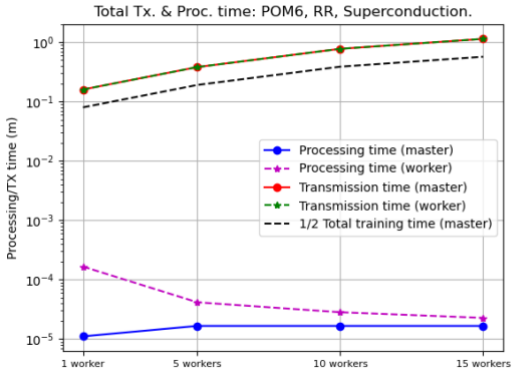
Superconduction



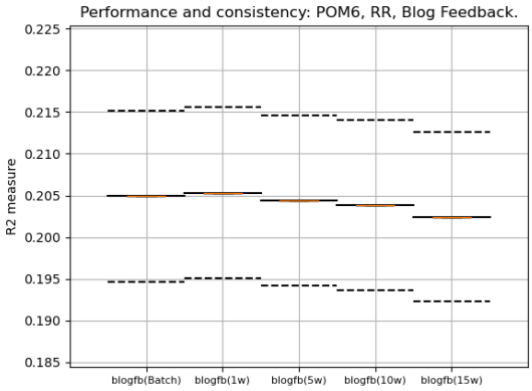


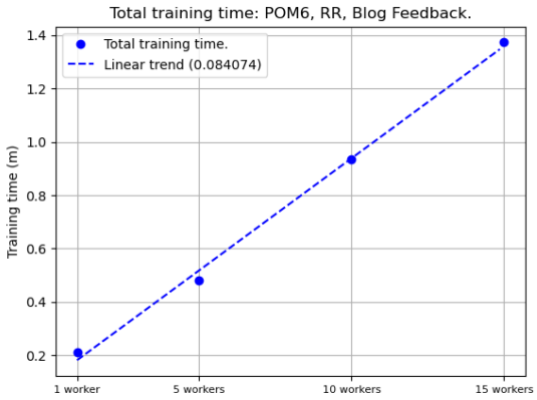


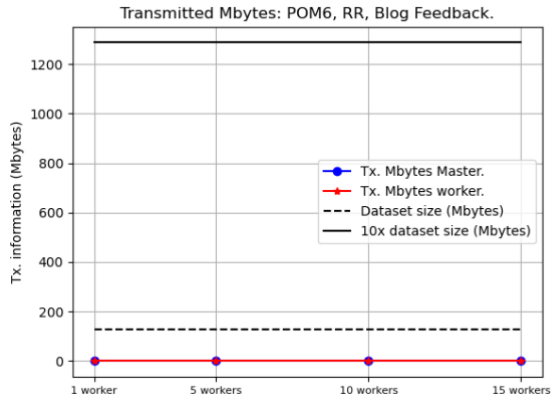


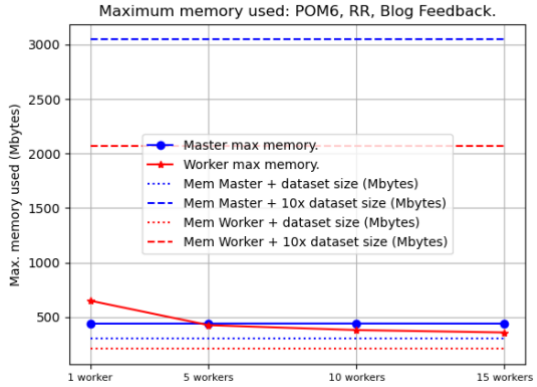


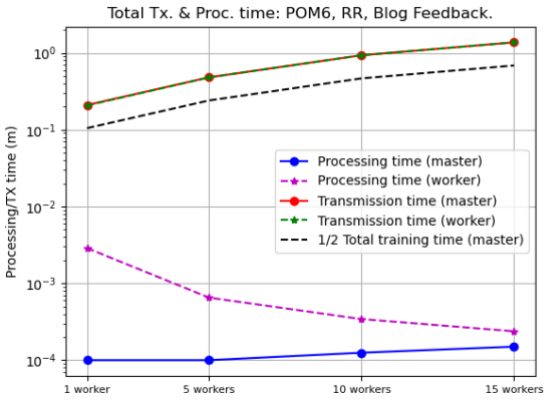
Blog feedback



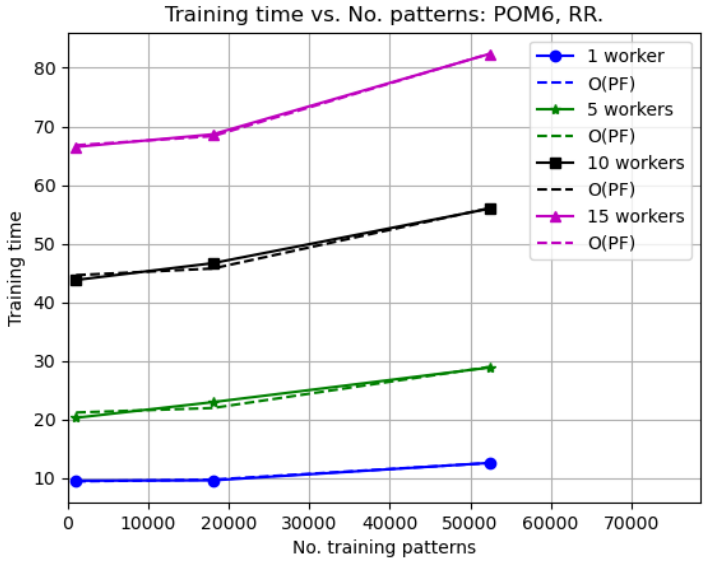


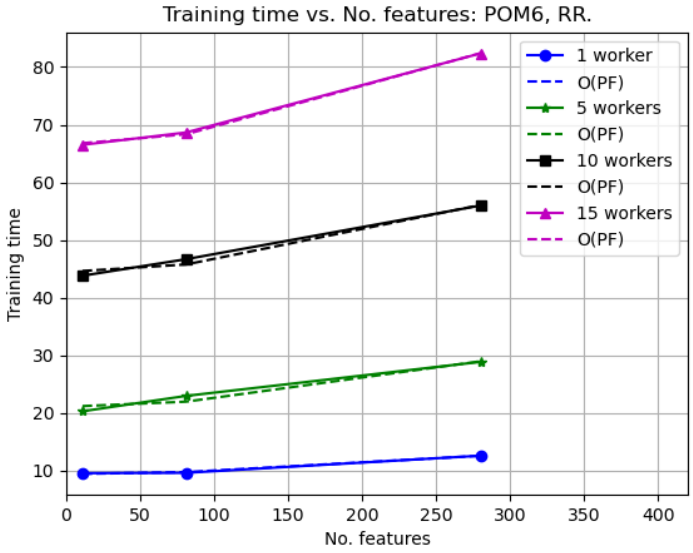






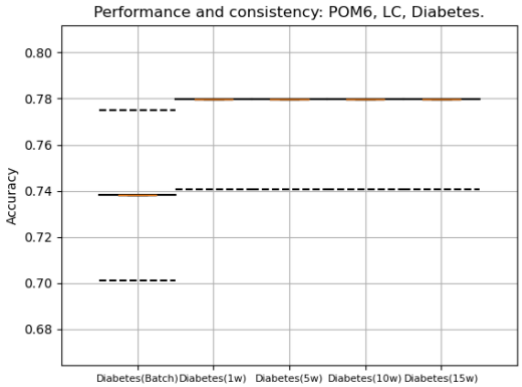
Scalability with respect to No. training patterns, number of features/model size.

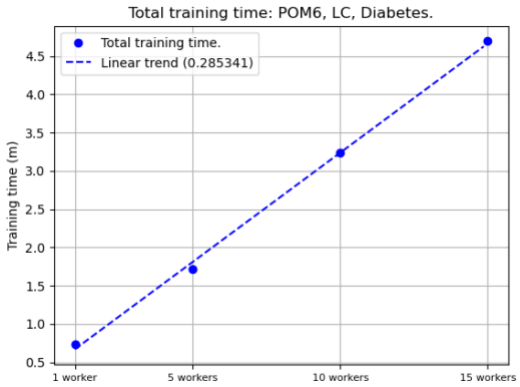


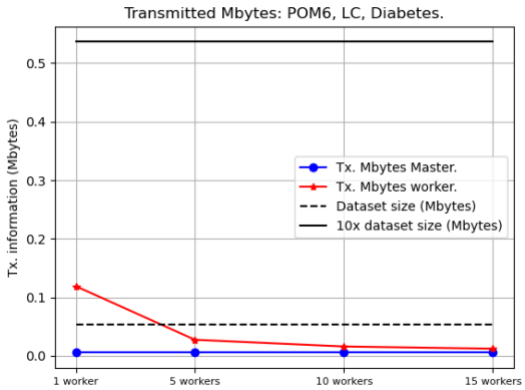


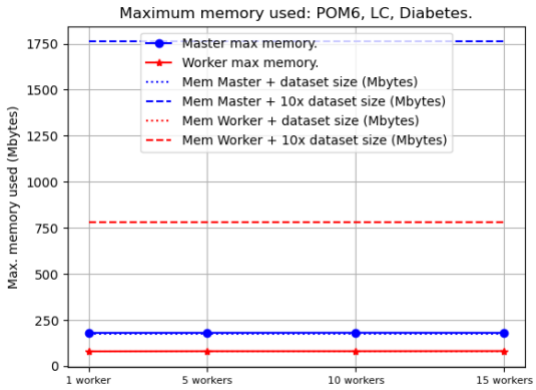
10.3.2 Logistic Classifier (LC)

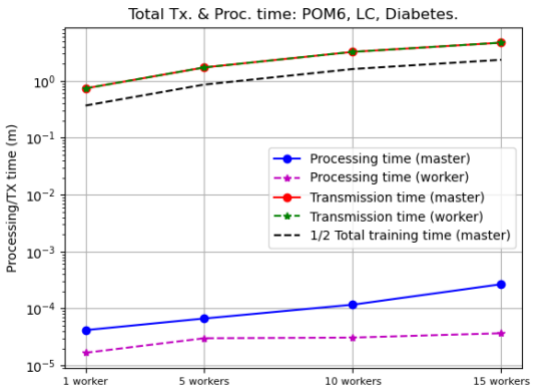
Diabetes



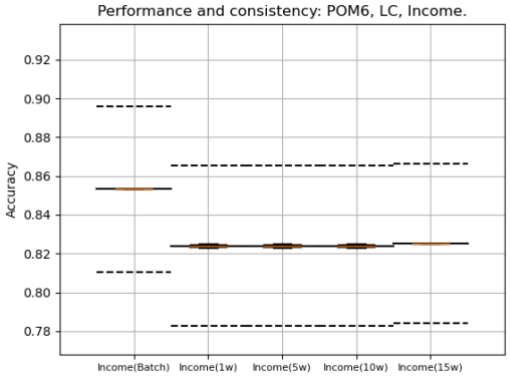


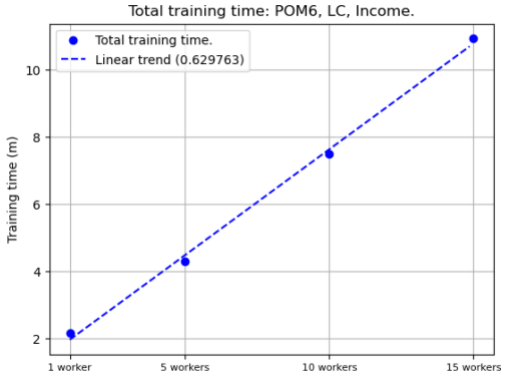


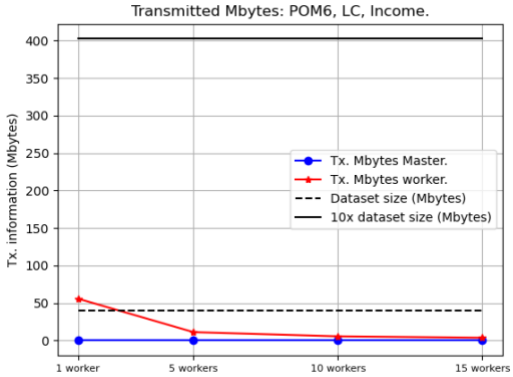


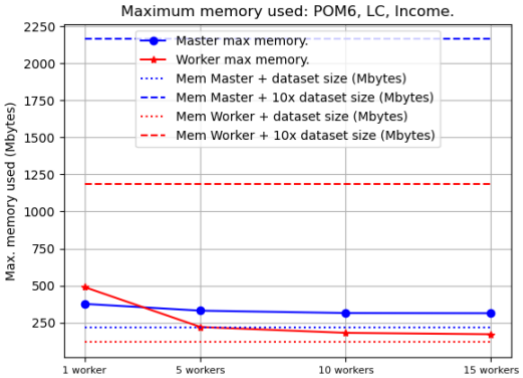


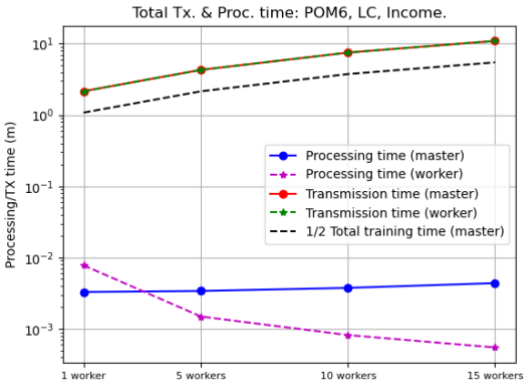
Income


Performance and consistency: POM6, LC, Income.
Accuracy
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
Income(Batch)
Income(1w)
Income(5w)
Income(10w)
Income(15w)

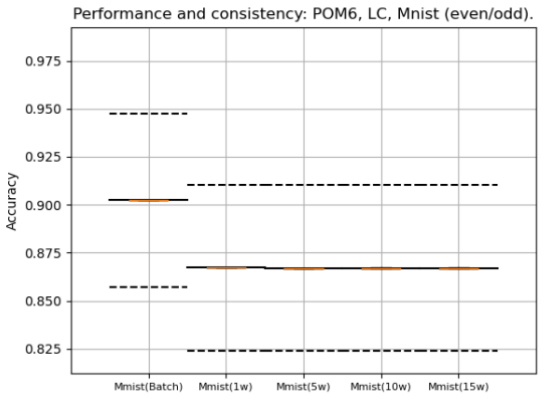

Total training time: POM6, LC, Income.
Total training time.
Linear trend (0.629763)
Training time (m)
10
8
6
4
2
1 worker
5 workers
10 workers
15 workers

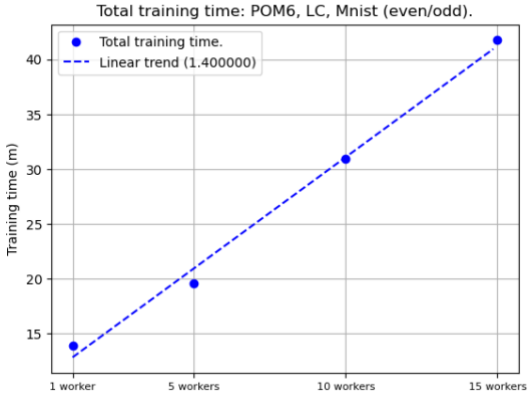

Transmitted Mbytes: POM6, LC, Income.
Tx. information (Mbytes)
400
350
300
250
200
150
100
50
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

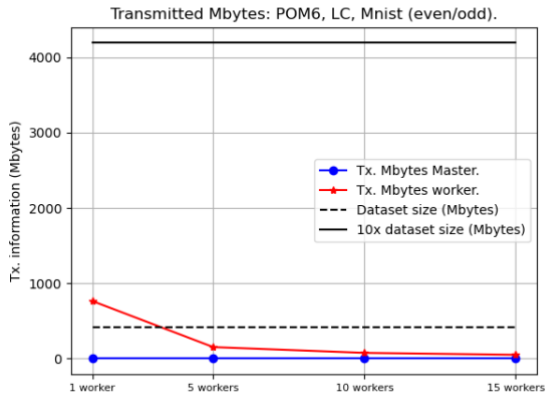

Maximum memory used: POM6, LC, Income.
Max. memory used (Mbytes)
2250
2000
1750
1500
1250
1000
750
500
250
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

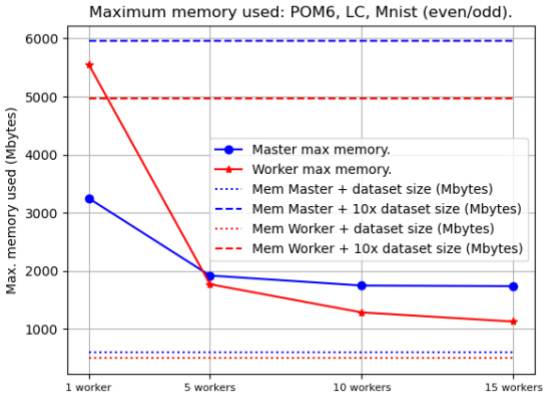

Total Tx. & Proc. time: POM6, LC, Income.
Processing/TX time (m)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

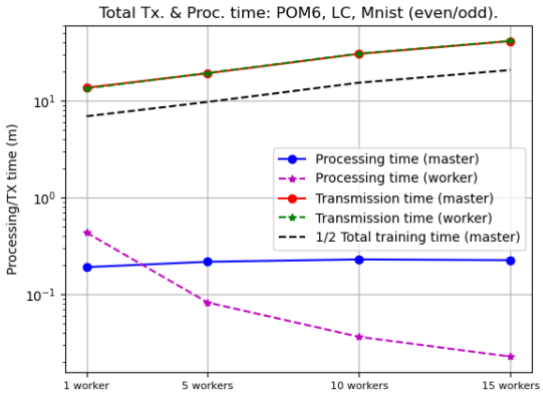
MNIST handwritten digits (even/odd)



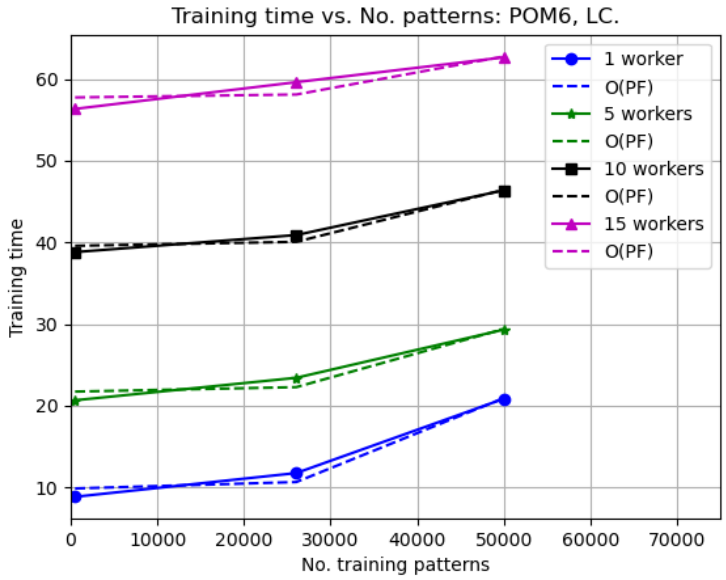


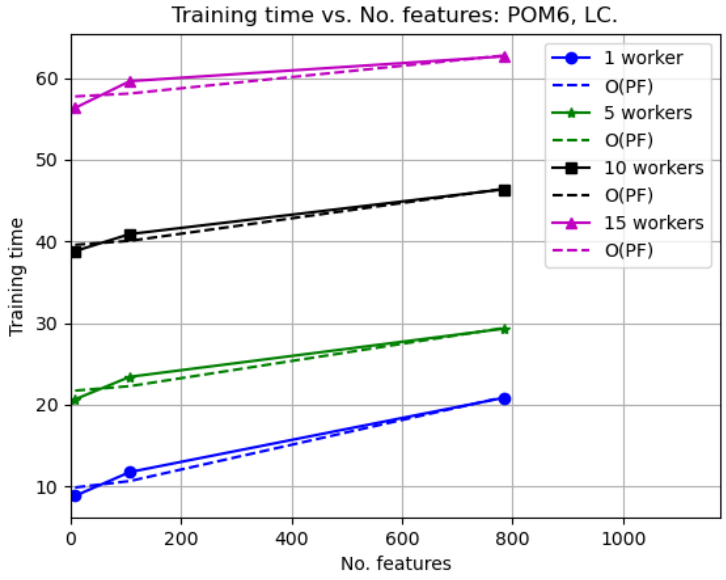






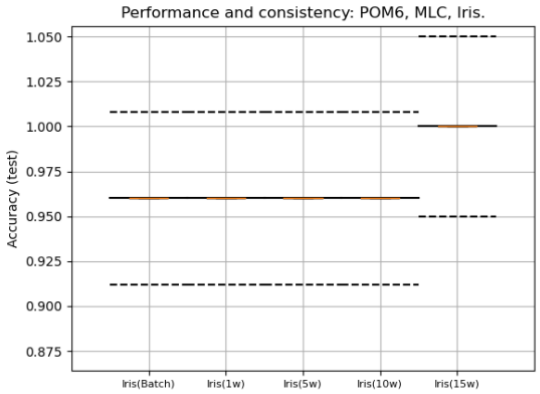
Scalability with respect to No. training patterns, number of features/model size.

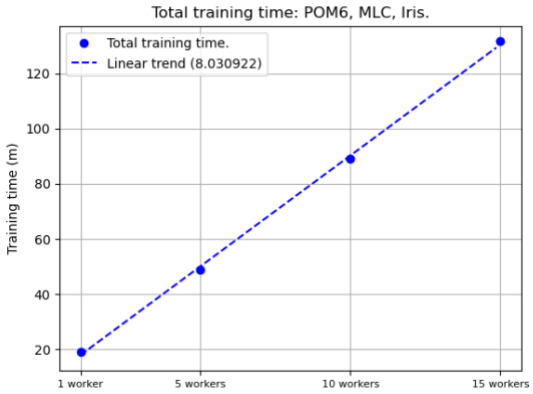


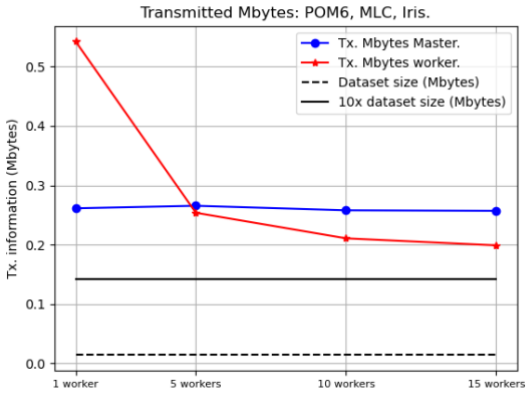


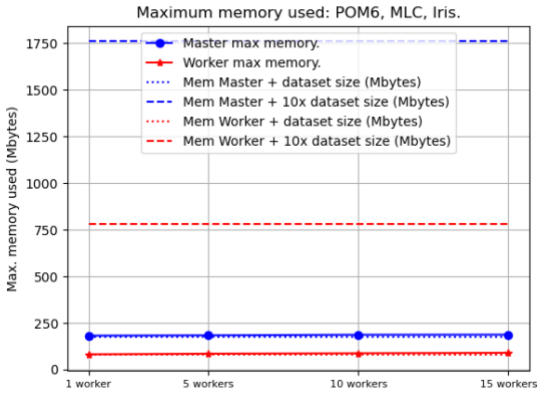
10.3.3 Multiclass Logistic Classifier (MLC)

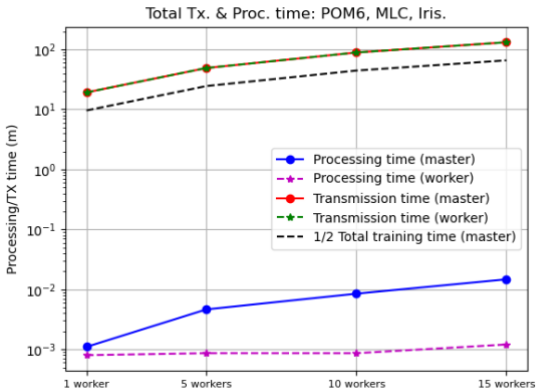
Iris



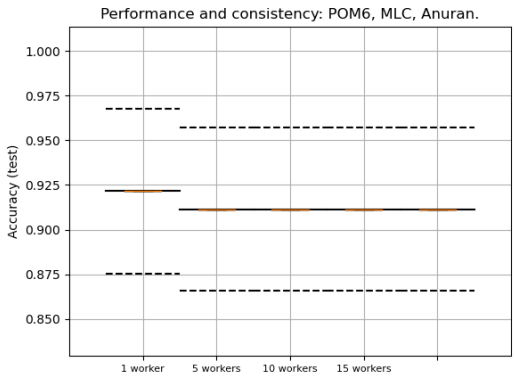


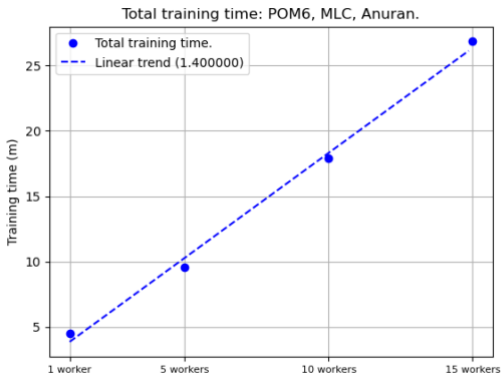


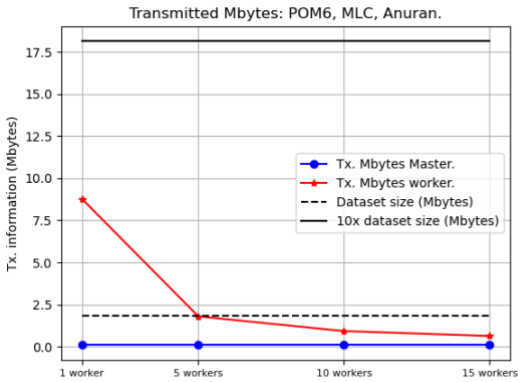


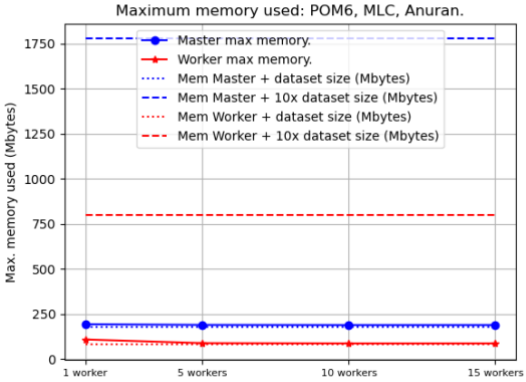


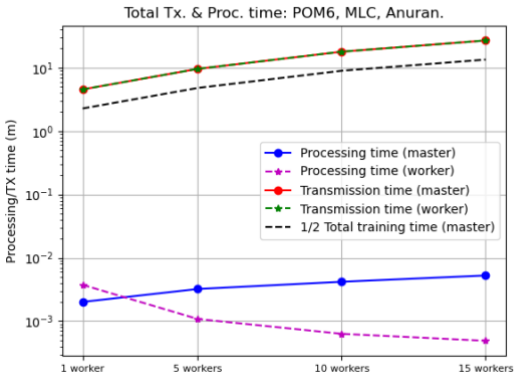
Anuran


Performance and consistency: POM6, MLC, Anuran.
Accuracy (test)
1.000
0.975
0.950
0.925
0.900
0.875
0.850
1 worker
5 workers
10 workers
15 workers

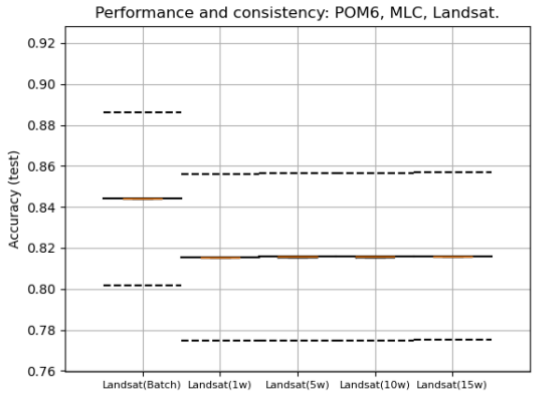

Total training time: POM6, MLC, Anuran.
Total training time.
Linear trend (1.400000)
Training time (m)
25
20
15
10
5
1 worker
5 workers
10 workers
15 workers

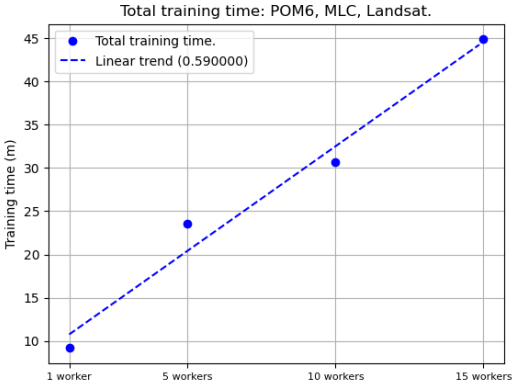

Transmitted Mbytes: POM6, MLC, Anuran.
Tx. information (Mbytes)
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

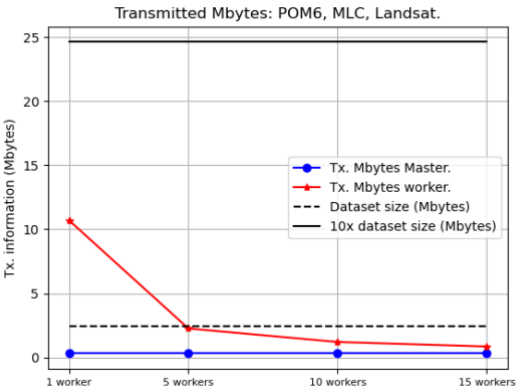

Maximum memory used: POM6, MLC, Anuran.
Max. memory used (Mbytes)
1750
1500
1250
1000
750
500
250
0
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

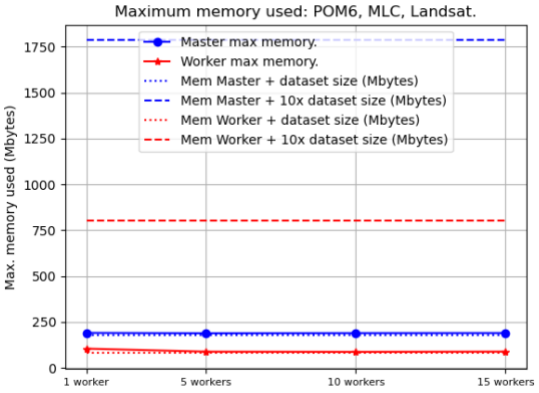

Total Tx. & Proc. time: POM6, MLC, Anuran.
Processing/TX time (m)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

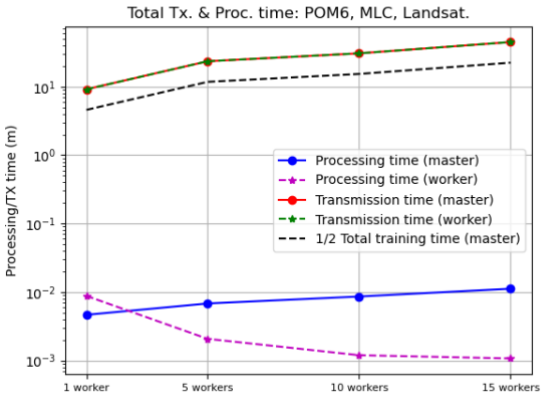
Landsat


Performance and consistency: POM6, MLC, Landsat.
Accuracy (test)
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.76
Landsat(Batch)
Landsat(1w)
Landsat(5w)
Landsat(10w)
Landsat(15w)

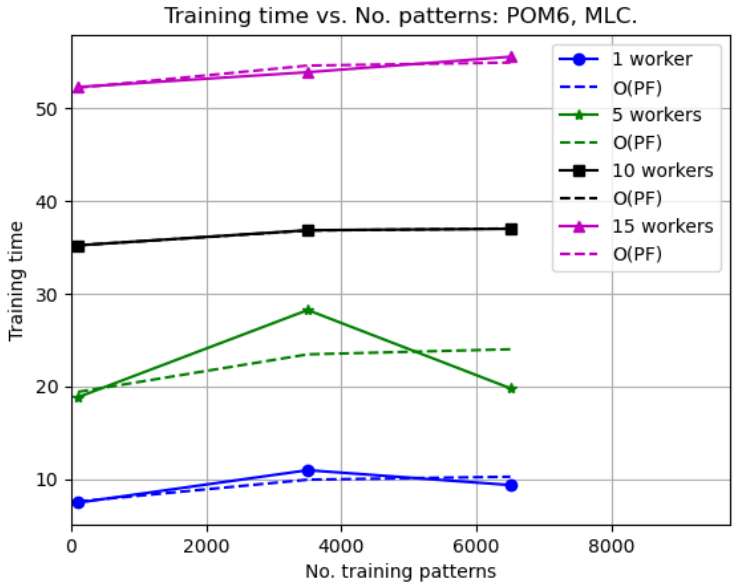

Total training time: POM6, MLC, Landsat.
Total training time.
Linear trend (0.590000)
Training time (m)
45
40
35
30
25
20
15
10
1 worker
5 workers
10 workers
15 workers

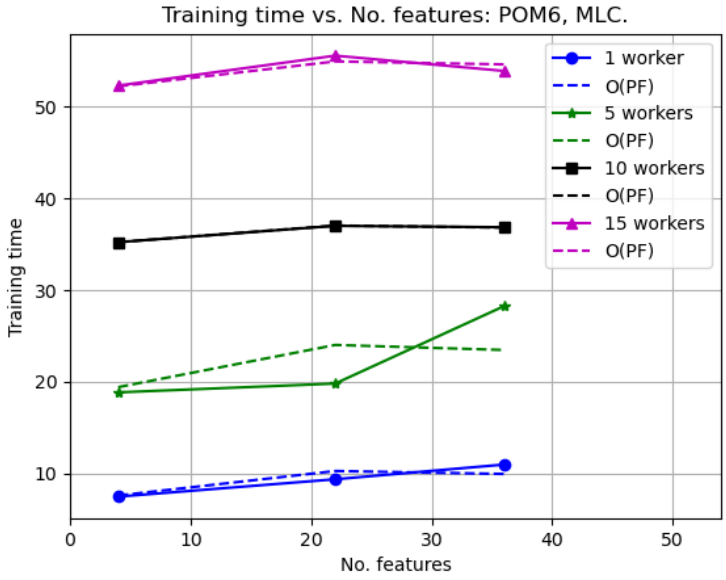

Transmitted Mbytes: POM6, MLC, Landsat.
Tx. information (Mbytes)
25
20
15
10
5
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Maximum memory used: POM6, MLC, Landsat.
Max. memory used (Mbytes)
1750
1500
1250
1000
750
500
250
0
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM6, MLC, Landsat.
Processing/TX time (m)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

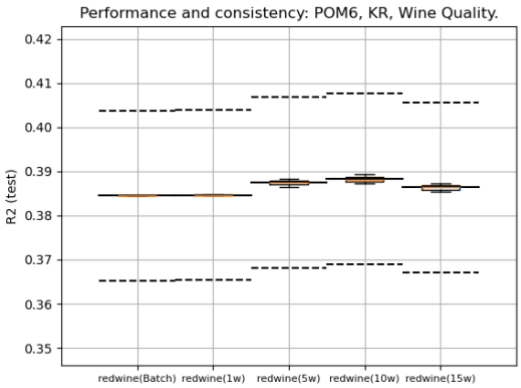
Scalability with respect to No. training patterns, number of features/model size.

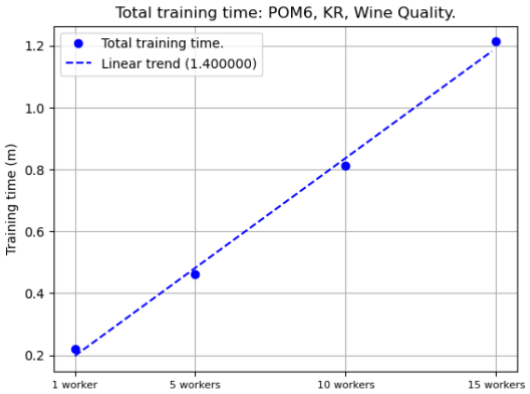


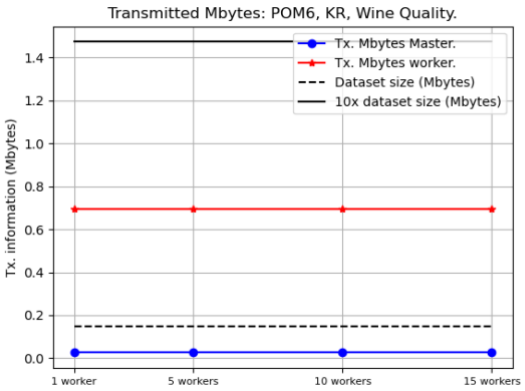


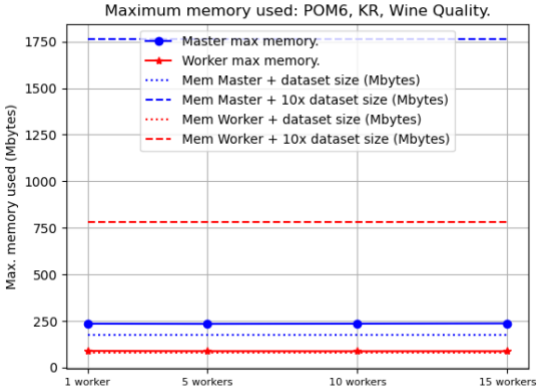
10.3.4 Kernel Regression (KR)

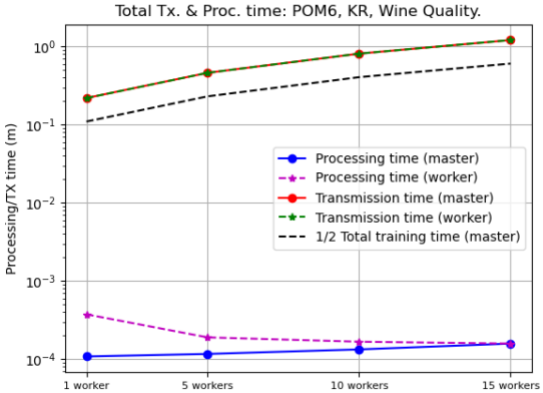
Wine quality



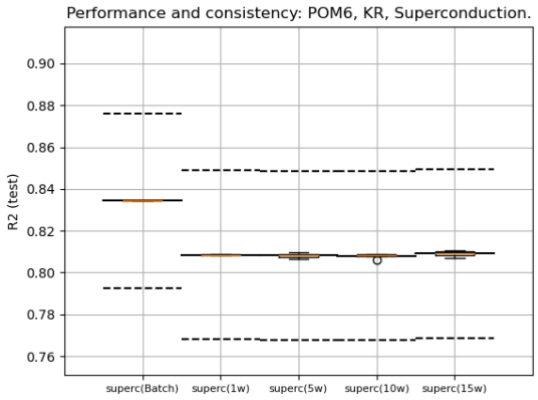


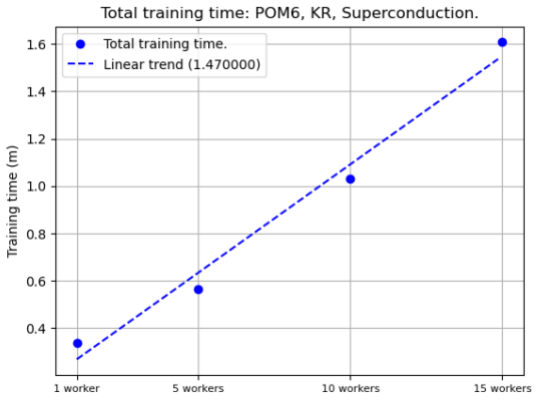


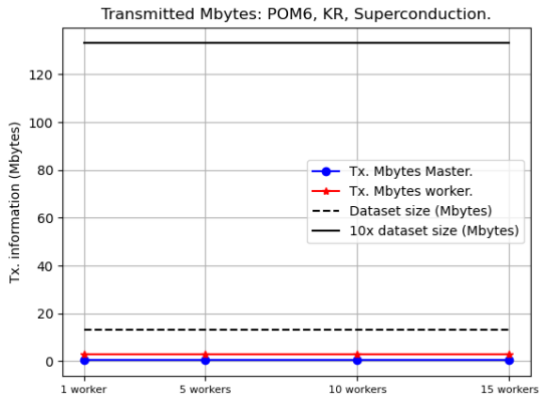


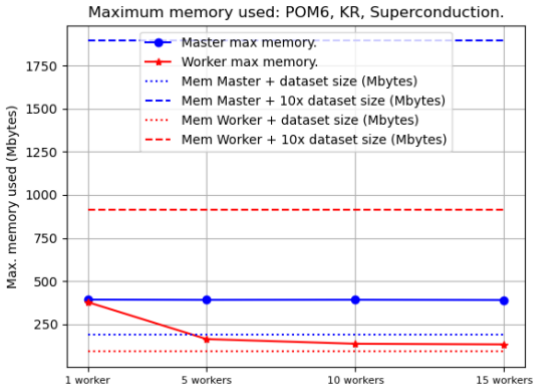


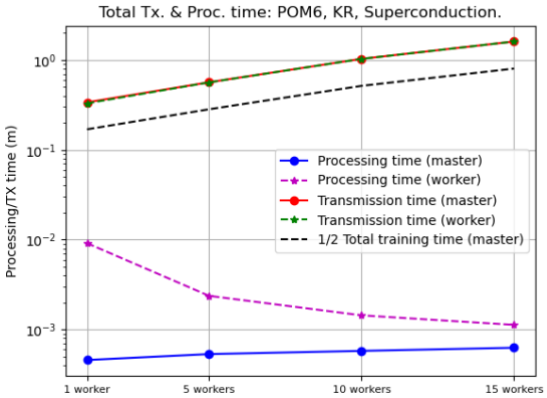
Superconduction



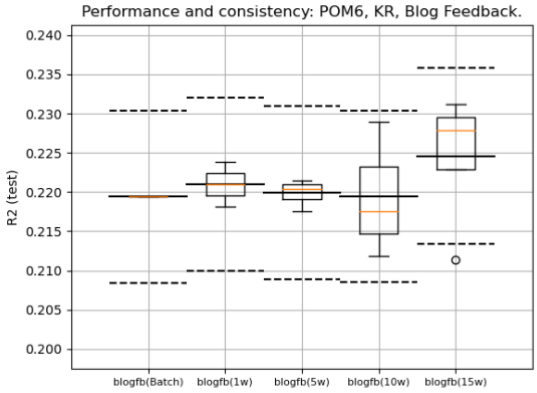


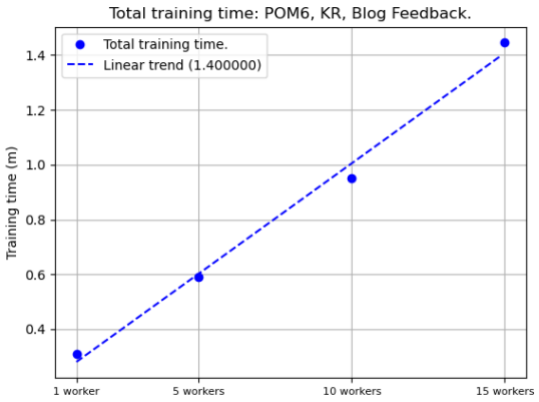


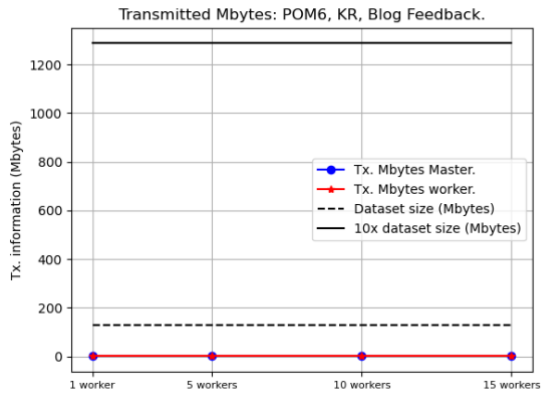


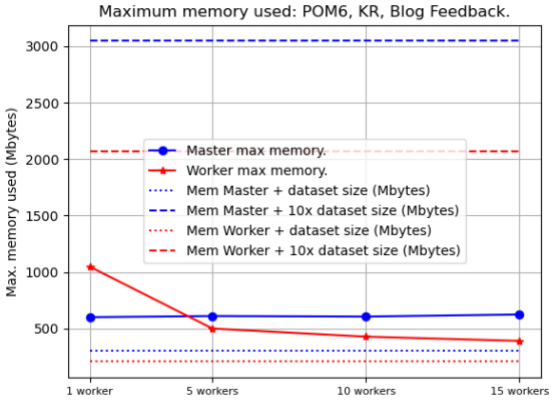


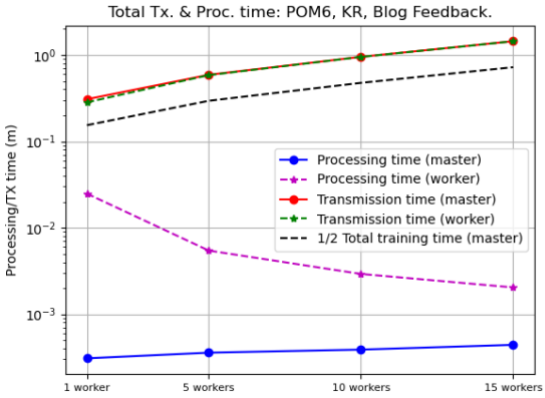
Blog feedback



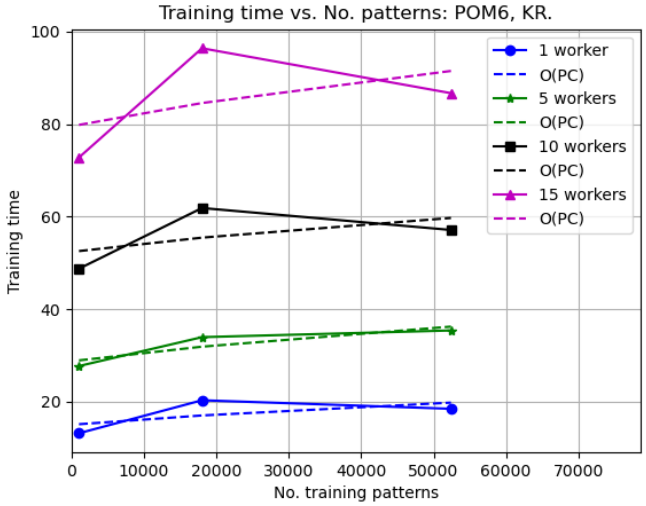


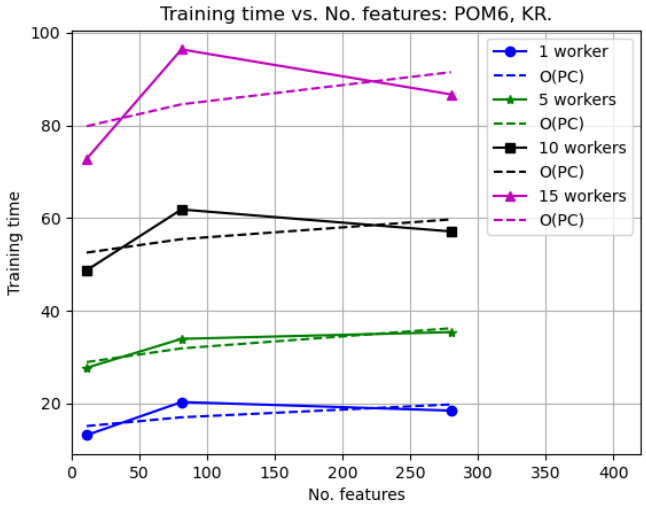






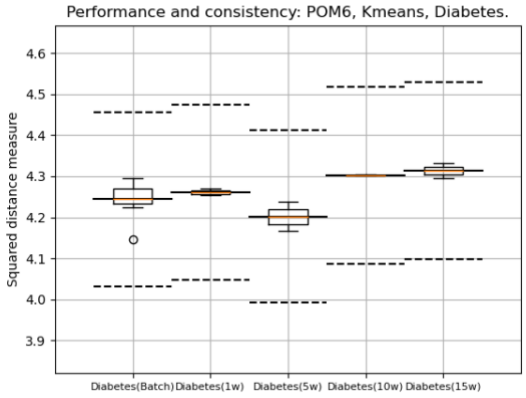
Scalability with respect to No. training patterns, number of features/model size.

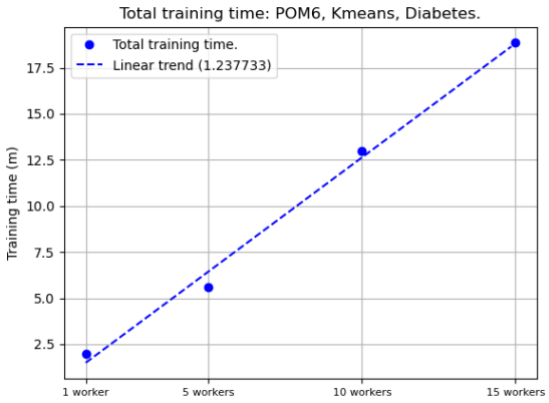


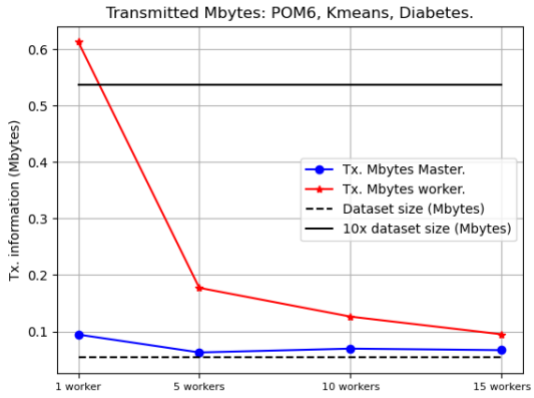


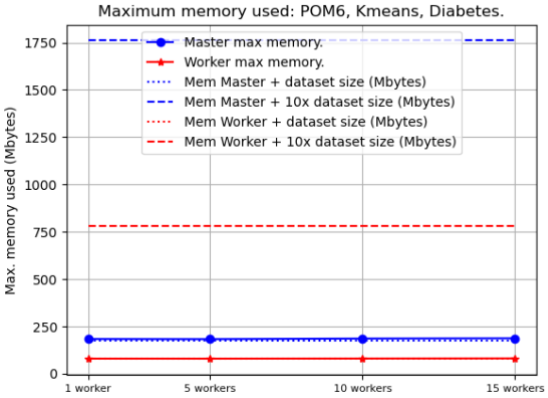
10.3.5 Clustering K-means (Kmeans)

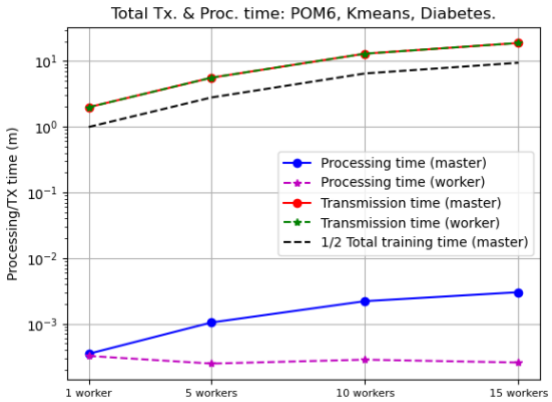
Diabetes



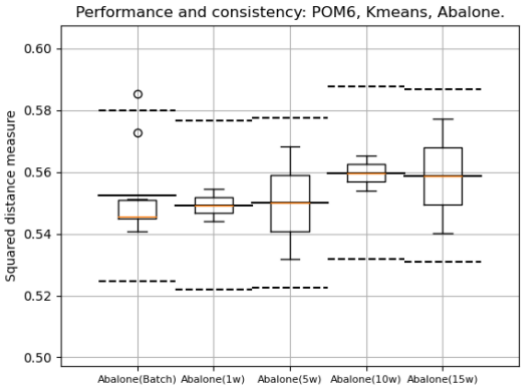


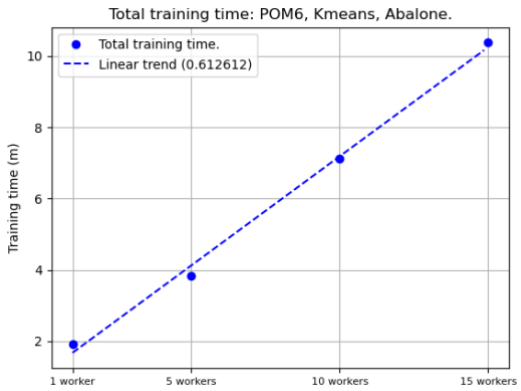


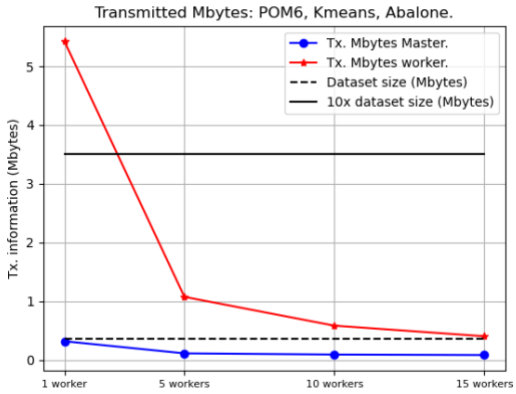


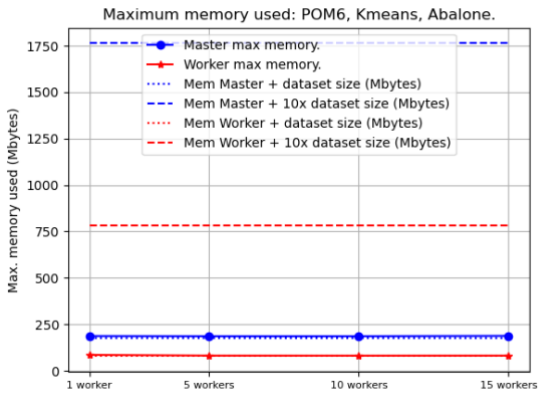


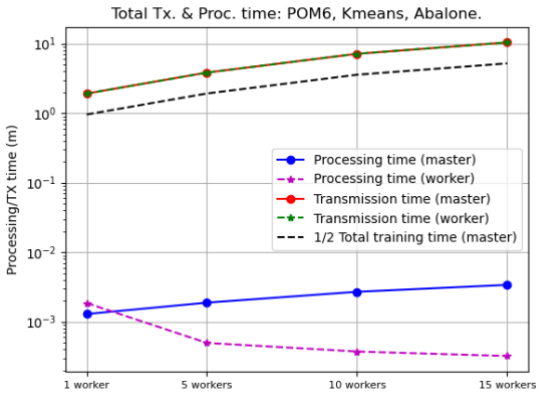
Abalone


Performance and consistency: POM6, Kmeans, Abalone.
Squared distance measure
0.60
0.58
0.56
0.54
0.52
0.50
Abalone(Batch)
Abalone(1w)
Abalone(5w)
Abalone(10w)
Abalone(15w)

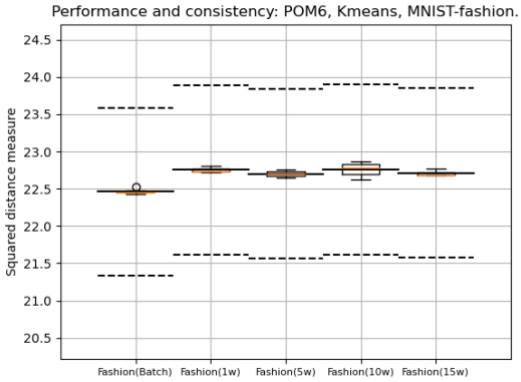

Total training time: POM6, Kmeans, Abalone.
Total training time.
Linear trend (0.612612)
Training time (m)
10
8
6
4
2
1 worker
5 workers
10 workers
15 workers

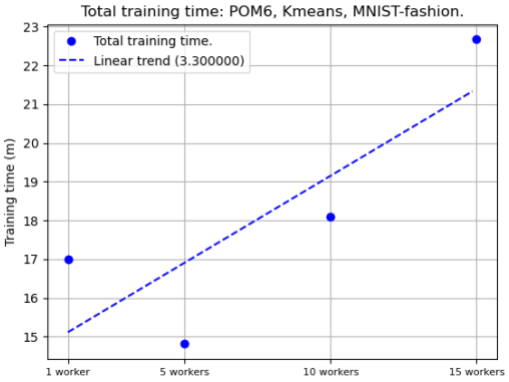

Transmitted Mbytes: POM6, Kmeans, Abalone.
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
Tx. information (Mbytes)
5
4
3
2
1
0
1 worker
5 workers
10 workers
15 workers

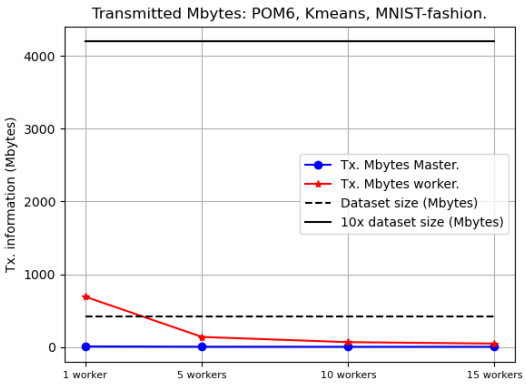

Maximum memory used: POM6, Kmeans, Abalone.
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Max. memory used (Mbytes)
1750
1500
1250
1000
750
500
250
0
1 worker
5 workers
10 workers
15 workers

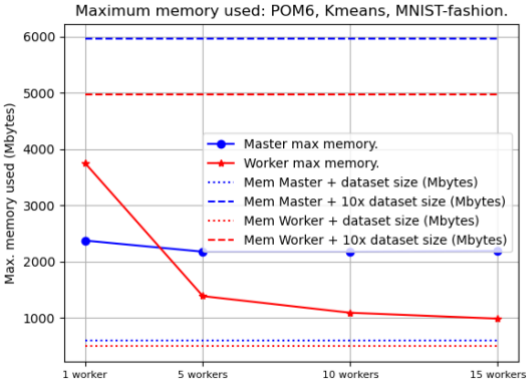

Total Tx. & Proc. time: POM6, Kmeans, Abalone.
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
Processing/TX time (m)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
1 worker
5 workers
10 workers
15 workers

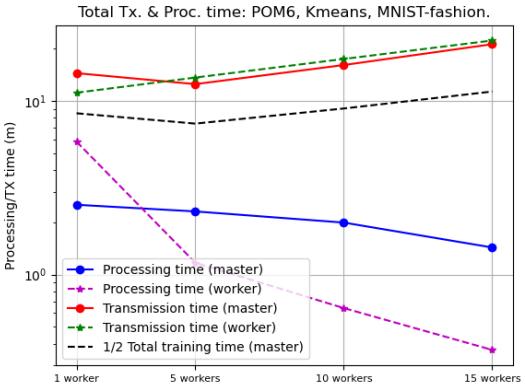
M-fashion


Performance and consistency: POM6, Kmeans, MNIST-fashion.
Squared distance measure
Fashion(Batch)
Fashion(1w)
Fashion(5w)
Fashion(10w)
Fashion(15w)

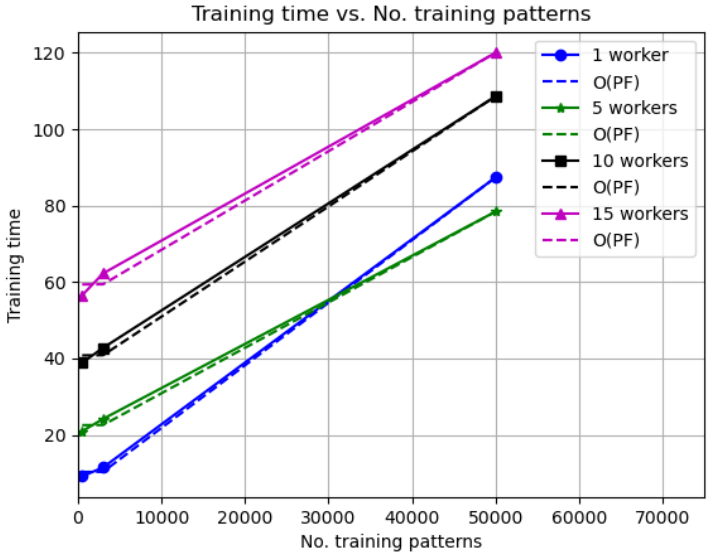

Total training time: POM6, Kmeans, MNIST-fashion.
Total training time.
Linear trend (3.300000)
Training time (m)
1 worker
5 workers
10 workers
15 workers

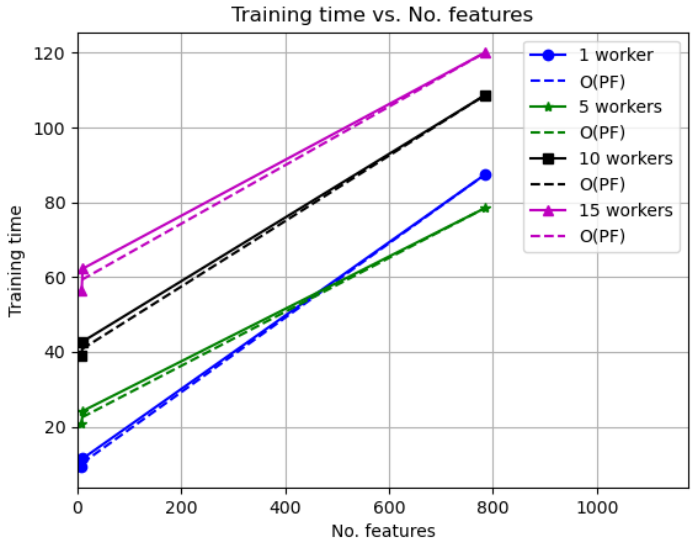

Transmitted Mbytes: POM6, Kmeans, MNIST-fashion.
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
Tx. information (Mbytes)
1 worker
5 workers
10 workers
15 workers


Maximum memory used: POM6, Kmeans, MNIST-fashion.
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Max. memory used (Mbytes)
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM6, Kmeans, MNIST-fashion.
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
Processing/TX time (m)
1 worker
5 workers
10 workers
15 workers

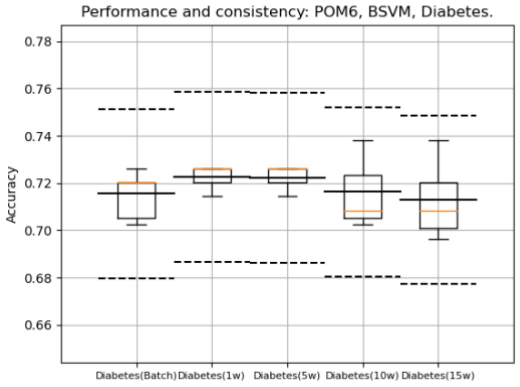
Scalability with respect to No. training patterns, number of features/model size.

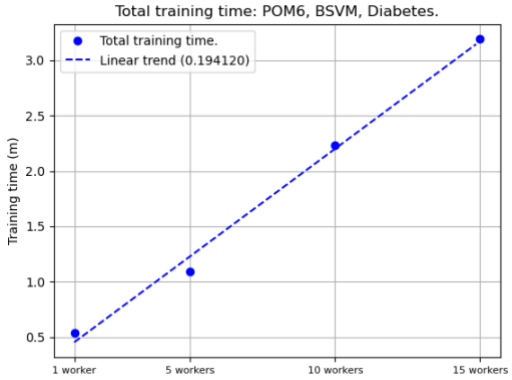


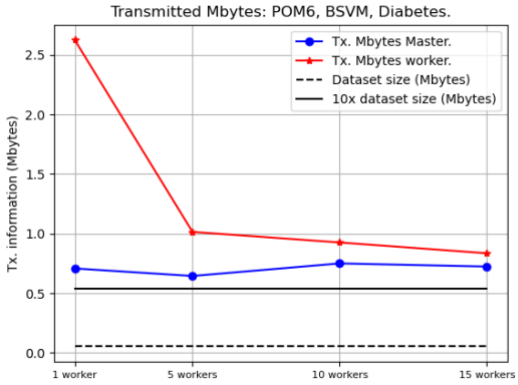


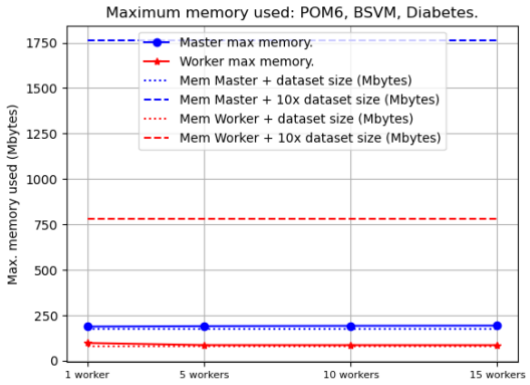
10.3.6 Budget Support Vector Machine (BSVM)

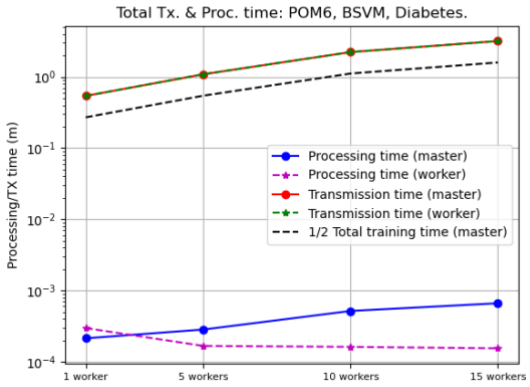
Diabetes



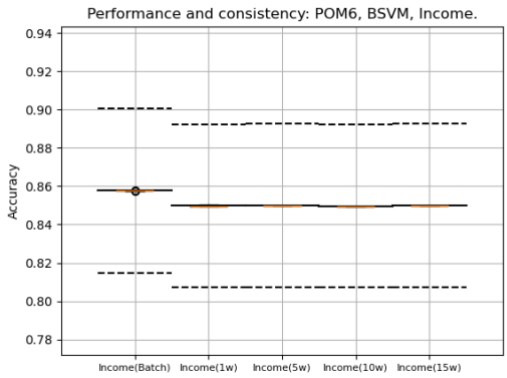


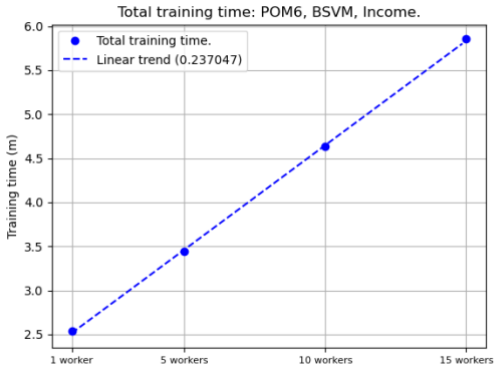


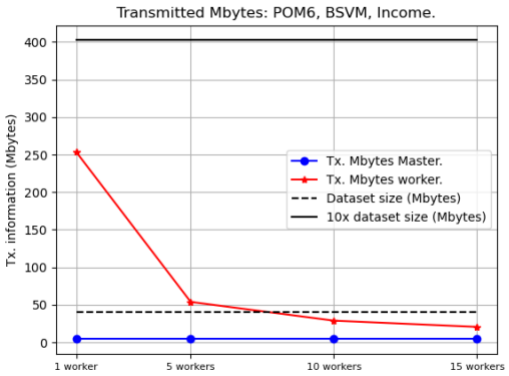


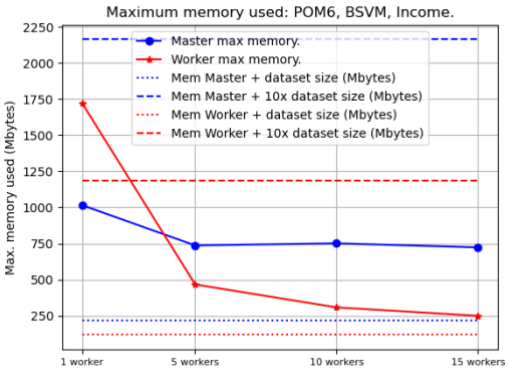


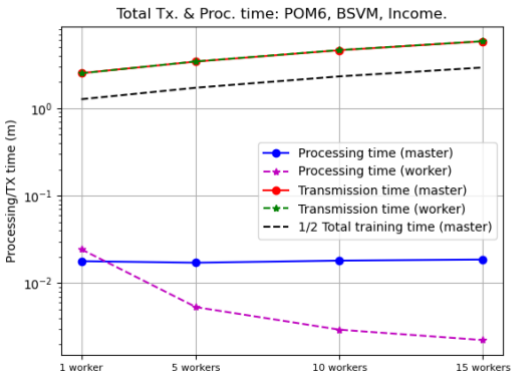
Income


Performance and consistency: POM6, BSVM, Income.
Accuracy
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
Income(Batch)
Income(1w)
Income(5w)
Income(10w)
Income(15w)

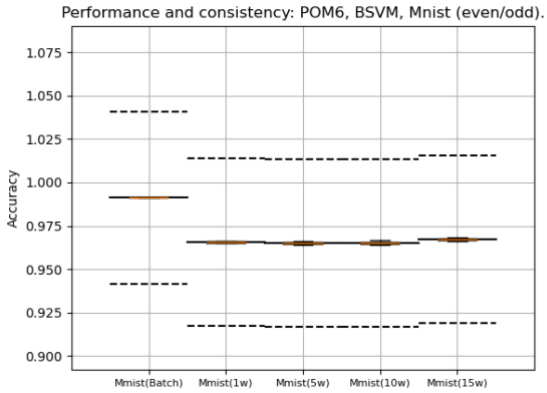

Total training time: POM6, BSVM, Income.
Total training time.
Linear trend (0.237047)
Training time (m)
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
1 worker
5 workers
10 workers
15 workers

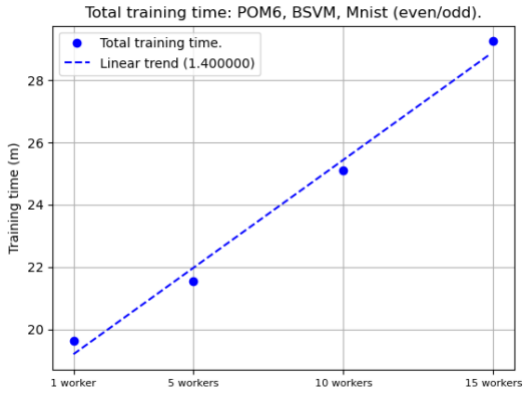

Transmitted Mbytes: POM6, BSVM, Income.
Tx. information (Mbytes)
400
350
300
250
200
150
100
50
0
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

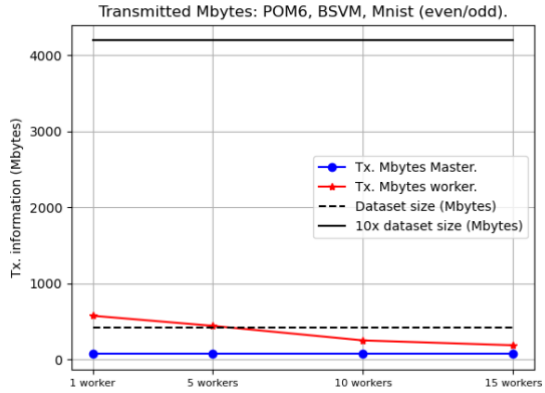

Maximum memory used: POM6, BSVM, Income.
Max. memory used (Mbytes)
2250
2000
1750
1500
1250
1000
750
500
250
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
1 worker
5 workers
10 workers
15 workers

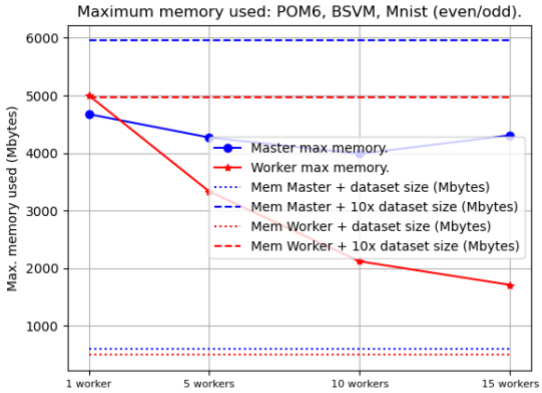

Total Tx. & Proc. time: POM6, BSVM, Income.
Processing/TX time (m)
10^{0}
10^{-1}
10^{-2}
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
1 worker
5 workers
10 workers
15 workers

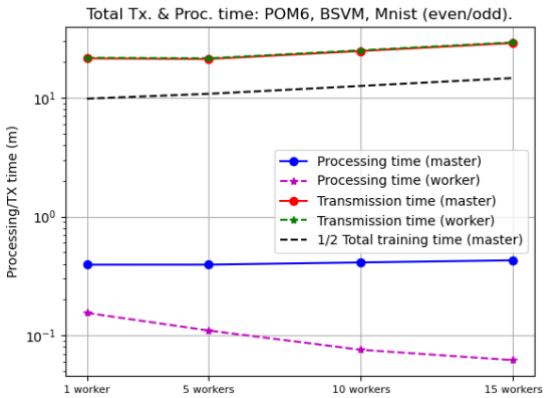
MNIST handwritten digits (even/odd)



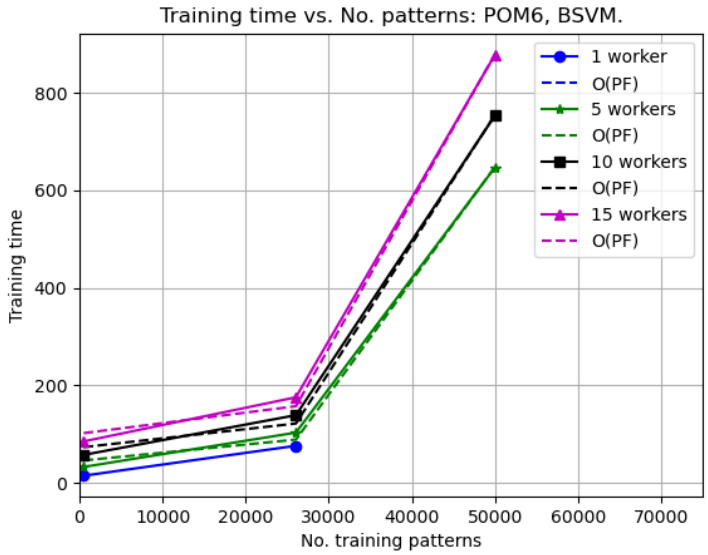


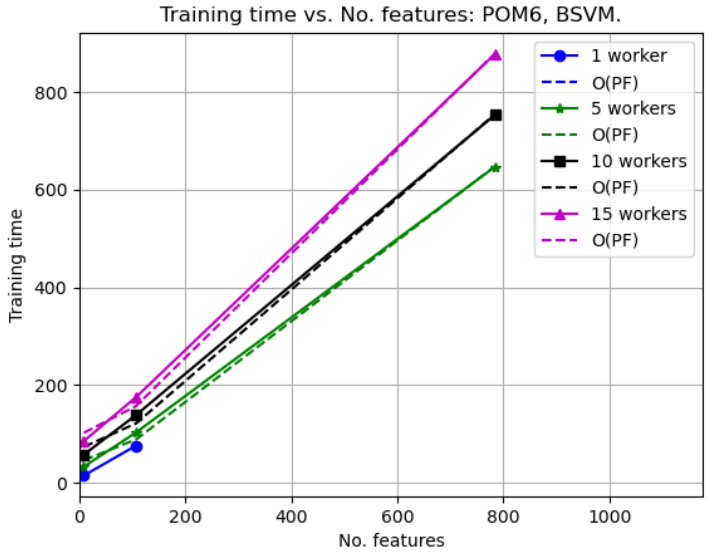






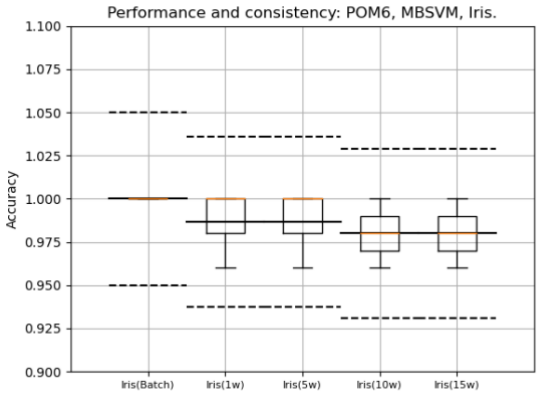
Scalability with respect to No. training patterns, number of features/model size.

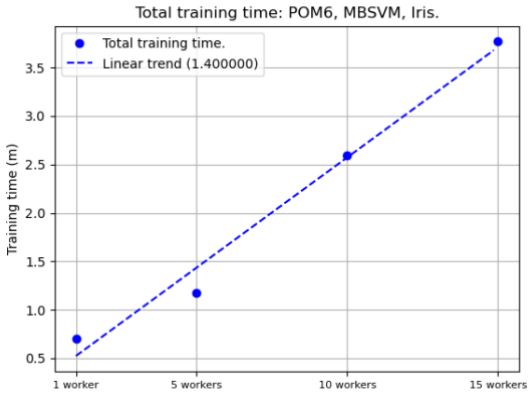


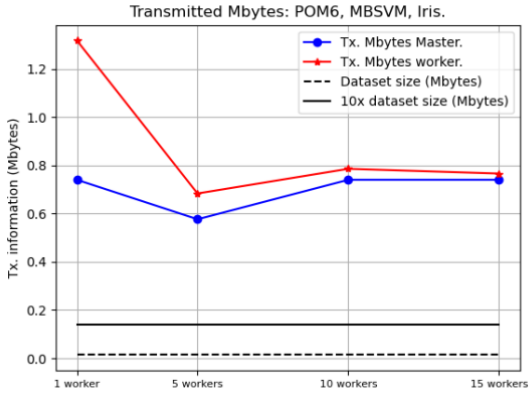


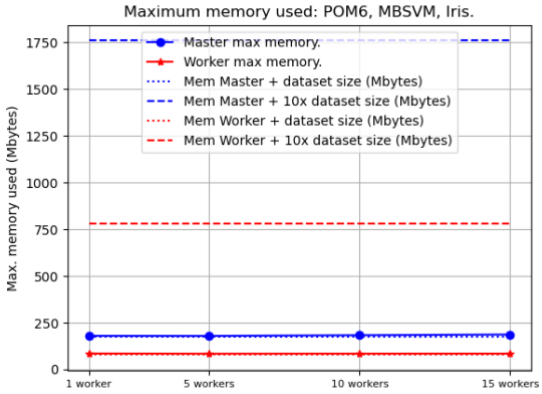
10.3.7 Multiclass Budget Support Vector Machine (MBSVM)

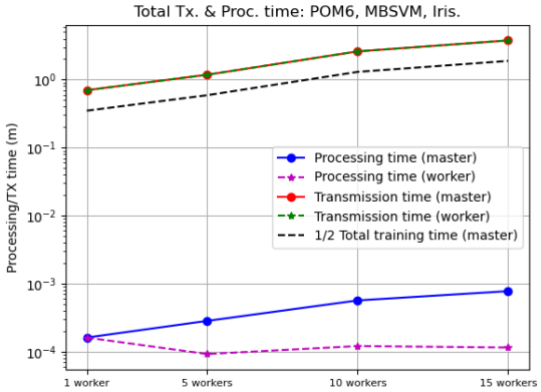
Iris



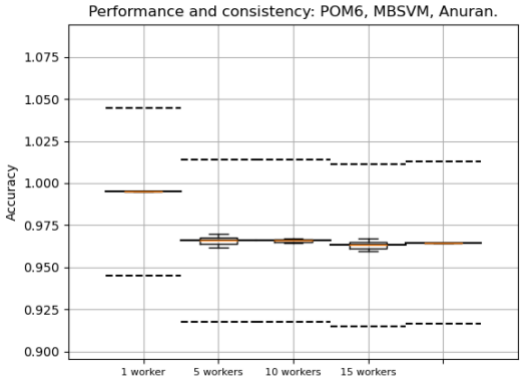


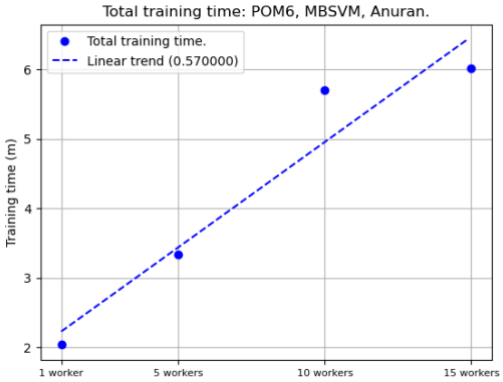


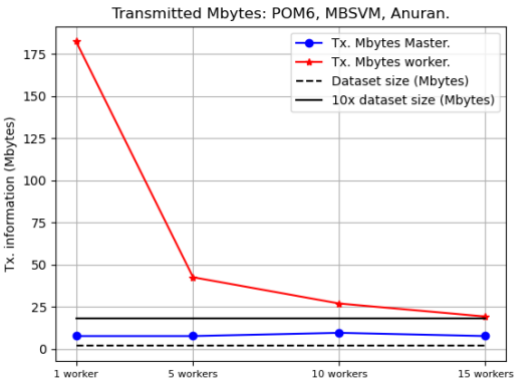


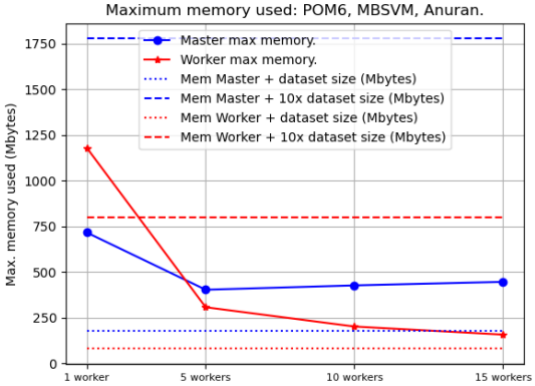


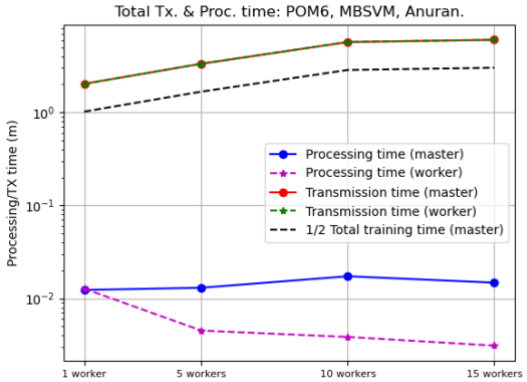
Anuran


Performance and consistency: POM6, MBSVM, Anuran.
Accuracy
1.075
1.050
1.025
1.000
0.975
0.950
0.925
0.900
1 worker
5 workers
10 workers
15 workers

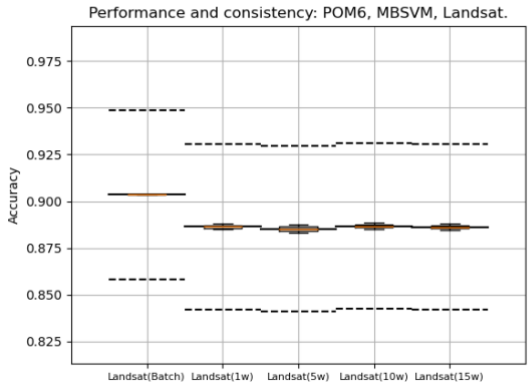

Total training time: POM6, MBSVM, Anuran.
Total training time.
Linear trend (0.570000)
Training time (m)
6
5
4
3
2
1 worker
5 workers
10 workers
15 workers

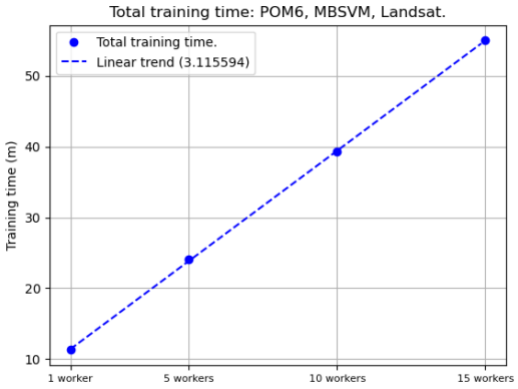

Transmitted Mbytes: POM6, MBSVM, Anuran.
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
Tx. information (Mbytes)
175
150
125
100
75
50
25
0
1 worker
5 workers
10 workers
15 workers

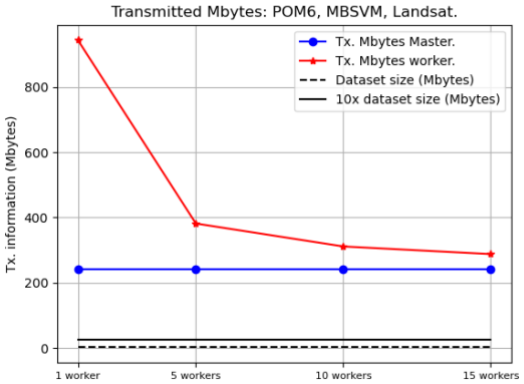

Maximum memory used: POM6, MBSVM, Anuran.
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Max. memory used (Mbytes)
1750
1500
1250
1000
750
500
250
0
1 worker
5 workers
10 workers
15 workers

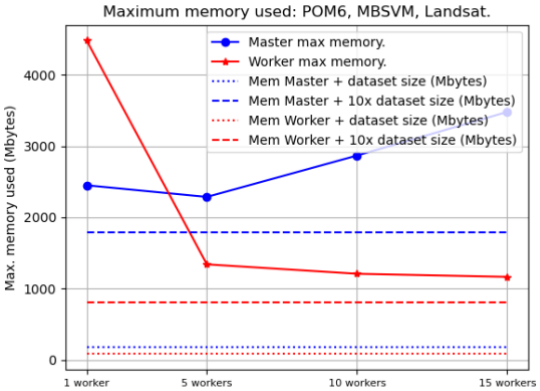

Total Tx. & Proc. time: POM6, MBSVM, Anuran.
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
Processing/TX time (m)
10^{0}
10^{-1}
10^{-2}
1 worker
5 workers
10 workers
15 workers

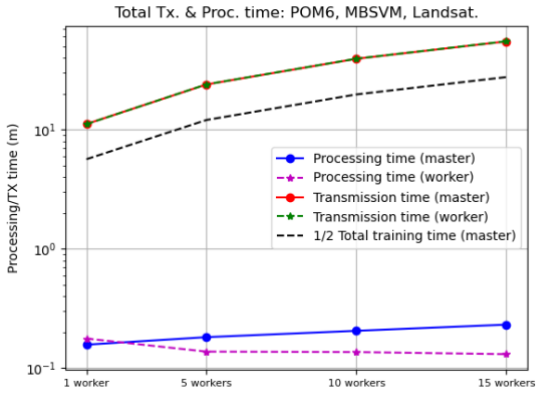
Landsat


Performance and consistency: POM6, MBSVM, Landsat.
Accuracy
0.975
0.950
0.925
0.900
0.875
0.850
0.825
Landsat(Batch)
Landsat(1w)
Landsat(5w)
Landsat(10w)
Landsat(15w)


Total training time: POM6, MBSVM, Landsat.
Total training time.
Linear trend (3.115594)
Training time (m)
50
40
30
20
10
1 worker
5 workers
10 workers
15 workers


Transmitted Mbytes: POM6, MBSVM, Landsat.
Tx. Mbytes Master.
Tx. Mbytes worker.
Dataset size (Mbytes)
10x dataset size (Mbytes)
Tx. information (Mbytes)
800
600
400
200
0
1 worker
5 workers
10 workers
15 workers


Maximum memory used: POM6, MBSVM, Landsat.
Master max memory.
Worker max memory.
Mem Master + dataset size (Mbytes)
Mem Master + 10x dataset size (Mbytes)
Mem Worker + dataset size (Mbytes)
Mem Worker + 10x dataset size (Mbytes)
Max. memory used (Mbytes)
4000
3000
2000
1000
0
1 worker
5 workers
10 workers
15 workers


Total Tx. & Proc. time: POM6, MBSVM, Landsat.
Processing time (master)
Processing time (worker)
Transmission time (master)
Transmission time (worker)
1/2 Total training time (master)
Processing/TX time (m)
10^1
10^0
10^{-1}
1 worker
5 workers
10 workers
15 workers

Scalability with respect to No. training patterns, number of features/model size.