Horizon 2020 Program (2014-2020)

Big data PPP

Research addressing main technology challenges of the data economy

Industrial-Driven Big Data as a Self-Service Solution

D3.3: Batch Processing Analytics module implementation final report[1†]

Abstract: The advances of WP3 from M19 to M30 are described in this deliverable, thus with respect to the Batch Processing Analytics module. The module includes two main parts: a pool of Machine Learning algorithms, and the software stack that supports executing them or any other user analytics tasks.

Contractual Date of Delivery	30/06/2020
Actual Date of Delivery	30/06/2020
Deliverable Security Class	Public
Editor	*Raül Sirvent (BSC)*
Contributors	BSC, FORTH, IBM, UNSPMF
Quality Assurance	*Omer Boehm (IBM)* *Dusan Jakovetic (UNSPMF)* *Kostas Lampropoulos (FORTH)*

[1†] *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780787.*

The *I-BiDaaS* Consortium

Foundation for Research and Technology – Hellas (FORTH)	Coordinator	Greece
Barcelona Supercomputing Center (BSC)	Principal Contractor	Spain
IBM Israel – Science and Technology LTD (IBM)	Principal Contractor	Israel
Centro Ricerche FIAT (FCA/CRF)	Principal Contractor	Italy
Software AG (SAG)	Principal Contractor	Germany
Caixabank S.A. (CAIXA)	Principal Contractor	Spain
University of Manchester (UNIMAN)	Principal Contractor	United Kingdom
Ecole Nationale des Ponts et Chaussees (ENPC)	Principal Contractor	France
ATOS Spain S.A. (ATOS)	Principal Contractor	Spain
Aegis IT Research LTD (AEGIS)	Principal Contractor	United Kingdom
Information Technology for Market Leadership (ITML)	Principal Contractor	Greece
University of Novi Sad Faculty of Sciences (UNSPMF)	Principal Contractor	Serbia
Telefonica Investigation y Desarrollo S.A. (TID)	Principal Contractor	Spain

Document Revisions & Quality Assurance

Internal Reviewers

1. *Omer Boehm (IBM)*
2. *Dusan Jakovetic (UNSPMF)*
3. *Kostas Lampropoulos (FORTH)*

Revisions

Version	Date	By	Overview
0.1	21/4/2020	Raül Sirvent	ToC first version
0.2	28/4/2020	Dusan Jakovetic, Raül Sirvent	ToC updated version, expected lengths and deadlines
0.3	2/6/2020	Miquel Martínez, Yolanda Becerra, Omer Boehm, Dusan Jakovetic, Srdjan Skrbic, Nemanja Milosevic, Lidija Fodor, Dusan Stamenkovic, Aleksandar Armacki, Raül Sirvent	All inputs from partners integrated into a single version
0.4	8/6/2020	Ramon Martin de Pozuelo, Dusan Jakovetic, Giuseppe Danilo Spennacchio, Raül Sirvent	Updates from first round of reviews
0.5	16/6/2020	Omer Boehm, Ioannis Arapakis, Ramon Martín de Pozuelo, Nemanja Milosevic, Dusan Jakovetic, Cesare Cugnasco, Miquel Martínez, Yolanda Becerra, Raül Sirvent	Updates from IBM review, TID, CAIXA, UNSPMF and BSC. Completed visualization section, Intro, Conclusions
0.55	17/6/2020	Raül Sirvent	Cleaned-up version
0.6	19/6/2020	Dusan Jakovetic, Raül Sirvent	Second round of review
1.0	23/6/2020	Raül Sirvent	Final version
1.1	30/6/2020	Raül Sirvent	Ready for submission version

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADMM: Alternating Directions Method of Multipliers
CC: Call Center
CEP: Complex Event Processing
COMPSs: COMP Superscalar
CQL: Cassandra Query Language
CSI: Customer Satisfaction Index
CSP: Constraint Satisfaction Problem
DB: Database
DBSCAN: Density-Based Spatial Clustering of Applications with Noise
KPI: Key Performance Indicator
ML: Machine Learning
MPI: Message Passing Interface
MQTT: Message Queuing Telemetry Transport
MVP: Minimum Viable Product
PCA: Principal Component Analysis
TDF: Test Data Fabrication
t-SNE: t-Distributed Stochastic Neighbour Embedding
UM: Universal Messaging

Executive summary

This report describes the advances in the work package 3 (WP3) of the I-BiDaaS project: Batch processing innovative technologies for rapidly increasing historical data. The work described in this report summarises the tasks developed from month 19th until month 30th.

Work package 3 focuses on the design and development of the batch module of the I-BiDaaS platform. The main components of this module are:

- A pool of machine learning algorithms implemented to provide users with tools to perform their data analysis
- The software stack that supports the execution of those algorithms and all the user analytics.

The work on the Batch module is driven by the requirements of the use cases provided by the industrial partners of I-BiDaaS. For this reason, we include in this report the description of these use cases as well as how they motivate each of the advances of the work package.

Regarding the pool of machine learning algorithms, we report the incorporation of new algorithms needed by the use cases provided by TID (Accurate location prediction with high traffic and visibility and Optimization of placement of telecommunication equipment) and CRF (Production process of Aluminium Die Casting and Maintenance and monitoring of production assets) as well as additional COMPSs-based machine learning implementations, also available at the I-BiDaaS knowledge repository. We also provide a mapping between the algorithms used and implemented in I-BiDaaS and the industrial I-BiDaaS use cases, and we report on additional applications of the developed algorithms on the I-BiDaaS industrial data sets. With respect to the software stack, we have completed the integration of the main components (Dislib, Hecuba and TDF) as well as the implementation of a new tool to guide users with their data analysis. This tool is a visualizer that is integrated with Qbeast to implement an efficient and scalable sampling mechanism.

1 Introduction

WP3, Batch processing innovative technologies for rapidly increasing historical data, is focused on developing the Batch Processing module of the I-BiDaaS platform. The main goal of this work package is to design and to develop the component of the platform that provides users with a pool of machine learning algorithms and a batch analytics software stack. In this report, we describe the work that we have completed from month 18th to month 30th of the project. We also describe briefly the advances that we have made in the development of the use cases of the project that exploit the Batch Processing module of the I-BiDaaS platform.

The work developed in this work package can be divided into two main groups:

- Advances related to the algorithms provided by the I-BiDaaS platform and how the use cases can benefit from these algorithms: these advances contribute to the goals of Task 3.1 ("Advancing data analytics via structured (non) convex optimization"), and Task 3.2 ("Innovative Distributed solvers library implementation with COMPSs programming framework"). In this report, we describe the batch data analytics solutions offered for the respective use cases; additional COMPSs-based machine learning implementations that are made available at the I-BiDaaS knowledge repository[2]; and the mapping between all batch algorithms considered and developed within I-BiDaaS with the I-BiDaaS industrial use cases. Namely, the use cases that are relevant for the Batch Processing module and that have been improved or developed within this reporting period are: Advanced analysis of bank transfer payment in financial terminal (provided by CAIXA), Enhanced control on online Banking (CAIXA), Accurate location prediction with high traffic and visibility (TID), Optimization of placement of telecommunication equipment (TID), Production process of aluminium die casting (CRF) and Maintenance and monitoring of production assets (CRF). Notice that this report describes the use cases focusing on their relations with the algorithms and the software stack of the Batch Module of the I-BiDaaS platform. The descriptions of the use cases from the domain perspective can be found in Deliverable D6.4.
- Advances related to the software components of the I-BiDaaS platform: these advances contribute to the goals of Task 3.3 ("Data management: Hecuba") and Task 3.4 ("Hecuba interaction with TDF"). During this period, we have completed the integration of dislib [4] (a library that contains the set of machine learning algorithms in COMPSs [1] that the I-BiDaaS platform offers to users), with Hecuba[3] (the component of the software stack that implements the access to the data). With this integration, all applications using the algorithms provided by I-BiDaaS will be able to work on data stored using Hecuba without requiring any modification. We also have completed a visualization tool based on Qbeast [3] that has been added to the I-BiDaaS platform and helps users to interpret the results of the data analytics algorithms. This visualization tool benefits from the highly scalable indexing mechanism implemented by Qbeast, which is also integrated with the database of the platform. During this period of the project, we also have completed a software component that feeds TDF[4] (the module of the I-BiDaaS platform that fabricates synthetic realistic workloads) with statistics computed on data stored in the database

[2] Knowledge repository of I-BIDaaS project: https://github.com/ibidaas/knowledge_repository
[3] https://github.com/bsc-dd/hecuba
[4] https://www.ibm.com/il-en/marketplace/infosphere-optim-test-data-fabrication

of the platform. With these statistics, TDF can produce the data generation rules that will fabricate realistic workloads.

1.1 Structure of the report

The rest of this report is organized as follows. Section 2 summarizes the architecture of the I-BiDaaS platform and the role of the Batch Processing module. Section 3 reports the advances related to the algorithms that the Batch Processing module provides to users as well as the relation with the use cases of the project. In this section, we also describe briefly the advances related to each use case. Section 4 describes the work done on each component of the software stack of the Batch Processing module. Finally, Section 5 shows the conclusions of this report.

2 The role of the Batch Processing module in I-BiDaaS architecture

The architecture of the I-BiDaaS platform was solidly defined very early in the project (at M8), and, although some interactions between components may have been modified/enhanced, its essential parts have not changed. The I-BiDaaS architecture and, in particular, the Batch Processing module, have been already described in D3.1 and D3.2. This section intends to be a summary of those previous detailed explanations, thus, for further details, we encourage the reader to refer to the corresponding deliverables. A detailed description of the components included in the Batch Processing module can be found in D3.1, and insights on the desired features can be found in D3.2.

Components

The mission of the Batch Processing module is to deliver batch data analytics and machine learning capabilities. As depicted in Figure 1, the module belongs to the Application Layer in the architecture, and includes two main components:

- The Advanced Machine Learning algorithms: a pool of innovative data analytics and Machine Learning algorithms, using Python and COMPSs as basic programming models.
- The COMPSs [2] programming model: a simple and sequential programming model able to automatically parallelise and distribute the algorithms implemented with it.

Tools

Apart from these main components, the Batch Processing module also takes advantage in some use cases of the following tools:

- The Qbeast [3] indexing system: enables interactive querying to facilitate data exploration and experimentation.
- The dislib library [4]: its objective is to facilitate the execution of big data analytics algorithms in distributed platforms, such as clusters, clouds, and supercomputers. Dislib has been implemented on top of the COMPSs programming model. Some algorithms from the Advanced Machine Learning pool have been contributed to dislib.

Connections

The connections of the Batch Processing module to other I-BiDaaS architecture layers are as follows:

- The distributed large-scale layer: the storage of data and distributed execution is commanded from the Hecuba interface to the Hecuba framework (using a storage such as Cassandra), and from the COMPSs programming model to the COMPSs runtime, respectively.
- The user interface: a web portal with different operation modes (Self-service, Expert, Co-Develop), where the end user can select available algorithms for their data, or implement their own. More in detail: ready-to-use algorithms in Self-service mode (where inputs and parameters can be tuned) or code templates in Co-Develop mode (to develop modifications of existing algorithms).

- The data ingestion and integration subsystem: the data (either real or synthetically generated) is ingested in the system through the Universal Messaging, and stored in the Hecuba back-end database (Cassandra). Also, there is a strong interaction between the Test Data Fabrication tool (TDF) (provided in this layer) with the Batch Processing module, since some analytics and meta-data are fed back from the Batch Processing module to the TDF for automatic data fabrication.
- The streaming analytics module: the real-time processing module can feed the Batch Processing module with inputs that enable periodic refinements of models used in machine learning methods. A section with a more detailed description is included in Deliverable D4.3.

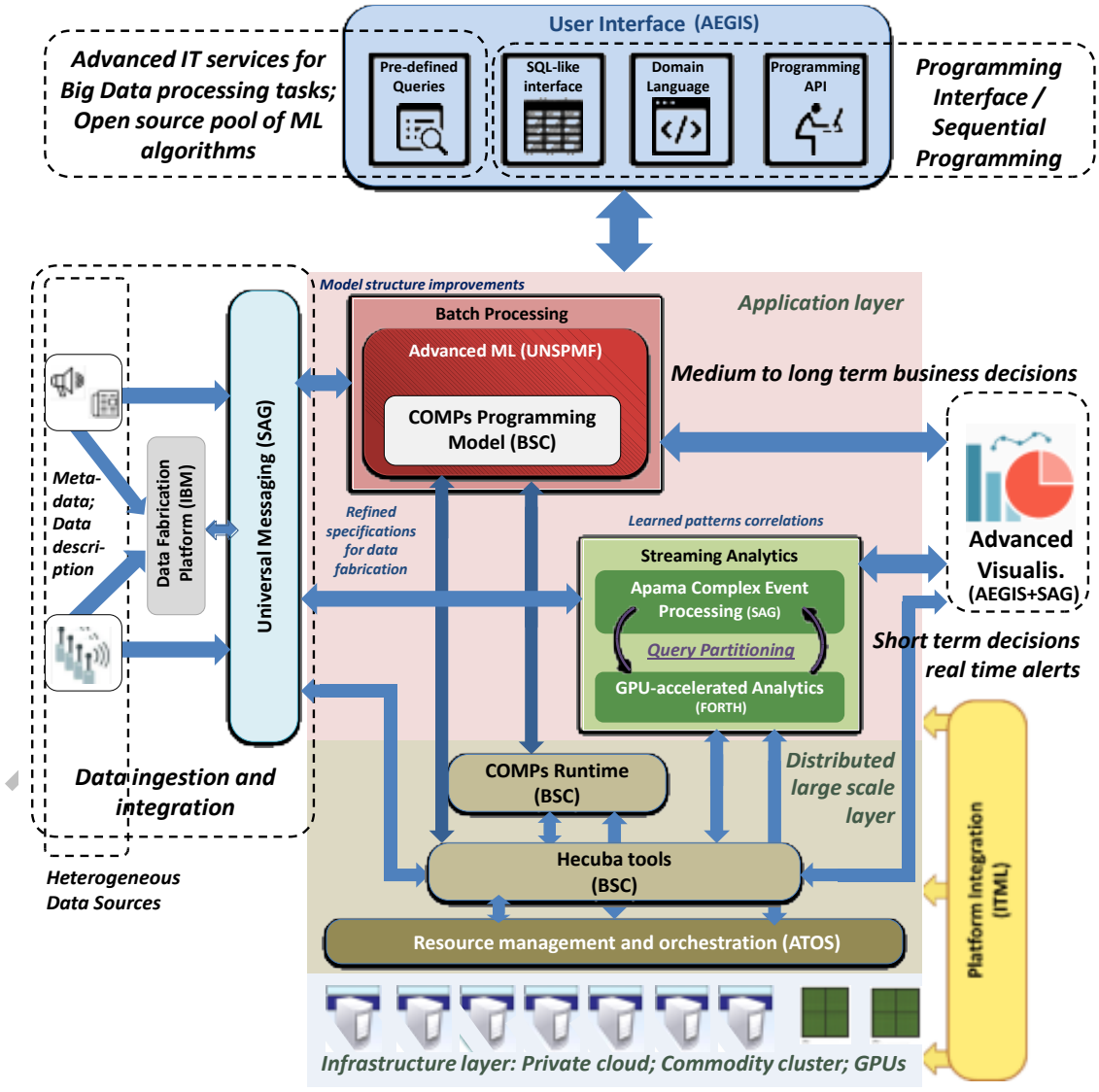

Figure 1: The I-BiDaaS platform architecture

Feedback from analytics results to problem modelling

Additionally, we have included in the module a way to provide feedback from analytics results to problem modelling, through visualization with Qbeast. The user is able to represent their results in a 3D visualization interface that helps to correctly interpret the results of any analysis task previously run. Once results are interpreted, users can go back to the originally provided model and modify it according to the conclusions extracted from the visual analysis.

Preparation and pre-processing

We have also included basic data preparation capabilities, such as:

1. Data cleaning
    - Removal of null/NaN (not a number) values from the dataset;
    - Column removal/selection – not all columns are relevant so we can choose to remove certain columns;
2. Data transformation
    - Categorical to numerical data – useful for ML algorithms that can work only with numerical data;
    - Normalization – possible normalization to, e.g., [0,1] values by row or by column in the dataset, again useful for ML tasks;
    - Column type transformations – e.g., Date to Day, Month, Year, and similar.

More precisely, we have implemented some source code in the Batch Processing module that is able to do this preparation and pre-processing of data automatically.

Packaging

The packaging of the Batch Processing module has not changed. As stated in previous deliverables, the module consists of a Python package with Python modules to run data analysis tasks (which also serve as a guide for new algorithms developed). Its invocation is through the I-BiDaaS end user interface (the web portal), with appropriate parameters. For visualization purposes, the analysis results are written back to the Cassandra back-end in the platform, under user's request, which makes it more flexible since different visualizations could be submitted to the data in parallel.

3 Use Cases that Leverage the Batch Processing Module

In this Section, we report the new developments conducted from WP3 with respect to the I-BiDaaS project use cases from our industrial partners. Although some of the cases were already presented in previous deliverables (D3.1 and D3.2), here we include the advances with respect to what was previously reported. More precisely, it includes:

- Advanced analysis of bank transfer payment in financial terminal (CAIXA): look for possible fraud in bank transfers, now using Hecuba and dislib.
- Enhanced control on online banking (CAIXA): control the correct access to the operative.
- Optimization of placement of telecommunication equipment (TID): analyse KPIs from Mobile Network Operators to understand their performance, using algorithms Random Forest [18], XGBoost [8] and CatBoost [9].
- Accurate location prediction with high traffic and visibility (TID): track the number of users on the network and predict the future number of users on the various parts of the network, using a time series analysis.
- Production process of aluminium die-casting (CRF): analyse annotated thermal images of the engine casting process, together with sensor data for the engines, using neural networks.
- Maintenance and monitoring of production assets (CRF): use the data from the production process to prevent faults before they happen using an outlier detection analysis, so sensor and machine maintenance can be done at the right time.

The intention of this Section was to provide insights on how the use cases have been implemented from the Batch Processing module perspective. Further analysis of the use cases from the domain perspective will be provided in Deliverable D6.4.

3.1 Advanced Analysis of Bank Transfer Payment in Financial Terminal

This CAIXA use case is focused on advanced analysis of bank transfers executed by employees on financial terminals, to detect possible fraud, or any other potential anomalies that differ from the standard working procedure. The used dataset is composed of different attributes which record the different steps that the employee performs and also other important data like: the account, client or amount of money transferred. All the data is encrypted using the Dice Coefficient [17], which codifies the data without losing important information. CAIXA has implemented different types of encryption, as it will be described in more detail in Deliverable 6.4: Experiments implementation, mainly Format-preserving encryption, Order-preserving encryption and Privacy-preserving encryption of text using Bloom filters.

In this new deliverable, all data processing techniques, like the K-means[5], PCA[6] (Principal Component Analysis) and DBSCAN[7] (Density-Based Spatial Clustering of Applications with Noise) have been performed using the dislib library. Also, the data structure used by dislib has been modified in order to be stored on a Cassandra Database using the Hecuba library. These are the main novelties with respect to what was reported previously in Deliverable D3.2.

[5] https://en.wikipedia.org/wiki/K-means_clustering
[6] https://en.wikipedia.org/wiki/Principal_component_analysis
[7] https://en.wikipedia.org/wiki/DBSCAN

The main goal is to transform data in order to be able to apply an unsupervised clustering to the dataset. The received dataset cannot be processed using the data transformations techniques from dislib, and for this reason, a pre-processing has been applied:

- First of all, the attributes which only contained the same value for all the registers have been deleted, as they do not give any relevant information.
- Also, all nulls and blank registers have been transformed into 0 values.
- Next, for those categorical attributes, we have applied a one-hot encoding[8], which has added many columns (one per each unique category value) to the dataset. With this encoding technique, all possible values keep an identical weighting.
- Finally, string attributes composed by multiple values, delimited by an especial character, had been separated and also encoded using the one-hot encoding technique.

Due to the encoding transformations, the amount of attributes has increased considerably from 89 to 501. This large amount of attributes made difficult to perform a K-means, and for this reason, it was decided to apply a PCA transformation to reduce the amount of dimensions down to only 3, to be able to represent it graphically.

Before applying the PCA transformation, and due to the differences in the magnitude of the attributes, we have standardized the data, using the scikit-learn [7] method *StandardScaler*.

```
bn, bm = 250, 167
dataset = ds.array(x=x, block_size=(bn, bm))

pca = PCA(n_components=3)
dataset_x_pca = pca.fit_transform(dataset)

dbscan = DBSCAN(eps=0.9, min_samples=50).fit(dataset_x_pca)
n_clusters=dbscan.n_clusters

kmeans2 = KMeans(n_clusters=n_clusters, random_state=500)
```

Figure 2: Code to reduce the data dimensionality

Finally, we have executed two different clustering algorithms: DBSCAN and K-means. As K-means requires the desired number of clusters as an input parameter, we have executed first DBSCAN and we have used the obtained number of clusters as the input parameter of K-means. Figure 3 shows the graphical representation of the clusters generated by DBSCAN in a two-dimensional space. Also, in the figure on the right, the third dimension of the PCA is shown as the Z axis. The result of K-means can be examined in Figure 4, where the center of each cluster is indicated with a red dot.

[8] https://en.wikipedia.org/wiki/One-hot

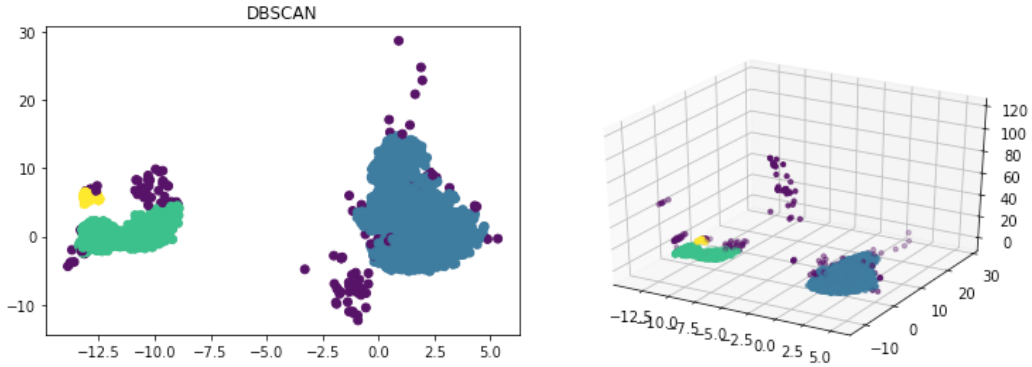

Figure 3: DBSCAN representation 2D and 3D

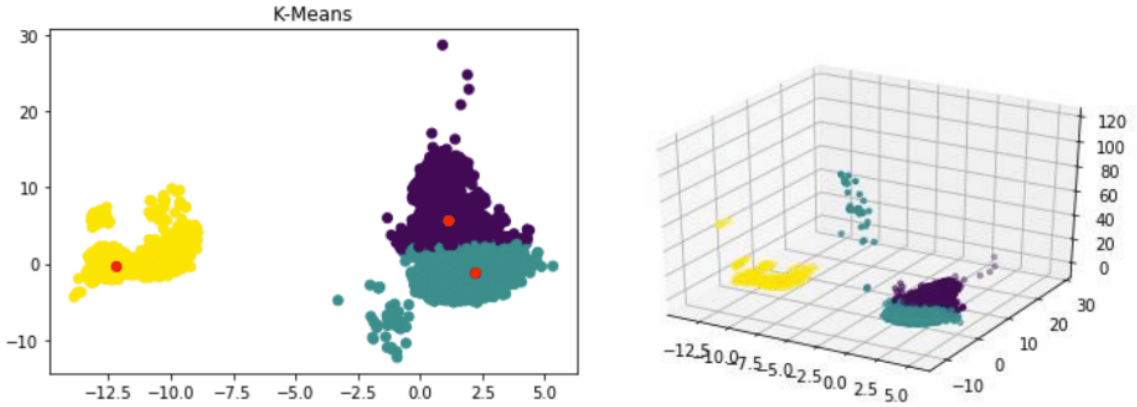

Figure 4: K-means representation 2D and 3D

In Figure 5, a more detailed vision of these differences is shown. We can observe that some values are far away from the cluster center and thus, they are potential anomalies in the data.

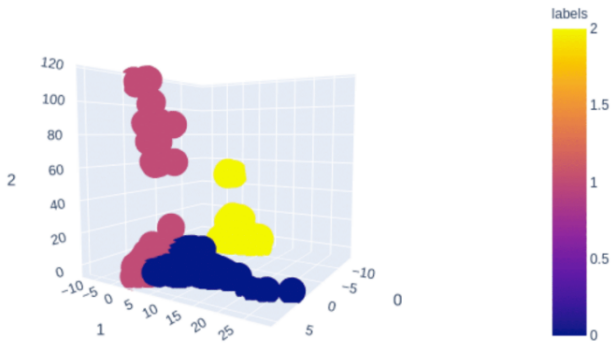

Figure 5: 3D K-means representation from a different point of view

The PCA reduced the attributes from 501 to 3, thus, it is difficult to understand which is the correlation between the resultant three dimensions and the 501 original attributes. In Figure 6, we have printed the mentioned correlation, so we can understand how the different attributes have influenced in each PCA dimension. We only show the first 84 because they are the most interesting with respect to the 3rd dimension, as seen in the Figure. We can appreciate that this 3rd dimension is heavily influenced by attributes from 64 to 82.

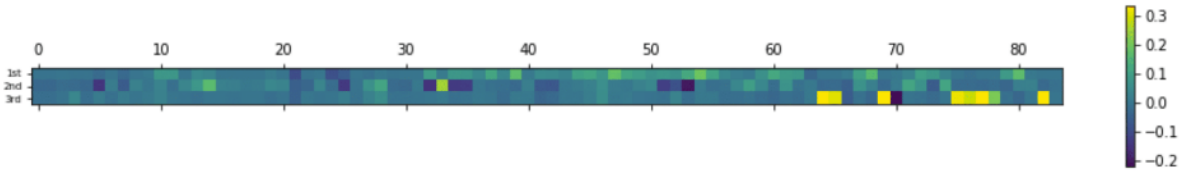

Figure 6: Heat-plot of the 84s most relevant of the first 501 original attributes

3.2 Enhanced Control on Online Banking

This use case of CAIXA focuses on analysing the mobile-to-mobile bank transfers ordered through online banking (web and application). It targets on assessing that the controls established to authenticate the user are applied adequately (e.g. second factor authentication) according to PSD2 regulation and depending on the context of the bank transfer.

For conducting this analysis, we are faced with the need for clustering on a categorical database, so that the vast majority of known algorithms lost efficacy. Initially, an attempt was made to apply a K-means by transforming the variable categories into columns (1, 0), a transformation known as one-hot encoding (as described in the previous Section), but the results were not satisfactory. The *k-modes* library[9], that includes algorithms to apply clustering on categorical data, was a more convenient match for this problem.

The *k-modes* algorithm [19] is similar to K-means, but with some modification that allows us to work with categorical variables. The *k-modes* algorithm uses a simple matching dissimilarity measure to deal with categorical objects, replaces the means of clusters with modes, and uses a frequency-based method to update modes in the clustering process to minimize the clustering cost function.

Once the algorithm has been decided, we must calculate the optimal number of clusters for our use case. For this, the method known as the *elbow method* [20] is applied, which allows us to locate the optimal cluster as follows; we first define:

- Distortion: it is calculated as the average of the squared distances from the cluster centers of the respective clusters.
- Inertia: it is the sum of squared distances of samples to their closest cluster center.

Then, we iterate the values of k from 1 to 10, calculating the values of distortions for each value of k, and calculating the distortion and inertia for each value of k in the given range. The idea is to select the number of clusters that minimize inertia (separation between the components of the same cluster).

[9] https://pypi.org/project/kmodes/

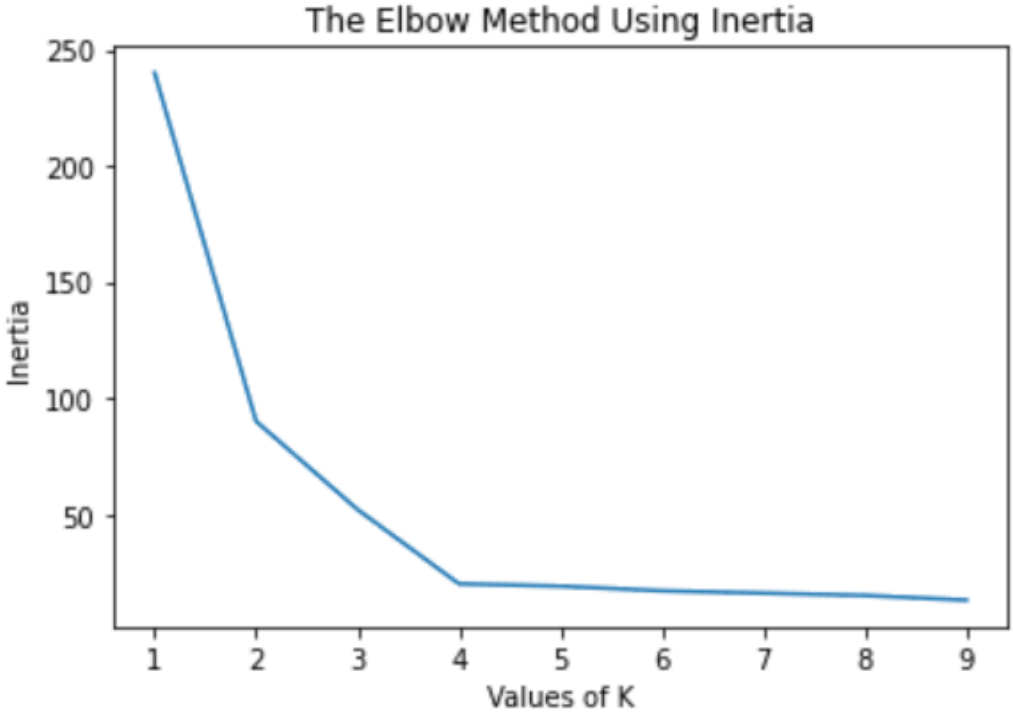

Figure 7: Representation of inertia using different values of *k* for the Elbow Method

To determine the optimal number of clusters, we have to select the value of k at the "elbow", i.e., the point after which the distortion / inertia starts decreasing in a linear fashion. Thus, for the given data, we conclude that the optimal number of clusters is 4. Once we know the optimal number of clusters, we apply *k-modes* with $k = 4$ and analyze the results obtained.

3.3 Optimization of Placement of Telecommunication Equipment

We briefly describe this use case provided by TID. Mobile Network Operators (MNOs) continuously collect various Key Performance Indicators (KPIs), which are one of the key information for MNOs to understand large-scale cellular network performance. Those KPIs can be grouped into the following categories: coverage, accessibility, retainability, mobility, availability, and congestion. In order to manage a network, operators need these KPIs to find out which cell sites underperform at any given time. By predicting which cell sites will become the next "hot spot" (a cell site with high traffic and congestion), operators can distribute telecommunication equipment in the optimal fashion (more equipment for hot spots).

The dataset contains anonymized, encrypted data for 24 hours, for over 40K antennas in an anonymized city. Each sample contains 16 antenna KPIs, whose names were also anonymized. We carry out the modeling approach which predicts based on the available KPIs which cell sites will become a hot spot or not; in other words, a supervised binary classification is carried out.

The data was split into a training set and a testing set in the following fashion. We group the antennas by their ID, so that each antenna can be either in the train set or in the test set. Then, we use 80% of the groups as a train set, and the rest as a test set. When doing so, we sought the splitting that yields as similar percentage of the positive class as possible in the train and test sets.

We used the following classification algorithms: Random Forest (implemented both in Scikit Learn [7] and in PyCOMPSs [1]), and two gradient boosting algorithms, namely XGBoost [8] and CatBoost [9]. Table 1 shows the achieved results.

Table 1: Performance of binary classification algorithms for the antenna KPIs use case

Algorithm	Accuracy	Precision	Recall
XGBoost	0.999	0.998	0.998
CatBoost	0.999	0.977	0.961
Scikit-Learn Random Forest	0.999	0.998	0.975
PyCOMPSs Random Forest	0.999	0.995	0.997

3.4 Accurate Location Prediction with High Traffic and Visibility: Time series analysis

The *'Accurate location prediction with high traffic and visibility'* use case with the real data provided by TID is concerned with tracking the number of users on the network and also predicting the movements of the users, by predicting the future number of users on the various parts of network. The dataset provided by TID is a large anonymised and encrypted collection, with more than 120,000 samples, where one sample represents a time series for a specific antenna. The time series duration was for one month, and the points in the series were divided into four-hour periods (yielding 186 points in total). Even though the dataset is of high quality and volume, one problem is that there is a very high degree of sparsity or missing data. The dataset was provided on a TID's hardware in JSON format, where every antenna is a JSON object containing its time series for the period. The values in the time series represent the number of users using the antenna at some point in time, so the values are strictly positive.

Our approach when dealing with this dataset is to train one time series model per antenna or per time series. We use the state-of-the-art time series model tool Prophet [10] developed by Facebook AI. Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality. Prophet is available for both Python and R programming languages, and we used the Python version.

In our implementation, the data is first loaded from the raw JSON file and pre-processed with pandas library[10]. No special pre-processing is done apart from joining the data into one large table where columns represent time steps and the rows represent specific antennas. Every time series is then split into training and validation sets. We split the data so that the last five points in the time series are used just for validation purposes.

Subsequently, the time series are fitted and tested with the respective Prophet models. We forecast the next five points and compare the values with the known true values. We calculated the mean absolute error between the forecasted values and the real values.

Due to the large number of time series models to train and the significant computing power provided by TID, we parallelize the training process. We used joblib[11], a Python library for both thread and process parallelization. We used process-based parallelization where every CPU core was assigned a process that trained a time series model for one antenna. Overall, to

[10] https://pandas.pydata.org/
[11] https://github.com/joblib/joblib/

train around 120,000 time series, it took three hours on the TID server, which is an impressively short time.

The last step in training was to select the best models (based on the mean average error), as agreed with the TID team, so that they can be used for visualization and prediction purposes. When selecting the best 1000 models, we obtained a baseline metric of the average mean absolute error that equals to 1.2565. The baseline model accuracy is good, although we suspect that a more accurate model can be made with more data pre-processing (e.g., through the imputation of missing values). Figure 8 shows a visual example of one time series obtained by the Prophet model for one of the antennas.

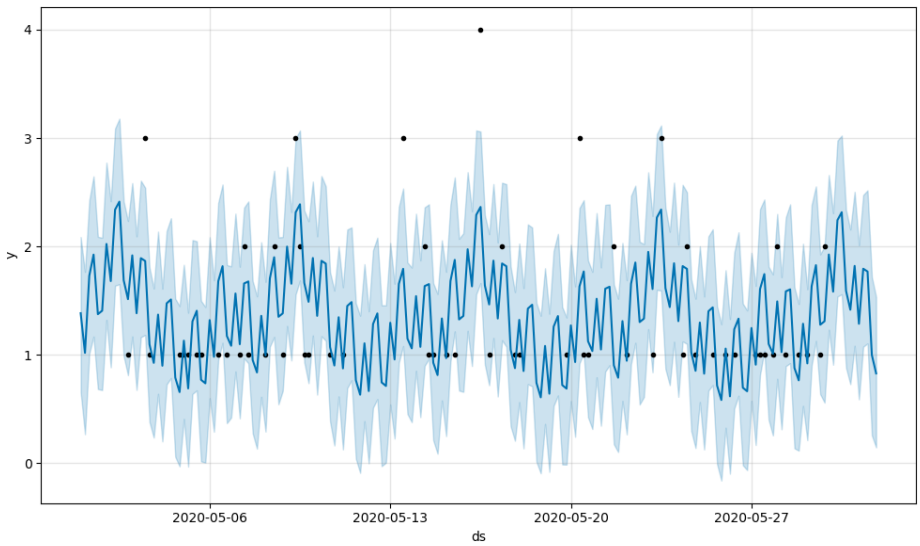

Figure 8: Illustration of a time series obtained by the Prophet model for one of the antennas within the TID mobility data.

3.5 Production Process of Aluminium Die-Casting: Thermal images data analysis

The CRF's *'Production Process of Aluminium Die-Casting'* use case studies one of the production processes of the engine blocks. During the casting process, molten aluminium is injected into a die cavity where it solidifies quickly. The die cavity is mounted securely in a machine and equipped with many sensors and thermal cameras. The process is complex and it is important to not only carefully design parameters and temperatures but also to control them because they have a direct impact on the quality of the casting. The goal of the use case is to predict whether an engine block will be produced correctly, i.e., without anomalies, during the casting process. The correct prediction during the casting process would avoid subsequent reworks and scraps, which would lead to financial savings for the manufacturer. We report here the advances from M18 to M30 in the analysis for the use case. The main improvements are due to the availability of an additional, thermal images, data set.

In more detail, the data provided by CRF for the analyses consist of a collection of casting process parameters monitored by sensors (sensor data), such as piston speed in the first and second phase, intensification pressures and others. In addition to the sensors data, CRF also provided a large dataset of annotated thermal images of the engine block casting process, under a hypothesis that there is a correlation among sensor data, thermal data and the outcome of the process.

Formally, the considered use case corresponds to a supervised learning M-ary classification problem [22]. To solve the problem, several models have been developed. We started from sensor-only based models, as reported in D3.2. In this deliverable, we describe the newly developed models that utilize both sensor and thermal images data.

First, we describe our neural network model based on thermal images only.

The neural network model based on thermal images data

Several data preprocessing steps were carried out. First, images were cropped to remove non-useful information and were re-sized to better fit the models. We also converted the images into grayscale that turned out to help the model to converge faster. Finally, we transform the image to a tensor of appropriate dimension.

We discovered through experimentation that simple shallow neural network models do not work satisfactorily on the data set, hence we consider more complex and deeper models. We used the ResNet18 network, see for example [11] which has been modified to accept our image dimensions and also to have the appropriate number and format for the output classes. We tried both the pretrained model (trained on the ImageNet dataset [23]) and the normal non-trained model and both behaved similarly in terms of accuracy. For training and validation, we used an 80/20 dataset split. Training is done for 300 epochs with a learning rate of 1e-3. The loss function used is the cross-entropy, commonly used for image classification. The entire implementation is written with pytorch [12]. The achieved results are the following: training average loss: 0.0618; accuracy: 1053/1195 (88%); validation average loss: 0.5642; accuracy: 169/299 (57%).

The joint neural network model based on sensors and thermal images data

The second model that we consider is a joint model based on thermal images and sensor data. The joint model uses an even deeper neural network for thermal images data, combined with a fully-connected network for sensor data.

The sensor data cleaning and preparation is carried out in the same way as with our previous random forest model reported in D3.2 and consists of removal of NaN values, column-wise normalization and time-stamp conversion. We also introduce standard image channel normalization, which helped in getting faster convergence.

We adopt here the DenseNet201 neural network model, see for example [11]. Due to the fact that the analyzed data set is imbalanced, we developed two different models: 1) the model that uses the full imbalanced dataset; and 2) the model that uses a random under-sampling technique to produce a balanced dataset.

The architectures of both models are presented in Figure 9.

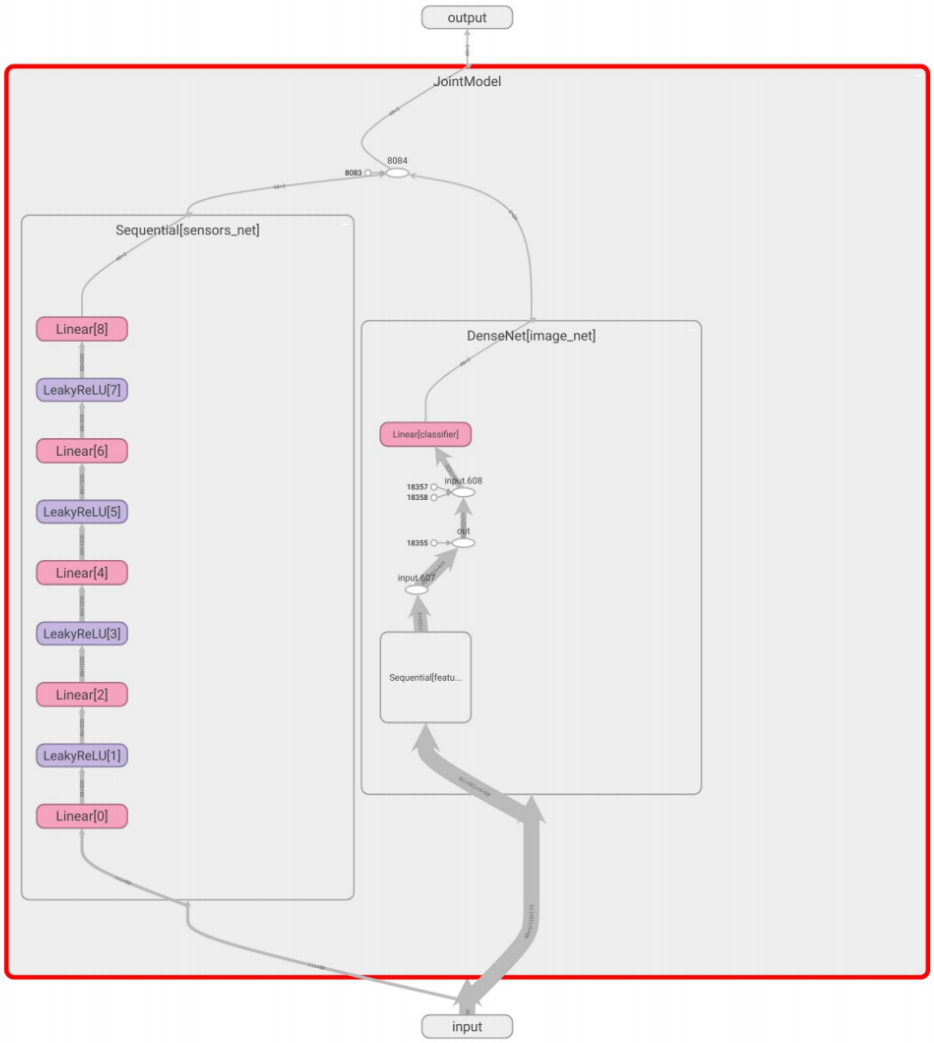

Figure 9: Architecture of the joint neural network models for the CRF aluminium casting use case

As shown in Figure 9, the DenseNet201 model was paired with a 7-layer fully connected neural network which is used for handling sensor data (without quality assurance parameters). Both models show good accuracy on training and validation datasets, as shown in Figure 10.

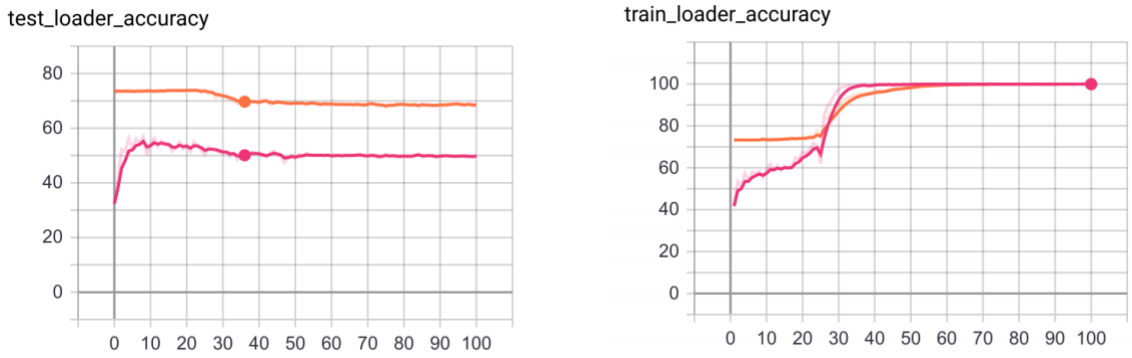

Figure 10: Training and testing accuracy for the two joint neural network models. Orange line: the model trained on full imbalanced data; Pink line: the model trained on sub-sampled balanced data.

As expected, the model trained with the full dataset (orange line) outperforms the model trained on the balanced dataset (pink line). It is important to note that both models converge fully and quickly on the training dataset. We see a 73.54% validation accuracy on the first model and 54.06% validation accuracy on the second model. Figure 11 shows the precision and recall values for the different predicted classes and the two models. We can see that the second model (with sub-sampled data) performs better over certain output classes.

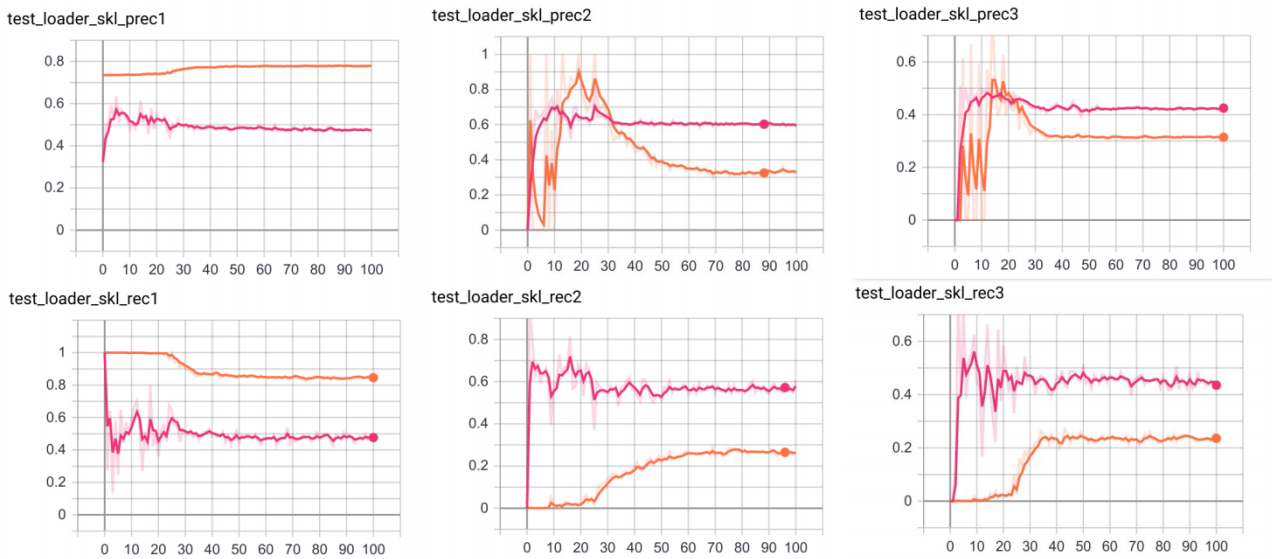

Figure 11: Precision and recall values for different classes accuracy for the two joint neural network models. Orange line: the model trained on full imbalanced data; Pink line: the model trained on sub-sampled balanced data.

3.6 Maintenance and Monitoring of Production Assets

The *'Maintenance and monitoring of production assets'* use case by CRF is concerned with using the data from the production process to prevent faults before they happen by doing sensor and machine maintenance at the right time. Different sensors are installed on the production line, acquiring different data types (e.g., sensors mounted on different machines, accelerometers mounted on linear stages). The sensors are connected to the control units on the line, while the control units are connected to a server that manages the data. The data consists of two different sets, the SCADA and the MES data. The SCADA data contains production, process and control parameters of the daily vehicle production. The MES data contains details of the type of the vehicle being produced. More details on the datasets can be found at Deliverable D2.1: Data assets and formats.

We carry out an outlier detection analysis on each sensor separately. Subsequently, we compare the time stamps of the detected anomalous measurements across the results for different sensors. The analysis did not require any parallelization, everything was done on a single GPU (NVIDIA RTX2070).

The outlier detection analysis was performed using a modified interquartile range[12] (IQR) test. The outlier detection was performed on a subset of 16 sensors. It was established that almost all sensors have around 500 different days with anomalous measurements, and almost

[12] https://en.wikipedia.org/wiki/Interquartile_range

all of them were common to different sensors (more than 90% on average). Two more similar tests were performed, where Q1 was calculated as the 10^{th} (5^{th}) quantile and Q3 was calculated as 90^{th} (95^{th}) quantile. The most informative results were obtained for Q1 = 5^{th}, and Q3 = 95^{th} percentile. The results read as follows:

- Sensors with the IDs 46, 47, 64, 68, 84, 92, 101, and 109 had between 485 and 512 days with anomalous measurements. All these sensors had more than 93.5% of shared days that correspond to the anomalous measurements;
- Sensor with the ID 113 had 5 days with anomalous measurements;
- Sensors with the IDs 19, 130, and 146 had two days with anomalous measurements. Sensor 19 and 49 shared both days;
- Sensors with the IDs 120 and 137 had one day with anomalous measurements. They shared one date with the sensor 146;
- Sensor 141 had no days with anomalous measurements.

One possible reason for the difference between the number of "anomalous" days for different sensors could stem from the fluctuation of sensors measurements since they are used for different types of vehicles.

4 Advances in the Batch Processing module software architecture

This section reports the advances in the development of the components of the Batch Processing module of the I-BiDaaS platform. First, we describe the advances related to the Machine Learning components and, second, we describe the advances in the technological components of the platform.

4.1 Machine Learning advances

4.1.1 Algorithms

We describe here the advances in the algorithm development that were carried out after month 18 of the project. Specifically, we present several additional algorithmic implementations, present the mapping of how the I-BiDaaS developed algorithms and the I-BiDaaS industrial use cases are linked, and we report on the novel testing that was carried out for the previously implemented algorithms on various I-BiDaaS industrial data sets. In summary, the main contributions after month 18 of the project are three-fold: 1) novel algorithmic implementations; 2) a comprehensive mapping of the algorithms considered in I-BiDaaS to the I-BiDaaS industrial use cases; and 3) additional testing and validation of previously developed algorithms on the I-BiDaaS industrial use cases. We refer to the I-BiDaaS public knowledge repository for the source code of the algorithmic implementations[13].

4.1.1.1 Algorithmic implementations

We describe several additional implementations of machine learning algorithms, namely 1) the ADMM-based K-means, 2) the ADMM-based logistic regression, 3) the stochastic gradient-based logistic regression, 4) the ADMM-based ridge and elastic net regression and 5) differentially private K-means and random forest algorithms. The first four algorithms are based on structured (non)convex optimization, mainly ADMM, reported in D3.1 and D3.2. For the differentially private K-means, we provide more details of implementation here, while the differentially private random forest implementation is similar to the differentially private K-means.

ADMM-based K-means

As part of broadening the pool of algorithms offered by the platform, a convex version of the renowned K-means clustering algorithm is implemented. The implementation is based on [13] and relies on the ADMM framework[13], [5], reported in D3.2, to find the optimal cluster centroids. Unlike the standard K-means algorithm, which requires the user to provide the desired number of clusters, the convex version of K-means algorithmically determines the optimal number of clusters. The framework also contains a regularization parameter that determines whether a smaller or a larger number of clusters is found. This allows for some flexibility and prior knowledge to be introduced by the user, while still keeping the most difficult and essential part of the K-means – the proper number of clusters – automated. An initial version of the algorithm is implemented. The algorithm implementation is Python-based; it relies on CVXPY [21] to solve the intermediate, ADMM-based problems for updating the solution variables. The K-means algorithm is run the same as previously

[13] https://github.com/ibidaas/knowledge_repository/blob/master/tools_technologies/sources/batch_processing/unspmf/

implemented ADMM-based algorithms (see D3.2): the ADMM-related parameters can be set by the user, while default values are offered as well, while the only parameter to be tuned is the K-means-related regularization parameter, impacting the number of clusters found.

ADMM-based Logistic Regression

We include a logistic regression algorithm for supervised binary classification, based on the ADMM framework[14], [5]. An in-depth description of the ADMM framework and its advantages is offered in D3.2. Logistic Regression is a (binary) classification model known for its simplicity and effective performance. As detailed below, the ADMM-based logistic regression methods have been applied to the CRF production process of aluminium die-casting use case. The current implementation offers a fully distributed, ADMM-based Logistic Regression algorithm. The implementation relies on PyCOMPSs for parallelization, as well as CVXPY for solving the ADMM-based intermediate optimization problems used for updating the solution variables. The underlying Logistic Regression problem uses a L2 regularization, therefore preserving convexity, but allowing for a tunable parameter that controls the norm/size of the obtained solution.

Stochastic gradient-based Logistic Regression

The stochastic gradient-based Logistic Regression algorithm offers another variation of the Logistic Regression model. The implementation is based on [6]. The algorithm is based on a distributed gradient descent scheme, which is known to be simple and effective in practice. A distinctive feature of this implementation represents the variable number of workers that are active at each iteration. Each worker is active with a certain probability. The probability of activity increases slightly with each iteration. A similar concept is prevalent in federated learning, where a subset of workers is active at each iteration. Such a scheme greatly alleviates the communication cost of the underlying optimization algorithm (e.g., as compared with the ADMM). The described stochastic gradient framework is then used in training of a Logistic regression model used for (binary) classification. The current implementation offers a fully distributed, gradient-based Logistic Regression algorithm. The algorithm relies on PyCOMPSs for parallelization. Since the algorithm is gradient-based, with a closed form update at each iteration, there is no need to solve an intermediate optimization problem, as it is the case for the ADMM settings. This makes the algorithm computationally more efficient. The underlying Logistic Regression problem uses a L2 regularization, therefore preserving convexity, but allowing for a tunable parameter that controls the norm/size of the acquired solution.

ADMM-based Ridge and ElasticNet regression

We provide another pair of regression algorithms, based on the ADMM framework[14], [5], namely Ridge and ElasticNet. Ridge and ElasticNet algorithms are part of the family of regularized regression algorithms. Both are very similar to the Lasso algorithm, with the main difference being the regularization: while Lasso uses a L1 regularizer to induce sparsity, Ridge regression uses a L2 regularizer, while ElasticNet combines both L1 and L2 regularization. The implementation relies on PyCOMPSs for parallelization, as well as CVXPY for solving the ADMM-based intermediate optimization problems used for updating the solution variables. The Lasso, Ridge and ElasticNet methods demonstrate the high code reusability of the ADMM-based I-BiDaaS algorithmic implementations, since the basis of the

[14] https://github.com/ibidaas/knowledge_repository/tree/master/tools_technologies/sources/batch_processing/unspmf

models is the same for all three of them, and the only part of the code subject to change is the code part that specifies the regularization.

Differentially private K-means and random forest algorithms

We have implemented differentially private K-means [24] and random forest algorithms [14], [15], [16] in the COMPSs framework. With K-means clustering, vectors within a data set *D* need to be placed in *K* clusters. The *K* centroids (abbreviated as *C*) are selected at random with *K* vectors from *D*. In order to allow for parallel execution, *D* is partitioned into *n* subsets $S_0,\ldots,S_{n-1}$. The next step is to perform the first clustering iteration. Figure 12 shows the COMPSs-relevant task dependency graph for the clustering process.

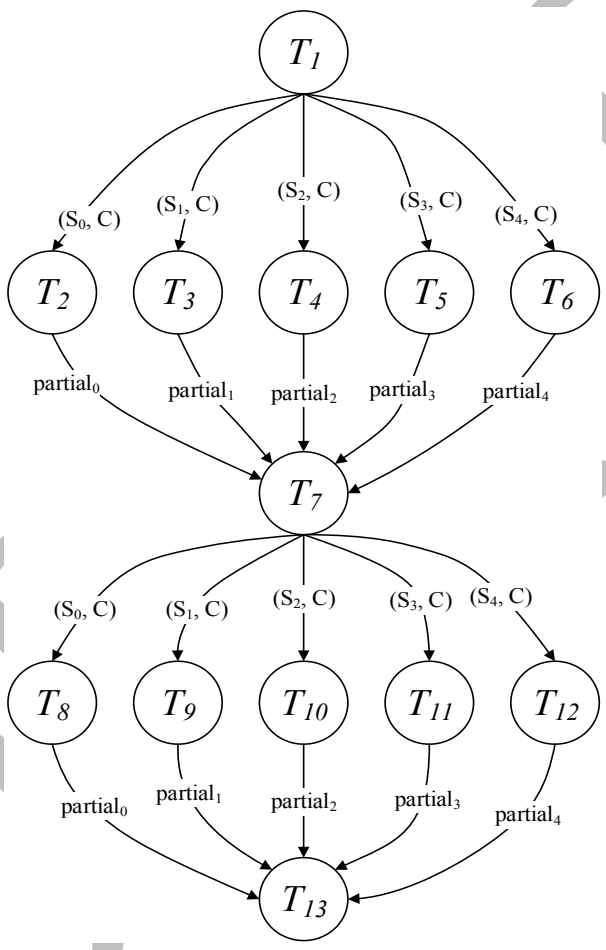

Figure 12: Tasks dependency graph of the clustering process

Let the initialization process be processed by task T_1. Then, the *n* subsets of *D* are passed to the remaining n tasks. Process T_2 handles S_0, process T_3 handles S_1, etc. Then, each of the tasks looks for the nearest centroid for the vectors it processes, then it performs the sum of the dimensions, and then it finds the size of each cluster. Upon completion of this process, the additional task T_7 synchronously collects the information from T_2 to T_6, combines this information and finds positions of the new centroids $\acute{C}$. The described procedure is repeated iteratively until a stopping criterion is met. In the updating centroids step, we add the Laplace noise to make the implementation satisfy ϵ-differential privacy.

4.1.1.2 I-BiDaaS algorithms' mapping to the I-BiDaaS use cases

An overview of the I-BiDaaS use cases, as well as the algorithms that are applied or can be applied on them, is presented in Table 2 below. The Table also provides a rationale why the respective algorithm is selected, as well as potential extensions.

Table 2: Mapping between algorithms and I-BiDaaS industrial use cases

I-BiDaaS use case title, relevant data provider, and a brief description	Algorithm(s) applied within I-BiDaaS, relevant partner who developed/applied the method, and relevant I-BiDaaS deliverable (if applicable); relevant I-BiDaaS related library (e.g., I-BiDaaS knowledge repository, PyCOMPSs dislib, etc.)	A rationale why and how the algorithm applies to the specific industrial use case.	Potential extensions and reference to COMPSs, dislib, or other libraries
CAIXA Analysis of relationships through IP addresses – detect users that are related and connected, based on the IP address pattern of their transactions.	1) K-means (available in dislib) (UNSPMF, D3.2) 2) Differentially private K-means (I-BiDaaS knowledge repository) 3) DBSCAN (available in dislib) (UNSPMF, D3.2) 4) t-SNE (UNSPMF, D3.2) 5) Batch application for the MVP (rule-based tailored algorithm) (BSC, D3.1)	1) K-means is a well-established clustering algorithm, known to scale well to a large number of samples. Due to the nature of the use case, as well as the possibility of analyzing the pattern of a huge number of users, clustering analysis and K-means in particular are a solid choice. 2) Represents a version of K-means that provides some additional privacy guarantees with respect to the data used for training, i.e., the algorithm makes it impossible to perform reverse engineer the model and exploit the data. 3) Another well-known clustering algorithm. Unlike K-means, which assumes convex-shaped clusters, DBSCAN can find clusters of any shape. Additionally, DBSCAN does not require a user provided estimate of the number of clusters, unlike K-means. 4) An algorithm used mostly for visualization of high-dimensional data. It projects the data in 2D or 3D, with the aim of keeping a neighbouring and clustering structure of the data as close to original as possible. We used t-SNE to inspect if there is an intrinsic cluster-like structure of the original data. 5) The goal of the application is to find relations between people, given a set of connections to IP addresses. We want to maximize the detections of close relations between users. Also, the objective of this use case is to validate and test if synthetic data generated with the same characteristics, provides the same or additional insights, and in particular if the number of relationships detected by CAIXA could be detected using synthetic data.	Gaussian mixture model for clustering (available in dislib)
CAIXA Advanced	1) K-means (available	1) K-means is a well-established clustering algorithm, known to scale well to a large number of	Added

analysis of bank transfer payment in financial terminal	in dislib) (BSC, D3.2) 2) DBSCAN (available in dislib) (BSC, D3.2) 3) PCA (scikit-learn library)	samples. Due to the nature of the use case, as well as the possibility of analyzing the pattern of a huge number of users, clustering analysis and K-means in particular are a solid choice. 2) Another well-known clustering algorithm. Unlike K-means, which assumes convex-shaped clusters, DBSCAN can find clusters of any shape. Additionally, DBSCAN does not require a user provided estimate of the number of clusters, unlike K-means. 3) After several changes, the number of attributes increased from the original 87 to 234. This dimensionality complicated the interpretation of the clustering results. For this reason, the second step was to reduce the attribute vector dimension to 4, applying a Principal Component Analysis (PCA) method from the scikit-learn library.	visualization with Qbeast scikit-learn library
CAIXA Enhanced control on online banking (Bizum)	1) K-modes algorithm (from K-modes library) (CAIXA, D3.3) 2) Elbow method (CAIXA, D3.3)	1) K-modes is a clustering algorithm similar to K-means, but that allows to work with categorical data 2) The Elbow method is a heuristic to determine the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use.	K-modes library
TID Accurate location prediction with high traffic and visibility – track the number of users on the network and predict their future number	1) Time-series modelling approach via Facebook's Prophet tool (UNSPMF, D3.3)	1) The adopted approach based on Prophet [10], unlike conventional time series models, does not require the values to be regularly spaced, nor there is need to interpolate missing values, which is suitable for the considered data set with a high degree of missing values.	Missing values imputation
TID Optimization of placement of telecommunication equipment – determining binary antenna KPI labels	2) Random Forest (dislib, D3.3) 3) XGBoost (D3.3) 4) CATBoost (D3.3)	2) Well-established classification algorithm. It represents an ensemble algorithm, as each "forest" consists of a number of decision trees. We chose this algorithm as the use case was modelled as a supervised learning classification problem, as well as due to the interpretability and practical performance of the algorithm. 3) Distributed variant of gradient boost used to train ensemble algorithms, like Random forest. Known to be a highly efficient, flexible and portable library for machine learning algorithms under the gradient boosting framework. 4) CATBoost represents a variant of gradient boost used to train ensemble algorithms, like Random forest. Its main advantages are an innovative, fully automated algorithm for processing categorical features, order boosting, a permutation driven alternative to the classical boosting algorithm as well as fast and easy to use training.	Cascade Support Vector Machine (available in dislib)
CRF Production process of	1) Random Forest (UNSPMF, dislib,	1) By modelling the problem as a supervised learning classification one, where we aim to classify the engines according to their control class, Random Forest comes as a good choice.	Cascade Support Vector Machine

aluminium die-casting – improve the quality of the aluminium die casting process and make it more stable, by finding the most significant process parameters to monitor and control	CRF hackathon, D3.2) 2) Weighted Random Forest (UNSPMF, CRF hackathon, D3.2) 3) Differentially private Random Forest (UNSPMF, I-BiDaaS knowledge repository) 4) ADMM logistic regression (UNSPMF, I-BiDaaS knowledge repository, D3.3) 5) Stochastic gradient logistic regression (I-BiDaaS knowledge repository, D3.3) 6) Deep Neural Network (UNSPMF CRF hackathon, D3.2) 7) DenseNet-201 (UNSPMF, D3.3)	2) Weighted Random Forest represents an extension of basic Random Forest. When a problem is imbalanced, i.e., there are more samples belonging to class "a" than to class "b", the basic random forest algorithm is susceptible to discriminate and favor the larger class. Weighted Random Forest manages to avoid this scenario, by associating different weights (or rewards) for different classes in the objective function. This use case corresponds to an imbalanced problem. 3) Differentially private Random Forest is an extension of the basic RF algorithm that provides privacy guarantees with respect to the data used for training, i.e., it makes it impossible to perform reverse engineering on the model and exploit the data. 4) ADMM is a general optimization framework, well suited to training a wide variety of machine learning models, as well as distributed implementation. Treating the problem in the use case as a binary classification problem, with the aim of identifying defunct and proper engines, a binary classification algorithm can be applied. Logistic regression is a binary classification algorithm, known for its good performance in practice. Its suitability to the problem, as well as the ease of implementing under the distributed ADMM framework made it an obvious choice. 5) Stochastic gradient based logistic regression offers the advantage of a communication efficient solver, alleviating the communication cost of training a distributed logistic regression model. 6) Since the data contains a large number of process parameters, and because of their proven performance in practice, we chose the Deep Neural Network framework. Deep Neural Networks contain a large number of layers that enable them to learn relationships between the input data and the target values, whether that relationship is linear or highly non-linear. 7) DenseNet-201 is a convolutional neural network that is 201 layers deep. It is used mostly in image classification. Considering that the part of the data for this use case consisted of thermal images of engines, convolutional neural networks, and DenseNet-201 in particular, were an obvious choice.	(dislib)
CRF Maintenance and monitoring of production assets - prevent faults before they happen by doing maintenance at the right time	1) Outlier detection using IQR test (UNSPMF, D3.3)	1) IQR (interquartile range) test is used for outlier detection. Outliers represent values that fall outside of the overall pattern of the data. IQR effectively checks how the data is spread about its median. Since it is less susceptible to outliers than the range (difference between the maximum and the minimum value), it is a more useful measure here.	Multi-dimensional outlier detection methods

4.1.1.3 I-BiDaaS algorithms tested on I-BiDaaS industrial data sets

Several recently developed machine learning algorithms by I-BiDaaS were successfully applied on I-BiDaaS data sets. The developed differentially private K-means parallel (COMPSs-based) algorithm was tested on the CAIXA real tokenized IP addresses data set. The data preparation process involved taking into account only those samples for which the respective IP address was used by at least 2 and no more than 10 users. The desired number of clusters was set to 13, according to some previous tests on the same data. After executing the algorithm, we get a silhouette score value of 0.5.

The developed ADMM logistic regression and stochastic gradient descent logistic regression algorithms using the COMPSs framework were tested on the CRF Aluminium casting (sensors) data set. After the necessary pre-processing steps, the data set was divided into two parts: 80% for the training phase and 20% for the predict phase. This means that the data set for fitting contained 51280 samples with 136 features. The resulting problem is a supervised binary classification problem. For both algorithms, the achieved accuracy is 86%. We also compared the accuracy of our implementations to the accuracy of the scikit-learn logistic regression on the same data. We set the parameters for the scikit-learn logistic regression according to the value of the regularization parameter for our ADMM logistic regression implementation. The achieved accuracy of the scikit-learn logistic regression algorithm is 60%.

We further illustrate the scaling results for the ADMM logistic regression algorithm. Figure 13 shows the scaling properties of this implementation. The algorithm was tested for 2, 4, 8 and 10 workers. The execution time scales down up to the number of workers equal 8, after which it starts to grow. The reason for the latter is that the benefit of dividing the work across more workers is eliminated due to the increased inter-worker communication cost.

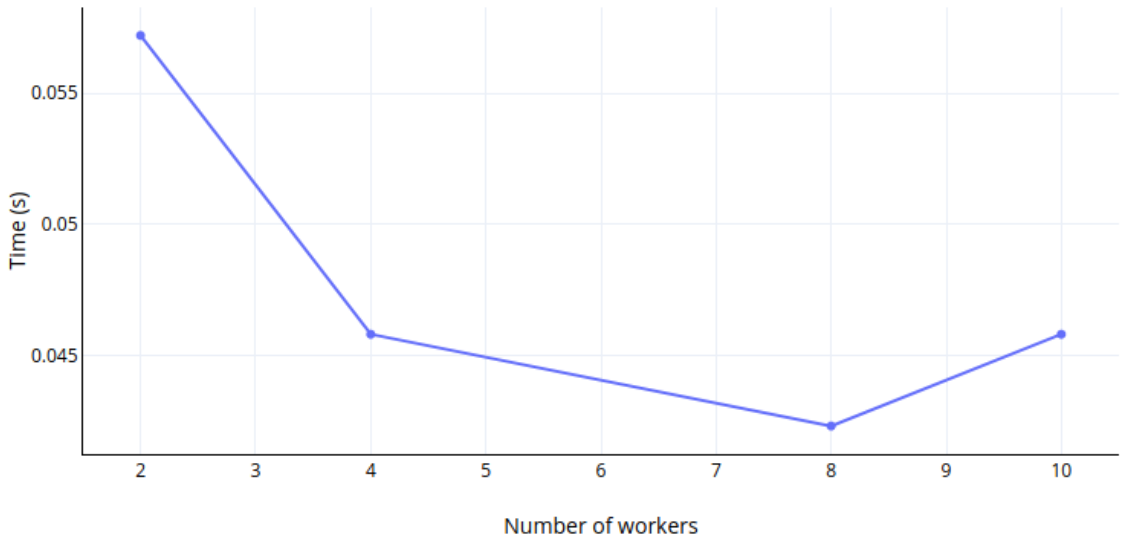

Figure 13: Scaling properties of the ADMM logistic regression algorithm on CRF Aluminium die-casting data set

4.1.2 Dislib integration

The I-BiDaaS consortium established collaboration with the BSC's dislib project [4]. Dislib is an effort to adapt the highly efficient scikit-learn machine learning implementations to the highly parallelizable COMPSs runtime. The I-BiDaaS team has initiated collaboration with dislib, in order to use and test the dislib library within I-BiDaaS, but also to contribute to its development. It was decided that the I-BiDaaS project can contribute to the implementation of the ADMM family of machine learning algorithms to the dislib library, so that they can be used in an open source fashion. We have worked closely with dislib to adapt the ADMM code to the dislib platform, as well as make it compatible with dislib interfaces. The dislib team also contributed greatly to the improvement of the resulting ADMM code.

4.1.3 Integration with data management

In order to fully integrate the dislib library, and all the machine learning algorithms that we have developed as part of the I-BiDaaS platform, we have been working to implement their integration with Hecuba.

The algorithms that dislib implements work on numpy ndarrays, called DS-Array. Hecuba already provides an implementation of numpy ndarrays with a data structure called StorageNumpy. These data structures are quite similar, and both can work with numpy ndarrays. The idea behind this integration is that once the users make the data persistent in Cassandra using Hecuba, they can use all the dislib algorithms in the same way and Hecuba work will be completely invisible for them.

Hecuba stores data in a distributed way, keeping the blocks division from original DS-Array structures. The dislib DS-Arrays store the data in two different ways: in a big structure and dividing data in blocks with the shape defined by the user. When the users call the method *make_persistent*, both arrays lists are transformed to StorageNumpys, which can be identified by the defined name and a unique ID. Once the DS-Array is persistent, the data that it contained before as numpy array has been transformed to StorageNumpy, which can be operated in the same way and do not require as much disk space, because they are a reference to the Cassandra database, thanks to Hecuba.

The method *load_from_hecuba()* can be used to load the stored data from the database, and the received data structure will be also a DS-Array. In Figure 14, it can be seen how a DS-Array is created and then stored in Cassandra using Hecuba. This is how all data arrays are transformed into StorageNumpys, which are a reference to the database and are always synchronized.

```
block_size = (2, 10)
x = np.array([[j for j in range(i * 10, i * 10 + 10)]
              for i in range(10)])
data = ds.array(x=x, block_size=block_size)
data._blocks
```

```
[[array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
         [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]])],
 [array([[20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
         [30, 31, 32, 33, 34, 35, 36, 37, 38, 39]])],
 [array([[40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
         [50, 51, 52, 53, 54, 55, 56, 57, 58, 59]])],
 [array([[60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
         [70, 71, 72, 73, 74, 75, 76, 77, 78, 79]])],
 [array([[80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
         [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])]]
```

```
data.make_persistent(name="hecuba_dislib.test_array")
data._blocks
```

```
[StorageNumpy([[[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
                [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]]]),
 StorageNumpy([[[20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
                [30, 31, 32, 33, 34, 35, 36, 37, 38, 39]]]),
 StorageNumpy([[[40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
                [50, 51, 52, 53, 54, 55, 56, 57, 58, 59]]]),
 StorageNumpy([[[60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
                [70, 71, 72, 73, 74, 75, 76, 77, 78, 79]]]),
 StorageNumpy([[[80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
                [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]]])]
```

Figure 14: Creation of a persistent DS-Array

4.2 Advances in the technological components of the platform

4.2.1 Big Data Interactive Visualization

One of the goals of the I-BiDaaS platform is to empower users with different levels of expertise and backgrounds to gather insights from Big Data. Our platform provides different methodologies to do so, ranging from the expert, the self-service and the co-develop modes. All these modes use underneath a combination of batch and streaming techniques to select, transform and aggregate data that the user later consumes via static and interactive visualization. In this setting, the visualization is the end of the analytical process. While such a linear approach is common and sufficient for many applications, some cases require intertwining computation and visualization in a close and fast loop.

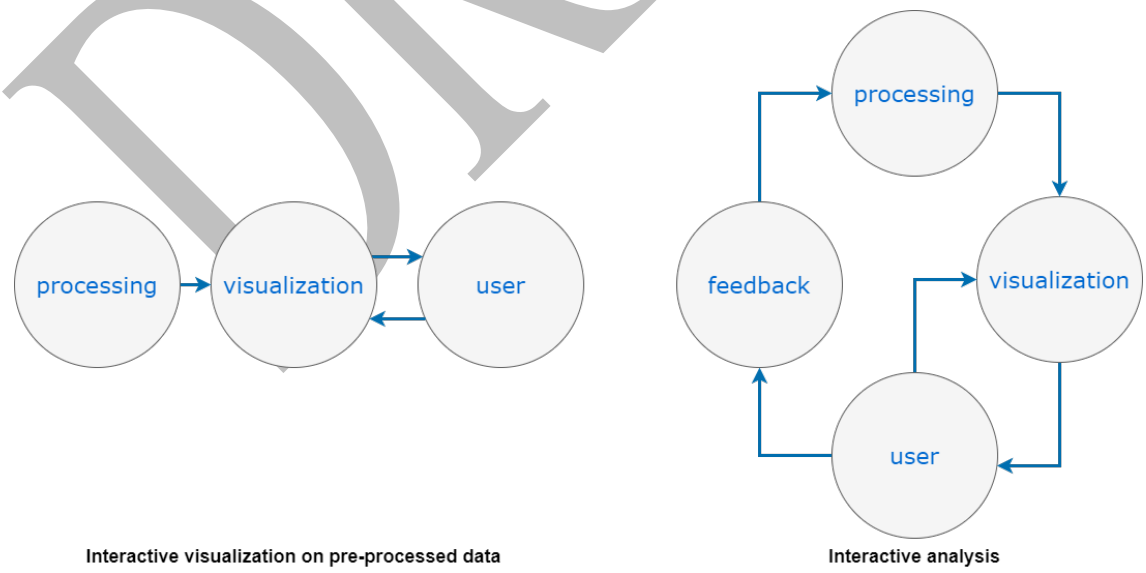

Figure 15: Linear vs Interactive analysis workflow

Figure 15 shows at high-level the different role of users in "linear" and "interactive" analysis. In the first case, users interact with a report or visualization built using a limited set of pre-

processed information. In contrast, in the second, users can provide feedback that triggers additional analysis which may run distributed on a remote cluster of computers. The main difference is that in the first case, what the users can analyse is defined at the time the report is developed, while in the second case, users have the possibility of changing and improving the output analysing new data, tuning algorithms and updating prediction models.

There are several applications in which having a close interaction between users and computation can be beneficial. For instance, when an analyst is approaching a new data set for the first time; or when we are dealing with a dataset that is rapidly changing and where new patterns and behaviour can emerge. Another typical situation is when we are looking for new anomalies in a large dataset. In all these cases, it is crucial to put the user into a loop between computation, visualization and analysis, so that she or he can focus, select, transform and run ML algorithms on large quantities of information interactively.

The key to achieving interactivity is to manage information smartly, reducing at a minimum possible the data managed using indexing, sampling and approximation teaching. For instance, if we are approaching a large unknown dataset, following the first approach, we can try to represent it via an interactive three-dimensional visualization where only a sample of the data is shown. While the user interacts with the graphical interface, new data is fetched from the storage to increase the resolution and to present additional information. Thanks to this approach, users can then give feedback to the system to improve the accuracy of the analysis. For instance, users can label some elements to train a supervised classifier, or they can apply a different algorithm on a specific subset of the dataset that shows a different pattern.

Moreover, to achieve interactivity on Big Data, all computation must run seemingly in multiple machines using any of the distributed machine learning algorithms available in the I-BiDaaS platform.

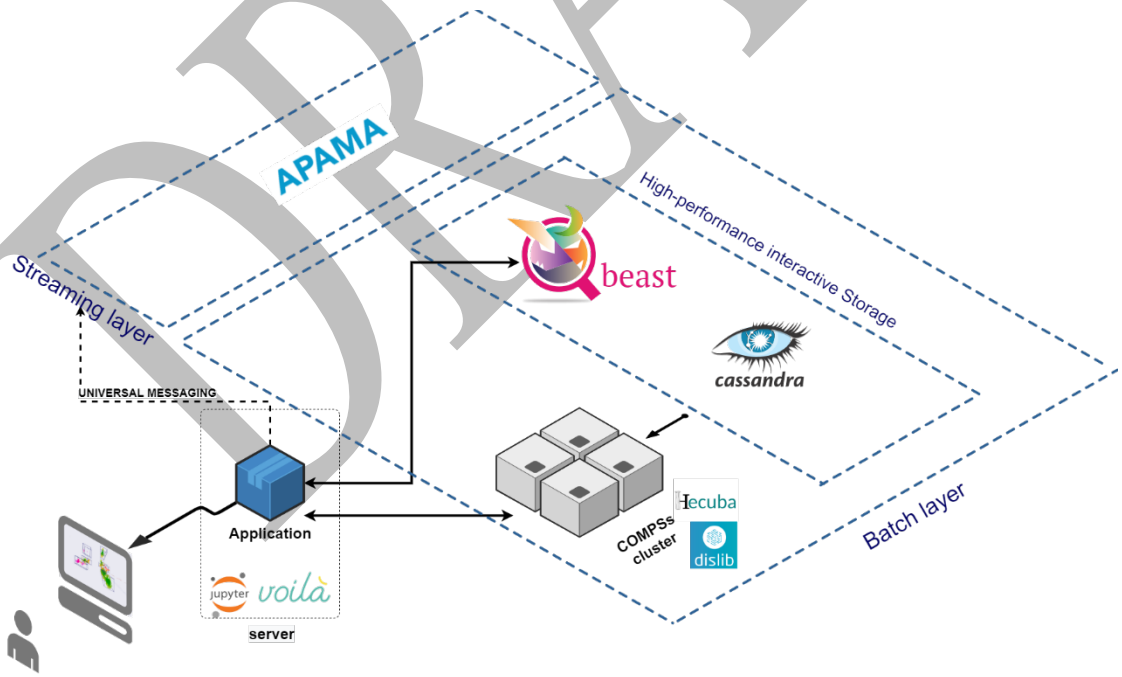

Figure 16: I-BiDaaS interactive analysis architecture

To ensure low latency and interactivity and consistency between the backend data and frontend visualization, we designed a new architecture that leverages both the advantages of parallel distributed computing and stateful applications. An essential requirement is to allow tight integrations between the backend used in distributed processing, the state of the

application, and the UI interface. To do so, we opted for an architecture that combines COMPSs (with dislib and Hecuba), with the Jupyter ecosystem (Voila[15] and Widgets[16]) and javascript frameworks such as Three.js[17] and Backbone.js[18]. The synchronization between frontend and backend is maintained using the Jupyter Widgets, which are eventful python objects that have a graphical representation action in the browser.

Figure 16 shows a diagram of the architecture designed to support interactive Big Data analysis. Users can access to the interactive visualization through their favourite browser. When they do so, a new python process is created in the Jupyter/Voila server. This process is responsible for keeping the current state and synchronizing what the user sees in the browser and what is computed in the remote COMPSs cluster. When the user interacts with a visual component in the webpage, the Widget, a message over WebSockets[19] is sent to the application to update the application status and trigger new computation if necessary. An application can use and combine a wide variety of Widgets, from both the open-source community and the ones we developed in the I-BiDaaS project that allows a smooth interaction with our platform and with our indexing system Qbeast [3].

A positive side effect of our approach is that in most cases, it is possible to develop both the frontend and backend of the application with the same python code. In such a way, the development of a new application is greatly simplified, and we can empower expert users to extend, configure, and customize analysis and visualizations in a simple and integrated manner.

4.2.2 Test Data Fabrication Tool production integration

Data analysis carried out by the Batch Processing module can be leveraged by the TDF module [20] to fabricate synthetic realistic workloads. Synthetic datasets can be used in various use cases e.g., application testing, performance evaluations, data augmentation, evaluation studies and more. The synthetic dataset, in order to be realistic, must resemble some ground truth i.e., a real-world dataset, as much as possible.

Typical data characterization would look for the simplest abstract mathematical model that has certain properties that are considered as most important. A common approach used in descriptive modelling (often called exploratory data analysis) is to create a statistical summary of the observed dataset. This can be further extended with the identification of distributions that constitute the model, correlations, relationships, patterns, types, formats etc. found in the data.

The Batch Processing module is currently able to produce various descriptive statistics metrics for Dates, Integers, Floats, Decimals, Longs, and Bigints. The analysis is performed per column, and written into a JSON file with agreed schema, containing its count, mean, stdev, variance, max value, min value, count of null, distinct and empty values, length, etc.

In the current workflow, data is analysed by a new Hecuba module called *hecuba.Stats*, where the various descriptive statistics metrics have been implemented to run in a distributed environment.

The current implementation supports:

[15] https://github.com/voila-dashboards/voila
[16] https://jupyter.org/widgets
[17] https://threejs.org/
[18] https://backbonejs.org/
[19] https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
[20] https://www.research.ibm.com/haifa/dept/vst/eqt_tdf.shtml

- for Dates, Integers, Floats, Decimals, Longs, and Bigints:
    - count
    - mean
    - stdev
    - variance
    - max
    - min
    - count of empty values
- for Booleans:
    - count
    - total true
    - total false
    - count empty values
- for text fields:
    - count
    - max length
    - min length
    - total empty values.
    - For each word in the field:
        - count
        - distinct count

Once the batch application terminates, the same code uses the computed information to generate a JSON file, which follows a pre-determined schema.


```
{
   "columnName":"fk_cod_operacion",
   "columnAnalysisResults":{
      "nbRows":799999,
      "nbOfNullValues":0,
      "nbOfEmptyValues":0,
      "maxDistinctFrequency":36939,
      "minStringValue":"00000",
      "maxStringValue":"validaUsuarioWS",
      "nbOfUniqueValues":282,
      "nbDistinctValues":1570,
      "lengthDistribution":[ {
            "count":560,
            "distinctCount":2,
            "value":23
         },{
            "count":307296,
            "distinctCount":980,
            "value":5
       }],
      "wordsDistribution":[{
            "count":534,
            "distinctCount":0,
            "value":"ARQDECInformeReexecucio"
         },{
            "count":2652,
            "distinctCount":0,
            "value":"01471"
      }]
   }
}
```


Figure 17: An example of JSON file structure for a text field

As shown in Figure 17, the JSON file contains a set of statistics for each column of the target dataset.

TDF can ingest these descriptive statistics using a REST API, to automatically create a set of corresponding constraint rules and add them into a TDF project, and then to start the automatic generation of synthetic data. This module supports various interfaces for analyses results produced by various sources (including the analysis results produced by the Batch Processing module)

For more information, see Deliverable D2.5, Sections 2.3 and 5.2 ('Data Analysis', 'Modeling Automation' and 'Integration with Hecuba and COMPSs').

4.2.3 Integration between Batch Processing and Streaming Analytics modules

In this final period of the work package (M19-M30), we have been able to interconnect both Batch Processing and Streaming Analytics processing modules, which worked independently in the past. With this, we enable the possibility of storing the processed streaming data for later usage, where more complex processing (impossible to be done in real-time) can be applied.

Essentially, a connector listens to the MQTT (Message Queuing Telemetry Transport) queue and writes the needed data to the selected persistent storage (Apache Cassandra) using the Hecuba framework to ease its implementation and increase the portability of the code. More details about this specific connector can be found at Deliverable 4.3: Streaming Analytics and Predictions.

5 Conclusions

The objective of this deliverable is to describe the progress achieved in WP3 in this last period of its lifetime (i.e., from M19 to M30), and as a continuation of the previous work progress reported in D3.2. From what has been described in this document, we can see that all the objectives of WP3 and the Batch Processing module have been accomplished without deviations (all deadlines and milestones have been met).

Section 2 has described how the Batch Processing module is one of the two core parts on how the I-BiDaaS platform performs data analytics, contributing with the advanced Machine Learning algorithms, the COMPSs programming model, the dislib library for distributed big data analytics and Qbeast for visualization. Features to feedback from analytics results to problem modelling and to data fabrication preparation and pre-processing and packaging are also offered.

Section 3 has presented the use cases from the perspective of the Batch Processing module, thus, from the point of view of the technology, not the specific domains targeted. More details related to the domain, impact and results obtained on each use case will be detailed on WP6 deliverables D6.4 and D6.5 (due M32 and M36 respectively). A total number of 6 use cases are presented, with specific details on how the Batch Processing module has helped on each case (i.e., specific tools and specific algorithms).

The specific technical innovations in the Batch Processing module have been described in Section 4, and include contributions in Machine Learning algorithms (novel algorithmic implementations; a comprehensive mapping of the algorithms considered in I-BiDaaS to the I-BiDaaS industrial use cases; and additional testing and validation of previously developed algorithms on the I-BiDaaS industrial use cases). It is of special importance to highlight Section 4.1.1.2, which synthesizes in a single table what algorithms have been used on each use case, and reasoning on why the algorithms have been selected as the ones required on each case. Technical innovations for each software component have been also presented, such as the dislib and Hecuba integrations, a Big Data interactive visualization tool based on the Qbeast software, and the latest innovations on the Test Data Fabrication tool (TDF). We finally highlight that we have enabled the integration of the Batch Processing module with the Streaming Analytics module.

Although WP3 work finishes officially here (M30), there are still tasks that we need to do in order to finish the preparation of showcases for the use cases that demonstrate how the different domains have gained capacities in terms of functionality on data analysis. We will provide full support to the use cases to achieve stunning demonstrators for the end of the project. Also, even when it was not in the original work plan thus it has been contributed as extra work, we have tried to integrate the algorithms produced in this WP in dislib, so they will continue being available for other projects / communities even when I-BiDaaS finishes. While, in some cases, we have finished the integration, in some others pending steps were missing, so this is something more that we can do in the future.

References

[1] "PyCOMPSs: Parallel computational workflows in Python", Enric Tejedor, Yolanda Becerra, Guillem Alomar, Anna Queralt, Rosa M. Badia, Jordi Torres, Toni Cortes, Jesús Labarta, IJHPCA 31(1): 66-82 (2017), DOI: 10.1177/1094342015594678

[2] "COMP Superscalar, an interoperable programming framework", Badia, R. M., J. Conejero, C. Diaz, J. Ejarque, D. Lezzi, F. Lordan, C. Ramon-Cortes, and R. Sirvent, SoftwareX, Volumes 3–4, December 2015, Pages 32–36, DOI: 10.1016/j.softx.2015.10.004

[3] Cugnasco, C., Calmet, H., Santamaria, P., Sirvent, R., Eguzkitza, A. B., Houzeaux, G., Becerra, Y., Torres, J. & Labarta, J. (2019, December). "The OTree: Multidimensional Indexing with efficient data Sampling for HPC". In 2019 IEEE International Conference on Big Data (Big Data) (pp. 433-440). IEEE.

[4] J. Álvarez Cid-Fuentes, S. Solà, P. Álvarez, A. Castro-Ginard, and R. M. Badia, "dislib: Large Scale High Performance Machine Learning in Python", in Proceedings of the 15th International Conference on eScience, 2019, pp. 96-105.

[5] "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers", S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Foundations and Trends in Machine Learning, 3(1):1–122, 2011.

[6] Sahu, A. & Jakovetić, D. & Bajović, D & Kar, S. "Communication efficient distributed weighted non-linear least squares estimation", EURASIP Journal on Advances in Signal Processing, (2018): 66.

[7] Scikit-learn: machine learning in python. https://scikit-learn.org/stable/

[8] T. Chen, C. Guestrin, "XGBoost: A Scalable Tree Boosting System", KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2016 Pages 785–794.

[9] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, Andrey Gulin, "CatBoost: unbiased boosting with categorical features", NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, December 2018, Pages 6639–6649.

[10] Taylor, Sean J., and Benjamin Letham. "Forecasting at scale", The American Statistician 72.1 (2018): 37-45.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[12] Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library", 2019, available at: https://arxiv.org/abs/1912.01703

[13] "Just Relax and Come Clustering! A Convexification of K-means Clustering", by Fredrik Lindsten, Henrik Ohlsson, Lennart Ljung, Technical report from Automatic Control at Linköpings universitet, 2011.

[14] C. Dwork, "Differential privacy" Automata, Languages and Programming, 33rd International Colloquium, ICALP2006, Venice, Italy, Proceedings, Part II, pp. 1–12, 2006.

[15] Y. Zhang, N. Liu, and S. Wang S, "A differential privacy protecting K-means clustering algorithm based on contour coefficients", PLOS ONE, vol. 13(11), e0206832, 2018.

[16] C. Dwork, "A firm foundation for private data analysis", Communications of the ACM, vol. 54, pp. 86-95, 2011.

[17] "A Variation Coefficient Similarity Measure and Its Application in Emergency Group Decision-making". Xuanhua Xu, Liyuan Zhang and Qifeng Wan. Systems Engineering Procedia (5); 119 – 124. 2012

[18] Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest", R news 2.3 (2002): 18-22.

[19] Huang, Z.: "Extensions to the k-modes algorithm for clustering large data sets with categorical values", Data Mining and Knowledge Discovery 2(3), pp. 283-304, 1998.

[20] Thorndike, Robert L. "Who belongs in the family", Psychometrika. 1953.

[21] "CVXPY: A Python-Embedded Modeling Language for Convex Optimization", S. Diamond and S. Boyd, Journal of Machine Learning Research, 17(83):1-5, 2016.

[22] C. Bishop, "Pattern Recognition and Machine Learning", Springer, 2006.

[23] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge", International journal of computer vision 115.3 (2015): 211-252.

[24] Sukpisit, S., Skrbic, S., Jakovetic, D., "Parallel Differentially Private K-Means Implementation Using COMPSs Framework", 10th International Conference on Information Society and Technology, Serbia, 2020, in print.