Abstract

PROTEUS project has been focused on applying Big Data technologies to ArcelorMittal data with the aim of reducing the cost derived from material rejections due to defects. Thus, previous deliverables (D2.8, D2.9 D2.10) presented three uses cases aligned with this goal. Both deliverables described the iterative evolution of the prototype designed to cover the use cases (control of the key process parameters, better quality of the flatness of the products and better quality of the flatness and process).

The current deliverable D2.11 is the final one of this series and in the same line as previous deliverables, it contains the improvements that have been reached. Therefore, it maintains the same structure with the three use cases, but with an additional section, the pilot evaluation one, which includes the potential, pros and cons of the achievements under a business point of view.
Executive summary

The proposed prototype had to tackle the problem of early-stage detection of defects in the Hot Strip Mill. In deliverables D2.8, D2.9 and D2.10, the general requirements and objectives to this goal were presented. The problem of early detection of defects is represented by three different industrial use cases. These three use cases cover three different goals: 1) Control of the key process parameters, 2) Better quality of the flatness of the products and 3) Better quality of the flatness and process. They will be describe in detail in Section 2 of this deliverable.

The current deliverable contains the last advances in the prototype before the final demonstrator. Thus, it includes a general review of the three industrial use cases (Section 2), summarizing the main achievements, introduces the new advances in each of them, and the main pros- and cons- obtained during the whole project under a business point of view.

The current deliverable details the software implementation and setup of the first prototype in PROTEUS following the above mentioned objectives according to the business use cases covering some alarming issues and pattern discovery aiming to detect and understand the appearance of flatness defects. Besides integrating the technology components produced by PROTEUS as a result of the research WPs (WP3, WP4 and WP5), the work performed in this task and documented in this deliverable also included the implementation of specific components that are required for the recreation of the operational environment, following the actual data production workflow and timeline, as a simulation of the actual production environment in the industrial context (i.e. in the form of configurable data producers, tailored to the specific performance of the real scenario).

Finally, some Key Performance Indicators (KPIs) have been defined as the ways to measure to what extent the objectives, both business and technical objectives, have been guaranteed and fulfilled with the current technology.
Document Information

IST Project Number | 687691 | Acronym | PROTEUS
--- | --- | --- | ---
Full Title | Final demonstrator |  |
Project URL | www.proteus-bigdata.com |  |
EU Project Officer | Martina EYDNER |  |
Deliverable Number | D2.11 | Title | Final demonstrator
Work Package Number | WP2 | Title | Industrial case: requirements, challenges, validation and demonstration
Date of Delivery | Contractual M36 | Actual M36
Status | version v.1 final ■ |  |
Nature | report □ demonstrator ■ other □ |  |
Dissemination level | public ■ restricted □ |  |
Authors (Partner) | AMIII |  |
Responsible Author | Name | E-mail | Partner | AMIII |  |
Abstract (for dissemination) | PROTEUS project has been focused on applying Big Data technologies to ArcelorMittal data with the aim of reducing the cost derived from material rejections due to defects. Thus, previous deliverables (D2.8, D2.9 D2.10) presented three uses cases aligned with this goal. Both deliverables described the iterative evolution of the prototype designed to cover the use cases (control of the key process parameters, better quality of the flatness of the products and better quality of the flatness and process)
The current deliverable D2.11 is the final one of this series and in the same line as previous deliverables, it contains the improvements that have been reached. Therefore, it maintains the same structure with the three use cases, but with an additional section, the pilot evaluation one, which includes the potential, pros and cons of the achievements under a business point of view.
Keywords | Use cases, KPIs, prototype

Version Log
<table>
<thead>
<tr>
<th>Issue Date</th>
<th>Rev. No.</th>
<th>Author</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>30/09/2018</td>
<td>0.1</td>
<td>TREE</td>
<td>ToC and initial contents update, starting from the previous D2.10 (third prototype)</td>
</tr>
<tr>
<td>02/11/2018</td>
<td>0.2</td>
<td>AMIII</td>
<td>First draft, pending to review Section 4</td>
</tr>
<tr>
<td>20/11/2018</td>
<td>0.3</td>
<td>AMIII</td>
<td>New draft with deep revision from TRI</td>
</tr>
<tr>
<td>22/11/2018</td>
<td>0.4</td>
<td>TRI</td>
<td>Major revision of the draft</td>
</tr>
</tbody>
</table>
# Table of Contents

Executive summary .................................................................................................................. 2  
Document Information ............................................................................................................ 3  
Table of Contents .................................................................................................................. 3  
List of figures .......................................................................................................................... 4  
Abbreviations .......................................................................................................................... 5  

1 Introduction .......................................................................................................................... 7  

2 Description and update of industrial use cases .............................................................. 8  
  2.1 Data description (casuistic) ......................................................................................... 8  
  2.2 Specific business use cases for the prototype (V3) .................................................... 8  
    2.2.1 Use case 1: Control of the key process parameters ........................................... 9  
    2.2.2 Use case 2: Better quality of the flatness of the products ............................... 11  
    2.2.3 Use case 3: Better quality of the flatness and process ................................... 12  
  2.3 Pilot evaluation ............................................................................................................. 14  
    2.3.1 Pilot evaluation methodology ........................................................................... 14  
    2.3.2 PROTEUS evaluation results ......................................................................... 14  
    2.3.2.1 Relevance of the tool’s functionalities ......................................................... 14  
    2.3.2.2 Potential improvements ........................................................................... 14  
    2.3.2.3 The commercial viability ......................................................................... 15  

3 Technical description ...................................................................................................... 16  
  3.1 Overview ....................................................................................................................... 16  
  3.2 Cluster infrastructure .................................................................................................. 17  
    3.2.1 Hardware Architecture .................................................................................. 17  
    3.2.2 Base Software & Services Architecture: services under Hortonworks data platform and CouchBase ................................................................. 19  
    3.2.3 The PROTEUS core contributions ................................................................. 22  
  3.3 Tuning for the final demonstrator ............................................................................ 24  
    3.3.1 HDFS + Hive ..................................................................................................... 25  
    3.3.2 Kafka cluster ....................................................................................................... 25  
    3.3.3 PROTEUS producer and consumer ................................................................. 25  
    3.3.4 Couchbase cluster ........................................................................................... 25  
    3.3.5 Flink cluster ........................................................................................................ 25  
    3.3.6 PROTEUS Flink job ......................................................................................... 26  
    3.3.7 PROTEUS dashboard ...................................................................................... 26  
  3.4 Visual Analytics Platform ............................................................................................. 27  
    3.4.1 Backend ............................................................................................................. 27  
    3.4.2 Front-end ............................................................................................................. 28  
  3.5 Setup ............................................................................................................................. 28  
    3.5.1 Required packages before installation ................................................................ 28  
    3.5.2 Operating System ............................................................................................... 29  
    3.5.3 Server and Agents ............................................................................................. 29  
    3.5.4 Hortonworks Data Platform deployment ........................................................ 31  
    3.5.5 Proteus Environment .......................................................................................... 31  
    3.5.6 Additional Storage System: Couchbase ............................................................ 34  
    3.5.7 Visual Analytics Infrastructure ......................................................................... 35  
    3.5.8 Setup Summary ................................................................................................... 35  

4. Validations & KPIs ............................................................................................................ 37  

5 Conclusions ....................................................................................................................... 44  

6 References .......................................................................................................................... 45
List of figures

Figure 1: Current implementation of the confidence band board ................................................................. 9
Figure 2: example of the “pause” interaction feature implementation in PROTEIC ..................................... 10
Figure 3: Current implementation of the parallel coordinates chart in PROTEIC (example) .................. 10
Figure 4: Representation of the approach proposed .................................................................................. 11
Figure 5: ONLMSR vs LASSO .................................................................................................................. 12
Figure 6: Representation of the SAX model ................................................................................................. 13
Figure 7: Example representation of a heatmap used for flatness map visualization .............................. 13
Figure 8: The overall distributed architecture of PROTEUS .................................................................. 16
Figure 9: Cluster details ............................................................................................................................ 18
Figure 10: Kafka producer ......................................................................................................................... 19
Figure 11: HDFS: Block replication ........................................................................................................... 21
Figure 12: HIVE architecture ..................................................................................................................... 22
Figure 13: The PROTEUS, Flink-based architecture ............................................................................. 23
Figure 14 ONLMSR vs LASSO .................................................................................................................. 39
Figure 15 Scatter plot example in PROTEIC (Iris dataset) ..................................................................... 40
Abbreviations

KPI: Key Performance Indicators
SOLMA: Scalable Online Machine Learning Algorithm
PEACH: Proteus Elastic Cache
SAX: Symbolic Aggregate approXimation
HDFS: Hadoop Distributed File System
ONLMSR: Online Normalised Least Mean Squares regression
ETL: Extract, Transform and Load
AMIII: ArcelorMittal Innovacion Investigacion e Inversion SL
DFKI: Deutches Forschungszentrum Fuer Kuenstliche Intelligenz GMBH
BU: Bournemouth University
TRI: Trilateral Research & Consulting LLP
1 Introduction

In the previous deliverables (deliverables D2.8, D2.9 and D2.10), we detailed the list of general requirements and objectives in order to address the defects at an early stage in the Hot Strip Mill. The industrial partner, ArcelorMittal, proposed three main industrial use cases (control of the key process parameters, better quality of the flatness of the products and better quality of the flatness and process, see deliverable D2.8 section 1.1) which will help to address the problem with different approaches and technologies.

In deliverable D2.7, we specified the software prototypes to be deployed and integrated for the ArcelorMittal use case, describing the infrastructure to get actual data in real-time from the factory sensors and historical registers from the systems without affecting the daily functioning of steelmaking process. Hardware and software requirements and restrictions to the integration with existing systems were also specified.

In deliverables D2.8, D2.9 and D2.10, we presented the different goals to be achieved by ArcelorMittal. Among them are the description of the industrial use cases which derive from the available data-at-rest and data-in-motion with the aim of early detection of defects and minimizing the effect of having missed any of them.

This deliverable describes the final prototype. It is a continuation of the steps of progress made and detailed in D2.8, D2.9 and D2.10. The deliverable provides a final description of the current status of the prototype. It is organized in five sections. An introduction to the deliverable (this section), the description and update of industrial use cases, the technical description of the prototype, the validation and KPI’s section and finally a conclusions section.
2 Description and update of industrial use cases

In this section, a brief reminder of the description of the three industrial use cases is provided. It is important to recall that one of the goals of WP2 was to achieve the functionality suggested by AMII in these three proposed business use cases but also to build a complete solution ready to be used in a production environment.

Moreover, section 2.1 is devoted to describing the casuistic that has been faced by the PROJECT in terms of data and that deserve to be highlighted, as they have been a source of continuous discussion and bottlenecks in certain cases.

Finally, the implementation of all the work and models developed within PROTEUS project directly in the facility seems to be unfeasible at this point, and PROTEUS has had to re-think the pilot methodology to evaluate the utility of the PROTEUS solution. AMIII - and most manufacturing enterprises in general- has a conservative tendency in terms of applying new models in plant, unless they have been widely tested/validated. This does not mean that all the work developed within PROTEUS project is not going to be used but that we have to wait until we can state that the prototype is running online in plant. As such, the project evaluated the possibilities and decided to evaluate the PROTEUS integrated prototype with one, key expert within the plant. The following outlines the contents of that evaluation.

2.1 Data description (casuistic)

It is important to highlight that there has been a significant amount of additional effort derived from the quality of the current plant information systems. AMIII has already detected, thanks to its R&D institution, that although they generate and save a huge amount of data, this might not be as representative and rich as it would desire. With this, we refer to faulty data, missing values and incoherent events that happen in its facilities. This is a problem that ArcelorMittal has been and is still working on, and that will represent an important digitalization strategy in which it has been immersed from the last couple of years and that will continue in the future. Nonetheless, meanwhile this digitalization comes to a reality, it is worth the effort of working directly with the data that it is available nowadays. In fact, it will help ArcelorMittal to distinguish some bottlenecks and improvement points under a data analysis point of view. This data is the data available within PROTEUS project, and has represented a challenge for analysis models and also for the consortium members, that might be used to reach richer results in general. Nonetheless, and once again, it is important to highlight that this is an industrial environment and having “dirty” data is a common issue that adds another complexity layer.

Another important remark derives from the agreement of sharing only anonymized data. This is a standard procedure in terms of IP but has constrained the use of intuition and common sense by the ML models. AMIII has tried to minimize the impact of this lack of semantic information as much as possible, but this is a handicap that we have been struggling with.

2.2 Specific business use cases for the prototype (V3)

In deliverable D2.7 we discussed economic loss due to production of defective coils. It was highlighted the necessity of using sensors in the Hot Strip Mill to detect and overcome the problematic coils as soon as possible.

The fundamental objective of the third version of the prototype is to detect the defective coils to decrease the cost by:

1. Stopping the production at an early stage.
2. Using the defective coils, where the quality standards are not so stringent - allowing new industrial routes.
This prototype contains the models and technologies developed within PROTEUS project to accomplish the requirements derived from the specific business use cases which were proposed. This section will briefly recall the three use cases and will highlight the main achievements in each of them.

2.2.1 Use case 1: Control of the key process parameters

As it was explained in D2.9 and D2.10, the first use case aimed to extract knowledge from the net of sensors spread along the HSM measuring online data parameters with the goal of detecting possible problematic process configurations. To be more precise, the final goal is to control the process parameters to detect problematic and unbreakable margins that these parameters should avoid in order to prevent the appearance of defects, breakages, etc.

Given the large amount of parameters provided, it is not feasible to make a manual inspection to detect deviations in some of the variables. Therefore, a crucial goal has been to develop a methodology able to monitor all these stream data and trigger alarms when the values of the variables are out of a “safety zone”. These alarms would represent a useful and agile tool for the plant operators that would allow them to take decision and develop actions. According to these decision and actions, it would be possible to prevent the appearance of a defect or, in the worst case, if the defect is already present, would allow them to evaluate which is the best way to proceed. Thus, the development of the prototype has achieved a more intuitive and fast visualization of the results by the plant experts.

The original dataset which was provided by ArcelorMittal was divided in two fragments. The first fragment is composed of sensor and flatness data and it has been used to train and make predictions with online machine learning algorithms. The second fragment is composed of other variables about coils obtained after they have been finished. This second fragment is called historical variables in the prototype. The idea was to save already visualized coils and be able to retrieve them after a given time frame. This function provides the powerful capability of crosschecking coils from different time periods, different castings or different steel grades, or simply to re-visualize a coil which was problematic and compare it with another coil with “normal” behaviour.

In D2.9 and D2.10 the main achievements have been explained. They can be summarised as follows:

- Development of a visualization tool capable of monitoring a large amount of signals coming from the plant sensors and trigger alarms when the values of the signals are exceeding a given boundary.
- Implementation of a safety zone, which is a region that adjusts automatically as a new point is included (initially the region had a fixed rectangular shape). In other words, the tool computes the interval in each point and then connects the different points.
- As well as the confidence band, red dots are representing the “problematic” exits from the exit zones. This is very visual and can help the operators to focus on these risky points.

![Figure 1: Current implementation of the confidence band board](image)

- Possibility of pausing the monitoring of the online signal. This makes that, by the action of pressing a simple button the operator can freeze the image and have a deeper analysis of a particular part of the signal. This is something really desirable as the online monitoring, especially when the frequency
rate is high, sometimes can hinder the visualization of trends or interesting parts for the expert. Furthermore, after the expert validations the normal online representation can be reactivated.

- A “zoom” of a desired area. The operator just needs to interactively mark the region that he/she wants to zoom and the screen is automatically adjusted. This functionality is especially useful in combination with the pause, as it allows to emphasize and have a detailed inspection, without the rush of seeing the time series moving or with an axis scale too imprecise.

![Figure 2: example of the “pause” interaction feature implementation in PROTEIC](image)

- Possibility of visualising coils and be able to retrieve them after a given time frame. This function provides the powerful capability of crosschecking coils from different time periods, different castings or different steel grades, or simply to re-visualize a coil which was problematic and compare it with another coil with “normal” behaviour.

- Introduction of the historical variables from the HSM dataset in the final analysis. The approach has been to represent parallel coordinates of several variables and several coils. This plot can be modified by the user, who can decide which are the variables that he/she wants to see as well as which coils are shown, and it has available to configure interaction options for filtering, ordering, and so on.

![Figure 3: Current implementation of the parallel coordinates chart in PROTEIC (example)](image)
Figure 3 shows information about variables V1825, V4018, V6679, V1827 and V1829 from coils 40304079, 40304076, 40304080, 40304082 and 40304085. Recall that these variables are anonymized for the consortium.

2.2.2 Use case 2: Better quality of the flatness of the products

There are three main parameters to take into account regarding the quality of the coils: thickness, width and flatness. However, the flatness is the hardest to predict due to its variability. There are different factors affecting the flatness of the coils although they are not known in advance.

The main goal of this use case is to predict the flatness of the coils and avoid expensive implications downstream in the production process or even at reception by ArcelorMittal clients.

In the prototype we aimed to achieve the following:

- develop predictive models which self-update sequentially as new flatness information become available. It can predict anytime, but learn (update the model’s parameters) only once the true output is obtained, hence the relevance of continuous learning.

The following diagrams (Fig. 2) highlight our approach:

![Diagram](Image)

**Figure 4: Representation of the approach proposed**

In deliverable D2.9, Lasso regression was described as a predictive model and it was totally integrated into SOLMA as it was explained in deliverable D2.10. The algorithm on the given scenario works as follows:

- The algorithm receives readings from various sensors such as temperature, pressure etc. on position \( x \) at each time step \( t = 1,2,3,\ldots \);
- The algorithm then calculates the flatness prediction \( y_t \) for position \( x \) at each time step \( t \);
- Once the actual flatness values become available, the algorithm updates its parameters.

This algorithm was tested on ArcelorMittal data showing that the algorithm can learn from the acquired information and provides a reasonable estimate for the flatness values in real time.

Since it was released deliverable D2.9 several improvements have been developed. A new algorithm, Online Normalised Least Mean Squares regression (ONLMSR) has been integrated into SOLMA and it has been possible to deploy it in the complete prototype. ONLMSR provides a significant improvement in terms of accuracy and time. Initially LASSO algorithm was used, as it was not clear which features are useful. It appears that the shrinkage feature of LASSO assigns zero weights to some features. ONLMSR was proposed as an alternative to LASSO. While it does not shrink the model (the weight of features do not go to zero), it has the ability to detect shifts in the data. LASSO time complexity is \( O(n^3) \). In contrast, ONLMSR time complexity is \( O(n) \). For more details, please see D4.3 and D4.4. The accuracy comparison on a sample of 300 coils of various lengths is as follows:
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RMSE</th>
<th>$R^2$</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASSO</td>
<td>24.48</td>
<td>0.09</td>
<td>9.51</td>
</tr>
<tr>
<td>ONLMSR</td>
<td>20.58</td>
<td>0.4</td>
<td>4.37</td>
</tr>
</tbody>
</table>

**Figure 5: ONLMSR vs LASSO**

ONLMSR outperforms online LASSO on all measures ($R^2$ and Mean Absolute Error). The plot shows the difference of the cumulative loss between ONLMSR and online LASSO. Notice the difference is negative for the entire stream implying that ONLMSR outperforms online LASSO.

### 2.2.3 Use case 3: Better quality of the flatness and process

This use case is a complement of the previous one. It tries to go beyond the prediction to find the cause of flatness problems. Thus, we aim to find patterns in the process variables that might explain/predict the appearance of flatness. These patterns would represent different aspects from the process variables, such as peaks, pronounced slopes, valleys and other statistical measurements which will be associated with certain degrees of flatness.

The SAX algorithm introduced in D2.9 is able to extract the relevant quantities from the time series signals and plot them in a two-dimensional graph as shown in Figure 6.
In Figure 6 each box represents the degree of flatness for each batch of points. Each batch is treated by the SAX one by one. In this example there are three degrees of flatness, A, B and C, where A is more flat and C less flat. The similarity of the set of points within each block is represented with a numerical value and a colour scale. In this colour scale, white colour means low similarity of the points in the block and dark blue ones higher similarity.

The graph is being created as the same time as the time series is evolving. Initially the points from the beginning of the signal are assigned a flatness degree A with a similarity of 0.4475. When the next batch of points is introduced to the model, these points differ from the initial ones, and the degree is assign to class C. From this graph it can be derived directly that the ending points of the coils have a different degree of flatness for the model in comparison with the rest of the signal (something which makes sense). It also happened that starting points have a different behaviour as the immediate posterior points, although the flatness prediction returns to this class later, in the moment when the polling process is more stable.

Moreover, as a complement, we have also created a flatness map. In this map we can visualize the evolution of the flatness in each of the parts of the coil with a colour scale. In Figure 7 it can be seen as the flatness is poorer in the borders of the coil whereas the inner part of the strip is more stable.

Figure 6: Representation of the SAX model

Figure 7: Example representation of a heatmap used for flatness map visualization
2.3 Pilot evaluation

As a result of continuous interaction within AMIII, we have developed a bespoke dashboard and an accompanying library of algorithms that will help tackle the flatness issue.

Conversations with HSM experts at AMIII have validated the results of the project and indicated that the outputs of the project have achieved improvements in the three use cases addressed by the PROTEUS solution. To be more precise, an expert has tested the visualization tool which contains the visualization charts described in use case 1, the Lasso model in use case 2 and the SAX algorithm described in use case 3. His feedback will be detailed in Section 2.2.1. Moreover, more plant experts have been involved during the project, sharing some feedback regarding the values reached by the model and the interpretability of some results. They felt pleased to have a model able to cope with the time series data and make on-line predictions, something which cannot be done nowadays in the plant. They would need to see performance evaluated in a live test scenario before fully installing the system in the factory. Nonetheless, since they last reviewed the algorithm, the accuracy of the models have improved, so they are optimistic with the performance of the developments.

The key expert’s evaluation of the PROTEUS integrated prototype is presented below.

2.3.1 Pilot evaluation methodology

The evaluation was carried out as follows: we contacted the expert, explained to him what we wanted from this consultation and arranged a physical meeting where we showed him the live visualization prototype running. We explained to him how the tool worked, what the plots mean and what capabilities the dashboard had. After this 15 minute introduction, we gave him some time (around 45 minutes) to play with the tool and share his feeling and questions about the usability. We had been taking notes during the whole period and transcribed them anonymised to the consortium afterwards.

The chosen expert for running the evaluation has been carefully selected due to his direct involvement with the chosen facility, the HSM. The person is in charge of the quality department in the HSM and thus, has a wide background knowledge of flatness and the variables involved in the process.

2.3.2 PROTEUS evaluation results

With respect to the findings of the evaluation, the expert provided feedback on three key factors:

1. The relevance of the tool’s functionalities for AMIII and its potential usability for employees
2. The potential improvements that would strengthen the case for implementation at AMIII or other facilities
3. The commercial viability of the toolset.

2.3.2.1 Relevance of the tool’s functionalities

With respect to the relevance of the tool for AMIII and the specific use cases examined there, the expert confirmed that the final, integrated prototype represents a significant improvement on AMIII’s current situation. People managing the HSM system only have historical visualizations of past coils to work with, and the streaming monitoring of certain crucial variables adds the ability to intervene much sooner. In addition, the expert indicated that the visualization dashboard and the way in which it presents the data is intuitive and self-explanatory. Workers who would need to use the tool should be able to do so without difficulty.

2.3.2.2 Potential improvements

In relation to potential improvements, the expert suggested three key improvements. First, it was suggested that other models could be introduced to the tool. Specifically, it would be nice-to-have a drop-down list with the algorithms that can be used and to be able to visualize them in a graph just by clicking on any of them.
Second, it would be nice to move the graph plots in the web just by dragging, and thus to be able to order the variables how the user chooses. Third, the expert mentions that it would be useful to obtain the value of the outlier as you are approaching the cursor, and moreover, to add the possibility of adding some notes somewhere, for instance, although there are many outliers in a certain region in a given coil, not all of them are critical or just a reminder to check this coil afterwards. The consortium members are considering each of these improvements in post-project versions of the toolset to enable these requested features to be integrated.

2.3.2.3 The commercial viability

Finally, with respect to commercial viability, the expert confirmed that it would be desirable to have a toolset like this installed in the plant in the future. The conservative nature of the AMIII management structure means that such an installation would take some time. However, the improvements it offers certainly warrant further investigation within the plant. In addition, the expert confirmed that there have not been any other similar systems that have come to the attention of AMIII, and that this product would be unique in the market. This further supports the unique value proposition of the PROTEUS toolset and visualizations for commercial manufacturing.

So, to sum up, AMIII is pleased with the work developed in PROTEUS. There are a wide range of models that can be tested and that will serve as a benchmark for future R&D projects. Moreover, the visualization tool, which has been customized with AMIII feedback and requirements, is already a useful tool that can be used by our operators and that can be generalized to other facilities. The validation of the tool by a volunteer expert demonstrated that the PROTEUS toolset was easy-to-use, that the visualisation of the results was intuitive for workers and that it was possible to use a solution like this in the plant.
3 Technical description

3.1 Overview

The Proteus environment will be deployed on a computer cluster located at Bournemouth headquarters. The cluster consists of four virtual machines with the Hortonworks Data Platform 2.5 distribution, which is capable of being integrated with external services such as Apache Flink and CouchBase. These four virtual machines are distributed on a master-slave architecture and are in charge of providing the hardware resources to achieve the full integration of the components of the demo. An overview of the architecture and the data flow is shown in the following image:

![Diagram of the Proteus architecture](image)

**Figure 8: The overall distributed architecture of PROTEUS**

The raw data starts its life within an ETL process, a process in database usage to prepare data for analysis, following these steps: (1) to send data to a distributed HDFS cluster along the four nodes. At this point Kafka takes the data (2) to start the ingestion on the processing side with Apache Flink Cluster (3) over Proteus Hybrid Engine, PEACH and SOLMA library developed on the project, or directly to CouchBase Server as another parallel ingestion (3) using Kafka Connect in order to persist data without any process.

The cluster deployment is based on a lambda architecture (i.e., a data-processing architecture designed to handle massive quantities of data by taking advantage of both batch and stream-processing methods) with two processing layers: a speed layer and a batch layer.
• Batch layer is in charge of processing historical data persisted on the CouchBase server. Using a specific Flink engine developed by Proteus team (SOLMA library, PEACH library and DSL), this layer will retrieve all data needed to improve the models that will be applied in the speed layer.

• Speed layer is in charge of processing all incoming data in a real-time fashion. This layer allows to use some specific models created or improved by the batch layer in order to process and analyse data, obtaining results in near to real time requests by operator user.

At the end of both Flink processing stages, data will be stored in the CouchBase cluster (4) and sent as a real-time stream to the Visual Analytics Platform where it is gathered by an Apache Kafka consumer at the back end (6).

The Visual Analytics Platform comprises a back-end application for authentication and data access; and a front-end client (8) where the user can create and customise visualizations of the historical data (7) and the results of the Proteus Hybrid Engine (6).

This is only an overview of the architecture of PROTEUS for a quick understanding. In the following sections the services, integration and deployment will be explained in depth.

3.2 Cluster infrastructure

All the services in charge of storing, loading, transporting and processing data are deployed in a cluster composed of four virtual machines in master-slave architecture

3.2.1 Hardware Architecture

The cluster is composed of one physical machine which is virtualized by VMWare Workstation 10.0.5.52125 in four virtual nodes.

<table>
<thead>
<tr>
<th>Table 1: Cluster Physical Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Physical Machine</td>
</tr>
<tr>
<td>Processor</td>
</tr>
<tr>
<td>RAM</td>
</tr>
<tr>
<td>HDD</td>
</tr>
</tbody>
</table>

The original machine is divided into four virtual nodes with the same hardware virtualized:
Figure 9: Cluster details

<table>
<thead>
<tr>
<th>Virtual Node 1: Master</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Centos 7</td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>12 GB</td>
<td></td>
</tr>
<tr>
<td>HDD</td>
<td>1 TB</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual Node 1: Slave 01</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Centos 7</td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>16 GB</td>
<td></td>
</tr>
<tr>
<td>HDD</td>
<td>1 TB</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster Physical Machine: Slave 02</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Centos 7</td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>16 GB</td>
<td></td>
</tr>
<tr>
<td>HDD</td>
<td>1 TB</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster Physical Machine: Slave 03</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Centos 7</td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>16 GB</td>
<td></td>
</tr>
<tr>
<td>HDD</td>
<td>1 TB</td>
<td></td>
</tr>
</tbody>
</table>
3.2.2 Base Software & Services Architecture: services under Hortonworks data platform and CouchBase

The solution selected to manage the cluster and the services is Hortonworks. The platform is flexible and can be extended with other data processing services. The services and packages deployed and their functionality are described below.

We use Hortonworks Data Platform on its version 2.5. This distribution packages the most typical services of the Hadoop Ecosystem. Among the services included by Hortonworks, we selected to install and keep active on the cluster only those necessary ones to achieve the needs of processes like data ingestion, data storage, data exploitation and management. These services are described in the following sections.

3.2.2.1 Data Ingestion: Apache Kafka

Apache Kafka is a distributed streaming platform which enables publication and subscription to streams of records. It gives the option of storing the records generated by a stream in a fault-tolerant way and it can process the stream of records as they occur. A typical Kafka cluster is composed of the following elements:

- **Brokers**: A Kafka cluster is formed by one to multiple brokers to maintain load balance. These broker instances can handle thousands of reads and writes per second and TB of messages. In the Proteus architecture there are three Kafka brokers located in the Slave 01, Slave 02 and Slave 03.

- **Kafka Producers**: These are the elements responsible for pushing data to the brokers. In the real environment only one coil is produced at a time. In Proteus deployment there will be a single Producer.

- **Kafka Consumer**: is responsible for pulling the data from a topic (a logical collection of queues to which messages will be written and read) with the help of the cluster coordinator. Although it belongs to the Kafka deployment, the Consumer is located in the Visual Analytics Platform, so it will be described in its corresponding section.

All the components of the Kafka cluster are coordinated by one instance of Apache ZooKeeper, another typical service of the Hadoop Ecosystem also included in the deployed Hortonworks Data Platform. ZooKeeper is responsible of coordinating the stateless brokers and for maintaining the cluster state. The architecture of the cluster is shown in Figure 8.

**Figure 10: Kafka producer**

Built using the API of Apache Kafka combined with the HDFS API, the Proteus Producer is a small implementation capable of taking formatted data from the HDFS and generating a simulated streaming of a real production environment. To achieve this, the Producer has one attribute called COIL_SPEED which is...
responsible for setting the time spent to produce one coil, which is by default 120 seconds, similar to the real industry scenario of AMIII.

The way to produce a coil in these 120 seconds is as follows: The producer reads the data of a coil from the HDFS (where all coils data is stored) and store it in a temporary buffer. Coil data is a sequence of sensor measurements. Each sensor measurement consists of an identifier of the observed variable, a value of this variable and the X coordinate of where this variable was read. It will be calculated all the distances between each two X coordinates, which result is multiplied by the result of the division between COIL_SPEED and the total count of sensor measurements of that coil. The final delay is calculated on this way. Once this delay is calculated, the producer is capable of producing all the values for the same X position in the same time.

As it is necessary to manage a big amount of data in real time, the producer is a multithreaded implementation, so it would be possible to manage a bigger number of coils on a production environment.

All the data is sent to one of the Kafka topics which is created with only one partition. Topics are divided according to the production flow settings:

- **Real-time**: manages all the time-series data with 1 dimension (position x) and 2 dimensions (position x and position y), produced and available in real time during the coil production.
- **HSM**: manages all HSM data, produced as aggregated information at the end of the process and available only once the coil production has ended.
- **Flatness**: manages the flatness data variables, produced as measures of the flatness of the resulting coil, and available only after a certain delay after the coil production has finalised.

The data stream is composed of messages which create objects of type Coil.

### Data Storage: HDFS

The Hadoop Distributed File System (HDFS) is a distributed file system that shares many similarities with traditional file systems. The HDFS have features like a highly fault-tolerance and a high throughput access to data, making it the most common choice for applications that have large data sets like the industry scenario subject of Proteus. The HDFS has a master/slave architecture composed of a single NameNode and a variable number of DataNodes.

The NameNode is the master server. Its function is to manage the file system namespace and to regulate the access to files while the DataNodes are responsible for serving read and write requests from the file system’s clients and to perform block creation, deletion and replication upon instruction from the NameNode.

In the Proteus architecture one NameNode exists located in the master node which coexists with one of the DataNodes. In addition, there are a DataNode for each slave node of the cluster. So, the HDFS cluster is composed by one NameNode and four DataNodes deployed with the following architecture.
3.2.2.3 Data Storage: CouchBase

In addition to HDFS, the prototype architecture uses CouchBase for data storage. Couchbase is responsible for storing the historical data in a reliable way. The applications developed with JSON are fully compatible because Couchbase can speak JSON natively and supports binary for all the data storage needs. Also, Couchbase comes with a developer-friendly query language, an optionally MapReduce for simple, efficient and comprehensive data retrieval.

A Couchbase cluster composed of four nodes, hosts the database, which store documents. Each of these documents is uniquely named in the database and it can be read, updated or written thanks a RESTful HTTP API provided by Couchbase. Each of these documents are the primary unit data in Couchbase and consists of any number of fields of varying types (text, number, Boolean, lists, etc.) and attachments with an extra metadata maintained by the database system.

3.2.2.4 Data Exploitation: Apache Hive

The Apache Hive data warehouse software facilitates reading, writing and managing large datasets residing in distributed storage using SQL language, the easiest way for a big majority of users. Apache Hive auto translates the SQL query to MapReduce, and its structure can be projected onto data already in storage. The Hive client provides a command line tool and a JDBC driver to make the queries.

Apache Hive is the selected tool to perform queries about the raw and log data. Apache Hive is capable of making queries against the data stored in the HDFS getting back the results in a low latency time, making it possible to compare them with the results of the processing.
3.2.2.5 Management Services: Apache Ambari and CouchBase Dashboard

The Apache Ambari project aims at making Hadoop management simpler by developing software for provisioning, managing and monitoring Apache Hadoop clusters. Ambari also provides an intuitive, easy-to-use Hadoop management web UI backed by its RESTful APIs.

Hortonworks Dataplatform has been used to deploy and manage cluster services. It includes Ambari as the default management tool. It is possible to make all type of services operations on this way, like changing configurations files, checking services status and logs or monitor resources consumption in the cluster. All of them can be realized from a friendly web interface.

In addition, the Web Console is the main tool for managing the CouchBase environment. It is accessed by navigating to the IP and PORT designated of a node running. From this dashboard it is possible to access to the Couchbase Cluster information, where it is possible to check, for example, RAM and disks usage among others are provided. Besides it is also possible to make queries in the database.

3.2.3 The PROTEUS core contributions

PROTEUS core technology is Apache Flink. It is an open-source framework for distributed stream processing for high-performing, always-available, and accurate data streaming applications. It is capable of providing results that are accurate even in the case of out-of-order or late-arriving data. It is stateful and fault-tolerant and capable of working at large scale and running on thousands of nodes with good throughput and low latency.

Because Flink Engine is capable to work with batch and streaming data, it is the core tool to make all the predictive processing. However the original Flink capabilities are being enhanced in PROTEUS project thanks the including of the consortium developed implementations:

- Firstly, a new Apache Flink version has been implemented as a fork of the original one. This new version is able to perform hybrid processing (batch and streaming in the same operator)
- A Scala-embedded engine-independent DSL called LARA. It allows to write machine learning algorithms in a high-level language.

- The SOLMA library: A set of online machine learning algorithms responsible to make all the operations of the predictive analysis of the data.

Each of these implementations, will be described in the next sections.

![Diagram of the PROTEUS, Flink-based architecture](image)

**Figure 13: The PROTEUS, Flink-based architecture**

### 3.2.3.1 Data Processing: the PROTEUS Hybrid Engine

PROTEUS Hybrid Engine is a fork of Apache Flink. It features an extended support to operations that mix streaming and batch processing. This is achieved through side inputs for stream operators, i.e., an operator processes records coming from the stream as well as preloaded historical data. As a result, side inputs enable hybrid processing within a stream operator.

### 3.2.3.2 Declarative Language: LARA

LARA is an engine-independent declarative language that allows programmers to author complete end-to-end data analysis programs that are automatically parallelized. LARA is based on EMMA, a deeply embedded in Scala domain specific language (DSL) which enables authoring scalable programs using an abstract data types (called DataBag) and control flow constructs. LARA aims to provide a common environment that allows to define pipelines that merge relational and linear algebra. To this end, LARA extends EMMA by introducing a Matrix data type meant for machine learning workloads. Secondly, LARA
enables joint optimizations over both relational and linear algebra as having a single pipeline made of both operations enables holistic optimizations.

LARA has not been added to this prototype at the moment. However it is possible to read about its use and performance in deliverable D3.9 and about its optimizers in deliverable D3.10.

### 3.2.3.3 PROTEUS distributed cache

This component stores temporarily some data (e.g. machine learning algorithm models) which is needed to be cached during machine learning algorithm performance. We trust in two solutions to implement this component. Firstly PEACH was developed to execute this mission (deliverable D3.5). PEACH (PROTEUS Elastic Cache) is a distributed key-value cache that can be used both inside and outside of the PROTEUS scope. The cache aims to provide low latency responses on a distributed elastic deployment with fault-tolerance capabilities. As a generic design, the cache could be integrated within Apache Flink to speed up computing processes. In addition to PEACH, Flink Parameter Server [9] also was used in the project to perform this important mission. It is an open source solution which has helped PROTEUS team in the development of some of online machine learning algorithms.

### 3.2.3.4 Scalable Online Machine Learning Algorithms: SOLMA

SOLMA is a scalable online machine learning library that consists of novel algorithms for classification, clustering, regression, and anomaly and novelty detection. All the algorithms are implemented on top of Apache Flink.

### 3.3 Tuning for the final demonstrator

The evolution of the PROTEUS prototype was explained in deliverables D2.8, D2.9 and D2.10. These deliverables explained each step of that evolution and what goals were reached in respect to the business use cases proposed by AMIII. This final prototype addresses the three business use cases proposed by AMIII and it forms the final demonstration of the PROTEUS project.

All efforts in the last months were dedicated to turning the prototype into a production ready solution able to scale in the most demanding circumstances.

As explained, the final demonstration is a complex distributed architecture formed of a wide variety of components. In this kind of architecture, a final tuning phase is always required. In deliverables D2.8, D2.9 and D2.10, the quality of each of the individual components was shown and explained, however the final solution has to be a complete solution and therefore all of the components have to be integrated together to produce a working system.

Below are the components which form the final demonstration:

- HDFS + Hive
- Kafka cluster
- PROTEUS producer
- PROTEUS consumer
- Couchbase cluster
- Flink cluster
- PROTEUS Flink job (SOLMA, Flink Parameter Server, PEACH)
- PROTEUS dashboard.
The following sections detail how each of these components is configured in the final demonstration to achieve optimal performance.

### 3.3.1 HDFS + Hive

The PROTEUS input dataset is a large number of sensor measurements produced in the AMIII factory. As it has not been possible to work directly with real-time information, AMIII provided a large number of sensor events collected during several months in the real factory. This events collection forms a huge CSV file of 16 GB. The use of a file of this size could be a headache, however we have used HDFS and Hive to help us in this task. HDFS is a distributed, scalable, and portable file system for the Hadoop framework. It allows us to store huge files in a distributed and scalable way.

The processing of files using HDFS is not very efficient, but Hive helped us to make it faster. Hive provided a data query and analysis solution with faster access to HDFS. Hive tables have allowed us to access sensor data in an acceptable time.

### 3.3.2 Kafka cluster

The final demonstration Kafka cluster is composed of 3 nodes. A Kafka broker (a server) is running in each of these nodes. Six Kafka topics are needed in the final demonstration, three of them are input data (proteus-realtime, proteus-flatness and proteus-hsm) and the other ones are output data (simple-moments, sax-results and lasso-results). When we create these topics we have to tune two important parameters:

- **Number of partitions**: This parameter is related to the number of simultaneous consumers attached to the topic. There is no fixed rule about this but we have to realise that tuning the number of partitions involves a trade-off between the throughput and the end-to-end latency. In the case of the topics of the final demonstration we decide to set 6 as the number of partitions.

- **Number of replicas**: This parameter is related to the fault tolerance of our solution. This number represents how many times the information of the topic is replicated. If the topic data is replicated and one of the Kafka broker fails our solution will continue working with the replicated data in other brokers. In the final demonstration topics we decided to use a factor of 2 replicas. Although, if the number of Kafka cluster nodes increase we could increase the number of replicas too and therefore improve the fault tolerance performance of our solution.

### 3.3.3 PROTEUS producer and consumer

The PROTEUS producer was built during the project development to simulate the real-time production of sensor events in the factory and the consumer was built to store historical data in a Couchbase database. They are very simple components and only need to be configured during the Kafka cluster configuration. The coding of the producer has been necessary to simulate the real-time producing of the sensor events of the factory. The coding of the consumer has been necessary to store historical data in Couchbase database. They are very simple components and only need to be configured in tune with Kafka cluster configuration.

### 3.3.4 Couchbase cluster

The final demonstration has a Couchbase cluster composed of 3 nodes. The Couchbase cluster stores all the historical data collected by the PROTEUS consumer (detailed above). We only tune one parameter in the cluster, the per node RAM Quota. Currently it is set to 12,000 MB.

### 3.3.5 Flink cluster

The big data processing in the final demonstration is principally based on Apache Flink. This is why the Flink cluster is one of the main pieces in the architecture. During PROTEUS project development an
overhauled Apache Flink version has been coded. This PROTEUS Flink version is included in the final demonstration. Apache Flink clusters are composed of two kinds of components: job managers and task managers. The final prototype is composed of one job manager and 3 task managers. There are two important issues to consider when tuning the cluster performance:

- **Number of task slots:** The number of parallel operator or user function instances that a single TaskManager can run. This value is typically proportional to the number of physical CPU cores that the TaskManager’s machine has. In this case we have used 4 machines with 2 CPUs per machine and therefore this parameter has been set to 2.

- **Task manager heap size:** This parameter is related to the memory which needs the Flink jobs executed in the cluster and, of course, with the available memory in the task managers. It has been set to 2048 MB in the final demonstrator which is enough to execute the three business use cases.

To get more details about how Flink parameter server works and how it is used in SOLMA library it is necessary to read deliverable D4.5

### 3.3.6 PROTEUS Flink job

This is the core component of the final demonstrator. It includes the implementation of the necessary algorithms to implement two of the business use cases (use cases 2 and 3). The implementation is based on SOLMA which is the scalable online machine learning developed in PROTEUS project. The parameters which need to be tuned are related to two issues:

- **Flink checkpointing:** Checkpoints allow Flink to recover state and stream positions to give the application the same semantics as a failure-free execution. In respect to this, there are two important parameters to configure:
  - Flink checkpoints interval
  - Flink checkpoints directory (HDFS)

- **Lasso algorithm tuning:**
  - Number of Lasso workers: Parallelism in the Lasso implementation. It is set to 4 in the final demonstration.
  - Number of parameter server nodes: Parallelism in the Flink parameter server used in Lasso implementation. It is set to 1 in the final demonstration.
  - Parameter server iteration wait time: It represents the maximum time to wait in an iteration of Flink parameter server. It is set to a large amount of time, concretely 86400000 seconds in the final demonstrator.
  - Parameter server pull limit: It represents the maximum pull operations which can be enqueue to be dispatched. It is set to 1000 in the final demonstration.
  - Flatness allowed lateness: It represents the maximum time to wait Flatness events of each coil. It is set to 5 seconds in the final demonstration.
  - Realtime allowed lateness: It represents the maximum time to wait Flatness events of each coil. It is set to 5 seconds in the final demonstrator.

### 3.3.7 PROTEUS dashboard

The PROTEUS dashboard is the face of the final demonstrator. As explained in deliverables D2.8, D2.9 and D2.10, PROTEUS dashboard is a web based component used to show the data analysis results processed in
the final demonstrator. From the perspective of tuning, the final solution is very simple and it doesn’t require special configuration.

### 3.4 Visual Analytics Platform

The analytics platform is designed as standalone software and can be run on commodity hardware and plugged to the processing engine and historical database, running on a cluster. Its user interface features a dashboard that provides querying, monitoring and visualization of both data-at-rest from the data storage layer and data-in-motion streaming from the data processing engine.

This analytics platform is implemented as a client-server architecture comprising a back-end server that provides user accounts and authentication; and a multi-platform client that can be used both as a web or a desktop application.

#### 3.4.1 Backend

The backend for the visual analytics infrastructure is implemented as a Java EE application, built on top of the Spring Framework and deployed in an Apache Tomcat Server. It provides user authentication, querying for the historical database using Spring Data and N1QL code in order to retrieve data from CouchBase server, and a Kafka consumer to collect the messages from the processing engine.

#### 3.4.1.1 User authentication and security

The visual analytics platform incorporates the Spring Security framework to preserve confidentiality of the Proteus dataset. This framework provides customizable authentication, authorization and protection against attacks like session fixation, clickjacking, cross site request forgery, etc.

#### 3.4.1.2 Kafka Consumer

As highlighted in Sec 3.3.2, data is read by a Kafka consumer before it is processed by the visualisation modules.

#### 3.4.1.3 Database connection

The visual analytics platform is able to query historical data stored in the CouchBase server using N1QL (declarative query language that extends SQL for JSON documents), but it could be connected to other kinds of databases for other use cases, both relational and non-relational. To enable these queries, the backend employs Spring Data, which provides a common API for the data access layer to abstract the programming model from the underlying data store.

#### 3.4.1.4Datasources API

The datasources API allows bidirectional communication of the visualization at the client side of the platform with the server side. This API can be used from the client-side to list all the available data sources, either batch or streaming, and receive data through various protocols, namely WebSocket, HTTP or HTTP/2. It can be extended to enable other protocols in the future.
3.4.2  Front-end

3.4.2.1  Proteus Dashboard

Proteus Dashboard is the user interface for the visual analytics platform which has already been described in section 2.2.1 and section 2.3. It is a cross-platform application that can run either as a web application or a desktop application by means of the Electron framework. The application allows the user to access a dashboard through a login screen where he can create and customize Proteic visualizations that can then be connected to a set of data sources in the back-end of the visual analytics platform. The dashboard itself is based in the ng2-admin framework, built with cutting edge, open source technologies like Angular, Bootstrap 4, Webpack and Electron.

3.4.2.2  Proteic

Proteic is the data visualization library for the web created for the interactive visualization of the Proteus data and the results of the incremental analytics engine. It is integrated in the Proteus Dashboard to display data retrieved from the back-end of the visual analytics platform. The library features a convenient declarative API for the creation of multiple kinds of visualizations suited for high volumes of both streaming and static data. Its API allows the retrieval of data over the various communication protocols available in the Datasources API of the back-end component.

This component is built and distributed as a standalone open source library that can be used in any data visualization scenario for the web. It is implemented in the TypeScript language, and follows the latest standards and good practices in web development to achieve high performance, compatibility and ease of development. The design process of the library accounts for usability and accessibility of the various visualizations, following design guidelines like responsive web design and offering alternative colour palettes for colour blind-users.

The Proteic library brings the following features to the visual analytics platform:

- Time series visualizations using dynamic and interactive line charts, stream graphs, and area charts. These charts include a set of tools for time series analysis such as dynamic thresholds, outlier alerts based on thresholds, dynamic error or confidence bands and pause / resume capabilities.
- User interaction features like brushing, zoom, highlighting, pause / resume; following the Shneiderman’s mantra: Overview first, zoom and filter, and details on demand.
- Usability and accessibility features, such as interactive legends, annotations, accessible colour palettes, customizable themes, error warnings and throbber (an animated graphical control element used to show that a computer program is performing an action in the background) for loading data sources.
- High-dimensional geometry and multivariate data visualization by means of parallel coordinates visualization, including features for visualization of missing data.
- Innovative real-time visualizations, namely real-time heatmaps and swimlane charts.

3.5  Setup

3.5.1  Required packages before installation

Just before deploying the Hortonworks Data Platform in the virtual nodes, it is necessary to check that the following software packages are installed in each of the virtual nodes:

- Yum
- Rpm
- Scp
- Curl
• Tar
• Wget
• Open SSL (v1.01, build 16 or later)
• Python 2.7

Also, all nodes must be in same network in order to communicate among themselves.

3.5.2 Operating System

The operating system in each of the nodes is Centos 7. By default, Centos 7 uses Chrony as synchronization service, which will be uninstalled by NTP synchronization.

3.5.3 Server and Agents

3.5.3.1 Ambari Server

The Ambari Server will be installed through yum. To achieve the deployment, it will be necessary download and install the repository from the Ambari Server will be downloaded:

```bash
wget -nv
```

After the properly check, the installation of Ambari-Server can be made:

```bash
yum install ambari-server
```

Before launch Ambari-Server, it is necessary make the setup:

```bash
ambari-server setup
```

In this setup, it will be checked the next points:

• SELinux it will be necessary to be deactivated.
• Ambari-Server will ask if the user want to “Customize user account for ambari-server daemon”. The answer is no, by the way the installation is made as root user.
• The iptables it will be necessary to be deactivated.
• The version of JDK to install, 1.8 will be the selected choice and it will be necessary to accept the user contract.
• When the Ambari-Server will ask by a database to store the server states, it will be selected the default option, where Ambari-Server uses the PostgreSQL by defect.

When the setup is ended, the Ambari-Server can be launched and checked.

```bash
ambari-server start
ambari-server status
```

3.5.3.2 Ambari Agents

With the Ambari-Server running properly, is moment to install the Ambari-Agents, which will be need to be installed manually. The first step consist in download the official repository from Hortonworks, like it was made in the Ambari-Server in the three nodes.
After check than the repository is correctly installed, it can be deployed the Ambari-Agent

```
yum install ambari-agent
```

When the agent finishes the installation, it will be needed to edit the next file

```
File: /etc/ambari-agent/conf/ambari-agent.ini

[server]
hostname=<FQDN_Selected_for_Master>
url_port=8440
secured_url_port=8441

[agent]
logdir=/var/log/ambari-agent
piddir=/var/run/ambari-agent
prefix=/var/lib/ambari-agent/data
:loglevel=(DEBUG/INFO)
loglevel=INFO
data_cleanup_interval=86400
data_cleanup_max_age=2592000
data_cleanup_max_size_MB = 100
ping_port=8670
cache_dir=/var/lib/ambari-agent/cache
tolerate_download_failures=true
run_as_user=root
parallel_execution=0
alert_grace_period=5
alert_kinit_timeout=14400000
system_resource_overrides=/etc/resource_overrides
    ; memory_threshold_soft_mb=400
    ; memory_threshold_hard_mb=1000
[security]
keysdir=/var/lib/ambari-agent/keys
server_crt=ca.crt
passphrase_env_var_name=AMBARI_PASSPHRASE
ssl_verify_cert=0
[services]
pidLookupPath=/var/run/
```
Finally, ambari agents need to be started in all the machines

```
ambati-agent start
```

### 3.5.4 Hortonworks Data Platform deployment

If the previous setup ends properly, a web interface will be showed in the FQDN selected for the Master in the port 8080 by default, for example:

```
cluster.master.local:8080
```

The web interface corresponds to Apache Ambari management tool, where will be possible to select the option “Create a Cluster” and follow all the steps defined by the “Install Wizard”. The principal decisions to take are the next options:

1. Select the Hortonworks Data Platform version. In this case, the choice is Hortonworks Data Platform 2.5.3.0 with the public repository.
2. The next step consists in define all the FQDN of each node who compose the cluster. One name by line and with the option of manual register for the Ambari agents. Then the agents will be registered against the server and the necessary checks will be made, to check than the process ends correctly.
3. Finally it will be needed to select all the services to deploy in the cluster. They are:
   - HDFS
   - Yarn + MapReduce 2
   - Apache Hive
   - Kafka

### 3.5.5 Proteus Environment

#### 3.5.5.1 Flink Hybrid Engine

First step is to download the official Flink Hybrid Engine DFKI repository on master node:

```
git clone https://github.com/proteus-h2020/proteus-engine.git
```

After that, it requires to be compiled and make and packaged:

```
cd flink
mvn clean package -DskipTests
```

Finally, on Build-Target folder, Hybrid Engine is installed

#### 3.5.5.2 PEACH

Peach works over Redis database, so first of all Redis would be deployed in PEACH node:

```
wget http://download.redis.io/releases/redis-<your_version>.tar.gz

tar xzf redis-<your_version>.tar.gz

cd redis-<your_version>
```
1. Compile the project to obtain the executable scripts:

```bash
mvn install –DskipTests
```

2. Move to the target directory to launch the PeachServer:

```bash
cd peach-redis-server-dist/target/peach-redis-server-dist-*/
```

3. Launch Peach with the Redis backend. Notice that Redis must be up and running

```bash
./bin/peach-redis-server-app
```

### 3.5.5.3 SOLMA

In order to use SOLMA library, in every maven project it is needed to add attached code inside pom.xml file:

```xml
<dependencies>
  <dependency>
    <groupId>eu.proteus</groupId>
    <artifactId>proteus-solma_2.10</artifactId>
    <version>0.1-SNAPSHOT</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-clients_2.10</artifactId>
    <version>1.4-SNAPSHOT</version>
    <scope>compile</scope>
    <exclusions>
      <exclusion>
        <groupId>log4j</groupId>
        <artifactId>*</artifactId>
      </exclusion>
      <exclusion>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
</dependencies>
```
<dependencies>
</dependencies>

<build>

<plugins>

<plugin>
    <groupId>net.alchim31.maven</groupId>
    <artifactId>scala-maven-plugin</artifactId>
    <configuration>
        <recompileMode>incremental</recompileMode>
    </configuration>
    <executions>
        <execution>
            <id>scala-compile-first</id>
            <phase>process-resources</phase>
            <goals>
                <goal>add-source</goal>
                <goal>compile</goal>
            </goals>
        </execution>
        <execution>
            <id>scala-test-compile</id>
            <phase>process-test-resources</phase>
            <goals>
                <goal>add-source</goal>
                <goal>testCompile</goal>
            </goals>
        </execution>
    </executions>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-assembly-plugin</artifactId>
    <version>2.4</version>
    <configuration>
        <descriptorRefs>
            <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
    </configuration>
</plugin>

</plugins>
</build>
<mainClass>solma.TestSolma</mainClass>

</manifest>
</archive>
</configuration>
<executions>
<execution>
<phase>package</phase>
<goals>
<goal>single</goal>
</goals>
</execution>
</executions>
</plugin>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-assembly-plugin</artifactId>
<executions>
<execution>
<id>copy-dependencies</id>
<phase>package</phase>
<goals>
<goal>copy-dependencies</goal>
</goals>
<configuration>
<useBaseVersion>false</useBaseVersion>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>

3.5.6 Additional Storage System: Couchbase

Couchbase installation requires next steps:

1. Disable THP (Transparent Huge Pages). THP feature of the Linux kernel must be disabled on systems running Couchbase Server. Transparent Huge Pages (THP) is a Linux OS feature that conceals much of the complexity of using actual Huge Pages as well as automates the creation of contiguous memory space. It is enabled by default in some Linux Operating systems, but not all.

2. Download official rpm package in all machines:
3. Install rpm package:

```bash
sudo rpm -i couchbase-release-1.0-2-x86_64.rpm
sudo yum update
sudo yum install couchbase-server
```

4. After that, access via dashboard manage interface on port 8091 in master node.

5. Check in “Start new cluster”

6. In port 8091 for all slave nodes, select “join a cluster now” option. Select FQDN master node and user name with password configured in that server.

3.5.7 Visual Analytics Infrastructure

The visual analytics platform is distributed as a standalone WAR file that contains both the front and back ends of the web application alongside all its dependencies. It requires Java 1.7 and Apache Tomcat 7 and it should be deployed by placing the WAR file in the webapps directory of the Tomcat 7 installation directory. The application requires setting some configuration values before running in a production environment and it is shipped with a default settings file that contains both the configuration values and its documentation. Its location can be passed to the application by setting the SPRING_CONFIG_LOCATION environment variable.

3.5.8 Setup Summary

The table provides a summary of the different technologies and versions used for the setup of the prototype:

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Technology</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop Ecosystem under Hortonworks Data Platform</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hortonworks Data Platform</td>
<td>2.5.3.0</td>
<td></td>
</tr>
<tr>
<td>HDFS</td>
<td>2.7.3</td>
<td></td>
</tr>
<tr>
<td>NameNode</td>
<td>Master</td>
<td></td>
</tr>
<tr>
<td>Secondary NameNode</td>
<td>Slave 01</td>
<td></td>
</tr>
<tr>
<td>DataNode</td>
<td>Master, Slave 01, Slave 02, Slave 03</td>
<td></td>
</tr>
<tr>
<td>NFSGateway</td>
<td>Slave 01, Slave 02, Slave 03</td>
<td></td>
</tr>
<tr>
<td>YARN + MapReduce2</td>
<td>2.7.3</td>
<td></td>
</tr>
<tr>
<td>ResourceManager</td>
<td>Master</td>
<td></td>
</tr>
<tr>
<td>App Timeline Server</td>
<td>Master</td>
<td></td>
</tr>
<tr>
<td>NodeManagers</td>
<td>Slave 01, Slave 02, Slave 03</td>
<td></td>
</tr>
<tr>
<td>YARN Clients</td>
<td>Master, Slave 01, Slave 02, Slave 03</td>
<td></td>
</tr>
<tr>
<td>Tez</td>
<td>0.7.0</td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>Hive</td>
<td>1.2.1</td>
<td></td>
</tr>
<tr>
<td>Hive Metastore</td>
<td>Slave 01</td>
<td></td>
</tr>
<tr>
<td>HiveServer 2</td>
<td>Slave 03</td>
<td></td>
</tr>
<tr>
<td>MySQL Server</td>
<td>Slave 03</td>
<td></td>
</tr>
<tr>
<td>WebHCat Server</td>
<td>Slave 03</td>
<td></td>
</tr>
<tr>
<td>Hive Clients</td>
<td>Master, Slave 01, Slave 02, Slave 03</td>
<td></td>
</tr>
<tr>
<td>HCat Clients</td>
<td>Master, Slave 01, Slave 02, Slave 03</td>
<td></td>
</tr>
<tr>
<td>Oozie</td>
<td>4.2.0</td>
<td></td>
</tr>
<tr>
<td>Oozie Server</td>
<td>Slave 01</td>
<td></td>
</tr>
<tr>
<td>Oozie Clientes</td>
<td>Slave 01, Slave 02, Slave 03</td>
<td></td>
</tr>
<tr>
<td>Zookeeper</td>
<td>3.4.6</td>
<td></td>
</tr>
<tr>
<td>Nodes</td>
<td>Master, Slave 01, Slave 02, Slave 03</td>
<td></td>
</tr>
<tr>
<td>Ambari Infra + Ambari Metrics</td>
<td>0.1.0</td>
<td></td>
</tr>
<tr>
<td>Infra Solr Instance</td>
<td>Master</td>
<td></td>
</tr>
<tr>
<td>Infra Solr Clients</td>
<td>Master, Slave 01, Slave 02, Slave 03</td>
<td></td>
</tr>
<tr>
<td>Kafka</td>
<td>0.10.0</td>
<td></td>
</tr>
<tr>
<td>Kafka Brokers</td>
<td>Master, Slave 01, Slave 02, Slave 03</td>
<td></td>
</tr>
<tr>
<td>External Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache Flink - Hybrid Engine</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>PEACH</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>SOLMA</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>DLS Language</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Couchbase</td>
<td>4.5.1-2844 Community Edition (build-2844)</td>
<td></td>
</tr>
<tr>
<td>Couchbase servers</td>
<td>Slave01, slave02, slave03</td>
<td></td>
</tr>
</tbody>
</table>

**Visual Analytics Infrastructure**

- Proteus Dashboard

**PROTEIC.JS**
4. Validations & KPIs

The identification of Key Performance Indicators (KPIs) within PROTEUS was an ongoing, iterative process that commenced with the work in Deliverable 2.6. In D 2.6, an initial set of KPIs was provided for measuring the performance of any future PROTEUS solution, within the parameters in which the project had progressed up to that point. This section provides a snapshot illustration of the final set of KPIs selected by the consortium. This will be used in the final impact assessment of PROTEUS, D 6.2, given the development stage of the project and the solution.

4.1 KPI review

In the final stage of the project, three technology components were finalised, namely: data process; visualisation; and online machine learning components. In addition, partners developed an industrial value KPI to assess the extent to which PROTEUS provided added-value to ArcelorMittal. For each one of these components’ partners selected the following KPIs:

Online Machine learning:
- True positive rate
- True negative rate
- False positive rate
- False negative rate
- Precision
- Recall
- Accuracy of flatness
- Coefficient of Determination ($R^2$)
- Algorithmic Complexity, e.g. O(n)
- Algorithmic Innovation, e.g. new algorithms surpassing Flink-ML state-of-the-art

Data Visualization:
- Number of functions e.g. zoom and notifications
- Number of plots that can be produced by PROTEIC
- Fulfilment of end users’ needs

Data process:
- Velocity
  - Throughput
  - Latency
  - Generation Rate
  - Velocity of algorithm
- Engine and abstraction innovation

Industrial value
- Process improvement KPI.

Data visualisation KPIs help assess the first business use case, Control of the key process parameters, as elucidated in section 2.2.1 above. Online machine learning KPIs help assess the second and third business use cases, Better quality of the flatness of the products and Better quality of the flatness and process, as illustrated in section 2.2.2 and 2.2.3 respectively.

In what follows, we recap the KPIs selected and assess the value of the final prototype presented in this deliverable against them.
4.2 PROTEUS components KPIs

The next set of KPIs reflect the technology components developed by PROTEUS, i.e. online machine learning, data visualisation and data process components. Below present KPIs related to these components.

4.2.1 Online Machine Learning KPIs

In this deliverable, D2.11, we report on algorithms for classifying points on the surface of a coil as flat or not. This latest development follows the collection of some new information from an outside company that shared their business logic for such a classification. If the steel coil exceeds 60 I-Units (in the middle zone which discards the head and tail parts of the coil which are not representative for the flatness) it is deemed to not be flat and thus defective for their purposes. We have thus used this value as a threshold for classifying the steel. The full details are reported in D4.4, but it suffices to say here that the algorithm itself is an innovative online weighted averaging passive-aggressive (WAPA) algorithm. More precisely, two variants of this model were explored, one in which a smoothing parameter is adjusted and tuned, which is referred to as DWAPA and one in which the smoothing parameter is exclusively set to 1, which is referred to simply as a distributed passive-aggressive (DPA) algorithm. Both these classes of model were further generalised by experimenting with the inclusion of another penalty term that was either linear (I) or quadratic (II). Experiments were performed using the PROTEUS data. As this was a classification problem, the appropriate evaluation metrics are the accuracy, precision and recall. A comparison of the accuracies for all model variants is shown below. The higher accuracy metric in the table shows that the DWAPA models outperform the DPA models when predicting flatness.

<table>
<thead>
<tr>
<th>Data</th>
<th>DWAPA</th>
<th>DWAPA-I</th>
<th>DWAPA-II</th>
<th>DPA</th>
<th>DPA-I</th>
<th>DPA-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>298coils</td>
<td>99.10</td>
<td>99.10</td>
<td>99.10</td>
<td>98.56</td>
<td>98.56</td>
<td>98.56</td>
</tr>
</tbody>
</table>

The accuracy metric alone however, is not an ideal indicator of performance owing to the significant class imbalance in the dataset. In fact, there were 22326 flat positions while there were only 213 non-flat positions, with the majority of them near the beginning of the coil. Consequently, simply predicting all positions as flat would return an accuracy of 99.055%. Nonetheless, the DWAPA models showed an improvement by predicting 22338 positions correctly giving an accuracy of 99.10%. To explain the KPI assessment in a more useful format we give the confusion matrix for this experiment below, where 1 indicates flat and -1 indicates non-flat:

<table>
<thead>
<tr>
<th>Label: 1</th>
<th>Label: -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted: 1</td>
<td>22317</td>
</tr>
<tr>
<td>Predicted: -1</td>
<td>9</td>
</tr>
</tbody>
</table>

From the confusion matrix, we can also report the following conventional metrics for classification problems:

<table>
<thead>
<tr>
<th>True +ve Rate (Recall)</th>
<th>True -ve Rate</th>
<th>False +ve Rate</th>
<th>False –ve Rate</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.999596883</td>
<td>0.098591549</td>
<td>0.90140845</td>
<td>0.000403117</td>
<td>0.525825397</td>
</tr>
</tbody>
</table>

From this table we see that although the true positive rate is high, the true negative rate is actually quite low and only around 10% of the non-flat positions are currently being predicted as non-flat, similarly the precision is not as high as we would hope owing to the high false positive rate. The data sets used for these experiments was saved and used to test other algorithms. Furthermore, while we predicted the flatness of positions on the coil, we also classified entire coils as flat or otherwise by defining a non-flat coil as a coil in which any position exceeds the flatness threshold of 60 I-Units in the central zone which discards the beginning and end of the coil which are often non-flat.

In D2.8 and D2.9 we also identified the following KPIs relating to data analysis:
Accuracy of flatness, Coefficient of Determination ($R^2$) and Algorithmic Complexity

Accuracy of the prediction models based on real-time variables such as tension and temperature and the flatness of previous coils.

A new algorithm, ONLMSR, has been integrated into SOLMA; this can be deployed in the final PROTEUS prototype. Below we show comparison between LASSO and ONLMSR algorithms, based on $R^2$, RMSE and MAE. ONLMSR provides a significant improvement in terms of accuracy and time. LASSO time complexity is $O(n^2)$. In contrast, ONLMSR time complexity is $O(n)$. The accuracy-of-flatness comparison is as follows:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RMSE</th>
<th>$R^2$</th>
<th>MAE</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASSO</td>
<td>24.48</td>
<td>0.09</td>
<td>9.51</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>ONLMSR</td>
<td>20.58</td>
<td>0.4</td>
<td>4.37</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

ONLMSR outperforms LASSO as shown by the $R^2$ statistics and the plot above. The plot shows the difference of the cumulative loss between ONLMSR and LASSO. Negative differences on the Y axis indicate that ONLMSR outperforms LASSO for the entire stream. In the context of the second business use case, clearly the ONLMSR algorithm performs better than the LASSO algorithm in accurately predicting flatness, thus better supporting ArcelorMittal in the improvement of the quality of its products.

**Algorithmic innovation KPI:** This is a qualitative KPI which assesses the extent to which PROTEUS contributed to innovate and expand upon FlinkML. This was discussed in greater detail in section 3.3 above. The following stream-data-specific components of SOLMA, were developed on top of Apache-Flink: Transformers (Reservoir samplers; Online PCA; Random Projection; SVD); Estimators (Moments; Heavy
hitters); Predictors (VHT). Using these stream-data-specific components, online Lasso and SAX algorithms for online learning were built on the SOLMA abstraction. These represent a great innovation as Flink-ML specialises solely on offline learning, while SOLMA is a library entirely built for online machine learning.

### 4.2.2 Data visualization KPIs

In this section we introduce updated KPIs for the visualization components of PROTEUS. To evaluate the performance of PROTEIC, the data visualization library, we have defined a set of performance indicators which we briefly evaluate here and assess more in depth in D 6.2.

**Number of functions.** These include: zoom, an alarm triggering system, and a window signaling errors.

**Types of plots that can be produced.** This indicates the number of different plots that can be produced by PROTEIC. PROTEIC can produce over 28 different plots. Some plots are shown below.

![Figure 15 Scatter plot example in PROTEIC (Iris dataset)](image)

Figure 2 shows what the operator sees when she/he pauses the online signal by pressing the pause button showed in section 2.2.1 above. This allows the operator to analyse in depth a particular part of the signal. These plots show that PROTEIC successfully helps HSM operators having full control over the key process parameters, thus successfully implementing the first business use case.

**Fulfilment of end users’ needs.** This KPI measures the extent to which the data visualisation platform met the end users’ needs. ArcelorMittal was a key contributor to the open-source PROTEIC library. ArcelorMittal worked very closely with Treelogic advising on the variables and the visual components to include in this tool (see section 2.3 above). For instance, following ArcelorMittal’s recommendations, Treelogic included in the visualisation dashboard a heating map, an alarm triggering system, and a window signaling errors. The final version of PROTEIC is thus the result of continuous interactions between Treelogic and ArcelorMittal and reflects the industrial expertise of the latter. PROTEIC thus brings together statistical/AI techniques for anomaly detection, i.e. the algorithms of the SOLMA library, and AMIII’s subject matter expertise (SME). PROTEUS, through its open-source nature, has thus contributed to advanced knowledge within the sector, providing a freely available template, PROTEIC, which other manufacturers can deploy in various industries. As a matter of fact, because of the way in which PROTEIC gathers coil data and displays coil variables, any manufacturer producing coils can potentially reuse this dashboard. PROTEIC thus provides the basis for further research on innovative ways to integrate AI expertise and industrial know-how in the smart manufacturing sector.
4.2.3 Data process KPIs

The final set of KPIs enables PROTEUS to demonstrate that the project solution represents improvements in big data analytics in general. This was the most challenging part of the development of the KPIs because of the relative immaturity of the field. The KPIs include indicators of system performance and the extent to which the tools can handle large, fast moving heterogeneous data.

Throughput and Latency relating to Flink: As clarified in D2.9, partners benchmarked the performance of windowed aggregation and join on Apache Flink, as core components of PROTEUS. On a 2-nodes cluster, partners found the average latency to be 0.5 and 4.3 seconds for windowed aggregation and join queries, respectively. The average throughput was 1.2 and 0.85 million tuples per second for windowed aggregation and join queries, respectively.

Latency of PROTEUS hybrid engine: In D3.9 experiments were performed to measure the latency of PROTEUS’ hybrid engine at maximum throughput. For full details the reader is referred to D3.9 but here it suffices to say that millisecond latencies were achieved while latencies of the order of seconds are generally deemed acceptable. Latency measurements are summarized in the table below. Latency reduces slightly when training data increases and more noticeably when more nodes are used.

<table>
<thead>
<tr>
<th></th>
<th>20% training data</th>
<th>40% training data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 nodes</td>
<td>4 nodes</td>
</tr>
<tr>
<td>End-to-end average latency (seconds)</td>
<td>0.69</td>
<td>0.54</td>
</tr>
<tr>
<td>Hybrid operator average latency (seconds)</td>
<td>0.81</td>
<td>0.67</td>
</tr>
</tbody>
</table>

Engine and abstraction innovation. Further innovations include a hybrid-computation model based on Apache-Flink and capable of processing data-at-rest and data-in-motion simultaneously, and LARA, a high-level language for online machine learning algorithms.

4.3 Industrial Value

Below we present a KPI computed to conduct a use case evaluation considering the impact of PROTEUS on ArcelorMittal’s industrial process. The KPI assesses the extent to which the PROTEUS solution provided added-value to ArcelorMittal through improving the coil production process.

4.3.1 Process improvement KPI

In this section, we report on an important process KPI which determines the economic benefit a predictive classifier would introduce to the steel manufacturing process, in comparison to the default scenario in which all coils are assumed to be high quality, flat and thus not defective. Although we do not install PROTEUS in the factory in this project, it is important to note that the metric for evaluating success is available for future reference.

The predictive classifier aims to identify non-flat coils early on so that they can be downgraded and assigned to a new use earlier than they would be otherwise. This will save the company the extra processing and handling costs associated with trying to mistakenly market defective coils as high quality only for them to be sent back. The algorithms defined in sections 2.2.2 and 2.2.3 would perform this function in a fully deployed version of PROTEUS.

If a coil is genuinely flat and the classifier predicts it to be so (true positive, TP), it would be handled (i.e. further processed, marketed, shipped and sold) as if it is high quality and the company will make the monetary value $V_{high}$. If a coil is genuinely non-flat and the classifier predicts it to be so (true negative, TN), it would be handled as if it is low quality and the company will make the monetary value $V_{low}$, where $V_{high} > V_{low}$.

If a coil is genuinely flat but the classifier predicts it to be non-flat (false negative, FN), it would be handled as if it is low quality and the company will make the monetary value $V_{low}$. If a coil is genuinely non-flat but the classifier predicts it to be flat (false positive, FP), it would be handled as if it is high quality only for it to
be sent back, downgraded and resold as low quality, and thus the company will make the monetary value $V_{low} - V_{process}$, where $V_{process}$ is the monetary value associated with the wasted additional processing and mishandling of the coil. This last scenario is the worst possible outcome.

If the factory operates in full accordance with the predictions of the classifier, the total monetary value made, $V_{classifier}$, will be:

$$V_{classifier} = TP \cdot V_{high} + TN \cdot V_{low} + FN \cdot V_{low} + FP \cdot (V_{low} - V_{process})P \cdot V_{high} + N \cdot V_{low} - FN \cdot \Delta V - FP \cdot V_{process}$$

Where $P = TP + FN$ is the total number of positive (flat) coils, $N = TN + FP$ is the total number of negative (non-flat) coils and $\Delta V = (V_{high} - V_{low})$ is the monetary value difference between the high and low quality coils. It is evident that FP introduce a larger penalty in value than FN when $V_{process} > \Delta V$. However, if the classifier worked perfectly such that $FP = FN = 0$, then the monetary value made would be:

$$V_{PerfectClassifier} = P \cdot V_{high} + N \cdot V_{low}$$

Alternatively, if the classifier simply predicted every coil to be flat such that $FP = N, FN = 0$, then the monetary value made would be:

$$V_{Pred.AllFlat} = P \cdot V_{high} + N \cdot (V_{low} - V_{process})$$

The condition set out in equation (3) represents the default current operation mode of the factory in that the absence of a classifier is equivalent to the assumption that all coils are flat at the section of the operation and so they continue to be processed accordingly right up until the final checks prior to dispatching to customers. It should be noted though, that at the very final checks for flatness and quality the company will operate a conservative approach when uncertain to the flatness so as to avoid sending out coils of insufficient quality but that is a separate judgement made much later to the early judgement made by the classifier that is the focus of this analysis.

Consequently it is possible to establish an upper bound on the value added by machine learning in the factory by comparing a classifier that works perfectly to the assumption that all coils are flat:

$$V_{PerfectClassifier} - V_{Pred.AllFlat} = N \cdot V_{process}$$

However, the value added by a general, possibly imperfect, classifier relative to one which predicts all coils to be flat, would be:

$$V_{Classifier} - V_{Pred.AllFlat} = TN \cdot V_{process} - FN \cdot \Delta V \cdot N \cdot V_{process} - FP \cdot V_{process} - FN \cdot \Delta V$$

Thus, a general classifier adds value if and only if:

$$\frac{FN}{TN} < \frac{V_{process}}{\Delta V}$$

Or alternatively in terms of the false omission rate $F. O. R$, value is added if and only if:

$$F. O. R = \frac{FN}{TN + FN} < \frac{1}{1 + \frac{\Delta V}{V_{process}}}$$

Finally, it is worth noting that it is often possible for a classifier to reduce FP at the expense of FN, or vice versa, by tuning a decision threshold parameter $\theta$. In which case, to minimise costs and maximise value one defines the optimal decision threshold parameter as:

$$\theta_{optimal} = \arg\min_\theta FP(\theta) \cdot V_{process} + FN(\theta) \cdot \Delta V$$
4.4 Summary

This section described the KPIs relevant to the final version of the prototype described herein. They will be discussed more in depth, together with other KPIs relevant to other deliverables, in the final impact assessment deliverable, D6.2.
5 Conclusions

This deliverable has detailed the final status of the demonstrator which has been accomplished since prototype 1. The advances of the final prototype are devoted to address the three industrial uses cases proposed by ArcelorMittal (section 2.1) with the aim of making an early detection of surface defects.

The main novelties come in terms of finishing some pending issues, both in data modelling and software implementation, and in the part of KPI definition. Another novelty is the pilot evaluation section, which shares the main pros and cons that have been reached under a business point of view. Special mention to the expert consultation, where a HSM worker has decided to evaluate the visualization tool.

Besides integrating the technology components produced as a result of the research done in the context of various WPs (WP3, WP4 and WP5), this prototype has been more focused on the integration of machine learning algorithms into SOLMA and enhancing the visualization tool to meet the requirements of the use cases.
6 References


