

## Deliverable 5.3: Workbook for the application of Big Data technologies

### Project information

Project title	Enhanced process stability and product quality in steel production by exploitation of break-through technologies for real-time monitoring, control and forecasting inspired by Big Data concepts
Project acronym	NewTech4Steel
Project call	RFCS 2017
Grant number	800677 RFCS-2017
Project duration	1.6.2018 - 31.05.2022 (48 months)

### Content

1.	Introduction and Overview about this workbook .....	1
2.	Use cases in NewTech4Steel .....	1
2.1.	Hot rolling at SZFG .....	2
2.1.1.	General tools for system organisation, administration and supervision .....	3
2.1.2.	Development systems (IDEs) .....	3
2.1.3.	Data ingestion .....	7
2.1.4.	Data storage .....	7
2.1.5.	Data transmission / streaming .....	8
2.1.6.	Data processing - analysis .....	11
2.1.7.	Data Processing - modelling .....	13
2.1.8.	Application for online visualisation .....	14
2.2.	Cold rolling at Marcegaglia .....	16
2.2.1.	General tools for system organisation, administration and supervision .....	19
2.2.2.	Development systems (IDEs) .....	19
2.2.3.	Data ingestion .....	20
2.2.4.	Data storage .....	21
2.2.5.	Data transmission / streaming .....	22
2.2.6.	Data processing - analysis .....	24
2.2.7.	Data Processing – modelling .....	26
2.2.8.	Application for online visualisation .....	28
2.3.	Wire rod production at JSW .....	29
2.3.1.	General tools for system organisation, administration and supervision .....	32
2.3.2.	Development systems (IDEs) .....	44
2.3.3.	Data ingestion .....	46
2.3.4.	Data storage .....	47
2.3.5.	Data transmission / streaming .....	55
2.3.6.	Data processing - analysis .....	57

2.3.7.	Data Processing – modelling .....	60
2.3.8.	Application for online visualisation .....	64
2.4.	Test bed implementation at iba AG .....	65
3.	Additional tools .....	66
3.1.	FADI .....	66
3.2.	iba tools for the transfer of measurement data to external systems .....	67
4.	Experiences and results made.....	68
4.1.	Experiences and implementation: Wire rod rolling use case.....	68
4.1.1.	Experiences: Implemented systems.....	70
4.1.2.	Prospects for the wire rod use case .....	76
4.1.3.	Efforts of system's implementation .....	76
4.2.	Experiences and implementation: Hot rolling use case .....	78
4.2.1.	Experiences: Implemented systems.....	78
4.2.2.	Experiences: Procedure of data processing and analyses.....	79
4.2.3.	Prospects for the hot rolling use case .....	90
4.2.4.	Efforts of system's implementation .....	90
4.3.	Cold rolling use case .....	91
4.3.1.	Experiences: Implemented systems.....	91
4.3.2.	Prospects for the cold rolling use case .....	95
4.3.3.	Efforts for implementation.....	95

## List of figures

Figure 1: General scheme of data processing for offline and online case .....	1
Figure 2: Architecture scheme at SZFG .....	2
Figure 3: Data sources of the hot rolling use case .....	3
Figure 4: Scheme of components' interrelation for modelling activities.....	10
Figure 5: Data sources at the cold rolling use case .....	17
Figure 6: Architecture scheme at the cold rolling use case .....	17
Figure 7: Implementation of the Spark Structured Streaming process .....	23
Figure 8: Sketch about data integration and alignment at Marcegaglia.....	24
Figure 9: Available data sources at JSW wire rod rolling mill.....	29
Figure 10: Architecture scheme at the wire rod rolling use case.....	30
Figure 11: Zabbix server dashboard .....	42
Figure 12 : Monitoring graph of CPU data .....	43
Figure 13: Flow chart for inference module at wire rod rolling.....	61
Figure 14: Example of inference results for monitoring surface quality (longitudinal defects) .....	63
Figure 15: Real time monitoring example at wire rod rolling through Grafana .....	64
Figure 16: Composition of FADI framework.....	66
Figure 17: Main dashboard – screen shot.....	70
Figure 18: List of Deployments – screen shot .....	71
Figure 19: List of Pods – screen shot .....	71
Figure 20: Cluster dashboard – screen shot.....	72
Figure 21: Dashboard Computer Resources of Pods – screen shot .....	72
Figure 22: Dashboard Computer Network Resources of Pods – screen shot .....	73
Figure 23: Azure Blob Storage Transaction Metrics – screen shot.....	73
Figure 24: Offset Explorer – screen shot.....	74
Figure 25: Example of real-time process data visualisation.....	74
Figure 26: Example of inference results for monitoring surface quality.....	75
Figure 27: Continuous signal compilation of the rolling process across two files .....	80
Figure 28: Example of window generation and feature extraction in time signal .....	80
Figure 29: Example for a one signal's categorised histogram .....	82
Figure 30: Ranking list by SOM, selected inputs from stand F7, statistical features .....	83
Figure 31: Example for summarising results at one sample for three data mining methods.....	84
Figure 32: Example for summarising results of 3 samples.....	84
Figure 33: Schematic confusion matrix .....	86
Figure 34: Example of MariaDB, results of rolling window (10 sec length) with aggregation – screen shot .....	88
Figure 35: Example of Spark Job Monitoring - screen shot.....	88
Figure 36: Example of Spark Streaming Monitoring - screen shot.....	89
Figure 37: Accurate cases .....	93
Figure 38: Shifted cases.....	93
Figure 39: Inverted cases.....	94
Figure 40: Cases with special causes of variation .....	95

**Glossary of tools**

Anaconda	4	MariaBD	8
Apache Hadoop File system (HDFS)	7, 21	Matplotlib	12, 25
Apache HTTP Server	28	Metabase	14
Apache Kafka	8, 22, 55	MinIO	51
Apache NIFI	20	MySQL	21
Apache Spark Structured Streaming	9, 23	NumPy	12
Apache Spark/PySpark	25	Pandas	12
Azure Blob Storage	50	PostgreSQL	52
Azure Data Bricks	60	Production data service –JSW	47
Azure Machine Learning Services	60	PyCharm	5, 45
C++ based processing modules	24	PySpark	58
Dedicated NT4S HMI	28	Python	3, 57
Grafana	64	Python code - SZFG	11
ibaAnalyzer	59	Rancher	38
ibaPDA-Data-Store-Kafka	7, 20, 56	RCIone	36
Inference module (self-written)	61	Redis	34
InfluxDB	47	Samba file system	8
Intellij IDEA	19	SciKit-learn	59
Jupyter notebook	5, 45	Spyder	45
Keras	26	TensorFlow	13
Kubernetes (K8s)	32	Visual Studio 2019	19, 44
LAP Laser data service - JSW	46	Visual Studio Code	19, 44
Loki	39	Wire Check data service - JSW	46
Longhorn	48	Zabbix	42

## 1. Introduction and Overview about this workbook

This workbook shall support future implementations of Big Data technologies for real-time monitoring, control and forecasting applications in steel production. It includes brief descriptions of the use cases investigated in the project NewTech4Steel and of the experiences and results made during the implementation of the selected technologies and architectures at those use cases.

This workbook will not (and cannot) deliver ready-to-apply solutions for specific problems and tasks, but it shall help operators and experts in the steel production when planning and executing similar tasks in their specific environment.

The tool description blocks are related to the use case, so readers may have a look firstly to that use case matching best their own task. This "tree-like" structure leads in some cases to multiple descriptions of a specific tool, but it preserves the relation to the depicted implemented schemes and will reflect the author's experiences according his/her specific application.

## 2. Use cases in NewTech4Steel

The described concepts and implementations of this workbook follow the below depicted general concept of data processing. For the real-time monitoring, control and forecasting of specific target features of processes and products in steel production, one needs two general chains of processing: the online and the offline path.

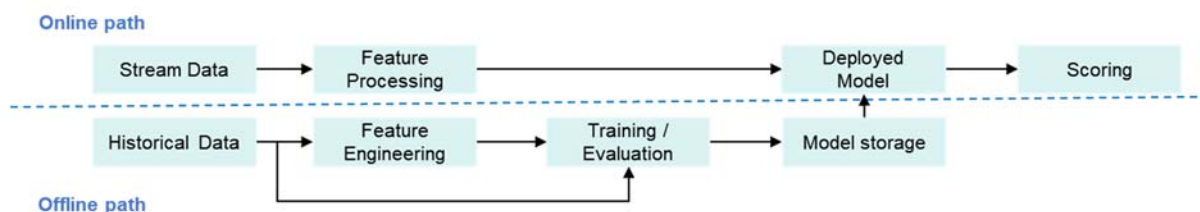


Figure 1: General scheme of data processing for offline and online case

The modules in the offline path corresponds to the tasks

- to collect a **historical data base** of a sufficient size. This data base has to be analysed concerning the correctness of data (e.g. time-series data, trigger signals to define start/end of batch processes, meta data including general plant/product properties).
- to extract **features** from the complete data set or from a subset (e.g. most important ones in relation to a target parameter) with regards to the later used analysis techniques.  
This may include
  - the derivation of features from the original data to extract ueseful information regarding the target parameter
  - statistical or Data Mining approaches to identify most important features/signals
  - to transform the original data in a shape that it can be used for for analysis (feature extraction, generation of rolling windows to select consecutive shorter time intervals)
- to **evaluate** the data and develop models (e.g. simple threshold comparison up to the **training** of deep neural networks) and to store these models for the online use.

The modules in the online path corresponds to the tasks

- to establish a data **stream** of measured data (near) online to the module
- where procedures of **feature engineering** are executed (defined in the offline evaluation)
- to **deploy** the stored **models** with this features to get a result about the target parameter
- to evaluate the model's output to achieve a **score** which will be used for information or dedicated actions

## 2.1. Hot rolling at SZFG

### Intention

At SZFG consent was found to enhance their ability to make use of collected data to reduce stability problems in the hot rolling area. When looking for promising technologies, going beyond the state that was existing at that time, the idea arose to apply architectures and tools from the field of Big Data. Those tools were promising fast and robust ways of data handling with the aim to online evaluate the fast incoming signals at the hot rolling mill in way. Corresponding to that, new approaches should be tested to collect, to store and to analyse the continuously growing amount of available data. Also here technologies from Big Data and Machine learning offered proven tools. The new system should be adaptable in aspects of hardware extensions, that means to extend storage capacities, processing memory and performance if it should be necessary in the future. In the same way, necessary software architectures and methodologies should be able to meet the existing and future requirements to exploit the occurring data as fast as possible.

The use case at SZFG consist of several sub use cases, e.g.:

- An initial proof of concept for the process stability of the hot rolled strip for each pass of the roughing mill for individual steel grades shows the functionality of the tool chain. For this purpose, a basic model evaluates the standard deviation of the rolling forces in the roughing mill according to steel grade and pass number.
- The investigated anomaly is to classify tails breaks in the finishing mill where short hot glowing metal pieces clip off the strip. The cause is believed to be an irregularity in differential rolling forces and the combination of various input factors like strip deviation, rolling forces, velocity and many more.

The following Figure 2 shows the architecture installed at SZFG.

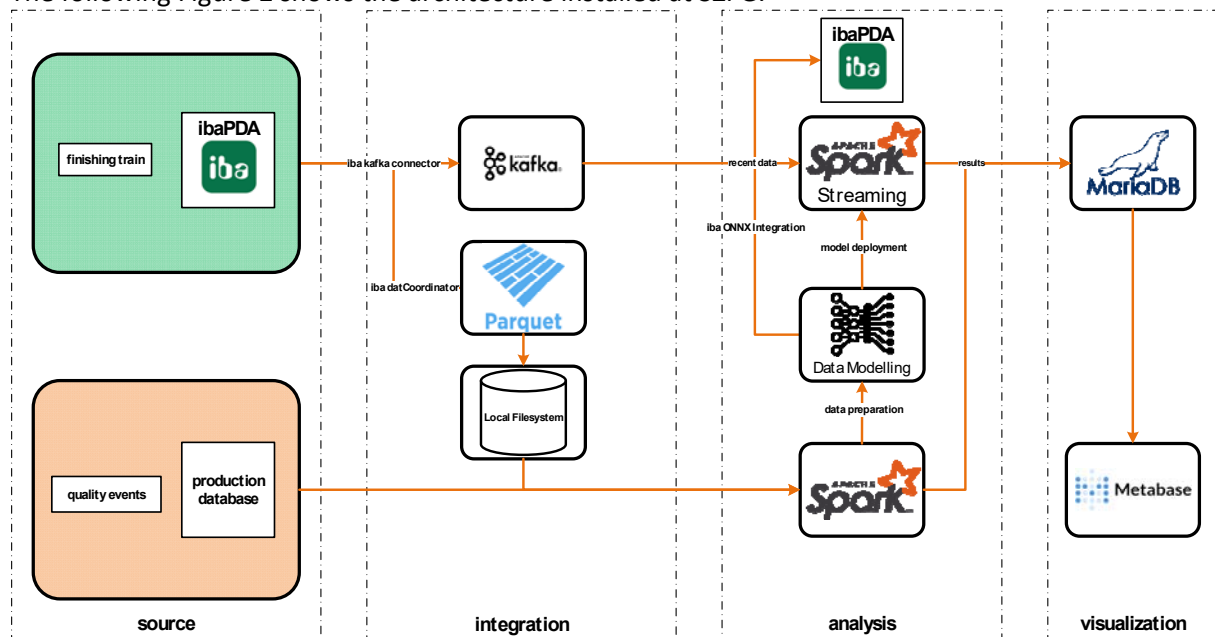


Figure 2: Architecture scheme at SZFG

The focus was put to data from the finishing train which are acquired via an ibaPDA system. This data includes more than one hundred different time signals measured at the seven stands of the hot rolling finishing train sampled at a rate of 100 milliseconds.

From the production database quality events, i.e., the occurrence of tail breaks, are stored in a MariaDB for further exploitation.

<b>PLC data from iba system</b> <ul style="list-style-type: none"> <li>• 353 process parameters of entry, 7 stands and exit of hot rolling mill</li> <li>• 7 binary signals (strip in stand x)</li> <li>• 1 identifier (coil number in stand 1)</li> <li>• sample rate 10 ms</li> <li>• exported every 5 minutes to Samba file system in Parquet format for off-line analysis</li> <li>• streamed to Kafka for online exploitation</li> </ul>	<b>Level 2 data via database connection</b> <ul style="list-style-type: none"> <li>• strip geometry (target values: width/thickness), steel grade</li> <li>• occurring tail break event</li> <li>• exported regularly into a MariaDB which is also used for result storage of streaming data evaluation</li> <li>• accessed by Kafka for online use</li> </ul>
---	--

Figure 3: Data sources of the hot rolling use case

Concerning file storage for the offline analysis of data, in a first approach collected data from ibaPDA-system were stored via Parquet format in a Hadoop based storage system (HDFS). This was changed into a simpler local file system which totally fulfils the necessary requirements of this task.

Offline analysis is done by tailored Python modules which analyses the data by different approaches. Here different Python libraries were applied (see below).

The online path for evaluating actual data of the finishing train is realised by the integration of Apache Spark which is directly connected to the ibaPDA system. With Apache Spark data blocks (windows of a defined time length) are compiled in real-time and send to the consumer "model evaluation". Here the models generated by BFI are processing the actual window data and the result is send to a MariaDB for storage and further visualisation by Metabase.

For this streaming path a second variant is implemented and tested by Salzgitter and iba. By storing developed models in ONNX format (Open Neural Network Exchange), a module in the ibaPDA system can execute these models directly without sending data to another instance in the system.

### 2.1.1. General tools for system organisation, administration and supervision

#### 2.1.2. Development systems (IDEs)

##### Python

###### General tool information

Python 3, installed as stand-alone or integrated into an IDE installation like Anaconda or PyCharm. It can be used under the OSI-approved open-source license. <https://www.python.org/downloads>

###### Tool/system description

Python is a "*high-level, interpreted, general-purpose programming language*", including "*structural, object-oriented and functional programming*" (see Wikipedia).

###### Tool implementation

In the hot rolling use case, there exist two development environments: one locally at BFI basing on Anaconda environment (Windows OS) and one at SZFG basing on a Jupyter Hub installation, both together with a PyCharm IDE installation.

Additional to the standard installations several Python modules (libraries) were installed for special purposes (see below). Their selection based on their popularity and an (expected) long-term life cycle to ensure the robust and future integration into the code development environment when updating versions of IDEs or libraries.

The installation of external (third-party) modules/libraries is supported by the Python Package Index (PyPI) which checks version compability of installed libraries.

###### Tool assessment

Pros:

Python is a good readable programming language, but users should have a basic understanding of programming. For certain applications like Data Analytics wider (theoretical) knowledge of the related methodologies is necessary.

There exist thousands of libraries covering many specialised questions, many of them supporting performance optimisation by exploiting multiple core hardware

Its integration into IDE packages like Anaconda simplifies its installation and use.

World-wide forums support developers by examples, experience reports and tutorials (see references below).

Cons:

Sometimes similar functionality can be found in several libraries with differences in function calls and the implementation of formats (e.g. Parquet file interfaces in PySpark and Pandas)

For packages and libraries which exist in free and commercial versions, sometimes the vendors' documentation of the free version is imperfect.

Related to the execution performance of written code, programmers have to look for an optimised choice of used data formats or procedures.

An example:

When adding calculated vectors like a set of derived statistical features to an growing matrix inside a loop, one may find that a variety of possible data types like data frames, lists and dictionaries or Numpy arrays. Here the user has to check for large data sets and many iterations which data type will be best for this task. That can be done easily by checking the internet, but one has to think about that possibility.

Another examples:

While-loops are faster than for-loops. list/dictionary comprehension is faster than executing a loop. But also such questions can be solved by an internet query.

## References

[www.python.org](http://www.python.org) for general information

<https://stackoverflow.com>, dedicated Python forum

[https://blog.feedspot.com/python\\_forums/](https://blog.feedspot.com/python_forums/), list of best Python forums

## **Anaconda**

### General tool information

Anaconda, licensed under the BSD, download at <http://anaconda.com/>

There exist also commercially distributed versions.

### Tool/system description

Anaconda is a Python distribution for several operating systems. It accesses repositories that allow an easy way to install and update Python libraries in packages. It also solves dependencies of the packages during installing. Using environments, you can have several combinations of Python libraries in different versions for different use cases or for backup purposes.

In the hot rolling use case, it is used for the Python backend by JupyterHub and the development of the source code of the different Python based modules for data ingestion and so on up to the implementation of Apache Spark for the (near) real-time processing.

### Tool implementation

Anaconda with its main tool conda is easy to use but powerful.

During the installation of new packages an environment can break and make it not usable anymore. Therefore, the use of environments is useful to always have a working environment and only install new packages in parallel to a working environment.

### Tool assessment

Anaconda uses hard links to reuse storage space of the same libraries in different environments.

A drawback is that you cannot rename environments.

## References

Installation: <https://docs.anaconda.com/anaconda/install/>, <https://docs.conda.io/en/latest/>



## Jupyter notebook

### General tool information

JupyterHub, licensed under the modified BSD license, download at <https://jupyter.org/hub>

### Tool/system description

JupyterHub is a web based IDE for developing in several programming languages but mostly Python.

Here it is used for writing the Python source code using the Pyspark library that run on the local Spark server.

### Tool implementation

These adjustments have been made:

- The JupyterHub server was configured that for each Linux system user an own JupyterHub process is spawned.
- Several Python kernels have been configured that start the Spark job with e.g. different amount of RAM and reserved computation cores .
- If a user runs one or several notebooks he does not know which CPU and RAM resources a notebooks uses. Using an extension like “jupyterlab-system-monitor” this information is displayed above each notebook.
- For several modelling tasks different python environments are necessary due to conflicting packages. Using the extension “nb\_conda\_kernels”, a user can chose the right Python environments for each notebook.
- A TLS certificate is used for encrypting the developed web browser that access the notebooks on the JupyterHub

Even if a notebook does not consume new messages via e.g. Kafka the RAM of the Spark job of a notebook is still used. So, the kernel of unused notebooks should be stopped in order to free RAM resources.

### Tool assessment

Jupyterhub does consume only few resources, but the Spark jobs it creates can use much more resources.

Juypter notebook could not be debugged by in-system tools, only by external tools like PyCharm Professional. This disadvantage was eliminated by JupyterLab 3.0 but is not tested.

### References

<https://jupyterhub.readthedocs.io/en/stable/quickstart.html>

<https://github.com/jtpio/jupyterlab-system-monitor> extension for Jupyter Hub

<https://jupyterlab.readthedocs.io/en/stable/user/debugger.html> Debugger front-end by default

## PyCharm

### General tool information

PyCharm is an integrated development environment (IDE) by the company JetBrains for the Python programming language, free version (PyCharm Community) only for Python projects at <https://www.jetbrains.com/pycharm/download/#section=windows>

### Tool/system description

PyCharm Community supports Python code development by

- project related programming environments
- comfortable code debugging
- word completion, showing library depending functions during typing, simplifies library installation etc

### Tool implementation

There exist installation packages for different operating systems, and PyCharm is included in the Anaconda IDE

### Tool assessment

PyCharm is quite helpful when developing Python modules:

in debug mode you can test variances of your code and you can overwrite variable values

- one can run multiple PyCharm instances concurrently
- one can execute multiple modules in one session in parallel
- code can be copied directly to Jupyter notebook. Only few IDE-dependent settings in the code have to be adjusted

### 2.1.3. Data ingestion

#### ibaPDA-Data-Store-Kafka

##### General tool information

ibaPDA-Data-Store-Kafka is an add-on to ibaPDA system and is commercially licensed by iba AG

##### Tool/system description

ibaPDA-Data-Store-Kafka stores data acquired by ibaPDA into a connected Kafka cluster. Since it uses messages according to Kafka's definitions these messages are handled by Kafka in the same way as every data producer.

##### Tool implementation

In the hot rolling use case, ibaPDA systems acquiring process data directly from measurement systems and other automation components (PLCs, Level2 computers) and send the selected data to an installed Kafka instance which delivers the data to Apache Spark Streaming component. This path is only for the online processing of the data, the storage of Parquet files is done separately from this path.

### 2.1.4. Data storage

#### Apache Hadoop File system (HDFS)

Rejected after tested

##### General tool information

Apache Hadoop , licensed under the Apache 2.0 license, download at <https://hadoop.apache.org/>

##### Tool/system name description

Apache Hadoop is a software framework that allows for the distributed processing of large datasets. It is designed to run on thousands of machines and offers sophisticated failure handling and data replication. Here it is used to store multiple terabytes of time series data. It is compatible to serve as datasource for Apache Spark distributed processing.

##### Tool implementation

These adjustments have been made:

- A minimal Hadoop implementation consists of one Namenode, storing metadata and handling write/read requests and at least 3 Datanodes, which serve as storage units.
- For each of the nodes a linux-machine has to be set up and configured.
- To orchestrate write/read or compute jobs Hadoop uses the included YARN Resource Negotiator.
- Since Hadoop is written in Java the host machines need to have a Java Runtime Environment installed. Compatible Version found on: <https://cwiki.apache.org/confluence/display/HADOOP/Hadoop+Java+Versions>
- Since the name node needs to access the data nodes a SSH public key authentication should be set up
- Stored data should be in a chunk size of 64Mb /128Mb in respective to the configured block size to optimize processing.

##### Tool assessment

The use of Hadoop for storing rather small datasets smaller 20 TB and with the requirement for real-time-analysis is not recommended. The efforts to configure and self-host a non-commercial version of Hadoop are greater than the benefit, since a manageable and scalable deployment also needs additional tools like Apache Ambari and Zookeeper which also have to be configured. The strength in scalability and performance of Hadoop come to fruition when handling really big or

heterogenous datasets. Concerning the shortcomings of Hadoop regarding the steel use case, the use of the local file system in combination with Apache Spark is to be preferred.

### **Samba file system**

#### General tool information

Samba, licensed under the GPL, download at <https://www.samba.org/>

#### Tool/system description

Samba is a file server that generally provides this service to Windows Clients and is mostly running on Linux servers.

At the hot rolling use case, Samba file system is used by the Linux host as a server where to the Microsoft Windows based ibaPDA software exports the large process data each few minutes after the production in the Parquet file format for the offline data path.

#### Tool implementation

Samba needs a special configuration which SMB standard should be supported since the Windows client showed errors in the first place with default configuration.

This additional configuration for /etc/samba/smb.conf was necessary:

client min protocol = SMB3

client max protocol = SMB3

ntlm auth = yes

#### Tool assessment

Samba provides its service without further problems and additional resources.

### **MariaDB**

#### General tool information

MariaDB, licensed under the GPL, download at <https://mariadb.com/>

#### Tool/system name description

MariaDB is relational database and is a successful fork of MySQL with most tools for accessing these databases are compatible.

In the hot rolling use case, MariaDB is used for storing evaluations and providing metadata and labeled process data.

#### Tool implementation

In contrast to most other used tools can MariaDB easily installed and updated by the Debian Linux package manager.

Besides the creation of users with their privileges and creation of databases and tables no additional configuration was necessary.

#### Tool assessment

Writing to and reading from the databases with the used data is very fast and used few CPU and RAM resources.

## **2.1.5. Data transmission / streaming**

### **Apache Kafka**

#### General tool information

Apache Kafka, licensed under the Apache License 2.0, download at <https://kafka.apache.org/>

#### Tool/system description

Apache Kafka is a distributed event store and stream-processing platform.

Here it is used as a message broker for the live data path that receives live process data with 100 or 1000 Hz. An Apache Spark job connects to this data messages and processes them.

### Tool implementation

Since Apache Kafka is also an event store it saves its data in its “log files” and the disk usage can add up quite fast. In order to limit the disk usage this configuration is useful to delete unused old data: `log.retention.hours=168`.

Messages are grouped into topics and these topics cannot be deleted by default. If you want to delete a e.g. testing topic this configuration has to be added: `delete.topic.enable=True`

### Tool assessment

CPU and RAM usage can hardly be noticed, so the performance is very good.

## **Apache Spark Structured Streaming**

### General tool information

Apache Spark, licensed under the Apache License 2.0, download at <https://spark.apache.org/>

### Tool/system description

Apache Spark is an analytical software for offline and online data processing.

In the streaming path it is subscribed to one or more Apache Kafka topics and receives the process data messages, pre-processes them, applies a model and generates an evaluation which can be stored e.g. using JDBC database driver.

### Tool implementation

Spark has several modes how to process live data: older “Spark Streaming”, here used “Spark Structured Streaming” and since some years experimental “Continuous Processing”. Documentation and examples are sometimes vague and difficult to use and new user should take some time for training. You often find the documentation with the same content, simple examples and very few complex ones.

For Spark to obtain Kafka messages several additional jar files must be installed into the jar subdirectory. The documentation here is also not clear in the first place and you must find the jar files on your own. The jar files with the version number in the file name should fit to the versions of Spark 3.a.b and Kafka 2.c, the necessary files seem to vary from version to version:

`commons-pool2-2.c.0.jar`

`spark-streaming-kafka-0-10-assembly_2.12-3.a.b.jar`

`kafka-clients-2.8.c.jar`

`spark-sql-kafka-0-10_2.12-3.a.b.jar`

`spark-token-provider-kafka-0-10_2.12-3.a.b.jar`

For accessing a database, a jar driver like `mariadb-java-client-x.y.z.jar` is also necessary and must be put into Spark jar directory.

It happens that Spark adds dozens of gigabytes of temporal data in the `/tmp` directory and the data is purged only after the complete job has been finished. If the Spark job should run without interruption for hours or days, there should be enough free space on `/tmp` or you have to deal with deleting old data using e.g. cron jobs.

Using an Apache Spark installation in Python modules makes the installation of the Python library PySpark necessary (see below at Data processing - analysis).

The streaming for the hot rolling use case is realised using Apache Spark. The development for the streaming processes were done using Jupyter Notebook and PySpark.

The source of the data is the messaging Apache Kafka which distributes the high-resolution data measured by the iba PDA system. See the Glossary to find the related sections of these tools.

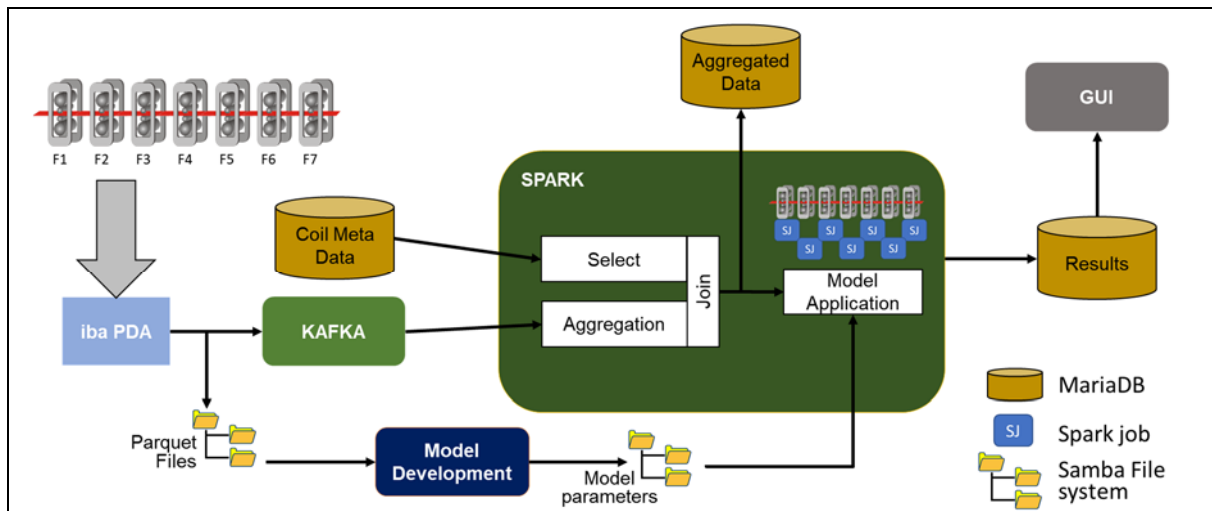


Figure 4: Scheme of components' interrelation for modelling activities

The implementation is realised using a separate Spark job (SJ) for each stand of the hot rolling mill. The definition of a common context for the project allows to share information between the different Spark jobs using a dedicated app name. Each new job gets this environment to share data.

```
# create spark session, connect to common environment
spark = SparkSession \
    .builder \
    .appName("bfi_stream_nt4s") \
    .config('spark.sql.caseSensitive', True) \
    .getOrCreate()
```

Furthermore, common functions e.g., to apply a model to the aggregated window data, can be defined and distributed to all jobs running in that environment

```
# broadcast the new function inside the spark context
sc = spark.sparkContext
bc_model_func = sc.broadcast(model_func)
print(bc_model_func.value)
```

Each job aggregates the related data using the Spark windowing feature. The trigger signal to define start and end of a rolling process at one stand is used for that aggregation. The aggregated data are stored into a MariaDB for the model development and later for the model implementation.

### Tool assessment

Spark can use all the CPU and RAM power it has if configured to use that can easily add up to 50-200 GB of RAM. In order to do testing, Python kernels with RAM limitation may be useful.

In the hot rolling use, the Spark windowing function is oriented to fixed-length time windows. The example to the right shows the results of a Spark job aggregating data for 5 second windows.

It can be seen that the data are aggregated e.g. from 10:49:50.000 to 10:49:54.999 (row 168 in the figure).

This leads to the effect that for the first and last window of a pass only partial data are used for the aggregation (see row 173 and 174, see column "count"). This behaviour has to be considered when using manually generated data for model development and stream processed data for online model application.

	123 count	time_start	time_stop	123 wafi
168	5.000	2022-04-19 10:49:50.000	2022-04-19 10:49:54.999	97
169	5.000	2022-04-19 10:49:55.000	2022-04-19 10:49:59.999	97
170	5.000	2022-04-19 10:50:00.000	2022-04-19 10:50:04.999	97
171	5.000	2022-04-19 10:50:05.000	2022-04-19 10:50:09.999	97
172	5.000	2022-04-19 10:50:10.000	2022-04-19 10:50:14.999	97
173	3.772	2022-04-19 10:50:15.000	2022-04-19 10:50:18.771	97
174	3.667	2022-04-19 10:50:31.333	2022-04-19 10:50:34.999	97
175	5.000	2022-04-19 10:50:35.000	2022-04-19 10:50:39.999	97
176	5.000	2022-04-19 10:50:40.000	2022-04-19 10:50:44.999	97
177	5.000	2022-04-19 10:50:45.000	2022-04-19 10:50:49.999	97
178	5.000	2022-04-19 10:50:55.000	2022-04-19 10:50:59.999	97

During the project also some benchmark tests were executed to check if limitations of the system occur at reduced window sizes. Therefore, the window size was set down to 0,1 seconds which lead to a frequency of the window calculation of 10 Hz (10 times of sample rate). At that value an asynchronous storage of Spark window data was detected. No data gets lost, but the aggregation results are not stored in the order they were calculated. As shown in the table excerpt below, in the row 'time\_db\_insert' the storage order differs from the calculation order as listed with the red numbers. This asynchronous storage of window data has to be considered when using the Spark window data in a stream for further calculations.

Depending on the computational capacity of the hardware and the load due to other processes, it has to be checked if the data processed in a stream are still delivered in the right order.

	ID	count	time_start	time_stop	time_db_insert	
1	5.019	10	2022-02-21 16:12:45.000	2022-02-21 16:12:45.090	2022-02-21 16:12:51.868	
2	4.999	10	2022-02-21 16:12:44.900	2022-02-21 16:12:44.990	2022-02-21 16:12:49.856	
3	5.006	10	2022-02-21 16:12:44.800	2022-02-21 16:12:44.890	2022-02-21 16:12:50.063	
4	5.016	10	2022-02-21 16:12:44.700	2022-02-21 16:12:44.790	2022-02-21 16:12:50.973	
5	5.003	10	2022-02-21 16:12:44.600	2022-02-21 16:12:44.690	2022-02-21 16:12:49.876	
6	5.005	10	2022-02-21 16:12:44.500	2022-02-21 16:12:44.590	2022-02-21 16:12:50.059	6
7	5.018	10	2022-02-21 16:12:44.400	2022-02-21 16:12:44.490	2022-02-21 16:12:51.005	5
8	5.012	10	2022-02-21 16:12:44.300	2022-02-21 16:12:44.390	2022-02-21 16:12:50.503	3
9	5.013	10	2022-02-21 16:12:44.200	2022-02-21 16:12:44.290	2022-02-21 16:12:50.615	4
10	5.001	10	2022-02-21 16:12:44.100	2022-02-21 16:12:44.190	2022-02-21 16:12:49.863	1
11	5.002	10	2022-02-21 16:12:44.000	2022-02-21 16:12:44.090	2022-02-21 16:12:49.865	2
12	5.014	10	2022-02-21 16:12:43.900	2022-02-21 16:12:43.990	2022-02-21 16:12:50.814	
13	5.015	10	2022-02-21 16:12:43.800	2022-02-21 16:12:43.890	2022-02-21 16:12:50.950	
14	5.010	10	2022-02-21 16:12:43.700	2022-02-21 16:12:43.790	2022-02-21 16:12:50.398	
15	5.011	10	2022-02-21 16:12:43.600	2022-02-21 16:12:43.690	2022-02-21 16:12:50.444	

sorted start/end time of windows  
 time of insertion  

5  
3

 order of insertion

### 2.1.6. Data processing - analysis

#### Python code - SZFG

##### General tool information

Python 3, installed as stand-alone or integrated into an IDE installation like Anaconda or PyCharm  
It can be used under the OSI-approved open-source license. <https://www.python.org/downloads>

##### Tool/system description

Python is a "*high-level, interpreted, general-purpose programming language*", including "*structural, object-oriented and functional programming*" (see Wikipedia).

##### Tool implementation

For the data tracking and data compilation mainly two development environments are used: one locally at BFI basing on Anaconda environment (Windows OS) and one at SZFG basing on a Jupyter Hub installation, both together with a PyCharm IDE installation.

At the hot rolling use case Python based modules are used to read hot rolling process data from the data lake and anomaly information from MariaBD to compile the data sample for offline analyses.

The modules include the separation of the data per coil and per stand (at SZFG the hot rolling line includes 7 stands) as well as the calculation of statistical features per coil and a rolling-window-base for analysis.

The data ingestion tools integrates 4 main parts:

- selection and reading reading data files (Parquet format) to enable gapless compilation of data belonging to one coil
- identification of trigger points start/stop of rolling to differe between idle times and real rolling operation
- extracting the selected process parameters per coil, calculating statistical features per coil/window
- storing the original process parameters per-coil as an interpolated length-fixed time series and as derived features as well as length-fixed and window-based time series (without interpolation) and derived features.

##### Tool assessment

During the development several problems occurred of which one had to be aware

- the mass of data makes it necessary to think early about possibilities to check automatically the correctness of data. That includes the automatic generation of data visualisaton, the assessment of data statistics, and sometimes simply checking the spelling of names (e.g. under Linux for manually generated file names due to upper/lower case differentiation)

- in batch-oriented processes (here per coil and per rolling stand) the identification of correct trigger points to define start and end of a batch (here: rolling process) can become complex, just at high-speed processes

## **Matplotlib**

### General tool information

Matplotlib is a plotting library for data visualisation in Python. Matplotlib only uses BSD compatible code, and its license is based on the PSF license, for download see <https://matplotlib.org/downloads.html>

### Tool/system description

Matplotlib offers static, animated, and interactive visualisations in Python. With its module PyPlot it offers a MATLAB-like interface which may help some users with existing Matlab knowledge. There exist a mass of plot types and functions for free use.

### Tool implementation

Matplotlib can be installed easily like other Python libraries via supported installers in the used IDE

### Tool assessment

Matplotlib works excellent with all common numerical data formats in Python, but best with types coming from NumPy library. Here one can find possibilities to accelerate the plotting of huge data enormously.

## **NumPy**

### General tool information

Numpy, under the BSD license, Repository: <https://github.com/numpy/numpy>

### Tool/system description

NumPy is an open-source Python library and highly optimized to manipulate numeric arrays for scientific computing. The core functionality of NumPy is its "ndarray" which is a (usually fixed-size) multidimensional array of items of the same type and size. Numpy arrays take less memory space and provide better runtime speed in comparison with similar data structures in Python (lists and tuples). Many advanced data science and machine learning libraries require data to be in the form of NumPy arrays before it can be processed.

### Tool implementation

After data pre-processing, actual data in the data frame format is converted into a Numpy array which is used as input to Tensorflow and Keras. Additionally NumPy offers mathematical functions like interpolation routines which were used in the tracking and data sample compilation (for the model development)

## **Pandas**

### General tool information

Pandas, BSD-licensed library, repository: <https://github.com/pandas-dev/pandas>

### Tool/system description

Pandas is an open-source library for working with data sets. It can handle data from various file formats like JSON, SQL, Microsoft Excel, etc. DataFrame is the powerful data format of Pandas. A DataFrame is structured like a table that contains two-dimensional data and potentially handle heterogeneous data and it's size is mutable. The rows and the columns both have indexes, and arithmetic operations can perform alignments on both row or column separately. Pandas is built on the NumPy library and written in languages like Python, Cython, and C. Pandas includes fast and performant I/O modules for Parquet file format.

### Tool implementation

Since in the hot rolling use case the process data is stored in Parquet file format, Pandas is the first choice to be used for reading and writing.



### Tool assessment

Pandas functions show a high performance at low memory requirements for the handling of the processed hot rolling data. Their coding is intuitive and easy to implement.

### **PySpark**

#### General tool information

PySpark is an interface for Apache Spark in Python, BSD-licensed library, licensed under Apache License 2.0, download at <https://spark.apache.org/>

#### Tool/system description

PySpark is a Python library which includes interfaces to Apache Spark, supporting Spark features like Spark SQL, DataFrame format, and MLlib, a scalable machine learning library

#### Tool implementation

In the hot rolling use case, after some practical experiences PySpark was not longer used for the offline data tracking and compilation.  
But for the online data streaming and model running PySpark is used to define and execute the rolling window generation and the aggregation of that window data.

### Tool assessment

For the offline handling of data in the hot rolling use case, PySpark shows some disadvantages:

- In terms of DataFrame format, there are differences in function calls and the implementation of formats to the library Pandas. This can be time-consuming and is prone to making errors.
- In case of long running processes, e.g. the iteratively reading of a mass of huge Parquet files (50.000 files, each containing 300.000 lines and around 100 columns), the underlying running Java machine produces errors like "stack overflow" or "heap memory limit passed"
- Some functions like the data aggregation in rolling windows for the calculation of statistical features was bad documented and less performant than coding this rolling window task by own and using Pandas NumPy libraries for calculations

In case of the online window generation and data aggregation one main disadvantage is the partially missing or incomplete official documentation. Here a lot of efforts are necessary to find for example the correct parametrisation of functions.

## **2.1.7. Data Processing - modelling**

### **TensorFlow**

#### General tool information

TensorFlow is an end-to-end open-source machine learning platform, under Apache License 2.0, repository: <https://github.com/tensorflow/tensorflow>

#### Tool/system description

The used Python library TensorFlow (including Keras, a deep learning library) includes functions for creating Deep Learning models and performs a variety of machine learning tasks. It utilizes various optimization techniques to make the computation of mathematical expressions easier and more efficient. TensorFlow also has a broad library of pre-trained models that can be used in your own projects. TensorFlow is supported on Python versions 3.3+.

#### Tool implementation

In the hot rolling use case TensorFlow and Keras are used inside the Jupyter notebook IDE to develop an anomaly detection with deep Autoencoder networks (AE).

Various variants of AE were tested:

- Convolutional AE (CNN-AE)
- Fully connected and convolutional AE (FCC-AE)
- Long-Short term memory AE (LSTM-AE).

AE networks are trained in a semi-supervised way. For the hot rolling use case only data of coils without occurring tail breaks were used for training i.e., AE is used as a one-class classifier. During the model run with new data there are two ways to identify irregularities in the evaluated data:

- The assessment of the middle layer (feature reduction layer) of the AE network by looking for outliers in the node values from the node value distribution after training,
- the evaluation of the reconstruction error and its comparison to limits derived from the training phase.

#### Tool assessment

In case of LSTM-AE and CNN-AE, the data must be imported as time series data. This can be done by the original data which have to be transformed to a fixed length by interpolation, or by splitting the signal into time windows of a fixed length.

For tests with LSTM-AE and CNN-AE complete coils were used and interpolated to a fixed-length of 1000 or 500 values per rolling one coil:

The memory required to import data for 15000 coils with 145 signals (each 1000 data points) is about 20 GB of memory.

The LSTM-AE is the slowest model type to train and requires significantly more memory than the other models types. For example, one iteration of the LSTM model with a training batch size of 256 coils with only 5 layers generating ~330000 adjustable parameters requires up to 50 GB of memory during training.

The memory requirement for an iteration of a CNN processing the same data is about 10 GB.

In the case of FCC-AE, input data can be statistical features derived from the time signals, e.g. the average, standard deviation etc. For FCC-AE, the training time and memory requirements are much lower than for the other methods.

### **2.1.8. Application for online visualisation**

#### **Metabase**

##### General tool information

Metabase is a visualisation tool, licensed under the AGPL, download at <https://www.metabase.com/>

##### Tool/system description

Metabase is a web-based dashboard and interactive exploration tool. Here it is used to display the evaluation of process data in a human friendly and configurable way.

##### Tool implementation

Metabase can be downloaded as a jar file that can be then easily run as a service. Several users with different access rights can be added.

##### Tool assessment

Metabase can scan and cache databases for faster access time but then it consumes more resources. This capability was not necessary for the hot rolling use case.



## 2.2. Cold rolling at Marcegaglia

### Intention

The industrial experience in steel production process matured at Marcegaglia Ravenna plant demonstrated that there is a strong relationship between the flatness features of a cold rolled strip and the processing ability of the same coil in the downstream processes. It is generally understood that small variation of flatness profile of the cold rolled strip could lead to dramatically different performance in HDG processing ability specifically for what concern the strip guidance and for centering devices of the looper section and furnace. This, of course, yields to significant issues in terms of global productivity of the HDG line and economic sustainability that can be estimated around 2%, due mainly to HDG line speed reduction or even stop of line in the worst case. In spite of the importance of this topic in the literature no detailed analysis can be found and at moment it is not at all clear which are the hot band characteristics and cold rolled strip flatness features, detrimental for the strip stability during processing in high speed HDG lines. The opportunity that the implementation of big data modelling could give to overcome the present limitations is just in the possibility to link digitally many processes, at moment completely disjointed, improving the understanding of the correlations that exists between raw material characteristics and process conditions. In addition to these aspects, it should be underlined that the involved process was already deeply covered by advanced sensors (Stressometer, Tensil-Pro, X-ray device, etc). Concerning the previous status of data organization at Marcegaglia Ravenna all the cold rolling and HDG processes were gathered and stored by means of IBA analyzer system in a dedicated server. The process data analysis were carried out by simple post evaluation of selected signals without any implementation of mathematical modelling. All these reasons prompted us to adopt this industrial and technological issue for the Marcegaglia Ravenna use case. The basic concept of the use case was therefore to implement a Machine Learning model, within an advanced big data lambda-architecture, to optimize the whole productivity of cold rolling process and the HDG line n.3 productivity.

The Marcegaglia cold rolling use case is related to production of galvanized coils with width from 900÷1530mm and thickness in range 0.25÷4mm covering steel families of carbon steels. The main components of the plant production line are: a pickling line, a cold rolling mill and a hot galvanizing line. The use case, on this plant, aims to increase the efficiency and quality of the process at the galvanization step in steel production, through the exploitation of high-sampled measurements provided by plant unit equipment by means of real-time processing and batch processing.

The batch elaboration has the objective of merging the data coming from different sources in the upstream galvanizing process to obtain a sort of classification of the coil with respect to the subsequent step. The classification procedure is based on ML and gives to coils a criticality index which arises from flatness problems generated at previously applied step. About the real-time processing, the objective is the online forecasting of specific process parameters which are particularly critical for coil continuous processing without having negative consequences.

The mentioned process parameters, in particular, are the positions of the coil guide and centering device placed into the galvanization line at the end of the looper. The analysis of the production and products problems revealed that these parameters are very important since they describes the limit of different processing conditions.

In the cold rolling plant data are collected from different data sources by means of various automatic systems: IBA acquisition system collecting high-sampled process data, IMS x-ray instrument providing profile measurements at pickling line, ABB Stressometer giving surface tensions detection, TensilPro mathematical model that computes mechanical characteristics at CRM. Moreover, part of production data is acquired from Level 3 systems stored in database. Below the summary of identified data source for the use case with the description of their characteristics .

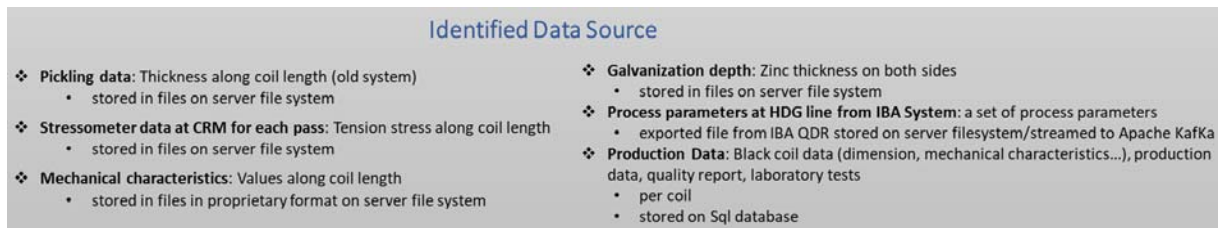


Figure 5: Data sources at the cold rolling use case

Then, continuous data are coming from IBA system sampled with 10 ms sample rates and stored to Kafka data stores to be distributed as a continuous stream to the real-time processing. Production and other equipment data are event data generated per coil and have a small impact on the total amount of traffic in the system. They are Ingested by means of NIFI tools and after that, the data flow continues until the call of the batch procedure.

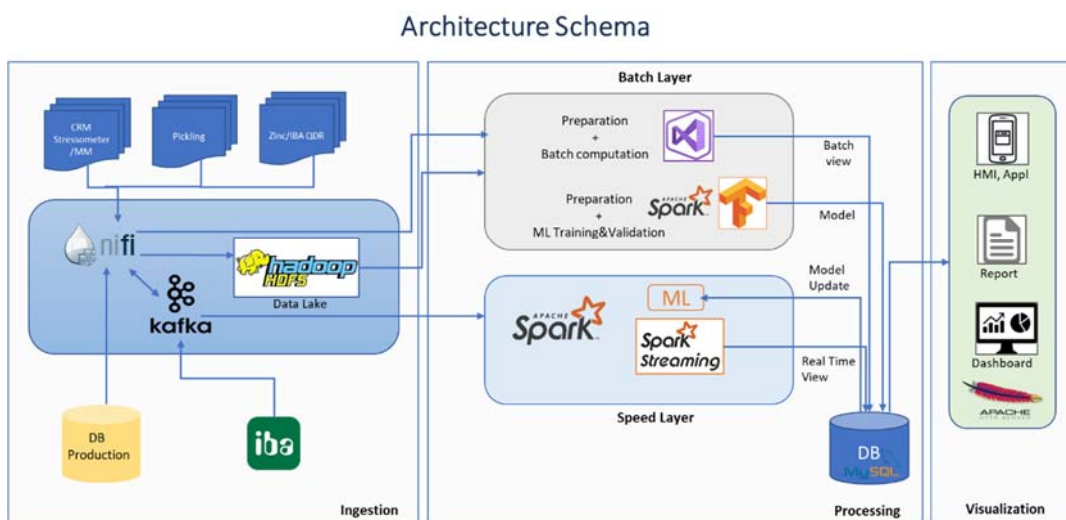


Figure 6: Architecture scheme at the cold rolling use case

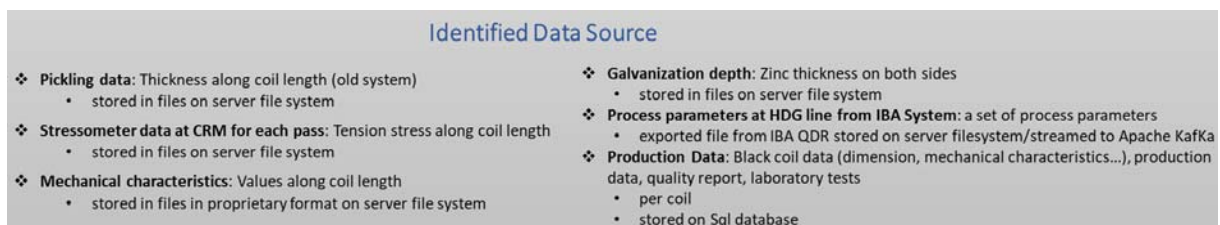


Figure 5Figure 6 illustrates the components and the logistics among them of the Big Data architecture for the cold rolling use case: the ingestion layer, the transmission module, the following data analytics units (Batch/Speed Layer), and the data visualization.

In such context of continuous process, the data transmission and acquisition method based on stream processing technology are used to obtain real-time monitoring data of the specified component on the production line. Facing this situation, the architecture framework is mainly based on Apache NIFI, Apache Kafka and Apache Spark and can combine the batch and real-time processing. The architecture includes data ingestion layer with redirection in data lake, data processing layer with result storage, and data visualization layer.

The Data ingestion layer uses:

- the NIFI distributed system which collects various data information and pushes it to the Hadoop Distributed File System (HDFS) data lake
- Kafka message cluster system as data storage in real-time for the collection of streaming data from IBA system.

The data processing layer consumes the message data in Kafka by means of Spark Streaming for real-time computing and uses, on the other side, the NIFI flow for the batch computation. In the result storage of the processing layer, the analysis results are output to the database. Finally, the analysis results can be read and displayed to users through the visual components in the visualization layer.

### 2.2.1. General tools for system organisation, administration and supervision

### 2.2.2. Development systems (IDEs)

#### Visual Studio Code

##### General tool information

Visual Studio Code, licensed under the MICROSOFT SOFTWARE LICENSE TERMS, download at <https://code.visualstudio.com/Download>

##### Tool/system description

A standalone source code editor that runs on Windows, macOS, and Linux with support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. It offers a lot of extensions to support just about any programming language.

In Marcegaglia development phase has been used for the coding of Spark and Spark Structured Streaming application by means the Python support.

##### Tool implementation

To enable the support of the Python coding, the Python extension has been installed and also all the necessary Python libraries required by the development.

##### Tool assessment

The Visual Studio Code is lightweight well-done editor that offers a good environment for the code writing and debug

#### Visual Studio 2019

##### General tool information

Visual Studio 2019, available under subscription, download at <https://visualstudio.microsoft.com/downloads/>

##### Tool/system description

Visual Studio is a comprehensive IDE for .NET and C++ developers on Windows. Fully packed with tools and features to elevate and enhance every stage of software development.

It has been used at the cold rolling use case for different scope:

- The development of analysis and pre-processing tasks in C++ language.
- The implementation of the standalone application for the integration of the Neural Network model in batch processing layer
- The development of the postprocessing classifier feature for the assignment of criticality index to coils

##### Tool implementation

It is useful to use the Visual Studio Installer for the installation

#### IntelliJ IDEA

##### General tool information

IntelliJ IDEA, licensed under the Apache License 2.0 as community edition, download at <https://www.jetbrains.com/idea/download/download-thanks.html?platform=windows&code=IIC>

##### Tool/system description

IntelliJ IDEA is an integrated development environment written in Java for developing computer software.

It was used in Marcegaglia cold rolling use case for the development of the web application based on PHP scripting language. The PHP script interact with MySQL db and through web server (Apache HTTP server) is produced the page to show.

##### Tool implementation

The standalone installation requires only the download of the executable and the run of the installer following the wizard steps

### 2.2.3. Data ingestion

#### Apache NIFI

##### General tool information

Apache NIFI, licensed under the Apache Licence 2.0, download at <https://nifi.apache.org/download.html>

##### Tool/system description

Apache NiFi is a software project from the Apache Software Foundation designed to automate the flow of data between software systems. It is a reliable device used as a real-time integrated data logistics and as a simple event processing platform. It comes with a web User Interface that helps to build data flows in real-time supporting flow-based programming by means of processors and connectors.

In the context of Marcegaglia cold rolling use case the NIFI tool has been used to access different data sources and build graphs of data routing, transformation, and software execution.

In detail, there are two purposes:

Firstly in order to get data from data source and transfer it as it is to the HDFS data lake and secondly to route the ingested dataflow to the batch processing procedure adding transformation and integration on the way.

##### Tool implementation

The tool runs on different OS but it requires Java 8 or Java 11 for the installation. The access to the web interface is supported by browser Microsoft Edge, Mozilla Firefox, Safari and Chrome.

During operation NiFi needs to have sufficient disk space allocated for its various repositories, particularly the content repository, the flow-file repository, and provenance repository then the system properties have been adjusted reflecting the use case data rates.

Data in NiFi is represented by abstraction called FlowFile and for the definition of the graph processors acting on the FlowFile, the NiFi Expression Language provides the ability to reference attributes, compare them to other values, and manipulate their values. This is a necessary skill to work with the tool.

##### Tool assessment

Apache NiFi supports clustering, so it can work on multiple nodes with same flow processing different data increasing the performance of data processing. A single node with 32 cores and 15GB RAM can handle a data rate/sec of 193MB. With 5 nodes the data rate/sec goes up to 881MB.

It provides security policies on user level and its UI can also run on HTTPS.

NiFi supports around 188 processors and a user can also create custom plugins to support a wide variety of data systems.

##### References

Processing one billion events per second with NiFi, <https://blog.cloudera.com/benchmarking-nifi-performance-and-scalability/>

Big Data Analytics: Analysis of Features and Performance of Big Data Ingestion Tools, Informatica Economică vol. 22, no. 2/2018

#### ibaPDA-Data-Store-Kafka

##### General tool information

ibaPDA-Data-Store-Kafka, Add-on license to ibaPDA basic license

##### Tool/system description

IbaPDA offers the streaming interface for online monitoring of processes captured to Apache Kafka.

In the context of Marcegaglia's use case this functionality is used to stream process variables related to galvanization operations directly to Kafka store using a specific topic. The expected signals are packaged in json messages and sent to the broker starting from the start of recording.



### Tool implementation

For the handling of ibaPDA-Data-Store-Kafka are required the basic knowledges about ibaPDA and cloud storage.

The system requirements necessary when using data storage in an Apache Kafka cluster are the ibaPDA v7.3.5 or higher and the License for the specific datastore then from Marcegaglia side, they updated the ibaPDA version and subscribed the license.

### 2.2.4. Data storage

#### **Apache Hadoop File system (HDFS)**

##### General tool information

Apache Hadoop Distributed File System (HDFS) part of Apache Hadoop project, licensed under the Apache Licence 2.0, download at <https://hadoop.apache.org/releases.html>

##### Tool/system description

The Hadoop HDFS is a distributed file system designed to run on commodity hardware. It is highly fault-tolerant and provides high throughput access to application data, suitable for applications that have large data sets.

In the context of Marcegaglia cold rolling use case the HDFS has been used as data lake then without a consistent data structure to allow the system to store raw data as it is; information is available to be analysed in its raw form or prepared for specific analytics uses as needed.

In particular data are extracted and processed outside in Spark processing framework.

##### Tool implementation

The installation requires the JDK 1.7 version and is important to set the required environment variables. It is necessary for the correct functioning of the file system, the writing of the three xml files with the configuration key of the HDFS properties.

The setting of the datalake required the adding of more TB of disk space for the data collection.

##### Tool assessment

HDFS is really not designed for many small files. Performances decrease in such cases because the client has to talk to the name node for each read.

During the setup particular attention was paid to the configuration of the name node that preferably should represent a separate physical server.

##### References

Hadoop: Addressing Challenges of Big Data, 2014 IEEE International Advance Computing Conference (IACC)

### **MySQL**

##### General tool information

MySQL, licensed under the GPLv2 or proprietary, download at <https://www.mysql.com/downloads/>

##### Tool/system description

MySQL is an open-source relational database management system (RDBMS) that organizes data into one or more data tables in which data may be related to each other; these relations help structure the data.

The Marcegaglia NT4S platform uses a MySQL server for the management of the processed data. It includes also configuration data for the postprocessing of obtained results.

##### Tool implementation

The integration of the tool has been completed by means of the installation of the XAMPP desktop application that is specific for local web development with Apache, MySQL, and PHP

## 2.2.5. Data transmission / streaming

### Apache Kafka

#### General tool information

Apache Kafka, licensed under the Apache Licence 2.0, download at <https://kafka.apache.org/downloads>

#### Tool/system description

Apache Kafka is a distributed, high throughput messaging system, a publish-subscribe environment that provide high availability. It can be used for stream processing, web site activity tracking, metrics collection and monitoring, and log aggregation. In general, it is designed for storing messages to be consumed by several applications. The principal elements of Kafka architecture are Producer, Broker, Consumer and Topic. Topics are used for feeding of messages. Producers send the messages to topics and consumers, who can subscribe to those topics consume the available messages.

In cold rolling Marcegaglia use case Apache Kafka is dedicated to the transmission of signal data from IBA system to the real-time processing component. In this case the producer is the ibaPDA-Data-Store-Kafka interface that send the json representation of the selected variables under the galvanization plant unit and the consumer is the Spark Structured Streaming script that elaborates in real-time the prediction of the target variables.

#### Tool implementation

To access broker is necessary to update the *advertised.listeners* in the *server.properties* file to give a host/IP and port combination that clients can resolve and connect to.

#### Tool assessment

Kafka possesses excellent speed and durability, and it is widely used for building real-time data pipelines and streaming applications.

The key feature of Kafka is its small number of features. It has few features and has a minimum of configurability.

With one broker handling hundreds of MB per second of reads and writes from several clients, Kafka is a very fast system. Replication is used for messages across the cluster and then stored on disk.

#### References

<https://kafka.apache.org/>  
<https://www.confluent.io/what-is-apache-kafka/>

## Apache Spark Structured Streaming

### General tool information

Spark Structured Streaming, licensed under the Apache Licence 2.0, download at <https://spark.apache.org/downloads.html>

### Tool/system description

Spark Structured Streaming is a stream processing layer on top of the core Spark data processing engine. The basic unit of processing in Spark Structured Streaming is a data frame object which is an abstraction for a collection of RDDs to be processed together. It follows a micro-batch policy for stream processing whereby streaming data coming into the systems is first collected in small batches (until the batch duration is reached) before being processed by the system.

The Spark Structured Streaming represents the architecture component covering the speed layer of the Lambda architecture in Marcegaglia use case. Spark streaming engine receives a stream of records from real-time data by means of Kafka topic messages and uses it to apply data prediction Neural Network model and produce the forecast of the target variables of the production process.

### Tool implementation

The implementation of Spark Structured Streaming continuous procedure requires the installation of Spark. This clearly requires Java 8, Python 3 and the setting of environment variables.

To make use of the built-in Kafka data source from Spark Structured streaming is also necessary to download the proper package spark-sql-kafka related to correct Kafka and Spark versions.

Below the image with the details about the Spark Structured Streaming process functioning implementation.

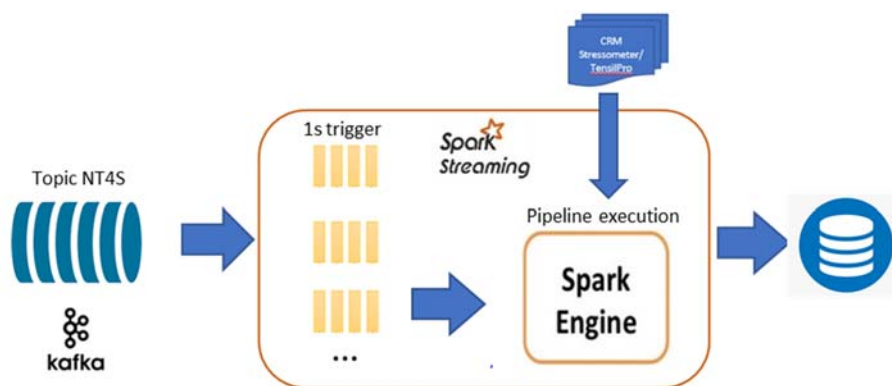


Figure 7: Implementation of the Spark Structured Streaming process

The implementation of the PySpark Structured Streaming script connects natively to the Kafka topic by means of the specific library spark-sql-kafka and foreach micro-batch of data prepare and integrate the variables necessary for the execution of the ANN model predicting the target process parameters. The script, finally, sink the predicted output directly to MySQL data-base for the online visualization in dedicated HMI.

### Tool assessment

Structured Streaming exhibits several advantages on streaming application but is also subject to certain weaknesses. The simple programming model and streaming API are countered by query limitation that restrict queries to supported operations. For this reason, we used the supplied function dedicated to specific user defined procedure but with the compromise that its operations are not parallelized at all but are managed principally by drivers rather than by executors.

## C++ based processing modules

Developed modules DataSetPreparation, ANN PrevisionMode, NT4S Classifier
--

C++ is a general-purpose programming language used to create two main components in the Marcegaglia use case batch computation and for the creation of the final dataset dedicated to the training of ML.

- The following image illustrates the summary of the step necessary to integrate and align all data from different sources.



- ✓ Rescale/Transform: Stressometer tension data are transformed to obtain IUnit values
- ✓ Data Alignment: data of different sampling on length position and width section of the coils are interpolated to obtain the same sample rate considering the material elongation and the head/tail cut, the latter by means of the weight of cut material.
- ✓ Missing values handling & Filter on noise: rows with missing values are dropped
- ✓ Outliers' detection and treatment: CRM Stressometer external sections are excluded because containing suspicious values.

- The results of the application are the prediction of the values of position guide (third guide of the looper) and position of centering device (third centering device of looper) that for the use case scope represent the indicators of the working condition for the next step in galvanization line. In particular the prediction of the two target variables is conducted on different settings of the group of process parameters then the results represent a list of matrices that are the

behavior predictions, in the centering device 3, of the entire strip length of each roll, simulating the main set of line setups

- The third component called *NT4S\_Classifier* is the procedure that assign a coil global index given the percentage thresholds defined for specific conditions of a combination of the target variables predicted with the *ANN\_PreviousMode* application. The module has the task of classifying each roll processed by the *ANN\_PreviousMode* through a severity index as a function of the percentages of stretch of strip for each severity index. The final result is a coil global index value in range from 0 to 3 which increasingly represents the criticality of the next galvanization phase under the predefined process conditions..

#### Tool implementation

The last two described software interacts with the database MySQL for the access to the configuration parameters and for the writing of results to make them available to the visualization web application.

#### Tool assessment

The time duration of the execution of the tool *ANN\_PreviousMode*, the ones that integrates the ML model, on a single coil is on the order of tenth of second. It depends on the length of the processed coil that can be in the range 4k until 7k meters. Instead, the performance of the *NT4S\_Classifier* are much shorter, processing times can last from 2 to 4 minutes to be able to produce the results of a single coil. This happens because it has to elaborate the global criticality for the coil but also the slices of subsequent criticality index for each coil process predefined setup and make them available for the visualization system. The values 0-3 of the criticality in visualization are mapped with colours to gives a glance on the specific value.

### **Matplotlib**

#### General tool information

Matplotlib is a plotting library for data visualisation in Python. Matplotlib only uses BSD compatible code, and its license is based on the PSF license, for download see <https://matplotlib.org/downloads.html>

#### Tool/system description

Matplotlib offers static, animated, and interactive visualisations in Python. With its module PyPlot it offers a MATLAB-like interface which may help some users with existing Matlab knowledge. There exist a mass of plot types and functions for free use.

#### Tool implementation

Matplotlib can be installed easily like other Python libraries via supported installers in the used IDE

#### Tool assessment

Matplotlib works excellent with all common numerical data formats in Python, but best with types coming from NumPy library. Here one can find possibilities to accelerate the plotting of huge data enormously.

### **Apache Spark/PySpark**

#### General tool information

Apache Spark, licensed under the Apache License 2.0, download at <https://spark.apache.org/downloads.html>

#### Tool/system description

Apache Spark is an open-source unified analytics engine for large-scale data processing. It is multi-language for executing data engineering, data science, and machine learning on single-node machines or clusters.

In context of cold rolling use case the Spark programming model has been used by means of PySpark, an open-source library for the Python programming language that allows to write Spark applications using Python APIs.

It supports most of Spark's features such as Spark SQL, DataFrame, Streaming, MLlib (Machine Learning) and Spark Core making possible the analysis of BigData in a distributed environment. In cold rolling use case, PySpark library has been used to access and elaborate the files into the Hadoop datalake saved by the ingestion process. It allowed mainly to query and aggregate large data during the development of procedure related to pre-processing step for the preparation of dataset dedicated to the training.

#### Tool implementation

The installation of PySpark on the development server, required the installation of jdk, the setting of JAVA\_HOME variable and the set of PATH variable to the location of Java bin file. The pySpark package has been installed by means of pip utility and finally, a new environment variable named HADOOP\_HOME to the location of the bin Hadoop folder was inserted.

#### Tool assessment

Running Spark jobs over HDFS data requires the setup of the HADOOP\_CONF\_DIR environment variable otherwise accessing HDFS file you might see errors.

### **2.2.7. Data Processing – modelling**

#### **Keras**

##### General tool information

Keras, licensed under the MIT license, is part of the TensorFlow library, so download at <https://www.tensorflow.org/install>

##### Tool/system description

Keras is the most used deep learning framework built on top of TensorFlow. It makes easier and faster to run experiments and can scale to large clusters of GPUs. In context of cold Rolling use case has been adopted for the definition, training and testing of the Neural Network models designed to the application of batch and speed layer processing.

##### Tool implementation

The installation of Keras requires Python3.6-3.9 and TensorFlow 2 package. Attention must be paid also for the compatibility between Keras and TensorFlow versions because some functionality may not work. Compatible operating systems are Ubuntu 16.04 or later, Windows 7 or later, macOS 10.12.6 (Sierra) or later.

Modelling with Keras, involved different steps of training. Apart the standard split of the available dataset for the testing and validation of data, a further data segregation has been applied to create cluster of data to train specific model and further validate how it performs against new data. Below the results of the different schemes of Back Propagation Neural Network considering also the clusterization of data. The table shows the different performances of trained ANN with different set of input variables and target variables considering that our main objective was the prediction of guide 3 and centering device 3 of the looper. The last four cases, as explained before, represent the results of training/evaluation on cluster of coils with same condition (performance grow in these cases).

	R2- Guida	R2- Centr.	INPUT							OUTPUT					
TF-ANN-0	0.750	0.012	SM	YS e UTS	Iba					Guida1	Centr.1				
TF-ANN-1	0.816	0.079	SM	YS e UTS	Iba	Guida1	Centr.1					Guida2	Centr.2		
TF-ANN-2	0.884	0.781	SM	YS e UTS	Iba	Guida1	Centr.1	Guida2	Centr.2					Guida3	Centr.3
TF-ANN-3	0.824	0.678	SM	YS e UTS	Iba	Guida1	Centr.1							Guida3	Centr.3
TF-ANN-3_0	0.81	0.663	SM	YS e UTS	Iba									Guida3	Centr.3
ANN-2 Cond-1	0.946	0.071	SM	YS e UTS	Iba	Guida1	Centr.1	Guida2	Centr.2					Guida3	Centr.3
ANN-2 Cond-2	0.902	0.035	SM	YS e UTS	Iba	Guida1	Centr.1	Guida2	Centr.2					Guida3	Centr.3
ANN-2 Cond-3	0.958	0.863	SM	YS e UTS	Iba	Guida1	Centr.1	Guida2	Centr.2					Guida3	Centr.3
ANN-2 Cond-4	0.944	0.905	SM	YS e UTS	Iba	Guida1	Centr.1	Guida2	Centr.2					Guida3	Centr.3

Currently in the batch prediction process, the cascade of network TF-ANN-0 and TF-ANN-3 is used to return at the output from the reversible rolling mill line the behavior forecast of the centering device 3 of the zinc painting line. In the Streaming mode, the TF-ANN-3 network is used, to give the operator the simulation of the centering device 3 in real time during the processing of the roll

#### Tool assessment

Keras includes a user-friendly API and it is very easy to create neural network models but is lacking in pre-processing functions. For this scope other libraries such as Scikit learn are more complete.

## 2.2.8. Application for online visualisation

### Dedicated NT4S HMI

#### Tool/system description

The dedicated NT4S HMI is a specialized web application to visualize the produced results in real-time processing modules. The data are provided by means the Spark Structured Streaming application running the ML model over the online production process. The produced results are stored on datastore with *foreachbatch* output sink, and the HTML/PHP web application is in continuous polling for new data to visualize. The visualized data are related to the online criticality of each coil under galvanization process. In details it shows the current length coil and the elaborated criticality colour until the reached galvanized length.

#### Tool implementation

The application was developed in PHP a server-side scripting language residing in our case on Apache HTTP Server. In the execution phase, it interprets the information received from the client (browser) and thanks to the Web server, processes it and returns a result to the client who made the request in a specific readable format.

### Apache HTTP Server

#### General tool information

The Apache HTTP Server, licenced under Apache License 2.0, download at <https://www.apachefriends.org/download.html>

#### Tool/system description

The Apache HTTP Server is a free and open-source cross-platform web server software, developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.

#### Tool implementation

Because Apache Friends developed XAMPP a free and open-source cross-platform web server solution stack package consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages, we optimized the installation directly using this solution. It integrates PHP language that is the solution adopted for the HMI development server side.

#### Tool assessment

Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server possible. It is simple to install and to configure thanks to the intuitive interface



## 2.3. Wire rod production at JSW

### Intention

At JSW wire rod rolling, the specific objective of the project was to improve the online supervision of selected processes and to forecast the wire rod profile and surface quality by applying performant new technologies.

Even if the plant was already equipped with high-resolution sensor systems and advanced online measuring systems providing for wire rod profile measurements and for the detection of different types of defects, relationships between dynamic process conditions of the rolling mill and quality features were not automatically identified online in the past. In fact, in the previous situation of the plant, when quality anomalies occur on products, plant operators were used to act manually on process parameters, such as stand rolling speeds and the height of loops, and the action to be performed was decided based on the arisen situation and the experience of the operators. No model was available for monitoring and predicting the wire rods quality in association with rolling conditions and the analyses were carried out by manually correlating the data.

Furthermore, high resolution process data were not directly exploitable in real time. However, the IBA acquisition system acquired high resolution process data by storing them on the filesystem in proprietary iba files (ten minutes duration) and a very powerful tool, called iba Analyzer, was used by operators to visualize process signals.

The wire rod rolling plant produces wire rod coils with profile diameter in range 5.5÷17 mm, covering different steel families, from low/medium carbon steels for drawing and cold rolling to other proper steels for wire rod for fasteners, nuts and bolts and for welding wire. The main components of the plant are a reheating furnace, a rolling mill and a Stelmor bed for wire rod air-cooling. The mill is a two-line continuous rolling mill, composed by rolling stands which are first shared between the two lines (roughing and first intermediate stands) and then separated on each line (second intermediate and finishing stands).

The use case aims to the development of analytical models for online detection of operational conditions in process data that can influence the rolled wire rod quality, in terms of profile and surface deficiencies, to enhance process stability and enable timely counter measurements.

In the wire rod rolling mill, data are collected online from different data sources by means of various automatic systems: IBA acquisition system collecting high-sampled process data, LAP Laser instrument providing profile measurements and Wire Check inspection system for surface defects detection. LAP Laser and Wire Check systems are positioned on the rolling mill at the end of each rolling line and their data are provided all at once per product at the end of the rolling process. Moreover, production data, concerning general characteristics and other information of products, are stored in the Level2-3 MES database of the plant, as well as some process data acquired via OPC/UA at specific events per product. Figure 9 shows a summary of available data sources and related details.

<b>PLC data from IBA System</b> <ul style="list-style-type: none"><li>• ~ 1000 process parameters</li><li>• sample rate 10 ms</li><li>• stored in files on IBA server filesystem / streamed to Kafka</li></ul>	<b>LAP Laser data</b> <ul style="list-style-type: none"><li>• wire rod profile and temperature</li><li>• sample rate 100 ms</li><li>• stored on SQL Server database</li></ul>	<b>Wire Check data</b> <ul style="list-style-type: none"><li>• defect details: size, position, classification and image</li><li>• stored in files on Wire Check server file system (<i>proprietary format</i>)</li></ul>
<b>Lev2-Lev3 PLC data (via OPC/UA)</b> <ul style="list-style-type: none"><li>• process parameters per billet</li><li>• at specific event</li><li>• stored on Oracle MES database</li></ul>	<b>Production Data</b> <ul style="list-style-type: none"><li>• wire rod coil data, operative practices, rolling equipment, laboratory tests etc..</li><li>• per billet</li><li>• stored on Oracle MES database</li></ul>	

Figure 9: Available data sources at JSW wire rod rolling mill

Data acquisition frequency varies from 100 Hz in case of rolling process data (like stand rolling forces, speeds and currents) to 10 Hz for wire rod profile measurements and finally 1 per product for production data and some process data acquired via OPC/UA.

The architecture implemented at JSW wire rod rolling plant is composed by on-premise and cloud infrastructures both used for computing and data storage. This hybrid IT structure combines the benefits of the security of on-premise systems and the flexibility of a cloud-based environment, with cost reduction and scalability advantages.

A conceptual sketch of the architecture is shown in Figure 10. The scheme separates the on-premise infrastructure (below dashed line) from the cloud infrastructure (above dashed line), giving an overview of the main tools used.

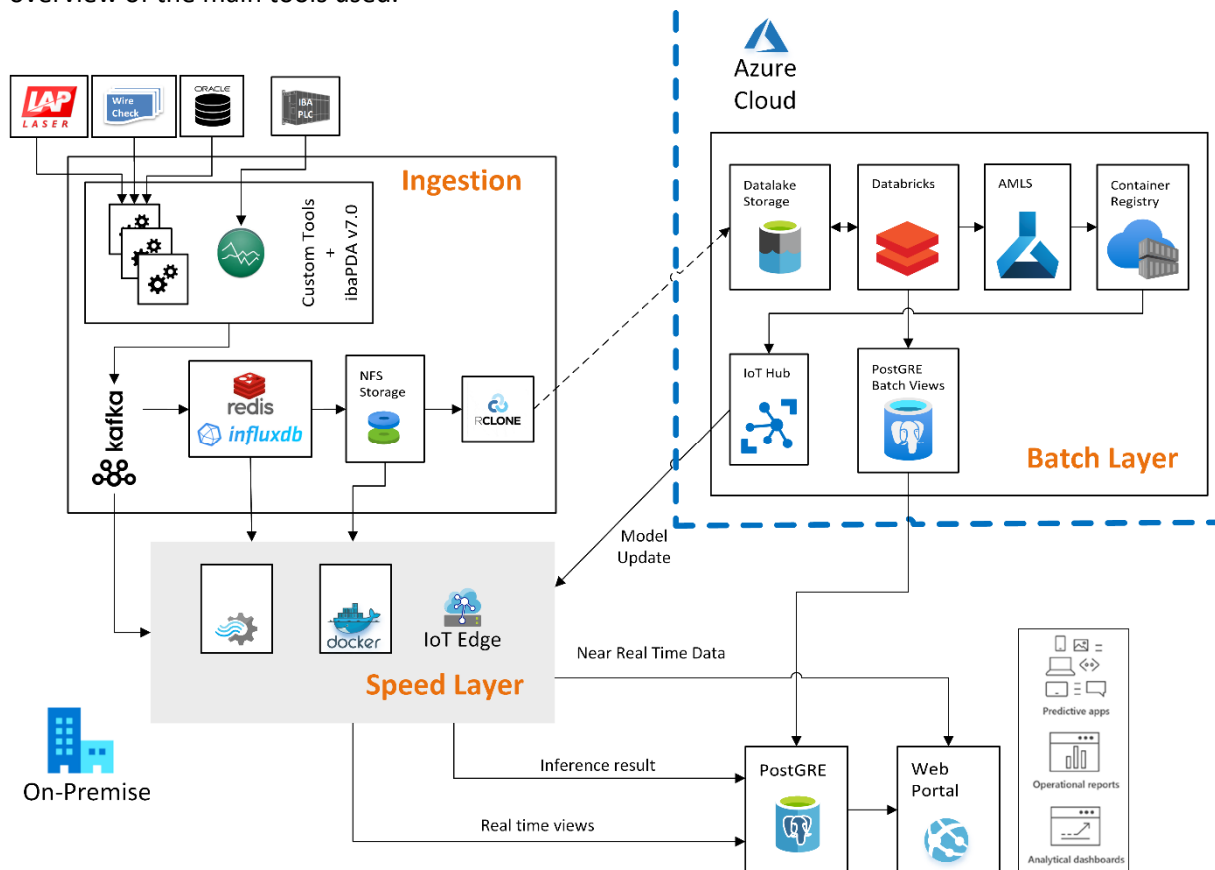


Figure 10: Architecture scheme at the wire rod rolling use case

The main activities performed by the on-premises infrastructure are the data ingestion, the inference of ML models to make predictions and the visualization of information to users through a suitable HMI, while the cloud infrastructure provides tools for historical data storage, data preparation, ML model training, validation, test and deploy.

During the data ingestion phase, quality data supplied by LAP Laser and Wire Check systems are collected by two specialized modules that decode them from their proprietary format for immediate fruition and send transformed data to the Kafka broker. On the other hand, production data and the L2/L3 PLC data (via OPC/UA) are directly sent to Kafka without any prior transformation, after being extracted from the MES database by means of another specialized module.

Moreover, the process data are continuously acquired by the IBA system and automatically sent to Kafka. A proper time-series database, InfluxDB, is used as a short-time buffer to store recent continuous data to be sent to the Speed Layer, for further speed processing, and to be loaded as historical data to the cloud for further batch processing. Before being transferred to the cloud, data are stored on the Network File System (NFS) and then packed in order to reduce communication latencies and optimize the throughput.

In the Speed Layer ML models are used to make predictions aimed at discovering operating conditions that can be risky for the quality of the product. Results of predictions are stored in a

database and can be displayed to users through dash applications viewed in the web browser. Moreover, process signal monitoring is supported through Grafana dashboards showing real-time streaming data from InfluxDB data source.

In the Batch Layer, recent data, loaded in batch from the on-premises infrastructure, are stored on the cloud in a historical data storage. Historical data are then conveniently prepared for the training and validation of ML models that will be deployed and used in the speed layer to serve the inference.

### 2.3.1. General tools for system organisation, administration and supervision

#### Kubernetes (K8s)

##### General tool information

**Kubernetes** (K8s) is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

Kubernetes, licensed under the Apache Licence 2.0, download at <https://kubernetes.io/releases/download/>

##### Tool/system description

Kubernetes provides you with:

**Service discovery and load balancing** Kubernetes can expose a container using the DNS name or IP address. If traffic to a container is high, Kubernetes can load balance and distribute the network traffic to stabilise the deployment.

**Storage orchestration**

Kubernetes allows you to automatically mount a storage system, such as local storage, public cloud providers, and more.

**Automated rollouts and rollbacks**

You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.

**Automatic bin packing**

You provide Kubernetes with a cluster of nodes which run containerised tasks. You can tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.

**Self-healing**

Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

**Secret and configuration management**

Kubernetes let you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images and exposing secrets in your stack configuration.

In the architecture implemented at the wire rod rolling, Kubernetes was exploited as an orchestrator that allows managing container lifecycle and all resources inside the cluster.

## Tool implementation

Kubernetes Cluster Requirements			
Role	Minimal required memory	Minimal required CPU (cores)	Components
Master node	2 GB	1.5	Kublr-Kubernetes master components (k8s-core, cert-updater, fluentd, kube-addon-manager, rescheduler, network, etcd, proxy, kubelet)
Worker node	700 MB	0.5	Kublr-Kubernetes worker components (fluentd, DNS, proxy, network, kubelet)
Centralised monitoring agent *	2 GB	0.7	Prometheus. We recommend limiting 2GB for a typical installation of a managed cluster with 8 workings and 40 pods per node with a total of 320 nodes. The retention period for the Prometheus agent is 1 hour.

## Tool assessment

All the services performing data ingestion (see below) are deployed as containers using Kubernetes manifests.

Different tools were used to manage and check Kubernetes cluster status.

**Rancher** (open-source) provides an interface that allows to manage the Kubernetes cluster and execute all the commands that you previously executed via the command line for example:

- nodes creation
- pods/Deployments/Jobs deploy and monitoring
- shell interface

In addition to Rancher, **Grafana** dashboards and **Loki logging** were exploited to verify the correctness and the performances. Some useful dashboards are available to display CPU, RAM and bandwidth utilizations details.

Using tools like Kubernetes and Rancher, it was possible to manage all cluster resources using command line or UI.

An important aspect concerns the nodes balancing. In JSW architecture, 5 nodes were provided : 1 master and 4 workers. There are pods that use different resources so it's important to specify which pods can be executed on specific nodes. In this way we can balance all the cluster load, for example isolating, in a specific node, a pod that requires a lot of resources.

### **Pros of Using Kubernetes**

- It has self-healing abilities. It's important to know that, by default, K8s do come with self-healing abilities, but only for pods. But a Kubernetes solution provider may have further integration of self-healing layers to ensure application reliability.
- K8s draw underlying computing resources and allow developers to deploy workloads to the entire cluster and not just a particular server.
- An administrator can single-handedly manage and monitor several simultaneously running containers. Also, K8s, in general, have minimal to no performance overhead.
- Kubernetes enables workload portability without limiting the types of applications it supports. Any application that a container can run, Kubernetes can too.
- It is an extensible platform. In January 2018, the Storage Orchestrator Runtime for Kubernetes was announced. STORK helps users run and monitor applications without disturbing their state more efficiently.
- Or a swift load-balancing, K8s provide individual IP addresses for every pod and a single DNS name for a set of pods.

- Kubernetes is a portable and cost-effective platform. It demands lesser computing resources to operate. Furthermore, you can set max and minimum for the CPU and memory resources accordingly for your containers.

#### **Cons of Using Kubernetes**

- Kubernetes can be oversized, especially considering the local developments and simple applications.  
This could affect efficiency and negatively affect the pre-set timelines of the organisation as a whole. K8s may become complex and unnecessary for a local development environment that could potentially diminish productivity.
- Making way across Kubernetes, the landscape could confuse new users because of its constant innovation and too many additions. The transition to Kubernetes can become slow, complicated, and challenging to manage.
- Kubernetes has a steep learning curve. It is recommended to have an expert with a more in-depth knowledge of K8s on your team, which could be expensive and hard to find.
- Adapting to a new system with many integrations and new technologies could be overwhelming for some deployment processes.
- Kubernetes requires experience and extensive training for its debugging and troubleshooting in due time.
- The benefits of K8s are in abundance but reaching that level might consume a lot of time, effort, and resources. Teams need to re-plan their time investing and familiarising themselves with new processes and workflow.

#### References

<https://kubernetes.io/releases/download/>

<https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>

<https://docs.kublr.com/installation/hardware-recommendation/>

#### **Redis**

##### General tool information

Redis is an open-source (BSD licensed) in-memory data structure store used as a database, cache, message broker, and streaming engine. Redis provides data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperlogs, geospatial indexes, and streams. Download at <https://redis.io/download/>

##### Tool/system description

Redis is used for multiple purposes.

##### **Session Cache**

One of the most apparent use cases for Redis is using it as a session cache. The advantage of using Redis over other session stores, such as Memcached, is that Redis offers persistence. While maintaining a cache isn't typically mission-critical regarding consistency, most users wouldn't exactly enjoy it if all their cart sessions went away, now would they?

Luckily, with the steam Redis has picked up over the years, it's pretty easy to find documentation on using Redis appropriately for session caching. Even the well-known eCommerce platform Magento has a plugin for Redis!

##### **Full Page Cache (FPC)**

Redis provides a straightforward FPC platform to operate outside of your basic session tokens. Going back to the consistency, even across restarts of Redis instances, with disk persistence, your users won't see a decrease in speed for their page loads—a drastic change from something like PHP native FPC.

To use Magento as an example again, Magento offers a plugin to utilise Redis as a full page cache backend.

For your WordPress users out there, Pantheon has an excellent plugin named wp-redis to help you achieve the fastest page loads you've ever seen!

### Queues

Taking advantage of Redis' in-memory storage engine to-do list and set operations makes it an amazing platform to use for a message queue. Interacting with Redis as a queue should feel native to anyone using push/pop operations with lists in programming languages such as Python.

If you do a quick Google search on "Redis queues," you'll soon see tons of open-source projects out there to make Redis an awesome backend utility for all your queuing needs. Celery, as an example, has a backend using Redis as a broker that you can check out here.

### Leaderboards/Counting

Redis does an amazing job at increments and decrements since it's in-memory. Sets and sorted sets also make our lives easier when trying to do these kinds of operations, and Redis just so happens to offer both of these data structures.

### Publish/Subscribe

Last (but certainly not least) is Redis's Pub/Sub feature. The use cases for Pub/Sub are truly boundless. I've seen people use it for social network connections, triggering scripts based on Pub/Sub events, and even a chat system built using Redis Pub/Sub! (No, really, check it out.)

## Tool implementation

We recommend these hardware requirements for production systems or for development systems that are designed to demonstrate production use cases:

Item	Description	Minimum Requirements	Recommended
Nodes per cluster*	At least three nodes are required to support a reliable, highly available deployment that handles process failure, node failure, and network split events in a consistent manner.	3 nodes	>= 3 nodes (Must be an odd number of nodes)
Cores* per node	RS is based on a multi-tenant architecture and can run multiple Redis processes (or shards) on the same core without significant performance degradation.	4 cores	>=8 cores
RAM* per node	Defining your RAM size must be part of the capacity planning for your Redis usage.	15GB	>=30GB
Ephemeral Storage	Used for storing <a href="#">replication files (RDB format)</a> and <a href="#">cluster log files</a> .	RAM x 2	>= RAM x 4
Persistent Storage	Used for storing <a href="#">snapshot (RDB format)</a> and <a href="#">AOF files</a> over a persistent storage media, such as AWS Elastic Block Storage (EBS) or Azure Data Disk.	RAM x 3	In-memory >= RAM x 6 (except for <a href="#">extreme 'write' scenarios</a> ); <a href="#">Redis on Flash</a> >= (RAM + Flash) x 5.
Network	We recommend using multiple NICs per node where each NIC is >100Kbps, but RS can also run over a single 1Gbps interface network used for processing application requests, inter-cluster communication, and storage access.	1G	>=10G

## Tool assessment

### **Advantages of using Redis:**

Redis has excellent read-write performance, fast IO read-write speed from memory, and supports read-write frequencies of more than 100k + per second.

Redis supports strings, lists, hashes, sets, ordered sets and other data type operations.

Redis supports data persistence, AOF and RDB.

Redis supports master-slave replication. The host will automatically synchronise the data to the slave, allowing read-write separation.

All Redis operations are atomic, and redis also supports atomic execution after full consolidation of several operations.

Redis is a single thread multi CPU, which is faster. Because of a single thread, there is no overhead of thread switching. There is no need to consider adding and releasing locks, so there is no deadlock problem. Single thread multiplex IO model. efficient.

### **Disadvantages of using Redis:**

Master-slave synchronisation and data synchronisation will be delayed. If the host goes down and some data is not synchronised to the slave before the shutdown, the data will be inconsistent.

It isn't easy to support online capacity expansion. Online capacity expansion will become very complex when the cluster capacity reaches the upper limit. When the system goes online, enough space must be ensured, which causes a great waste of resources.

## References

### **Libraries:**

In .NET developments we use "StackExchange.Redis" v2.2.4 library  
(<https://github.com/StackExchange/StackExchange.Redis/>).

### **Docs:**

<https://www.hitechnectar.com/blogs/pros-cons-kubernetes/>

<https://developpaper.com/introduction-advantages-and-disadvantages-of-redis-database/>

## **RClone**

### General tool information

Rclone is a command-line program to manage files on cloud storage. It is a feature-rich alternative to cloud vendors' web storage interfaces. Over 40 cloud storage products support rclone, including S3 object stores, business & consumer file storage services, and standard transfer protocols.

It is released under , released under MIT license, download at <https://rclone.org/downloads/>

### Tool/system description

Rclone is a command line program to manage files on cloud storage. It is a feature rich alternative to cloud vendors' storage interfaces. Rclone has cloud equivalents to the Unix commands (e.g. rsync, cp, mv) and can be used by the command line, in scripts or via its API. Rclone really looks after your data, preserving timestamps and verifying checksums at all times ensuring the integrity of files. It can transfer data over limited bandwidth, intermittent connections or subject to quota allowing the transmission to be restarted from the last good file transferred. Where possible, rclone employs server-side transfers to minimise local bandwidth use and transfers from one provider to another without using local disk.

Rclone is mature, open-source software originally inspired by rsync and written in Go and is widely used on Linux, Windows and Mac.

In wire rod rolling use case, Rclone is used to synchronize files from on-premise to the cloud and specifically to upload created parquet files from the local file system (NFS) to Azure Blob Storage.

The services that upload files to Azure Blob Storage are Kubernetes Cron Jobs with RClone container image. Different Cron Jobs were created, each uploading a specific data flow type: LapLaser, WireCheck, ProductionData, IBA and IBAPosition.



In production environment, using Grafana dashboards and Loki logging, it was possible to verify the correctness and the performance.

#### Tool implementation

The tool is available for Linux, Mac and Windows operating systems.

Rclone is single executable (Rclone or rclone.exe on Windows) that you can simply download as a zip archive and extract into a location of your choosing. Afterwards, run rclone config for setup and copy the .config file in your USERPROFILE folder.

For Linux, please verify that the Go runtime was installed (<https://go.dev/doc/install>).

#### Tool assessment

Using RClone approach we have different ways to customize our operations. For example, we can specify the “—no-traverse” option during copy/move tasks that allows decreasing the writes in Azure Blob Storage that have a cost, or with the “—bwlimit upload:download” option we can set the bandwidth upload and download limits.

We upload approximately 1.2GB of data every hour. The bandwidth in the upload is limited to 5mps and the transfer time is 4 minutes.

The following table shows a comparison of Rclone with two other similar tools.

	Rclone	Storj	pCloud
Score	4.5/5	4/5	4/5
Pros	Powerful command line interface (CLI) - Free and open-source - Works with over 40 cloud storage services	Encrypted and decentralized cloud storage - 150GB free plan - Open source	10GB free storage - Easy to use - File versioning tool - Multiple device two-way syncing
Cons	Steep learning curve - No customer support - No cloud storage	Lack of collaboration options - Command line interface can be difficult to learn	Limited customer support - End-to-end encryption is a paid add-on
Verdict	Rclone is made for power users who want total control over their data. Free and open-source, Rclone requires some upfront work and knowledge acquisition to unlock its full potential.	Storj offers best-in-class decentralized cloud storage at a reasonable price. Its robust command line interface is perfect for seasoned data professionals but may be difficult for the average user.	pCloud is a great cloud storage option for the everyday user. However, the lack of real-time collaboration capabilities and limited customer support is a common complaint.

Sometimes the transmission of files via RClone failed and produced corrupted files. A reason that RClone seems to produce this problem perhaps due to concurrent execution of tasks. The source files are not influenced.

#### References

<https://rclone.org/>

Comparison of Rclone with other tools: <https://www.itproportal.com/reviews/rclone-review/>

## Rancher

### General tool information

Rancher is an open-source software platform that enables organisations to run containers in production. With Rancher, organisations no longer have to build a container services platform from scratch using a distinct set of open-source technologies. Rancher supplies the entire software stack needed to manage containers in production.

### Tool/system description

Rancher is a complete software stack for teams adopting containers. It addresses the operational and security challenges of managing multiple Kubernetes clusters while providing DevOps teams with integrated tools for running containerised workloads.

### Tool implementation

#### **Hardware Requirements**

The following sections describe the CPU, memory, and disk requirements for the nodes where the Rancher server is installed.

#### **CPU and Memory**

Hardware requirements scale based on the size of your Rancher deployment. Provision of each individual node according to the requirements. The requirements are different depending on whether you are installing Rancher in a single container with Docker or installing Rancher on a Kubernetes cluster.

#### **RKE and Hosted Kubernetes**

These CPU and memory requirements apply to each host in the Kubernetes cluster where the Rancher server is installed.

These requirements apply to RKE Kubernetes clusters, as well as to hosted Kubernetes clusters such as EKS.

DEPLOYMENT SIZE	CLUSTERS	NODES	VCPUS	RAM
Small	Up to 150	Up to 1500	2	8 GB
Medium	Up to 300	Up to 3000	4	16 GB
Large	Up to 500	Up to 5000	8	32 GB
X-Large	Up to 1000	Up to 10,000	16	64 GB
XX-Large	Up to 2000	Up to 20,000	32	128 GB

### Tool assessment

**Advantages:** The undoubted advantage is the availability of the catalogue of ready-to-run services and applications. Rancher also has a straightforward web interface and can quickly launch other orchestrating solutions like Kubernetes and Docker Swarm.

**Disadvantages:** Still, Rancher is not an easy-to-use platform. To deploy your application with it, at a minimum, you must be able to create Docker images, which means that you have a specific technology entry limit.

### References

<https://d2c.io/post/comparison-docker-swarm-kubernetes-rancher-and-d2c#:~:text=Advantages%3A%20The%20undoubted%20advantage%20is,easy%2Dto%2Duse%20platform.>

## Loki

### General tool information

Loki is a horizontally scalable, highly available, multi-tenant log aggregation system inspired by Prometheus. It is designed to be very cost effective and easy to operate. It does not index the contents of the logs but rather a set of labels for each log stream.

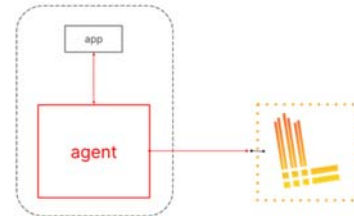
### Tool/system description

Grafana Loki is a log aggregation tool, and it is the core of a fully-featured logging stack.

Loki is a data store optimised for efficiently holding log data.

The efficient indexing of log data distinguishes Loki from other logging systems. Unlike other logging systems, a Loki index is built from labels, leaving the original log message unindexed.

An agent (also called a client) acquires logs, turns the logs into streams, and pushes the streams to Loki through an HTTP API. The Promtail agent is designed for Loki installations, but many other Agents seamlessly integrate with Loki.



Loki indexes streams. Each stream identifies a set of logs associated with a unique set of labels. A quality set of labels is key to creating an index that is both compact and allows for efficient query execution.

LogQL is the query language for Loki.

#### Loki features

Efficient memory usage for indexing the logs

The index can be significantly smaller than other log aggregation products by indexing on a set of labels. Less memory makes it less expensive to operate.

#### Multi-tenancy

Loki allows multiple tenants to utilise a single Loki instance. The data of distinct tenants are completely isolated from other tenants. Multi-tenancy is configured by assigning a tenant ID to the agent.

#### LogQL, Loki's query language

Users of the Prometheus query language, PromQL, will find LogQL familiar and flexible for generating queries against the logs. The language also facilitates the generation of metrics from log data, a powerful feature that goes well beyond log aggregation.

#### Scalability

Loki can be run as a single binary; all the components run in one process.

Loki is designed for scalability, as each of Loki's components can be run as microservices. The configuration permits scaling the microservices individually, allowing flexible large-scale installations.

#### Flexibility

Many agents (clients) have plugin support. This allows a current observability structure to add Loki as their log aggregation tool without needing to switch existing portions of the observability stack.

#### Grafana integration

Loki seamlessly integrates with Grafana, providing a complete observability stack.

### Tool implementation

#### Supported operating systems

The following operating systems are supported for Grafana installation:

- Debian / Ubuntu
- RPM-based Linux (CentOS, Fedora, OpenSuse, RedHat)
- macOS
- Windows

Installation of Grafana on other operating systems is possible, but it is neither recommended nor supported.

#### **Hardware recommendations**

Grafana does not use a lot of resources and is very lightweight in use of memory and CPU.

Minimum recommended memory: 255 MB Minimum recommended CPU: 1

Some features might require more memory or CPUs. Features that require more resources include:

- Server-side rendering of images
- Alerting
- Data source proxy

#### **Supported databases**

Grafana requires a database to store configuration data, such as users, data sources, and dashboards. The exact requirements depend on the size of the Grafana installation and the features used.

Grafana supports the following databases:

- SQLite
- MySQL
- PostgreSQL

By default, Grafana installs with and uses SQLite, an embedded database stored in the Grafana installation location.

#### Tool assessment

Pros:

- Open source
- High-speed ingestion
- Low resource footprint
- REST API
- Perfect fit for Kubernetes

#### References

<https://grafana.com/oss/loki/>

### **Prometheus**

#### General tool information

Prometheus is designed to track overall system health, behaviour and performance rather than individual events.

It discovers targets to scrape from service discovery. These can be instrumented applications or third-party applications that can be scraped via an exporter.

The scraped data is stored and can be accessed in a dashboard using PromQL.

#### Tool/system description

Prometheus is a free software application used for event monitoring and alerting. It records real-time metrics in a time series database built using an HTTP pull model, with flexible queries and real-time alerting. Its features are:

- Gathering of data through pull model over HTTP.
- A multidimensional data management module based on time series (metric name and key/value pairs).
- PromQL: an oriented language used for time series queries.
- Every single instance is autonomous (server contained in a single binary) and saves data directly on local storage.
- Dashboard support based on Grafana.
- Real-time warnings.

Typologies of logs that can be provided by the web portal are:

- Logs about applications software.

- Logs about services.
- Logs about the infrastructure of servers/clusters.

The web portal wants to show not only the raw logs but to even apply a BI correlating all logs, transforming the data logs into information.

### Tool implementation

The minimal requirements for the host deploying the provided examples are as follows:

- At least 2 CPU cores
- At least 4 GB of memory
- At least 20 GB of free disk space

### Tool assessment

#### **Writing applications to be monitored.**

Developers are often required to make their applications observable, even if they aren't the ones who'll be actively monitoring them once they go live.

Pros:

- Some excellent client libraries that allow you to add Prometheus metrics to your code, which deal with creating a /metrics endpoint and serving the data upon request. There are also guidelines for writing exporters.

Cons:

- Client libraries mean additional dependencies to keep track of. If you aren't maintaining your monitoring server, you'll require a Prometheus administrator to configure a job to retrieve your metrics.
- The Prometheus server needs access to connect with the server or container your application runs on. Ideally, it should be discoverable via service discovery, all things which the application developer might not have direct control over.

#### **Monitoring a single environment, e.g. container cluster**

Pros:

- Depending on the environment, there is a variety of monitoring agents which support Prometheus, including cAdvisor, which is one of the main tools for monitoring Docker containers.
- Getting the list of endpoints to scrape can be handled by service discovery - Prometheus was built for Kubernetes, and the containers in k8s pods and nodes are continually changing, but k8s can provide a list of the endpoints at any given moment.
- Scrape jobs mean if the metrics stop flowing, it can easily discover whether the endpoint has stopped responding - endpoint states are constantly monitored.
- Access to the endpoints is handled by running Prometheus locally, so no external firewall issues will likely arise.

Cons:

- Service discovery is supported for a limited number of services, but you're required to either use DNS-based discovery or generate a file with its details. That's a whole specialised service to create and maintain if you're not monitoring something standard.
- Either limited timeframe metric storage or using resources in your environment.
- No redundancy if there's an issue with the cluster since everything is running within it. Running outside the cluster may incur additional costs and/or have access issues for services within containers.
- It's difficult to run as a service for other teams across a global organisation. This is covered in the section below (monitoring multiple environments/locations).

#### **Monitoring multiple environments/locations**

Pros:

- All the benefits of a single environment setup.
- Prometheus installation and configuration can be automated easily, and that makes this kind of multi-site setup easy to maintain.

#### Cons:

- Often, environments need to be logically separated. Since Prom needs access to make HTTP connections into each service, it usually needs to be run inside the same network as the service.
- It's possible to consolidate metrics in one place using the Prometheus federation, but that usually means downsample or aggregating metrics since the volume of data being transferred can easily become too much to transfer it all to a central point.

#### References

<https://prometheus.io/>

<https://www.metricfire.com/blog/prometheus-or-graphite/#Pros>

### Zabbix

#### General information

Zabbix is an open-source monitoring software tool for diverse IT components, it is licensed under version 2 of the GNU General Public License (GPL), see <https://www.zabbix.com/>

#### Tool Description

Zabbix monitoring capabilities include networks, servers, virtual machines, and cloud services. Zabbix provides monitoring metrics, CPU load and disk space consumption. Zabbix can collect monitoring data from one or more monitored devices and send the information to the Zabbix server. Zabbix can be used to access the cluster and each server status data. Once the user is authenticated to the Zabbix service they can view Zabbix monitoring dashboards. The global view of Zabbix server that is running is shown below.

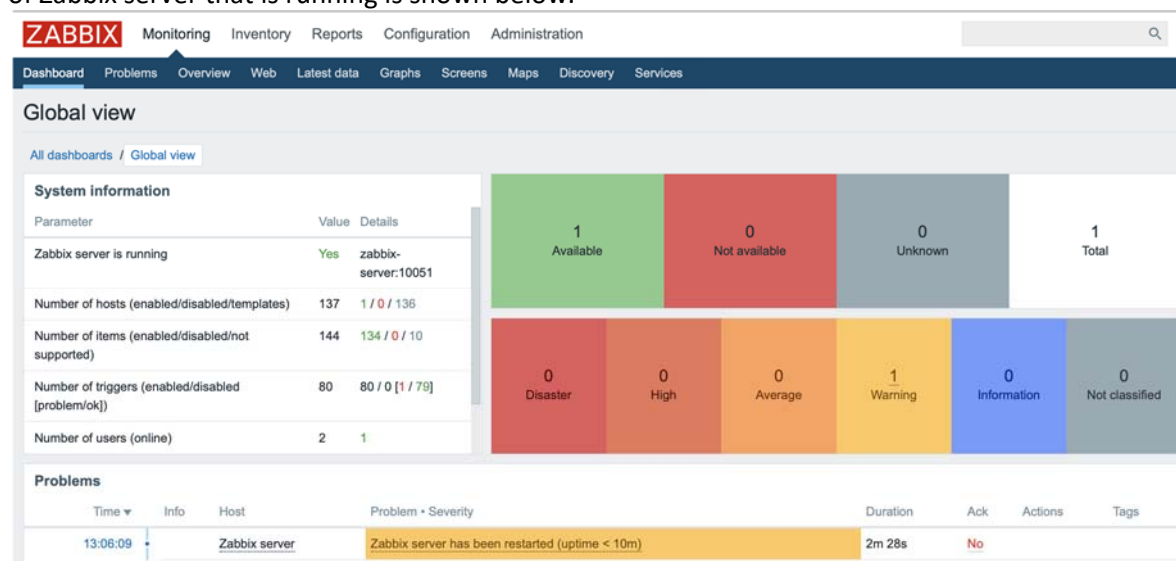


Figure 11: Zabbix server dashboard

Using the Zabbix tool bar, users can setup the “Host groups” and “Zabbix servers” to monitor, they can access available dashboards with the latest monitoring data. The application displays the list of the available dashboards for each host group and server to the user.

Dashboards display real time graphs of monitoring data, here's an example of the dashboard displaying the CPU utilization in the last 15 minutes in real time.

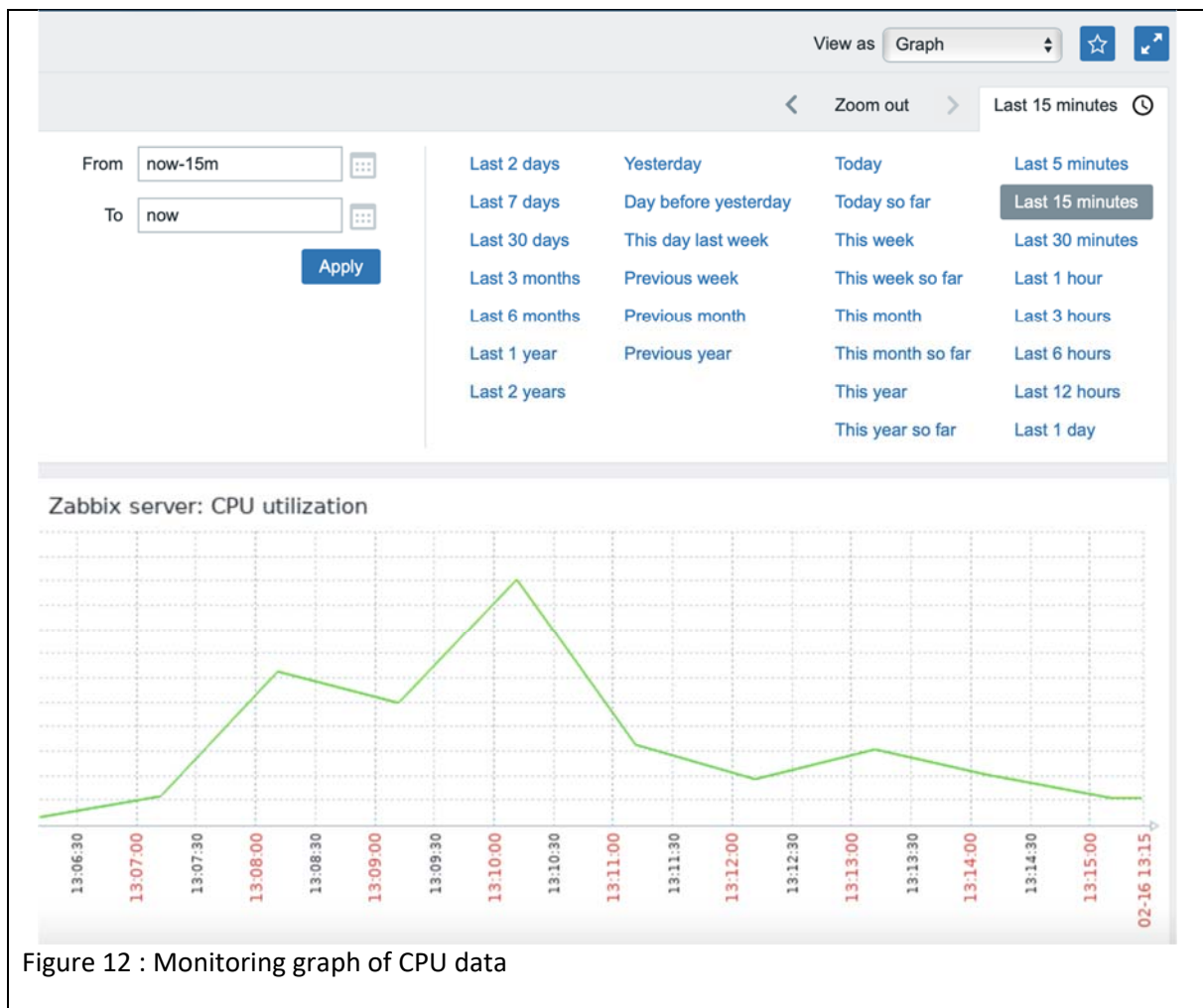


Figure 12 : Monitoring graph of CPU data

### Tool Assessment

A strong, adaptable, and configurable monitoring tool that can deeply monitors the entire infrastructure, services, applications as well as the entire network. One can easily add additional hosts, create a network map and monitoring a large number of operating systems, devices, and assets from a single interface.

### References

<https://www.zabbix.com/>

### 2.3.2. Development systems (IDEs)

#### Visual Studio 2019

##### General tool information

Visual Studio 2019, available under subscription, download at <https://visualstudio.microsoft.com/downloads/>

##### Tool/system description

Microsoft Visual Studio is one of the IDEs provided by Microsoft, used to develop desktop, web and mobile applications as well as websites and web services. Moreover, it allows to use different technologies provided by Microsoft platform like Windows Forms, Windows Store, Windows API, Windows Presentation Foundation and Microsoft Silverlight. Visual Studio supports 36 different programming languages and is available for Windows and macOS. It is released under a commercial licence nevertheless the Community edition is free for students, open-source contributors and individuals (Freemium licence).

In wire rod rolling use case, it was widely used in the data ingestion process for the development in C# language of modules such as LAPLaser, WireCheck and ProductionData services, transmitting data to Kafka, and also for subsequent modules collecting data from Kafka to InfluxDB and NFS.

##### Tool implementation

Install it using the Visual Studio installer by choosing those components you are interested in, otherwise the installation can become very heavy in terms of required memory.

##### Tool assessment

It is a very good development environment. Using NuGet, that is the package manager for .NET, it was possible to install and use all needed packages. It was used also to create and publish Docker Container images for modules directly from the IDE to the Azure Container Registry for deploy.

#### Visual Studio Code

##### General tool information

Visual Studio Code, licensed under the MICROSOFT SOFTWARE LICENSE TERMS, download at <https://code.visualstudio.com/Download>

##### Tool/system description

Visual Studio Code is a lightweight but powerful source code editor, made by Microsoft, that is free, open-source and runs on Windows, macOS and Linux. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. It comes with built-in support for JavaScript and many extensions are available for working with other languages such as C++, C#, Java, Python, PHP, Go and runtimes (such as .NET).

In wire rod rolling use case, it was used for the development of some specific parts, taking advantage of the python support:

- artificial reconstruction of billet signals that relate iba signals to the billet.
- data preprocessing module for features computation
- models inference module and components for the visualization of inference results

##### Tool implementation

It is a good development environment for writing code and debugging it and is very lightweight. It was exploited both on Windows and Linux machines.



## Jupyter notebook

### General tool information

The Jupyter Notebook is a web-based interactive computing platform, open-source under the modified BSD license, see <https://jupyter.org/>

### Tool/system description

Jupyter is a popular IDE in the form of a web application that allows to combine in a single document code, formatted textual description and visualize results (i.e tables, charts,...). These characteristics make it a suitable tool for data analytics and ML tasks. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed via the internet.

### Tool implementation

Jupyter Hub and Lab as well as the server app can be installed by the command shell or through package managers such as Anaconda (see references).

### References

<https://www.anaconda.com/>, a package including Python and R programming language as well as IDE Spyder, IPython interpreter and Jupyter notebook

## Spyder

### General tool information

Spyder is an open-source IDE dedicated to the development of Python code, open-source under the MIT License, download at <https://www.spyder-ide.org/>

### Tool/system description

Spyder is an open-source IDE dedicated to the development of Python code and oriented to scientific applications in the fields of engineering and data science. Spyder includes an editor, an interactive console (through IPython) and a powerful debugger together with facilities for the monitoring of variables and plots management. The IDE can be extended by adding some plugins developed either by Spyder team or community users.

### Tool implementation

Spyder IDE can be installed through a stand-alone installer under Windows and Linux OS, through the command shell or through package managers such as Anaconda.

### References

<https://www.anaconda.com/>, a package including Python and R programming language as well as IDE Spyder, IPython interpreter and Jupyter notebook

## PyCharm

### General tool information

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python programming language. PyCharm Community Edition is open-source and licensed under Apache 2, <https://www.jetbrains.com/pycharm/download/other.html>

### Tool/system description

PyCharm is developed by the Czech company JetBrains (formerly known as IntelliJ). It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django and data science with Anaconda.

### Tool implementation

PyCharm can be installed on Linux, Windows and macOS

### 2.3.3. Data ingestion

#### LAP Laser data service - JSW

##### Tool/system description

LAP Laser data service is a specialized module to ingest LAP Laser data. LAP Laser data are provided at the end of the rolling process by the LAP Laser instrument as blob files, one per product, in the database. This module was developed to collect these data from the data base as soon as they are stored, decode them from their proprietary format for immediate fruition and transmit transformed data to the Kafka broker, as json messages, into one topic: each json message wraps all profile and temperature measurements related to one product, originally supplied in the single blob file. The service has been designed to be very robust, recovering from failed transfers due to temporary networks problems or to timeouts. The module was developed in C# using .NET Core 3.1 and Visual Studio 2019 and is runnable both on Windows and Linux platform.

##### Tool implementation

The LAP Laser data service was installed as a Docker Container.  
**.NET Core** and **Docker**, a software package that creates containers, are required for the installation.  
To make it possible for LAP Laser specialized module to send big messages to Kafka, it was necessary to increase the message dimension on the topic and configure it also in the module.  
**Several Nuget** packages were exploited during the development: Oracle.ManagedDataAccess.Core to connect to the Oracle data base, Confluent.Kafka to manage the transmission to Kafka and Newtonsoft.Json to serialize C# objects to Json messages.

#### Wire Check data service - JSW

##### Tool/system description

Wire Check data service is a specialized module to ingest Wire Check data. Wire Check data are provided, at the end of the rolling process, by the Wire Check instrument on the Wire Check server filesystem by means of one single .def file containing all defect properties (position, classification etc..) of the product and different .bmp files for the defect images. This module was developed to collect these data from the filesystem as soon as they arrive, decode .def file from its proprietary format, wrap together information concerning single defects and finally send transformed data to the Kafka broker, as json messages. The ETL procedure produces different json messages, one for each defect (properties and compressed image) and one for a summary on the product defects. As the other specialized modules, the service has been designed to be very robust and it is able to recover from failed transfers due to temporary networks problems or to timeouts.  
The module was developed in C# using .NET Core 3.1 and Visual Studio 2019 and is runnable both on Windows and Linux platform.

##### Tool implementation

The Wire Check data service (one per line) was installed as a Docker Container and .NET Core and Docker are required for the installation.  
Several Nuget packages were exploited during the development: Oracle.ManagedDataAccess.Core to connect to the Oracle data base (to keep trace of transmissions status), Microsoft.Extensions.FileProviders.Physical to watch for arriving files on the filesystem, Confluent.Kafka to manage the transmission to Kafka and Newtonsoft.Json to serialize C# objects to Json messages.

##### Tool assessment

Even if everything worked fine in the development environment, the simple FileSystemWatcher library (from System.IO), originally used, was not working properly when the service container watches for files on a network drive outside of the cluster, as it happens in the production environment at JSW. For this reason, it was necessary to use PhysicalFileProviders (from Microsoft.Extensions.FileProviders package) instead.

## Production data service –JSW

### Tool/system description

Production data service is a specialized module to ingest Production data. Production data are provided during the rolling process on the Oracle MES database, one per product. This module was developed to collect these data from the data base as soon as they are stored and transmit them directly to the Kafka broker, as json messages. Different workers provide json messages for different categories of data. As the other specialized modules, the service has been designed to be very robust, recovering from failed transfers due to temporary networks problems or to timeouts. The module was developed in C# using .NET Core 3.1 and Visual Studio 2019 and is runnable both on Windows and Linux platform.

### Tool implementation

The Production data service was installed as a Docker Container and .NET Core and Docker are required for the installation. Several Nuget packages were exploited during the development: Oracle.ManagedDataAccess.Core to connect to the Oracle data base, Confluent.Kafka to manage the transmission to Kafka and Newtonsoft.Json to serialize C# objects to Json messages.

## 2.3.4. Data storage

### InfluxDB

#### General tool information

InfluxDB, licenced by InfluxData, download at <https://portal.influxdata.com/downloads/>

### Tool/system description

InfluxDB is an open-source time series database (TSDB) developed by InfluxData company, written in the Go programming language. It is purpose-built for storage and retrieval of time series data in various fields such as operations monitoring, application metrics, Internet of Things sensor data and real-time analytics. InfluxDB is a powerful engine that outperforms other commonly used technologies for time-series data (e.g. Cassandra, OpenTSDB and Elasticsearch) in terms of write performance, on-disk storage requirements and query performance and guarantees high data scalability and availability. It is available open-source, via the Cloud as a DBaaS option, or by an Enterprise subscription.

In wire rod rolling use case, influxDB is used to store recent continuous signals (time-series data), transmitted to the Kafka broker by the ibaPDA acquisition system. Data in InfluxDB are available for further speed processing and are loaded in batch as historical data to the cloud for further batch processing. Grafana can be directly connected to InfluxDB data source to show real-time streaming data, allowing process signal monitoring through dedicated dashboards.

### Tool implementation

Run InfluxDB on locally attached solid-state drives (SSDs). Other storage configurations have lower performance and may not be able to recover from minor interruptions in normal processing.

Estimated guidelines include writes per second, queries per second, and a number of unique series, CPU, RAM, and IOPS (input/output operations per second).

vCPU or CPU	RAM	IOPS	Writes per second	Queries* per second	Unique series
2-4 cores	2-4 GB	500	< 5,000	< 5	< 100,000
4-6 cores	8-32 GB	500- 1000	< 250,000	< 25	< 1,000,000
8+ cores	32+ GB	1000+	> 250,000	> 25	> 1,000,000

## Tool assessment

InfluxDB is a good time-series database (TSDb) for collecting sensor data with time stamps and the pros outweigh the cons, although it needs some time to become familiar with the flux language.

### **Pros:**

- open-source database.
- Windows, Linux and Docker version
- analysing data that requires accessing it over a window of time
- integration with external tools, such as Grafana
- designed for real-time application
- client libraries for the most popular programming languages

### **Cons:**

- not efficient to exporting big data
- limitation to writing high quantity data in a short time

In wire rod rolling scenario, 560 iba time series signals are sent to the Kafka broker, wrapped in json messages, by the ibaPDA acquisition system. The messages are distributed to a Kafka topic that is divided into 5 partitions and 5 consumers, each assigned to a single partition, are needed to keep up with the sample rate of data ingested into Kafka. Unexpectedly, it was also necessary to increase iba signals sample rate from 1ms to 10ms to prevent InfluxDB from being a bottleneck, demonstrating a limit of this tool.

Every 2 s, the producer sends 200 messages which will be distributed to the partitions.

Each consumer takes approximately 5 ms to insert a batch of messages every 2 s.

Furthermore, for InfluxDB management, the **NuGet** package used was “InfluxDB.Client”. Moreover, the monitoring of the performance was done by using Visual Studio debug tools during developing, while in the production environment, the correctness and the performance were verified using **Grafana** dashboards and **Loki logging**.

## References

<https://www.influxdata.com/>

More Pros&Cons at

[https://archive.docs.influxdata.com/influxdb/v0.9/concepts/insights\\_tradeoffs/](https://archive.docs.influxdata.com/influxdb/v0.9/concepts/insights_tradeoffs/)

<https://www.hitechnectar.com/blogs/pros-cons-kubernetes/>

Libraries:

In .NET developments we use “InfluxDB.Client” v2.1.0 library

(<https://github.com/influxdata/influxdb-client-csharp/tree/master/Client>).

Docs:

<https://www.hitechnectar.com/blogs/pros-cons-kubernetes/>

<https://wearecommunity.io/communities/india-java-user-group/articles/891>

## **Longhorn**

### General tool information

Longhorn delivers simplified, easy to deploy and upgrade, 100% open-source, cloud-native persistent block storage without the cost overhead of open core or proprietary alternatives.

Tool/system description

Longhorn is an official CNCF project that delivers a powerful cloud-native distributed storage platform for Kubernetes that can run anywhere. When combined with Rancher, Longhorn makes deploying highly available persistent block storage in your Kubernetes environment easy, fast and reliable.

## Tool implementation

Each node in the Kubernetes cluster where Longhorn is installed must fulfil the following requirements:

- Docker v1.13+
- Kubernetes v1.14+.
- open-iscsi is installed, and the iscsid daemon runs on all the nodes.

The host filesystem supports the file extents feature to store the data. Currently, we support:

- ext4
- XFS
- curl, findmnt, grep, awk, blkid, and lsblk must be installed.
- Mount propagation must be enabled.

## Tool assessment

Highly available persistent storage for Kubernetes

In the past, ITOps and DevOps have found it hard to add replicated storage to Kubernetes clusters. As a result, many non-cloud-hosted Kubernetes clusters don't support persistent storage. External storage arrays are non-portable and can be extremely expensive.

Longhorn delivers simplified, easy to deploy and upgrade, 100% open-source, cloud-native persistent block storage without the cost overhead of open core or proprietary alternatives.

Easy incremental snapshots and backups

Longhorn's built-in incremental snapshot and backup features keep the volume data safe in or out of the Kubernetes cluster.

Scheduled backups of persistent storage volumes in Kubernetes clusters are simplified with Longhorn's intuitive, free management UI.

Cross-cluster disaster recovery

External replication solutions will recover from a disk failure by re-replicating the entire data store. This can take days, during which time the cluster performs poorly and has a higher risk of failure.

Using Longhorn, you can control the granularity to the maximum, easily create a disaster recovery volume in another Kubernetes cluster and failover to it in the event of an emergency.

If your main cluster fails, you can bring up the app in the DR cluster quickly with a defined RPO and RTO.

## References

<https://longhorn.io/#:~:text=Using%20Longhorn%2C%20you%20can%20control,a%20defined%20RPO%20and%20RTO.>

## Azure Blob Storage

### General tool information

Azure Blob Storage, available under Azure subscription

### Tool/system description

Azure Blob storage is a distributed file storage, service-based for storing large amounts of unstructured object data, such as text or binary data. You can use Blob storage to expose data to the world publicly or store application data privately.

Typical uses of Blob storage include: serving images or documents directly to a browser.

Azure Blob storage provides scalable, cost-efficient object storage in the cloud. Store and unstructured access data for your most demanding workloads

### Tool implementation

To access Azure Storage, you'll need an Azure subscription. If you don't already have a subscription, create a free account before you begin. All-access to Azure Storage takes place through a storage account.

In wire rod rolling use case Azure Blob Storage is used to store historical data on the cloud infrastructure. Data are loaded in batch into the storage from the on-premise infrastructure through Rclone tool. Data are structured in parquet files with different schemas depending on the data source and are organized by day. In fact the ETL procedure, performed on premise by the ingestion process, creates one parquet per product (rolled billet) for LAP Laser, Wire Check, Production and IBAPOSITION data (iba continuous data related to the product where time has been synchronized to the position on the final wire rod) and in addition splits iba continuous data coming from InfluxDB into parquet files lasting 5 minutes.

### Tool assessment

#### **Pros**

Well put documentation, and quite easy to work with. It's not as complete as Amazon AWS, but it's enough to build around the API

The Hot class has pretty good prices (around 0,018 USD per GB/month). Archive class price is one of the lowest in the industry (about 0,002 USD per GB/month)

Different storage classes for each necessity: Hot (frequent use), Cool (infrequent use) and Archive (long-term storage)

High durability. Microsoft Azure Blob Storage provides 99.99999999999999% (16 9's) of the durability of objects over a given year with the default replication strategy: RA-GRS. This percentage beats every single provider on the list.

Downloading data from the Hot class, it's completely free. This is a fantastic deal if you fetch files frequently.

Many different storage options depending on your needs: Blob, Archive, Queue, File or Disk  
You receive \$200 to use Microsoft Azure services after you enroll (within 30 days) and during 12 months in selected services.

#### **Cons**

You need to purchase a support plan starting from 29 USD per month to get direct support.

Prices in other providers like Backblaze B2 are cheaper (around 0.005 USD per GB/month) for Standard access.

All the different storage options end up being confusing. For example, it's not straightforward for a beginner to know when to use Blob service over File service.

### References

<https://azure.microsoft.com/>

<https://itnext.io/microsoft-azure-blob-storage-pros-cons-and-how-to-use-it-with-javascript-ca5aaf5d5ffd>

<https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-portal>

## MinIO

### General tool information

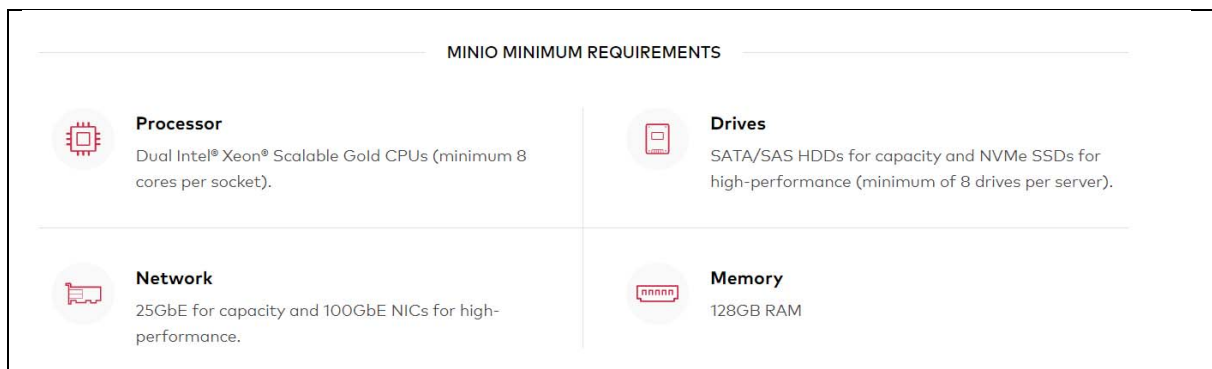
MinIO offers high-performance, S3 compatible object storage. Native to Kubernetes, MinIO is the only object storage suite available on every public cloud, every Kubernetes distribution, the private cloud and the edge. MinIO is software-defined and is 100% open-source under GNU AGPL v3.

### Tool/system description

MinIO is different in that it was designed from its inception to be the standard in private/hybrid cloud object storage. Because MinIO is purpose-built to serve only objects, a single-layer architecture achieves all necessary functionality without compromise. The result is a cloud-native object server that is simultaneously performant, scalable and lightweight.

While MinIO excels at traditional object storage use cases like secondary storage, disaster recovery and archiving, it is unique at overcoming the challenges associated with machine learning, analytics and cloud-native application workloads.

### Tool implementation



### Tool assessment

The benefits of Minio:

- S3 API compatibility.
- Data Redundancy.
- High Availability.
- Horizontal and vertical scaling.
- Supports multiple Pluggable storage backends.
- Data security using encryption on both server and client-side

Limits:

In Minio, there is the stand-alone mode; the distributed method as per usage requires a minimum limit of 2 and a maximum of 32 servers. But there is no limit on disks shared across the Minio server. If we need multiple setups, quickly spin up multiple Minio Instances managed by tools like Kubernetes.

### References

<https://www.xenonstack.com/insights/minio#:~:text=Minio%20is%20the%20best%20server,Redis%2C%20MySQL%2C%20and%20Gitlab.>

## PostgreSQL

### General tool information

**PostgreSQL** is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads.

Committed by PostgreSQL Global Development Group, download at <https://www.postgresql.org/download/>

### Tool/system description

**PostgreSQL**, also known as Postgres, is a powerful, open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. PostgreSQL features transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures. PostgreSQL supports both SQL (relational) and JSON (non-relational) querying. It is available for Windows, Linux and macOS.

In the wire rod rolling use case, a PostgreSQL database was used to store preprocessing results and model inference results. The python model inference module and visualization components takes advantage of psycopg2 library to access the PostgreSQL database.

### Tool implementation

PostgreSQL is available for download as ready-to-use packages or installers for various platforms. The installer allows to include also some useful tools during the installation:

- pgAdmin: a useful graphical tool for administrating and managing the PostgreSQL database
- StackBuilder: a package manager that can be used to download and install additional PostgreSQL tools and drivers. Stackbuilder includes management, integration, migration, replication, geospatial, connectors and other tools.

The minimum hardware required to install and run PostgreSQL are:

- 1 GHz processor.
- 2 GB of RAM.
- 512 MB of HDD.

### Tool assessment

PostgreSQL is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads.

PostgreSQL has been proven to be highly scalable in the sheer quantity of data it can manage and the number of concurrent users it can accommodate. There are active PostgreSQL clusters in production environments that manage many terabytes of data and specialised systems that manage petabytes.

### References

<https://www.postgresql.org/>



## NFS

### General tool information

Network File System (NFS) is a distributed file system protocol originally developed by Sun Microsystems (Sun) in 1984, allowing a user on a client computer to access files over a computer network much like local storage is accessed. NFS builds on the Open Network Computing Remote Procedure Call (ONC RPC) system like many other protocols. NFS is an open IETF standard defined in a Request for Comments (RFC), allowing anyone to implement the protocol.

### Tool/system description

The Network File System (NFS) is a mechanism for storing files on a network. It is a distributed file system that allows users to access files and directories located on remote computers and treat those files and directories as if they were local.

### Tool implementation

To install a dedicated NFS server, you can use any operating system that provides NFS. Additionally, the NFS server must meet the following hardware requirements: 16 GB RAM, 8 CPU cores, and 100 GB free disk space.

### Tool assessment

#### Pros:

- Low-cost solution for network file sharing.
- Easy to set up, uses existing IP infrastructure.
- Enables central management of shared files.
- Reduces disk space requirements for individual users (due to the reduced need to store software or other files locally).
- Lets users access remote files the same way they access local files.
- Reduces the need for removable media storage like DVDs or USB disks, which improves security.

#### Cons:

- Based on RPC, which is inherently insecure. RPC communication should only be allowed behind a firewall on a trusted network.
- NFSv4 and NFSv4.1 may have limits in maximal bandwidth so that NFS can slow down in the event of high traffic loads. This has improved in version 4.2.

### References

<https://cloud.netapp.com/blog/azure-anf-blg-linux-nfs-the-basics-and-running-nfs-in-the-cloud>

## Parquet (file format)

### General tool information

Apache Parquet is a file format designed to support fast data processing for complex data.

### Tool/system description

Characteristics of Parquet file format:

**Columnar:** Unlike row-based formats such as CSV or Avro, Apache Parquet is column-oriented – meaning the values of each table column are stored next to each other, rather than those of each record:

ROW-BASED STORAGE	COLUMNAR STORAGE
1 MARC, JOHNSON, WASHINGTON, 27	ID: 1 2 3
2 JIM, THOMPSON, DENVER, 33	FIRST NAME: MARC, JIM, JACK
3 JACK, RILEY, SEATTLE, 51	LAST NAME: JOHNSON, THOMPSON, RILEY
	CITY: WASHINGTON, DENVER, SEATTLE
	AGE: 27 33 51

**Open-source:** Parquet is free to use and open-source under the Apache Hadoop license and is compatible with most Hadoop data processing frameworks. To quote the project website, “*Apache Parquet is... available to any project... regardless of the choice of data processing framework, data model, or programming language.*”

**Self-describing:** In addition to data, a Parquet file contains metadata, including schema and structure. Each file stores both the data and the standards used for accessing each record – making it easier to decouple services that write, store, and read Parquet files. The Apache Parquet file format characteristics create several distinct benefits for storing and analysing large volumes of data. Let’s look at some of them in more depth

**Compression:** In Parquet, compression is performed column by column. It is built to support flexible compression options and extendable encoding schemas per data type, e.g., different encoding can be used for compressing integer and string data.

### Tool implementation

The Parquet file format can be used directly in several programming languages and IDEs like Python, Matlab or .NET

### Tool assessment

Besides the above mentioned characteristics a Parquet-based file system offers high performance I/O operations. Additionally, there are exist tools to handle Parquet-based file systems by SQL-based queries which eases the use of this file format for people familiar with ANSI-SQL (e.g. Apache Drill, see references)

### References

In .NET developments we use “Parquet.Net” v3.9.0 library

<https://github.com/aloneguid/parquet-dotnet>.

for Python based developments: Pandas, an open-source library for data analysis and manipulation, <https://pandas.pydata.org/>

**Docs:**

<https://parquet.apache.org/>

<https://www.upsolver.com/blog/apache-parquet-why-use>

**Tools:**

<https://drill.apache.org/docs/querying-parquet-files/>, SQL based access to data in a files storage system using Parquet files.

### 2.3.5. Data transmission / streaming

#### Apache Kafka

##### General tool information

Apache Kafka, licensed under the Apache Licence 2.0,  
download at <https://kafka.apache.org/downloads>

##### Tool/system description

Apache Kafka is an open-source stream-processing software platform developed by the Apache Software Foundation, written in Scala and Java. The software aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Kafka can connect to external systems (for data import/export) via Kafka Connect and provides Kafka Streams, a Java stream processing library. Kafka uses a binary TCP-based protocol that is optimized for efficiency and relies on a "message set" abstraction that naturally groups messages together to reduce the overhead of the network roundtrip. This protocol leads to larger network packets, larger sequential disk operations, contiguous memory blocks which allows Kafka to turn a burst stream of random message writes into linear writes.

So, in general Kafka is used to building real-time streaming data pipelines and real-time streaming applications. A data pipeline reliably processes and moves data from one system to another, and a streaming application is an application that consumes streams of data.

In wire rod rolling use case, Kafka broker is used for the transmission of time signal data acquired by the ibaPDA acquisition system and directly produced through iba-PDA-DataStore-Kafka. A json message containing all configured process parameters (name-value) is sent every 10 ms. These messages are distributed into multiple partitions of one topic and are consumed by multiple consumers (each subscribing to one partition) that provide to store them into InfluxDB. Moreover, Kafka broker was used to transmit data received from the three specialized modules for LAP Laser, Wire Check and Production data, using dedicated topics. Json messages, with different formats and related to the single product, are sent by the producer modules. Specific consumers provide to transform them and store information into NFS as parquet files per product. Specifically Wire Check data consumer must merge information from different messages related to different defects of the same product into the single billet parquet file to be stored.

##### Tool implementation

###### **Software remarks:**

To make it possible for clients to connect to the message broker, it was necessary to add listeners and advertised.listeners properties in the server.properties file specifying the host/IP, port and protocol.

###### **Hardware Requirements/Recommendations**

- Kafka Broker Node: 8 cores, 64 GB to 128 GB of RAM, two or more 8-TB SAS/SSD disks, and a 10-Giga Network Interface Card (NIC) .
- Minimum of three Kafka broker nodes
- Hardware Profile: More RAM and faster speed disks are better; 10 NICs ideal (75 MB/sec per node is a conservative estimate (can go much higher if there is more RAM and reduced lag between writing/reading), therefore 10GB NICc is required ).
- With a minimum of three nodes in your cluster, you can expect 225 MB/sec data transfer.
- You can perform additional further sizing by using the following formula:

$$num\_brokers = desired\_throughput (MB/sec) / 75$$

##### Tool assessment

In the ingestion process, Kafka senders/consumers were developed with Visual Studio 2019/2017 using C# language and .NET Core 3.1 and "Confluent.Kafka" NuGet package was used to connect to the Kafka broker.

Conduktor as a monitoring tool and Kafka tools fully integrate Kafka technologies and were

experimented to interface the Kafka broker and have a control on data. Conduktor is a good tool to verify the broker and topics settings. This tool allows also to increase the maximum message dimension for a specific topic. Kafka tool is useful to view the message content for each topic. To make it possible for LAP Laser specialized module to send big messages, it was necessary to increase the message dimension on the topic and configure it also in the producer module.

A reference to a comprehensive listing of pros&cons of Kafka is given below. Just the advantages like high-throughput, low latency or on-the-fly scalability of Kafka make it ideal for our data lake implementation

#### References

##### **Docs:**

<https://kafka.apache.org/>  
<https://www.confluent.io/what-is-apache-kafka/>  
<https://www.conduktor.io/>  
<https://www.kafkatool.com/>  
<https://data-flair.training/blogs/advantages-and-disadvantages-of-kafka/> (pros&cons  
[https://docs.cloudera.com/HDPDocuments/HDF3/HDF-3.1.0/bk\\_planning-your-deployment/content/ch\\_hardware-sizing.html](https://docs.cloudera.com/HDPDocuments/HDF3/HDF-3.1.0/bk_planning-your-deployment/content/ch_hardware-sizing.html)

##### **Libraries:**

In .NET developments we use “Confluent.Kafka” v1.7.0 library  
(<https://github.com/confluentinc/confluent-kafka-dotnet/>).

#### **ibaPDA-Data-Store-Kafka**

##### General tool information

ibaPDA-Data-Store-Kafka added to ibaPDA version 7. Both are licenced by IBA.

##### Tool/system description

ibaPDA-Data-Store-Kafka is used for data streaming into Apache Kafka cluster. In the wire rod rolling use case, ibaPDA, with the ibaPDA-Data-Store-Kafka addition, is used for the acquisition and direct transmission to the Apache Kafka cluster of time-based signal data. A json message containing all configured process parameters (name-value) is written into Kafka every 10ms.

##### Tool implementation

To run ibaPDA v7, it is required to have at least Microsoft .NET Framework 4.6 installed. ibaPDA-Data-Store-Kafka requires at least ibaPDA v7. When using ibaPDA v7 it is necessary to upgrade also other IBA components. At JSW a technical specification for the hardware and software upgrade of IBA system was shared between JSW and IBA Italy Technical Department. An upgrade of the iba system from v6 to v7 was carried out.

In the wire rod rolling use case, the data streaming was implemented taking advantage of ibaPDA-Data-Store-Kafka, used for data streaming into Apache Kafka cluster, and InfluxDB, used to store recent continuous signals (time-series data), transmitted to the Kafka broker by the ibaPDA acquisition system. These two tools have been already described in the above. Afterwards, process signal monitoring is supported through Grafana dashboards showing real-time streaming data from InfluxDB data source .

### 2.3.6. Data processing - analysis

#### Python

##### General tool information

Python, licensed under the PSF (Python Software Foundation) License Agreement, download at <https://www.python.org/downloads/>

##### Tool/system description

Python is an interpreted, object-oriented, high-level and general-purpose programming language with dynamic semantics, meaning that its variables are dynamic objects. Its high-level built-in data structures and its dynamic typing make it very powerful, flexible and very attractive for developing a wide variety of applications. Python supports modules and packages, facilitating program modularity and code reuse. The Python interpreter and the extensive standard library are freely distributed. Python is very popular in data science, Machine Learning and Artificial Intelligence, due to the widespread use of its libraries such as NumPy, Pandas, SciPy, and TensorFlow. However, Python is employed also in many other fields for the development of Web, Business and Embedded Applications.

In wire rod rolling use case, Python main packages were used for data handling, analysis and Machine Learning models development. Access to data stored in Parquet files was performed via Pandas and PySpark. Pandas was used for data pre-processing purposes as well and afterwards advanced data analysis tasks (outliers identification, imputation, clusterization) were performed by using SkLearn capabilities and a proprietary SSSA package. SkLearn was used also for the models development together with MiniSOM, a well-known library for the design, tuning and deployment of Self-Organizing Maps (SOM). SkLearn was used also for models hyper-parameters tuning purposes. Models development phase took advantage of the Conda framework for the management of the development environment. Data visualization exploits Matplotlib, Seaborn and Plotly packages that allow the creation of readable, expressive and interactive charts. JupyterLab/Notebook, Spyder and Visual Studio Code were used as IDEs (Integrated Development environment).

##### Tool implementation

Python is a cross-platform language and will work on different operating systems (Windows, macOS, and Linux). Python programs can run on a Windows computer, as long as the Windows machine has the Python interpreter installed (most other operating systems come with Python pre-installed).

To install the tool, simply choose the installer that suits the operating system, among the various executable installers that are available for different operating system specifications.

For Windows users, during the installation pay attention to check the Add PATH to Python checkbox otherwise it is necessary to add Python Path to environment variables afterwards.

Moreover, all packages can be easily installed when required through the command line using the command:

```
python -m pip install somepackage
```

This command install the latest version of a module and its dependencies from the Python Package Index. It's also possible to specify an exact or minimum version directly on the command line.

On the other hand, when Conda environment is used, to install a specific package into an existing environment "myenv", just use the command:

```
conda install --name myenv somepackage.
```

##### Tool assessment

Python is actually the programming language of Machine Learning due to the completeness of its packages and the wide eco-systems of libraries that researchers continuously build and maintain on top of it. The selected libraries offer all the facilities needed for the development and deployment of the ML-based tools envisaged within the project as well as its connection with industrial databases.

#### Pros

- up-to-date libraries continuously maintained and updated
- fast and efficient connection with industrial databases
- complete for the full-stack development of ML based solutions
- packages are central with respect to a wide eco-system of tools that effortlessly interface with them allowing to extend the developed tools easily
- no cost: all of them are free to use

#### Cons

- possible future problems among conflicting libraries (solved using Conda for environment management)
- Python may suffer from computational performance if not suitably optimized parameter settings are selected

### PySpark

#### General tool information

PySpark is an open-source ML library for the Python programming language

#### Tool/system description

PySpark is an interface for Apache Spark in Python. It allows to write Spark applications using Python APIs and it also provides the PySpark shell for interactively analyzing your data in a distributed environment. PySpark supports most of Spark's features such as Spark SQL, DataFrame, Streaming, MLlib (Machine Learning) and Spark Core.

In wire rod rolling use case, PySpark library has been used to access parquet files, both produced offline (converting .dat files generated with ibaPDA system) and those created by the ingestion process. The tool was used to query and perform aggregations on structured data exploiting SparkSQL facility during the development of preprocessing procedures. Moreover, the tool has been deeply used in the algorithm for the artificial reconstruction of billet signals, that relate iba signals to the single billet, starting from already existing time series signals and furnace exit times of billets.

#### Tool implementation

The installation of PySpark on the local machine, used for the development, requires the following steps:

- If not already installed, it is required to install jdk and set new related environment variables: set JAVA\_HOME variable to the location of the Java file and set PATH variable to the location of Java bin file.
- Afterwards the pySpark package can be installed with the pip utility
- Then download (at <https://github.com/steveloughran/winutils/tree/master/>) and setup winutils.exe. Make a new folder called 'winutils' and inside of it create again a new folder called 'bin'. Then put the file winutils.exe, recently downloaded, inside of it.
- Finally, set a new environment variable named HADOOP\_HOME to the location of winutils folder.

#### Tool assessment

Several kinds of problems were encountered while using this tool:

- Java Heap Space Out of memory problems, that were resolved configuring some special settings in the configuration of the Spark session, in order to regulate the memory allocated for spark driver and executors.
- Slow computation. This problem was resolved by using persisting techniques when facing with small dataframes or by changing some specific settings in the configuration of the Spark session (for example increasing allocated memory and for example spark.sql.shuffle.partitions).

- Spark incompatibility problems. Some parquet files produced offline converting .dat files acquired with ibaPDA system at JSW plant, were not Spark compatible, due to special characters in variable names, but unfortunately recommended solutions based on column rename (withColumnRenamed, toDF, alias..) were not resolute in our case. Concerning that point, a dedicated post was published by SSSA on stackoverflow.

Due to these problems, it was a bit troubling to use this tool, but SparkSQL facility was very useful to easily work with structured data stored in parquet by means of a SQL-like query language. In our opinion, skilled persons are required to take full advantage of the real capabilities of this tool, especially in a real distributed environment.

Main advantages of pySpark are:

- In-memory computation: with in-memory processing, the speed of processing can be increased, while thanks to data being cached, it allows not to fetch data from the disk every time.
- Fault tolerance: by means of abstraction-RDD, PySpark is specifically designed to handle the failure of any worker node in the cluster, ensuring that the loss of data is reduced to zero.
- Real-Time Stream Processing: PySpark is well-known also for its real-time stream processing support.

#### References

<https://spark.apache.org/>  
<https://stackoverflow.com/questions/67484166/use-parquet-file-with-special-characters-in-column-names-in-pyspark>  
<https://www.datacamp.com/tutorial/installation-of-pyspark>

### **SciKit-learn**

#### General tool information

Scikit-learn is an open-source ML library for the Python programming language

#### Tool/system description

SciKitLearn is module of the Python programming language that includes functions, classes and data structures devoted to data analytics and machine learning. Due to its wide usage, it is considered the standard module for performing such activities and many other libraries have been implemented on top of it. SciKitLearn content includes  
 packages for data preprocessing  
 a wide set of basic and advanced AI/ML models for regression, classification and clustering  
 model selection and evaluation

In wire rod rolling use case, SkLearn was used to perform advanced data analysis tasks (outliers identification, imputation, clusterization) and moreover for models hyper-parameters tuning purposes. Models development phase took advantage of the Conda framework for the management of the development environment.

#### Tool implementation

The library can be installed through the standard packaging tools designed to be used from the command line, such as pip utility. The requirements for Scikit-learn installation are:  
 Python (>= 2.6 or >= 3.3)  
 NumPy (>= 1.6.1)  
 SciPy (>= 0.9)

### **ibaAnalyzer**

#### General tool information

ibaAnalyzer, free of charge, download at <https://www.iba-ag.com/en/downloads>

#### Tool/system description

ibaAnalyzer is characterized by broad functionalities for analyzing and evaluating. The application offers an intuitive operation along with the complex scope of functions. The license for ibaAnalyzer is free of charge for analyzing data which have been acquired with iba systems like ibaPDA, ibaQDR or ibaLogic and were saved in the dat format as well as for data that have been recorded with ibaHD-Server.

#### Tool implementation

In the wire rod rolling use case, it was used to analyze the .dat files produced at JSW plant. It has been very helpful to understand the involved time series signals and more generally the plant process, but also during data verify operations performed on parquet files produced by the ingestion. Moreover, the tool has been deeply used in the part concerning the artificial reconstruction of billet signals that relate iba signals to the single billet.

#### Tool assessment

#### References

<https://www.iba-ag.com/en/ibaanalyzer>

### **Azure Data Bricks**

#### General tool information

Azure Data Bricks, available under Azure subscription

#### Tool/system description

Azure Data Bricks is a service to manage data transformation and analytics, built upon Apache Spark. It offers a notebook interface to write the code, manages spark clusters and the jobs that are to be run on the cluster and provides the user with visualization tools such as dashboards. Databricks can also implement Machine Learning pipelines, both for training models from data with parallel algorithms and for make batch processing of data using ML algorithms.

#### References

<https://azure.microsoft.com/>  
<https://docs.microsoft.com/en-us/azure/>

### **Azure Machine Learning Services**

#### General tool information

Azure Machine Learning Services, available under Azure subscription

#### Tool/system description

Azure Machine Learning service provides SDKs and services to quickly prepare data, train, and deploy machine learning models. The goal of the service is to improve productivity and costs with autoscaling compute and pipelines. It supports open-source Python frameworks, such as PyTorch, TensorFlow, and scikit-learn. It makes easier and convenient to train complex models using the computing power of the cloud without having to manage manually the hardware resources. It also provides services to manage the results of the models and build services to serve the inference, that can then be deployed in the cloud or downloaded to be run on premises.

#### References

<https://azure.microsoft.com/>  
<https://docs.microsoft.com/en-us/azure/>

### **2.3.7. Data Processing – modelling**

Same tools as Data Processing – Analysis section were adopted. Specifically, Python MiniSOM library was used for the development of SOM models.



Different models are provided for different quality targets and different lines and are installed as pickle files. Each pickle file associated to a model contains all the information needed to run it. The models wanted to be run are configured in the configuration file of the prediction models inference component within the speed layer, that is described hereafter.

### Inference module (self-written)

#### General tool information

Inference Module

#### Tool/system description

In the wire rod rolling case, the inference module is a specialized module aimed to perform the online inference of the models. Its main functionalities are those of reading the billet process data, preparing the input data for models, executing the models and finally storing the inference results in the PostgreSQL database. Different models are used for different quality targets and lines and, therefore, expected features, depending on the specific model, must be prepared through proper pre-processing procedures to run them. Process data are read from the iba position file, which is produced, at the end of billet, by the synchronization process, within the ingestion phase. Iba position files are stored on the NFS. Each iba position file contains iba continuous data related to the billet, where time has been synchronized to the position on the final wire rod. Inference procedure is activated by the arrival of a message in Kafka, specifying the identifier of the billet that has just finished.

Figure 13 shows a flow chart of the main activities of the inference module at wire rod rolling.

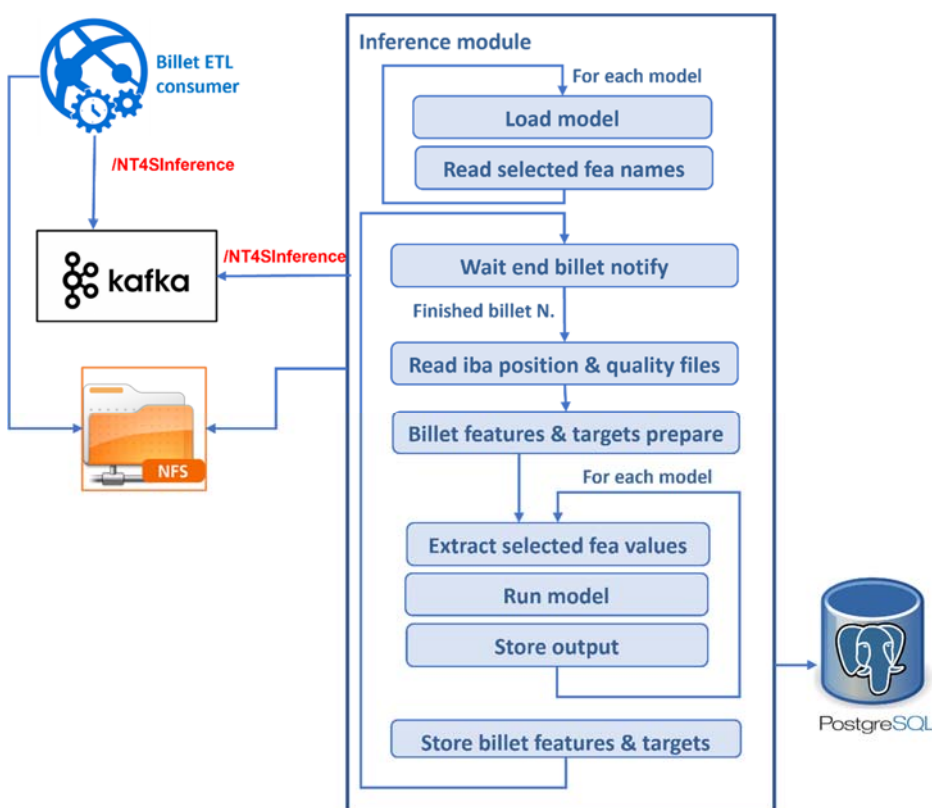


Figure 13: Flow chart for inference module at wire rod rolling

Specifically, when the module starts, all models are loaded and features names and data scalers expected by each model are identified. When a new billet is finished, the main operations performed by the inference procedure are the following:

- Read the billet iba position and quality files in NFS.
- Prepare all billet features from billet process data.

- For completeness, also the targets are computed from billet quality data files, stored in NFS by the ingestion modules (one parquet file for Lap Laser data and one for Wire Check data per billet).
- For each model, extract and pre-process those features expected as input by the model, run the model and finally store inference results into the PostgreSQL database.

The module was developed in Python/PySpark and VisualStudioCode as IDE and is runnable both on Windows and Linux platform.

#### Tool implementation

The Inference module has to be installed as a Docker Container and Docker is required for the installation.

Several python libraries were exploited during the development. Specifically, Python MiniSOM library was used to infer developed SOM models. Moreover, other specific libraries for Python language were used:

- KafkaConfluent library to connect to the Kafka broker in order to run models when the event that a new billet is finished has been notified
- psycopg2 library to store pre-processing and inference results into the PostgreSQL database (psycopg2 package).

Inference results are displayed through further Web applications that use Dash plotly library, see <https://dash.plotly.com/>.

#### Tool assessment

##### Model Inference Total Time (MIT)

An evaluation of times required for data preparation of model input data and inference of models for the just ended billet, was computed. The model inference total time (MIT) was calculated as the time required to receive the results of the inference since the time of the inference request, which is notified at the end of billet processing. This time was measured in terms of:

- Time for data preparation of all features and targets
- Time required for the model to calculate the inference response (only model time)
- Time to store the inference response

The average value for the Model Inference Total Time (MIT), from the request time of inference to the availability of results, is about 9 seconds, while the average value to run all models (Model Inference Calculation Time (MCT)) is less than 1 sec. The obtained results are considered satisfactory and above all in line with production process that requires about 1-2 minutes for one billet rolling.

##### Assessment of SOM performance

In order to assess the performance of the SOM -based monitoring system developed and applied to the monitoring of profile and surface defects, a dispersion index, calculated and averaged over all the clusters of the SOM, was adopted. Informally, such index measures the dispersion of the target measure associate to each cluster after the labelling procedure. For an arbitrary cluster C of the SOM, if we consider the set X of all the input observations associate to that cluster and the related set of labels Y, the dispersion of the cluster C is the standard deviation of Y. Such index was selected to express the precision of the label with respect to each cluster (the lower the better). The overall dispersion index is subsequently calculated as the average of the so-described index over all the clusters forming the SOM.

Values of Dispersion Index to define labelling quality of SOM

Target Id	Dispersion index
defc_binary_longdef	0.05
defc_ntotscabdef	0.56
prof_ova	0.06 mm

The dispersion index obtained by models monitoring respectively the presence/absence of longitudinal defects and the total number of scab defects correspond to a satisfactory result. More in detail, in the case of the longitudinal defect, which is a binary feature, the dispersion index corresponds to an average error of 5%. In the case of the total number of scab defects, which is an absolute feature, it corresponds to an average discrepancy slightly higher than half a defect. On the other hand, the profile ovality is defined as the difference between the maximum and the minimum values of actual diameter, averaged on all the axes. In the framework of ovality, the dispersion index measures in practice the difference between the actual bar ovality and the one associated to the SOM cluster the bar refers to. Therefore, results obtained when monitoring the profile ovality are perfectible, in the light of its distribution.

#### Example of inference results visualisation

Inference results are properly displayed through further Web applications to operators. Thanks to SOM data visualization technique, the high dimensional input space is mapped in a 2-dimensional real-time map of operating conditions, where critical conditions, that can be risky for the quality of the product, are depicted in a darker colour, depending on the quality indicator of interest, used to label the map. An example of results of inference achieved by executing the model monitoring the absence/presence of longitudinal defects is represented in Figure 14. In the interface, the unit map containing the actual operating conditions (related to the just finished billet) is marked. For correcting actions, the set of features selected by the model as those affecting the quality indicator are depicted in the table and their actual values are showed towards the reference values of the centroid of the cluster. The label associated to a unit map represents the quality indicator rate on all instances (of operating conditions) within the cluster. An indication of the percentage of instances of operating conditions falling into the cluster is indicated too with the caption %hits.

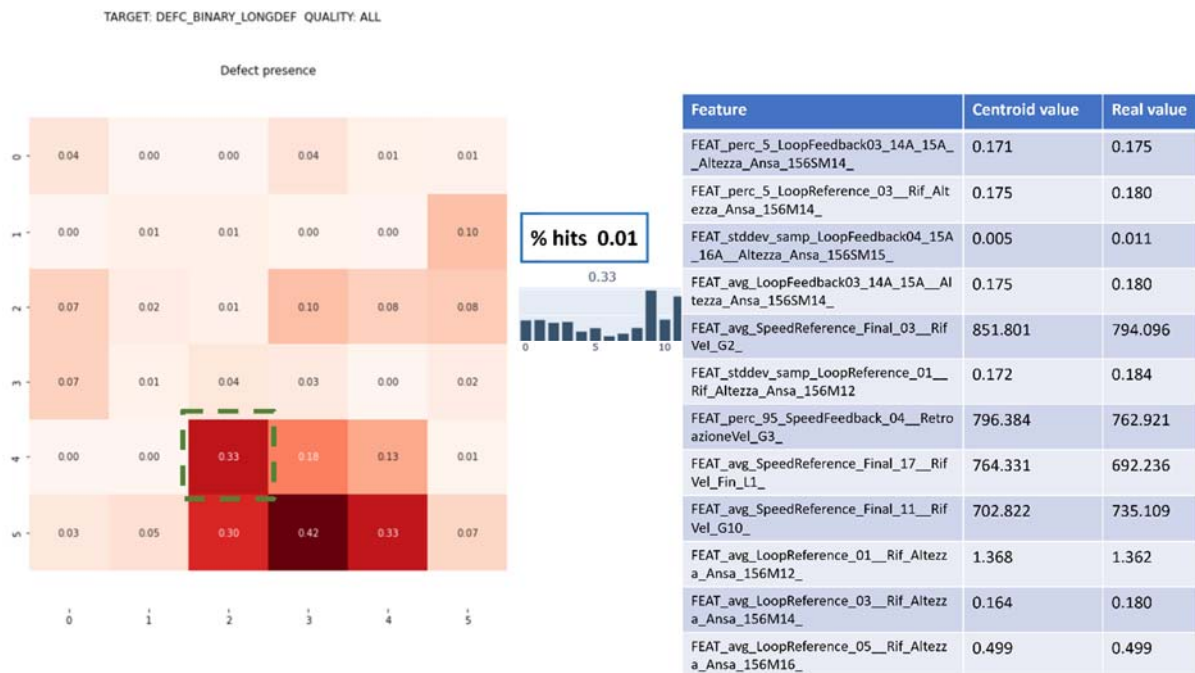


Figure 14: Example of inference results for monitoring surface quality (longitudinal defects)

## 2.3.8. Application for online visualisation

### Grafana

#### General tool information

Grafana is an open-source solution for running data analytics, pulling up metrics that make sense of the massive amount of data & to monitor our apps with the help of customisable dashboards. Besides an open-source version (GNU Affero General Public License v3.0), a managed Cloud version and a commercial Enterprise version exist. See <https://grafana.com/oss/grafana/>

#### Tool/system description

Grafana is a multi-platform open-source analytics and interactive visualization web application. It provides charts, graphs and alerts for the web when connected to supported data sources. A licensed Grafana Enterprise version with additional capabilities is also available as a self-hosted installation or an account on the Grafana Labs cloud service. It is expandable through a plug-in system. End users can create complex monitoring dashboards using interactive query builders.

In the wide rod rolling use case, process signal monitoring is supported through Grafana dashboards showing real-time streaming data from InfluxDB data source.

#### Tool implementation

Grafana connects with every possible data source, commonly referred to as databases such as Graphite, Prometheus, Influx DB, Elasticsearch, MySQL, PostgreSQL etc.

Grafana enables us to write plugins from scratch for integration with several different data sources.

The tool helps us study, analyse and monitor data over a period of time, technically called time-series analytics, e.g. by developing tailored dashboards to check real-time signal data.

It helps us track the user behaviour, application behaviour, frequency of errors popping up in production or a pre-prod environment, type of errors popping up and the contextual scenarios by providing relative data.

#### Example of real-time data visualisation

As already described in section 2.3.5, in the wire rod rolling use case real-time data are transmitted to the Kafka broker by the ibaPDA acquisition system and InfluxDB is used to store recent signals (time-series data), coming from the Kafka broker. Process signal monitoring is supported through Grafana dashboards showing real-time streaming data directly from InfluxDB data source. An example of a custom dashboard to monitor real-time signal data is shown below, where the user can choose and visualize any time-series signal he wishes.

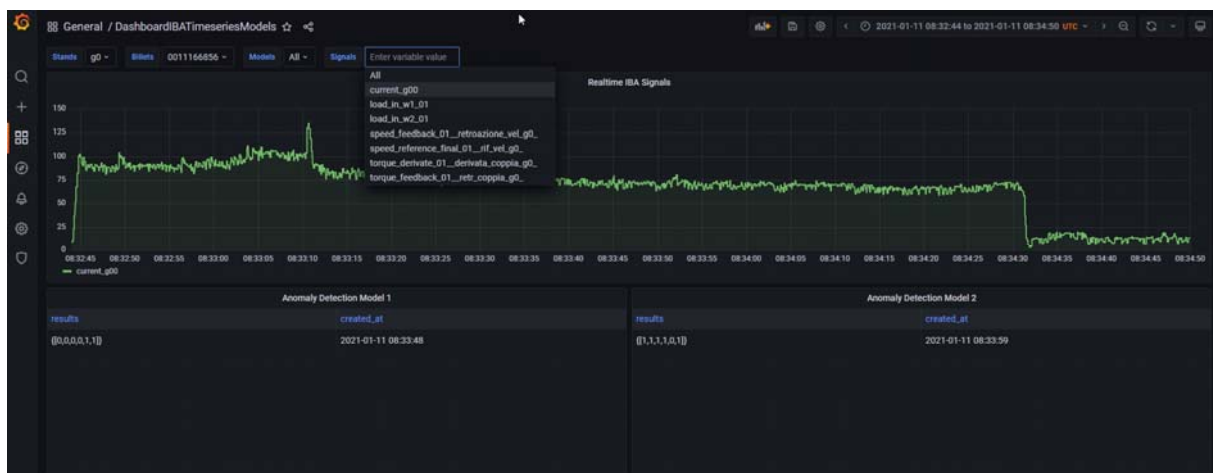


Figure 15: Real time monitoring example at wire rod rolling through Grafana

## 2.4. Test bed implementation at iba AG

### Intention

For iba, the intention to participate in this project was the development of new methods and technologies for analyzing and storing data. In previous projects, data analyses were mostly determined by proprietary formats of the data to be analyzed. Data from specific control systems or measurement data acquisition systems were stored in a format that was usually tailored to the analysis software in order to save storage space or optimize access. When transferring these proprietary data formats into a generic format, information was often lost due to algorithms for data compression or individual formatting problems or the necessary meta data. At other times, the disk space used by the data during export or conversion processes became extremely large, or the traceability of the data to its origin was lost. The analysis work becomes extremely difficult when data from different sources are to be put into relation with each other. A great effort was required to combine data in a suitably prepared analysis in a normalized form. Challenging here are above all cross-process and cross-plant analyses. In most cases, a complex chain of instructions is necessary just to load the data in a suitable form. Under certain circumstances, it can be very time-consuming to automate this, as the tools used do not offer suitable interfaces. Methods implemented in the iba software tools need to be enhanced in order to be able to deal with new data interfaces (ibaPDA - Kafka Interface, InfluxDB, MQTT, ibaAnalyzer - extraction as a parquet file) in order to make data retrievable in generic form. Due to the performance uncertainties and implementation risks, iba was very interested in participating in this project in order to proof these methods and technologies with real-world applications. It was the intention to migrate these methods later into further developments as products to be used worldwide. Another major challenge of the data infrastructure is the storage of data. At the time of the project start, little experience was available in dealing with distributed and horizontally scalable storage systems. This technology with completely new access methods and challenges for data analysis tools and techniques should be investigated with regard to possible performance improvements, especially with regard to cross-process analyses. The ibaHD server as a central element for storing data was compared with other time series specific data storage systems and subsequently we are working on an improvement regarding scalability and optimization of configurability and concepts to improve availability and further on develop concepts to ease the cross-process analysis. Within the framework of the project, various possibilities for the system infrastructure were considered in order to meet the requirements for long-term evaluation and archiving as well as live data evaluation by means of various algorithms, which can also originate from the AI environment.

The aim is to implement software and an infrastructure that can be easily integrated into scalable systems in order to effectively implement analysis tasks, both interactively and in automated form. The interfaces introduced within the framework of the project and subsequently to store process measurement values or to make them available to other systems can simplify the analysis situation of data that originate from several systems, for example successive systems. Also, the analyses themselves can no longer only be carried out on site, but also decentral, e.g. by a globally operating team of experts, through the use of cloud storage systems. In this context, new approaches also arise as to how business areas can be developed. Ideas on this have not been discussed far enough, but can be seen as an approach to answering the question of the goal for the use of the tools.

### 3. Additional tools

#### 3.1. FADI

FADI is a frame work for big data analytics and was partially optimised and adapted in the NT4Steel project.

##### General tool information

FADI - Ingest, store and analyse big data flows, see <https://fadi.cetic.be/>

##### Tool/system description

The purpose of FADI is to provide a straightforward way to deploy integrated and modular Big Data systems to various infrastructures (private and public clouds). FADI relies on the Kubernetes container orchestration engine for portability and scalability.

It comes with the following depicted components

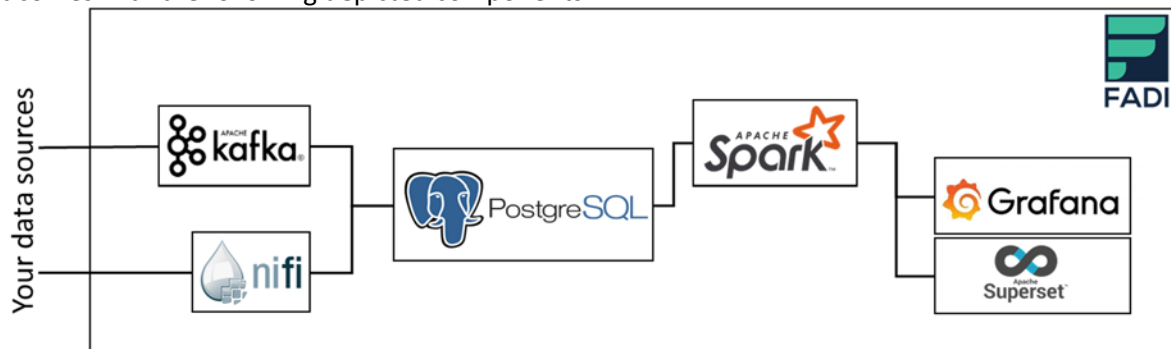


Figure 16: Composition of FADI framework

##### Tool implementation

FADI platform can be installed

- on a laptop/workstation, using Minikube for local development
  - on a generic Kubernetes cluster for a self-hosted installation
  - on Google Kubernetes Engine (GKE) as an example of public cloud,
- on Windows OS and MAC OS

### 3.2. iba tools for the transfer of measurement data to external systems

The commercial data acquisition system ibaPDA (together with the analysis tool ibaAnalyzer) is worldwide installed at nearly all steel plants. It includes the following features which were used inside this project:

- ibaPDA-Data-Store-Kafka: Data store to stream time-based measurement data in real-time directly to an installed Apache Kafka application
- ibaAnalyzer-File-Extract: Acquired time-series process data can be process-synchronously transferred directly into Parquet formatted files. Various trigger and aggregation functions are available to select the data and to reduce the amount of data to be transferred into the Parquet format.
- Additionally, a module in the ibaPDA system was developed which can execute data driven models saved in the ONNX format directly inside the ibaPDA software.

#### General tool information

**ibaPDA** as well as the integrated acquisition interfaces (southbound), data stores and output interfaces (northbound) are commercial tools which are licensed.

**ibaAnalyzer** is a free tool for data analysis, which is implicitly licensed by creating dat files using ibaPDA. Some components of ibaAnalyzer such as database access, transfer into other file formats as well as importing 3<sup>rd</sup> party formats to be used with ibaAnalyzer need a separate license.

#### Tool/system description

##### **ibaPDA-Data-Store-Kafka**

This data store allows streaming of time-based data directly from the data acquisition system ibaPDA to a Kafka cluster. The architecture of Apache Kafka consists of a cluster computer network. So-called brokers in this computer network store the received messages from ibaPDA with a time-stamp in so-called topics. Applications such as ibaPDA that write messages to a Kafka cluster are called producers. The data can be continuously sent or sent by trigger conditions (start/stop) only. Trigger conditions can be easily configured in ibaPDA and derived from all signals acquired in ibaPDA. For sending data to a cluster different encryption and authentication methods such as certificates for SSL communication are supported.

##### **ibaAnalyzer-File-Extract**

Enables the generation of a sample of historical data for further analysis. Data can be exported into (smaller) dat files, CSV or Parquet. Parquet is a common and performant file format to be used in Python applications and stores data in a table-like and condensed way. Using the expression editor of ibaAnalyzer, the data set can be reduced, aggregated or even enhanced by so-called virtual signals, i.e. combining arbitrary signals to new signals before exporting into the Parquet format. Furthermore start/stop trigger can be derived from the data only to export specific, interesting segments for further use into the Parquet format. This will help to focus on the right data and only store the essential data for long-term storage and long-term analysis.

##### **ONNX interface**

ONNX (Open Neural Network Exchange) is an open format for representing deep learning models. With ONNX, AI developers can exchange models between different tools. For ibaPDA an interface is under development to integrate an externally developed (deep) neural network model into the data processing stream of an installed iba ecosystem. This will allow real-time monitoring of the process under investigation. Based on the learned model, the process behaviour is analysed and process deviations and anomalies are detected in real-time.

#### Tool implementation

The software tools ibaPDA and ibaAnalyzer are running under Windows OS and can be downloaded from the iba AG website. Demo licenses are available from the iba support.

#### 4. Experiences and results made

This chapter will deliver the stories of the activities. Illustrated with many figures and screen shots, it will give an idea about what have been done and what are the results and experiences made.

##### 4.1. Experiences and implementation: Wire rod rolling use case

At JSW, new Big data tools were adopted for data transmission, data management (handling and storage) and data streaming to achieve near real time data processing, according to the wire rod rolling architecture. All the tools and developed components exploiting them were successfully integrated into the steel production IT systems.

In the wire rod rolling use case, the correct behaviour of ingestion services, that transmit quality and production data to Kafka (Lap Laser Data Service, Wire Check Data Service and Production Data Service), was verified thanks to the transmission tables stored in the Oracle data base were all transmissions and their outcomes are tracked by the services. A different transmission table is provided for each type of service. Moreover, file logging implemented within the services was very helpful during the test phase.

In the architecture implemented at the wire rod rolling, Apache Kubernetes was exploited as an orchestrator that allows managing container lifecycle and all resources inside the cluster.

Different monitoring tools were exploited depending on the implemented component. By using Kubernetes, it was possible to centralize the management for each of them, which allows fast and simple observation of the running processes.

Different and specific tools were used for monitoring processes and systems.

- Rancher (open source)
- Grafana (open source)
- Prometheus (open source)
- Loki (open source)
- Console InfluxDB (open source)

Different monitoring tools were exploited depending on the context:

- **Pods logs**
  - Loki = log aggregation in Kubernetes
  - Command line
  - Rancher = complete container management platform for Kubernetes
- **Resources**
  - Grafana = multi-platform open-source analytics and interactive visualization web application. Using Grafana dashboards we can monitor all the resources in the cluster. We can also create different Prometheus data sources that can be queried.

- **Kafka**

Using Kafka broker, useful tools, such as Conduktor and Kafka tool, were used to directly manage topics, to monitor the arrive of messages and view their contents.

- Conduktor: Good tool to verify the broker and topics settings.

This tool was used to increase the maximum message dimension for a specific topic.

- Kafka tool: Useful to view the message content for each topic.

- **Time series data**

- InfluxDB console: From version 2.0.x InfluxDB incorporates a console that previously had to be installed separately and which allows monitoring data

Concerning the upload of files from NFS to the Azure Blob Storage, it was possible to use RClone with proper options, in order to set the bandwidth upload and download limits and to decrease the writes in Azure Blob Storage optimizing the data transfer process.



The navigability and the correctness of schemas and content of different types of parquets provided by ingestion (IBA, IBA synchronized data and quality data) was checked by using pySpark exploiting SparkSQL facilities to navigate, query and elaborate the data.

All experiences made during the installation and use of tools adopted in the wire rod rolling use case are documented in the Workbook.

The assessment of results was done in terms of the following benchmarks that were transferable to operational conditions:

- An evaluation of the write time of a new time series record into the InfluxDB (time series database) (TSW).

The evaluation was done with an IBA acquisition system providing 560 time series signals every 10ms into a Kafka topic divided in 5 partitions and five consumers, each subscribed to each partition, writing concurrently into InfluxDB.

- An evaluation of the transfer rate performed by means of Rclone tool, during the upload of parquet files created by the ingestion components from the local file system (NFS) to Azure Blob Storage. (TRT)
- An evaluation of times needed for data preparation of model input data and inference of models for the just ended billet, within the inference module, considering that different models are provided for different quality targets (MIT and MCT from D2.1).

In this activity the model inference total time (MIT) was calculated as the time required to receive the results of the inference since the time of the inference request, which is notified at the end of billet processing.

The conclusions on inference module benchmarks were the following: the average value for the Model Inference Total Time (MIT), from the request time of inference to the availability of results, is about 9 seconds, while the average value to run all models (Model Inference Calculation Time (MCT)) is less than 1 second. The obtained results are considered satisfactory and in line with production process that requires about 1-2 minutes for one billet rolling.

#### Assessment of SOM performance

In order to assess the performance of the SOM based monitoring system developed and applied to the monitoring of profile and surface defects, a dispersion index calculated and averaged over all the clusters of the SOM was adopted. Informally, such index measures the dispersion of the target measure associate to each cluster after the labelling procedure. For an arbitrary cluster C of the SOM, if we consider the set X of all the input observations associate to that cluster and the related set of labels Y, the dispersion of the cluster C is the standard deviation of Y. Such index was selected to express the precision of the label with respect to each cluster (the lower the better). The overall dispersion index is subsequently calculated as the average of the so-described index over all the clusters forming the SOM.

*Table 1: Values of Dispersion Index to define labelling quality of SOM*

Target Id	Dispersion index
defc_binary_longdef	0.05
defc_ntotscabdef	0.56
prof_ova	0.06 mm

The dispersion index obtained by models monitoring respectively the presence/absence of longitudinal defects and the total number of scab defects correspond to a satisfactory result. On the other hand, results obtained when monitoring the profile ovality are perfectible, in the light of its distribution.

#### 4.1.1. Experiences: Implemented systems

##### Cluster management

In the architecture implemented at the wire rod rolling, Apache Kubernetes is exploited as an orchestrator that allows managing container lifecycle and all resources inside the cluster. All the developed services, performing data ingestion etc., are deployed as containers using Kubernetes manifests. Different tools are used to manage and check Kubernetes cluster status.

Rancher is an open-source platform that allows to manage the Kubernetes cluster providing the following functionalities:

- nodes creation
- Pods/Deployments/Jobs deploy and monitoring
- shell interface

In addition to Rancher, Grafana dashboards and Loki logging are exploited to verify the correctness and the performances. Specifically, some useful Grafana dashboards are available to display CPU, RAM and bandwidth utilizations.

This section shows a list of figures concerning the installations at JSW wire rod rolling, demonstrating the use of Rancher to manage the cluster resources, to monitor the running processes and the use of Grafana dashboards to monitor CPU, RAM and bandwidth utilizations.

Moreover, an example of a metrics collected for Azure Blob Storage is displayed. Finally, an overall overview of the Apache Kafka is given.

See also the video, showing the cluster exploring and monitoring, which is available on the project website <http://www.newtech4steel.eu/> in the Download Area

##### Main dashboard of the cluster

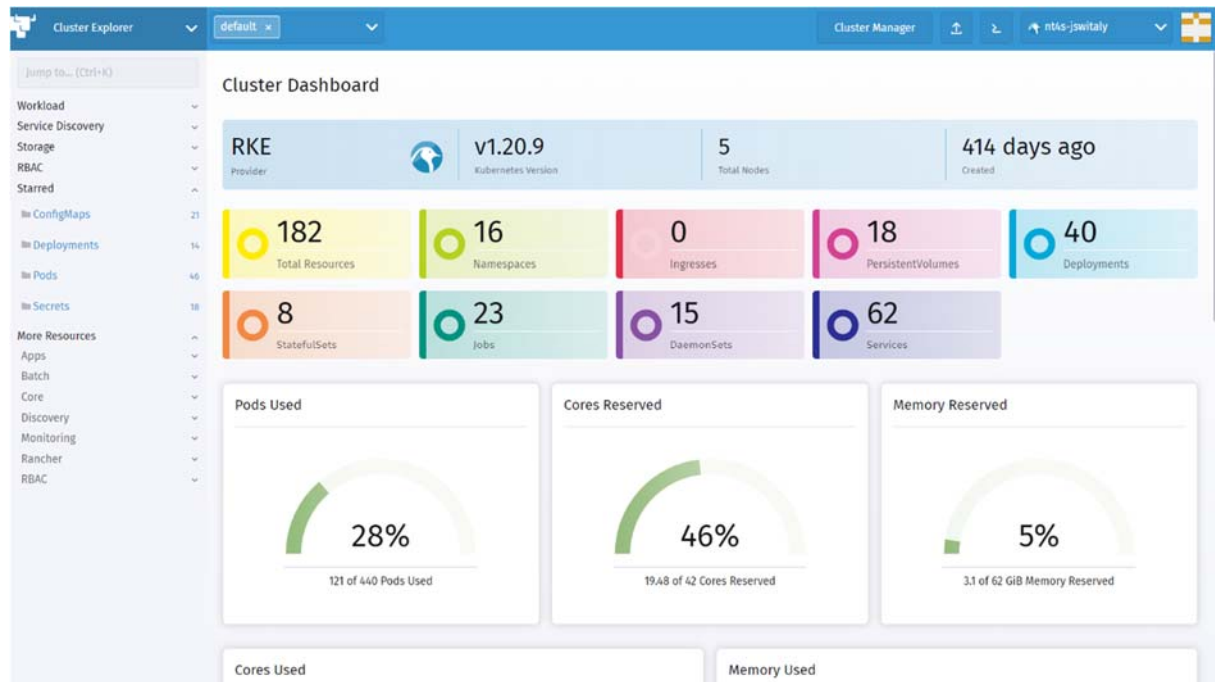
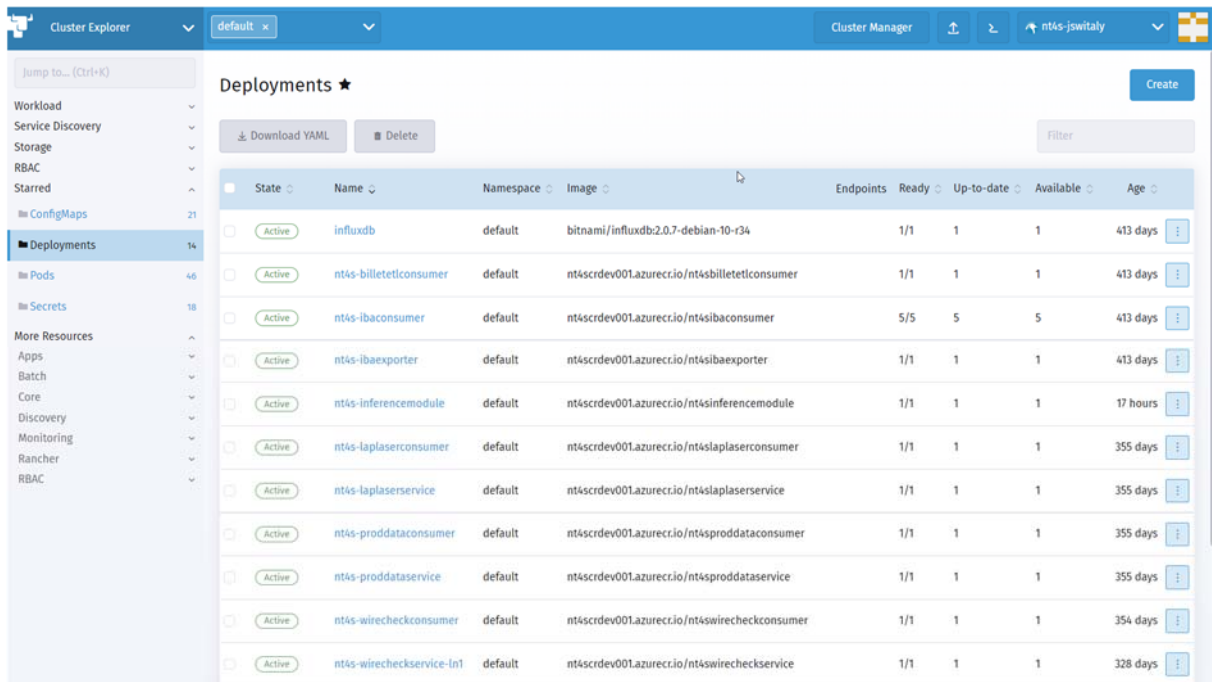


Figure 17: Main dashboard – screen shot

## List of Deployments

A Kubernetes deployment is a resource object in Kubernetes that provides declarative updates to applications.

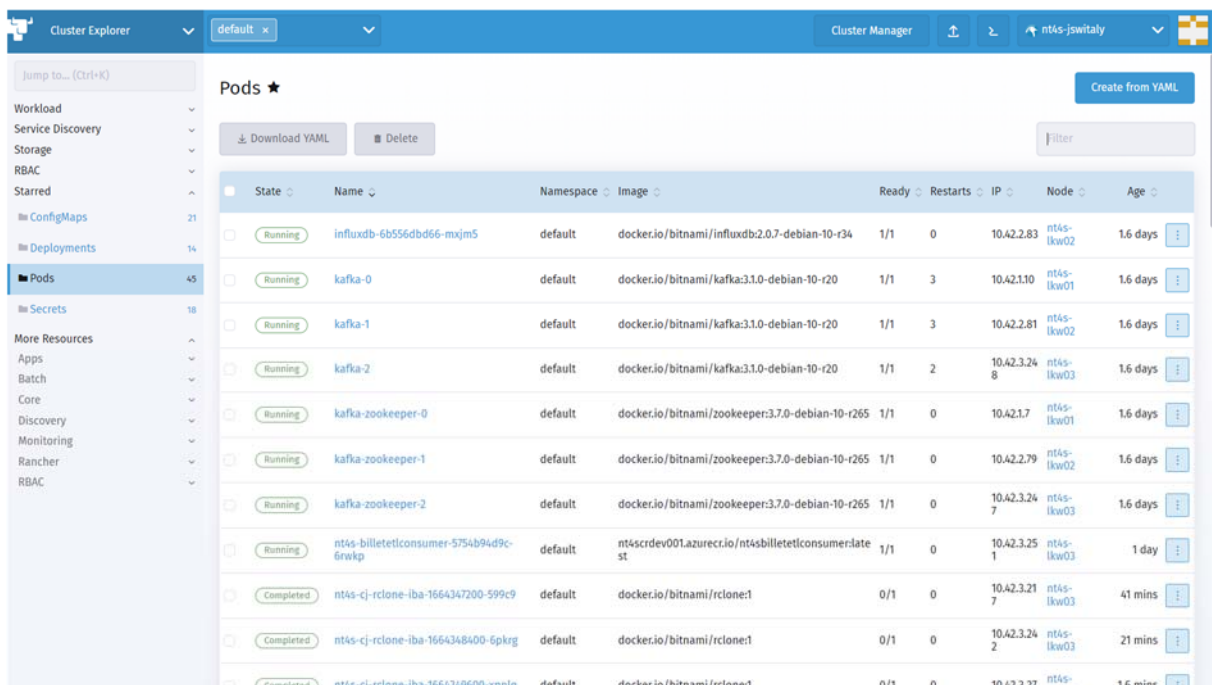


State	Name	Namespace	Image	Endpoints	Ready	Up-to-date	Available	Age
Active	influxdb	default	bitnami/influxdb2.0.7-debian-10-r34	1/1	1	1	1	413 days
Active	nt4s-billetetconsumer	default	nt4scrdev001.azurecr.io/nt4sbilletetconsumer	1/1	1	1	1	413 days
Active	nt4s-ibaconsumer	default	nt4scrdev001.azurecr.io/nt4sibaconsumer	5/5	5	5	5	413 days
Active	nt4s-ibaexporter	default	nt4scrdev001.azurecr.io/nt4sibaexporter	1/1	1	1	1	413 days
Active	nt4s-inferencemodule	default	nt4scrdev001.azurecr.io/nt4sinferencemodule	1/1	1	1	1	17 hours
Active	nt4s-laplasrconsumer	default	nt4scrdev001.azurecr.io/nt4slaplasrconsumer	1/1	1	1	1	355 days
Active	nt4s-laplaserservice	default	nt4scrdev001.azurecr.io/nt4slaplaserservice	1/1	1	1	1	355 days
Active	nt4s-proddataconsumer	default	nt4scrdev001.azurecr.io/nt4sproddataconsumer	1/1	1	1	1	355 days
Active	nt4s-proddataservice	default	nt4scrdev001.azurecr.io/nt4sproddataservice	1/1	1	1	1	355 days
Active	nt4s-wirecheckconsumer	default	nt4scrdev001.azurecr.io/nt4swirecheckconsumer	1/1	1	1	1	354 days
Active	nt4s-wirecheckservice-in1	default	nt4scrdev001.azurecr.io/nt4swirecheckservice	1/1	1	1	1	328 days

Figure 18: List of Deployments – screen shot

## List of Pods

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.



State	Name	Namespace	Image	Ready	Restarts	IP	Node	Age
Running	influxdb-6b556dbd66-mxjm5	default	docker.io/bitnami/influxdb2.0.7-debian-10-r34	1/1	0	10.42.2.83	nt4s-lkw02	1.6 days
Running	kafka-0	default	docker.io/bitnami/kafka:3.1.0-debian-10-r20	1/1	3	10.42.1.10	nt4s-lkw01	1.6 days
Running	kafka-1	default	docker.io/bitnami/kafka:3.1.0-debian-10-r20	1/1	3	10.42.2.81	nt4s-lkw02	1.6 days
Running	kafka-2	default	docker.io/bitnami/kafka:3.1.0-debian-10-r20	1/1	2	10.42.3.248	nt4s-lkw03	1.6 days
Running	kafka-zookeeper-0	default	docker.io/bitnami/zookeeper:3.7.0-debian-10-r265	1/1	0	10.42.1.7	nt4s-lkw01	1.6 days
Running	kafka-zookeeper-1	default	docker.io/bitnami/zookeeper:3.7.0-debian-10-r265	1/1	0	10.42.2.79	nt4s-lkw02	1.6 days
Running	kafka-zookeeper-2	default	docker.io/bitnami/zookeeper:3.7.0-debian-10-r265	1/1	0	10.42.3.247	nt4s-lkw03	1.6 days
Running	nt4s-billetetconsumer-5754b94d9c-6nwkp	default	nt4scrdev001.azurecr.io/nt4sbilletetconsumer:latest	1/1	0	10.42.3.251	nt4s-lkw03	1 day
Completed	nt4s-cj-clone-iba-1664347200-599c9	default	docker.io/bitnami/rc1one:1	0/1	0	10.42.3.217	nt4s-lkw03	41 mins
Completed	nt4s-cj-clone-iba-1664348400-6pkrq	default	docker.io/bitnami/rc1one:1	0/1	0	10.42.3.242	nt4s-lkw03	21 mins
Completed	nt4s-cj-clone-iba-1664349600-xpnlg	default	docker.io/bitnami/rc1one:1	0/1	0	10.42.3.27	nt4s-lkw03	1.6 mins

Figure 19: List of Pods – screen shot

## Cluster Dashboard

Shows overall cluster CPU/Memory/Disk usage in each node

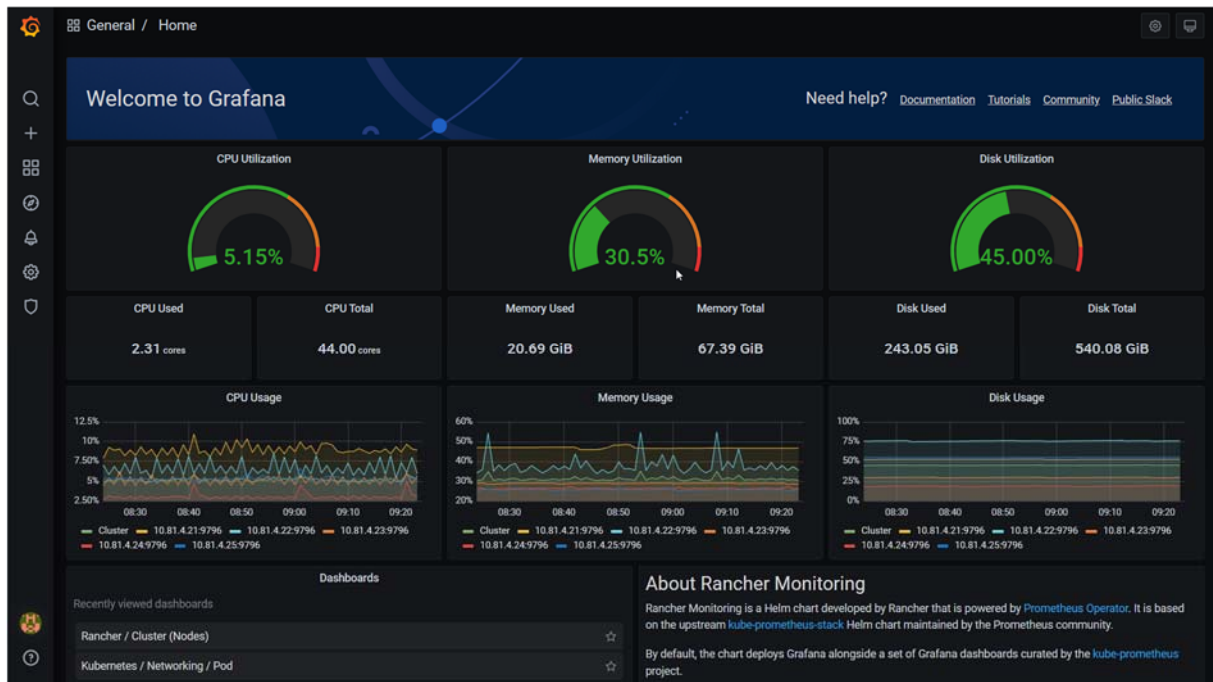


Figure 20: Cluster dashboard – screen shot

## Dashboard Computer Resources of Pods

Shows overall cluster CPU/Memory/Disk usage in each pod

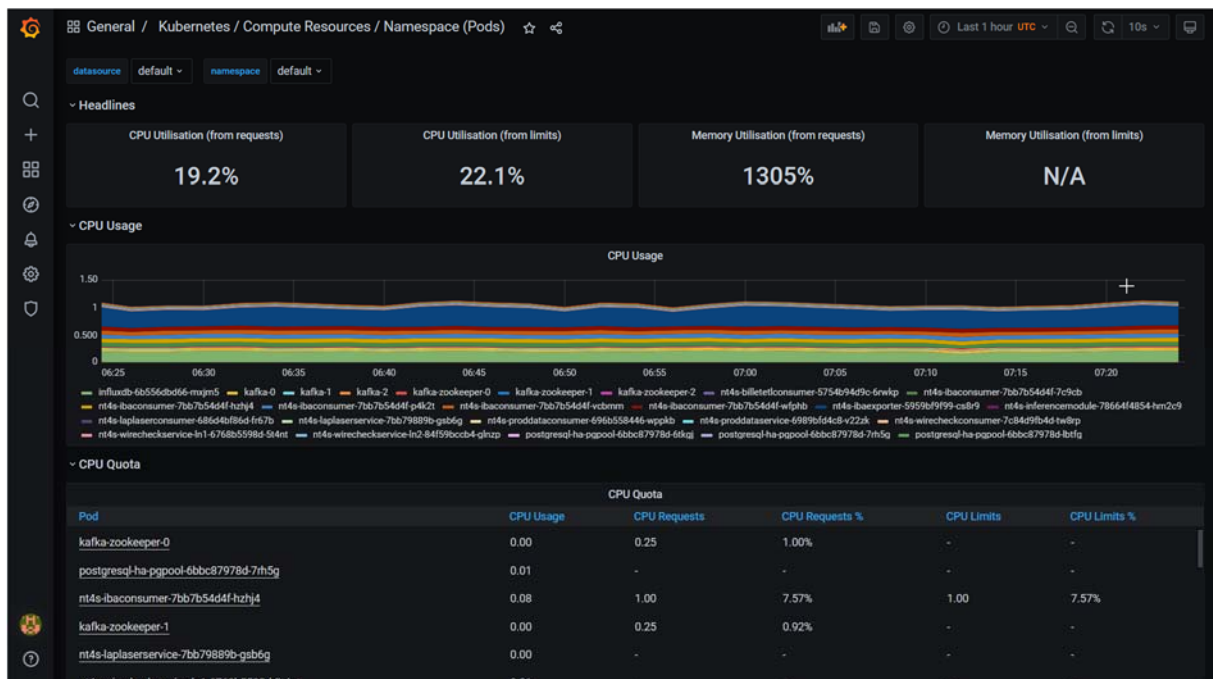


Figure 21: Dashboard Computer Resources of Pods – screen shot

## Dashboard Computer Network Resources of Pods

Shows overall cluster Receive/Transmit Bandwidth in each pod

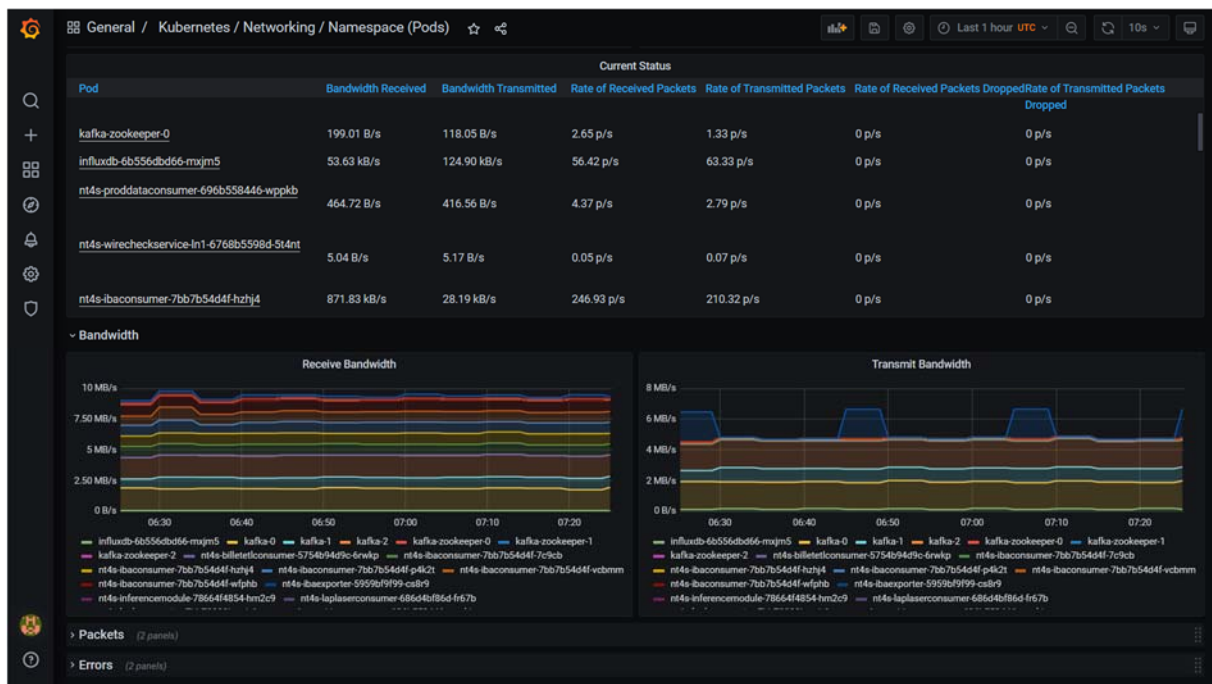


Figure 22: Dashboard Computer Network Resources of Pods – screen shot

## Azure Blob Storage Transaction Metrics

Shows the amount of ingress data into the Azure Blob Storage

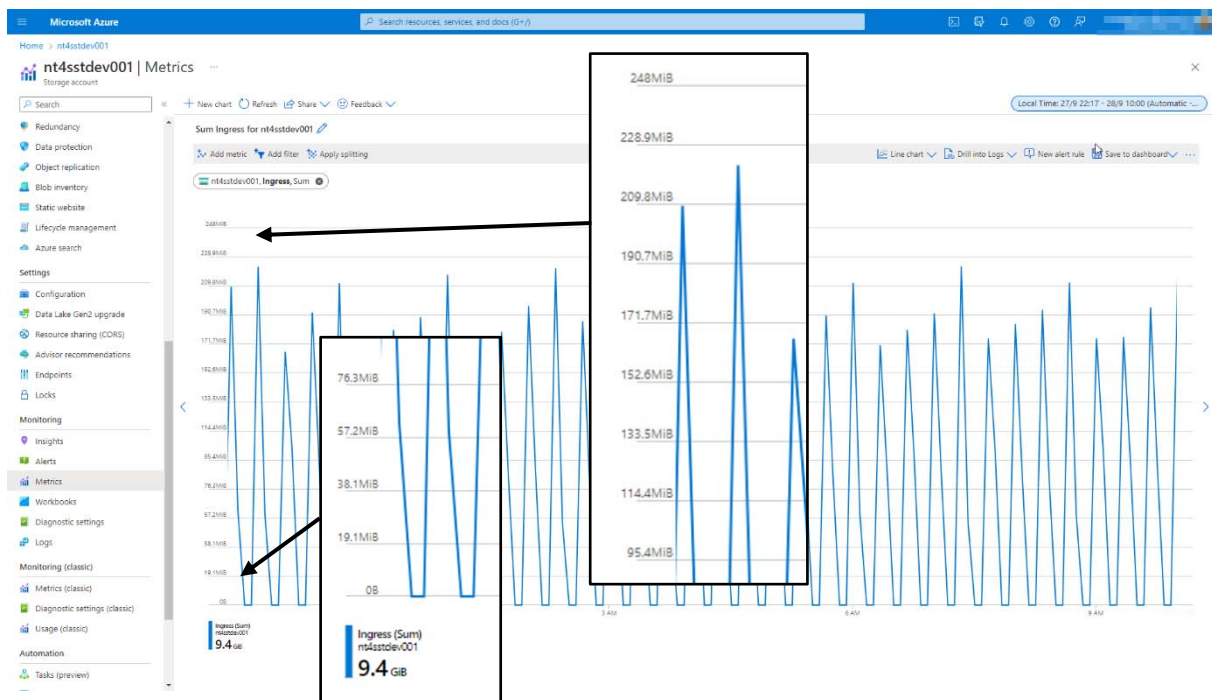


Figure 23: Azure Blob Storage Transaction Metrics – screen shot



## Offset Explorer

### Objects overview in the Apache Kafka cluster (Offset Explorer)

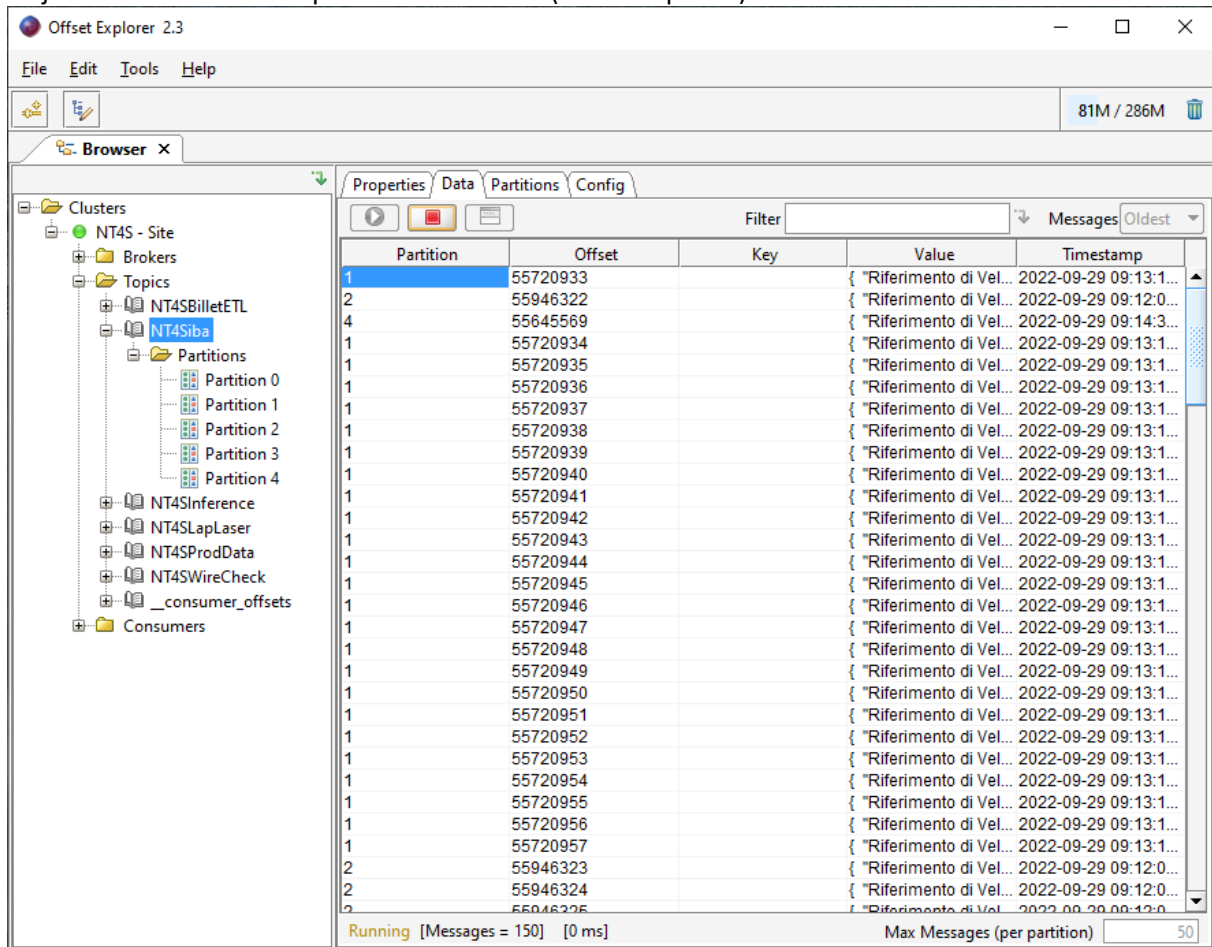


Figure 24: Offset Explorer – screen shot

## Process and wire rod quality monitoring system output results

In the wire rod rolling, real-time data are transmitted to the Kafka broker by the ibaPDA acquisition system and InfluxDB is used to store recent signals (time-series data), coming from the Kafka broker. Process signal monitoring is supported through Grafana dashboards showing real-time streaming data directly from InfluxDB data source.

An example of a custom dashboard to monitor real-time signal data is shown below, where the user can choose and visualize any time-series signal he wishes.

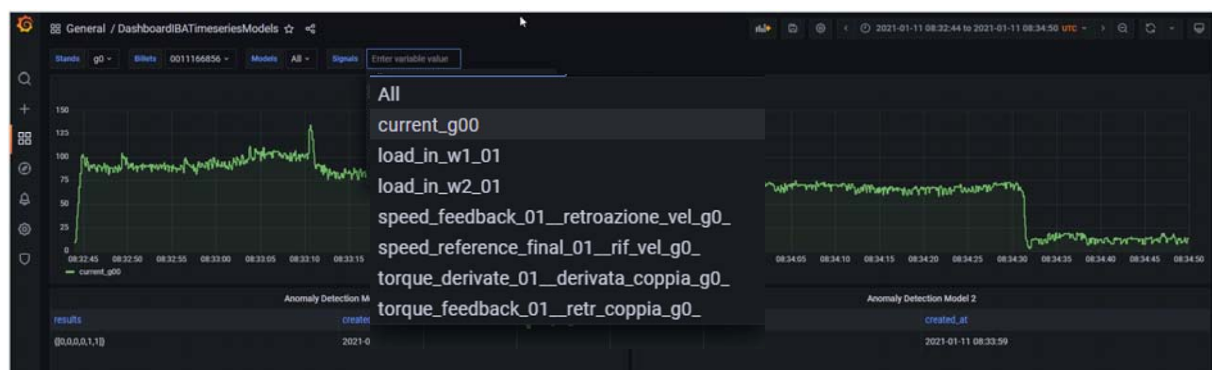


Figure 25: Example of real-time process data visualisation

Example of inference results for monitoring surface quality (longitudinal defects)

An example of the inference results achieved by executing the model monitoring the absence/presence of longitudinal defects is represented in the following figure. Thanks to SOM data visualization technique, the high dimensional input space is mapped in a 2-dimensional real-time map of operating conditions, where critical conditions, that can be risky for the quality of the product, are depicted in a darker colour, depending on the quality indicator of interest, used to label the map. In the interface, the unit map containing the actual operating conditions (related to the just finished billet) is marked.

For correcting actions, the set of features selected by the model as those affecting the quality indicator are depicted in the table and their actual values are showed towards the reference values of the centroid of the cluster. The label associated to a unit map represents the quality indicator rate on all instances (of operating conditions) within the cluster. An indication of the percentage of instances of operating conditions falling into the cluster is indicated too with the caption %hits.

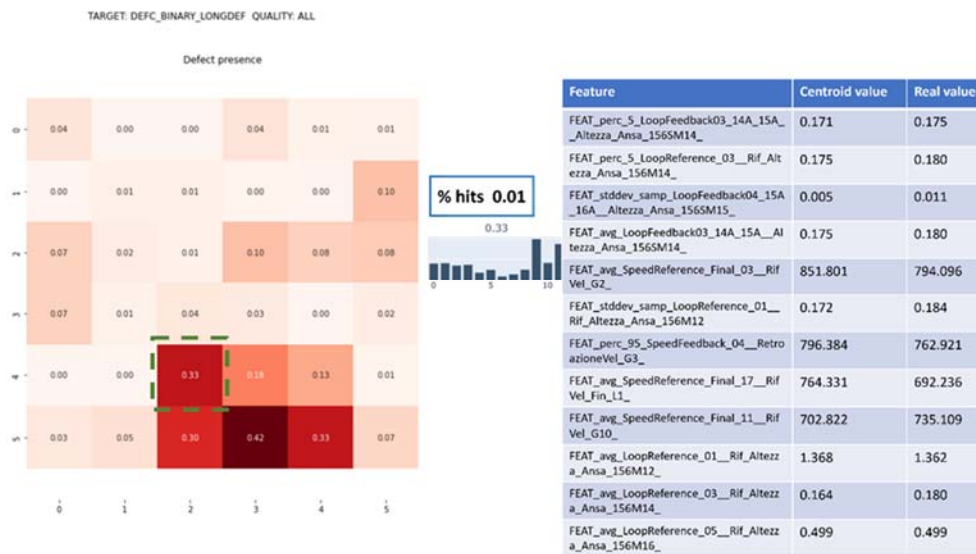


Figure 26: Example of inference results for monitoring surface quality

During the project, an advanced architecture for online data acquisition of high-resolution process and quality data was designed and implemented by means of innovative tools from Big Data technology to achieve near real time data processing.

An AI-based decision support system was developed to monitor and control the wire rod quality, in terms of profile and surface deficiencies. This support decision system exploits different SOM based models for online detection of operating conditions that can affect the rolled product quality, supporting plant operators to set up process parameters to improve it.

In addition, online visualization of real-time process signals was successfully supported through Grafana dashboards showing real-time streaming data.

With respect to the objective of reducing surface and profile defects during production, the new system allows the monitoring and modelling of the process and seems in perspective a powerful tool for identifying the appropriate corrective actions.

The set of variables collected by the system is exhaustive and complete, including both automation data in real time, production data and the outputs of the quality measurement systems involved in the process. Despite the large amount of input data and their heterogeneity, the processing times of models' inference component, resulting from the first tests, are compatible with production process times and therefore sufficient to allow corrective actions as required by the project objectives.

The decision support system helps the operators at discovering operating conditions that can be risky for the quality of the product, by identifying a set of variables that move away from the critical area, thus suggesting the operator the process parameters to be corrected. The corrections suggested are therefore much faster and more targeted.

In addition, online visualization of real-time process signals has been supported through Grafana dashboards that are very useful for controlling the signals in real time, a possibility previously limited to the IBA servers and with the signals present there not related to the product.

From the industrial point of view, a better exploitation of available data through Big Data tools enhances the insight into the steel making processes and the early detection of anomalies improves the product quality, resulting in a reduction of downgraded products.

Moreover, within the designed architecture, the separation of the cloud part, used for historical data storage and the training of the models, from the on-premises part to perform operational functions guarantees the security and availability required by a process control in real time and combines industrial needs with the advantages offered by the new cloud technologies.

#### 4.1.2. Prospects for the wire rod use case

Further steps of the new system are the following:

- to achieve the necessary reliability, it is necessary to carry out a proper test phase on a large number of productions, since the short production periods have limited the system test phase. In this way, it will be also possible to quantify an expected reduction of downgraded products, by using the new production process supervision and production quality monitoring system.
- how to integrate results of online modelling into the automation of process parameter control.

#### 4.1.3. Efforts of system's implementation

Since JSW and DA chose a cloud based solution, costs occurred for renting required hardware resources (IaaS: Infrastructure as a Service) as well as for using provided software tools (SaaS: Service as a service).

##### Storage Account

Configuration: West Europe, Block Blob Storage, General Purpose V2, LRS Redundancy, Cool Access Tier

Description	Quantity	Options	Monthly cost
Capacity	1 TB	Pay as you go	10.24 €
Write operations	10 x 10 000		1.00 €
List and Create Container Operations	10 x 10 000		0.54 €
Read operations	10 x 10 000		0.10 €
Archive High Priority Read	1 x 10 000		0.01 €
Data Retrieval	1 TB		10.26 €
Data Write	1 TB		0.00 €
			22.15 €

##### Azure Container Registry

Configuration: West Europe, Standard Tier

Description	Quantity	Options	Monthly cost
Registries	1	0.668 € x day	20.04 €



**Azure Machine Learning**

Configuration: West Europe

Description	Quantity	Options	Monthly cost
NC6s v3 (6 Core(s), 112 GB RAM)	1	Pay as you go	2 795.96 €

**Azure Databricks**

Configuration: West Europe, All-Purpose Compute Workload, Standard Tier

Description	Quantity	Options	Monthly cost
DS3V2 (4 vCPU(s), 14 GB RAM)	1	Pay as you go	198.93 €
DBU (Databricks Unit)	0.30075 x hours		219.41 €
			418.34 €

**Hardware**

Description	Quantity	Options	Cost
Cluster	1		11 800.00 €

**Installation and configuration**

Description	Quantity	Options	Cost
Cluster installation and configuration	1 Person		1 week

Moreover, the overall system needed an IBA system upgrade with the addition of the licenced component ibaPDA-Data-Store-Kafka to stream iba signals (time series data) directly to an Apache Kafka cluster.

## 4.2. Experiences and implementation: Hot rolling use case

### 4.2.1. Experiences: Implemented systems

The data management system implemented at SZFG's hot rolling use case consists of the following components:

- ibaPDA for acquisition of process data and works as the main data supplier for the system with its connection to Apache Kafka and the storage of process data via ibaAnalyzer-File-Extract tool. Since ibaPDA is installed world-wide at a lot of steel plants, the staff of SZFG is familiar with the system and the integrated tools. This made the exploitation inside the project relatively easy and fast. The performance of ibaPDA is very good because data handling and export are optimised at a high level (see table below). It could be easily integrated into the system via its ibaPDA-Data-Store-Kafka interface.
- Apache Kafka is used as a message broker and connects data producers and consumers. Apache Kafka's CPU and RAM usage is very low and its performance is high. Its default parametrisation leads to a fast increase of used disk, but this and other aspects can be improved by parameter settings.
- Samba File System is used as a data lake of non-processed data sent by ibaPDA Parquet export interface.

Samba File System was used after a first trial with a self-hosted Apache Hadoop installation for storage of the process data for offline analysis. Because it was found that Hadoop is not useful for the foreseen implementation for a relatively small amount of arising data, it was rejected. The efforts for configuration and administration were too high and too complex, and Hadoop's scalability and performance were not necessary. Therefore, with Samba File System a simple and fast solution was integrated which fulfils the demands on performance and easy administration.

- Apache Spark Structured Streaming to process online data streams coming from ibaPDA via Apache Kafka,

Apache Spark is the first choice for being applied to online data streaming.

Its performance is high with low latencies when getting data via Kafka and processes it by included functions or user defined routines. It is used at hot rolling use case to trigger the start and end of a single rolling process, divides the data stream into windows of a defined length and processes this window data.

To ensure its performance at hot rolling, an IT environment of 50-200 GB RAM and CPU power > 10 cores was installed.

For implementing processing tasks like pre-defined models in Spark Streaming, a Python library, PySpark, has to be installed.

- MariaDB is integrated into the system as a source of coil related metadata as well as a storage for results from online modelling

MariaDB is a free relational database and can be easily installed. No complex administration is necessary, and it covers all tasks of it was implemented for.

- MetaBase as a web-based dashboard and interactive exploration tool implemented for tool. For online visualisation MetaBase was integrated as service. Installation is easy and MetaBase can scan databases. It is used for the result visualisation of the online modelling.

These installed components are running under operational conditions at SZFG and fulfil the expectations of the operators.

Here some figures concerning the data transmission performance are listed.

Table 2: Data transmission times online run

		Data transmission from ... to in seconds			
	ibaPDA to Samba <sup>1)</sup>	ibaPDA to Kafka	Kafka to Spark	Spark to database	ibaPDA to database
median	< 2.0	0	1.07	0.50	1.57
mean		0	1.04	0.49	1.52

<sup>1)</sup>: Collected process data representing 5 minutes- time interval, size 40 MB

#### 4.2.2. Experiences: Procedure of data processing and analyses

First task to enable an offline analysis of data is to establish **a robust and useful data ingestion and storage**. Like mentioned above, during the project a first approach using Hadoop framework was discarded because the administrative and maintaining efforts were too high in case of the hot rolling use case. Since the number of data sources was manageable (only iba system for supplying time series data, and level 2 data regarding meta information about coils), and because the amounts of data to be handled and stored do not call for a solution which is able to handle Terabytes of data, another more simple storage solution was applied.

In steel production the single production steps from liquid steel making (blast furnace, DRI or EAF) via solidification (ingot or continuous casting) to transformation (rolling or forging) and finishing (e.g. coating, shaping etc.) are spatially and temporally separated. Additionally, the separation of steel production steps has historically been accompanied by separated systems of data collection and administration. Together with the temporal decoupling and against the background of examining a problem at one production step there was no reason found to enable the access to all data along the steel production in one storage system.

Therefore a simple solution by applying a powerful format for the data (found by Parquet format) and applying a file based storage system fulfils the requirements related to our use case.

Finally, **data ingestion and storage** was realised by applying iba Parquet export combined with a Samba file storage solution. Minor data amounts of Level 2-data were exported via data base export functions.

The **data compilation** for the offline analysis was realised mainly by Python scripts which were developed to generate data samples of

- interpolated time signals per signal, stand and coil,
- window-based signal data of equal length,
- derived statistical features on per-coil and per-window base.

These scripts include

- the reading of Parquet files containing each 5 minutes of time series signals,
- the examination of files' integrity,
- the enabling of a continuous data compilation of the rolling process of one coil at one stand across two files (see Figure 27 below),
- the identification of start and end of each single rolling process (triggering), the calculation of an interpolated signal representation of rolling processes of different coils to generate a data sample of same-length time signals for the coils with their variable rolling time,
- the generation of time intervals of equal length by rolling-window techniques,
- and the derivation of statistical features from the original data as well as of the window based data (see Figure 28 below).

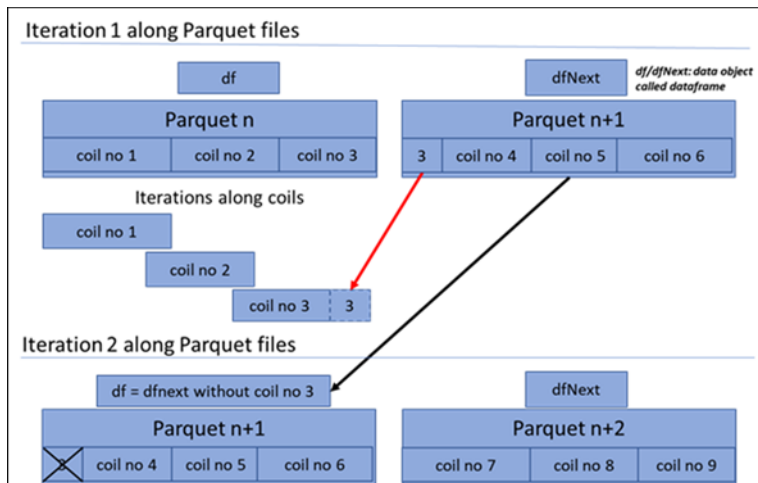


Figure 27: Continuous signal compilation of the rolling process across two files

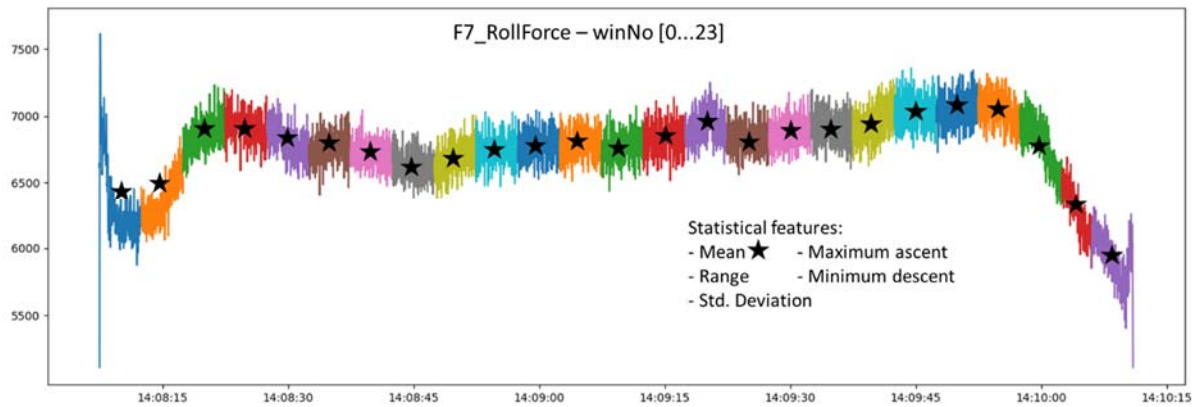


Figure 28: Example of window generation and feature extraction in time signal

To **ensure the correctness** of data, no strategies to identify and exchange outliers at original data were applied.

On the hand, no significant amount of irregular values during the investigated time periods of active rolling were found.

For that, a visual inspection was done by looking at a mass of diagrams which were transformed from bitmap format into a video stream. This enables a faster visual inspection then clicking through a mass of diagrams.

On the other hand, for the amplitude normalisation of used data, after a discussions with process experts for each signal a minimum and a maximum value was defined to apply a amplitude normalisation to values between 0 and 1. This minimum and maximum values covers the physical and technological plausible signal range. For groups of signals describing same/similar physical quantities the same normalisation parameters were set.

Furthermore with a view to the online exploitation of data, an online identification of outliers and their exchange by plausible values was thought of to be too time consuming. Therefore the data were checked after amplitude normalisation to be inside the defined min- max range, in case of an exceedance normalised values are set to 0 or 1 respectively. This corresponds to the procedure to compare the original data with their defined plausible range and then to exchange irregular values by the corresponding plausible minimum or maximum.

When producing the length-normalised signal data the interpolation algorithm itself produces a smoothing of the curves, so outliers are adjusted automatically. If the plausible range will be violated furthermore, during amplitude normalisation such values are exchanged.

**Main occurring problems or challenges** in that phase were

- wrongly generated trigger signals which made a work-around necessary for those data stored before this error was eliminated in the basing PLC system,
- misnaming of data files because of "manual" interventions in the procedure of data file storage what led to programme failures during correct reading of the data files. This made the check of the file names included in one subdirectory necessary.
- occurring memory leaks in the generation of data samples by long-running loops, e.g. when pre-processing a large number of Parquet files (compilation of a 6 months' period means 288 files of 5 minutes per day x 180 days x 50 selected signals is equal 2.592.000 iterations to calculate the interpolated representation of each selected time signal). Especially the application of the PySpark module for the reading and processing of the data files led to memory leaks in the underlying Java runtime kernel. This made a complete change to another Python module (Pandas) necessary with related additional programming work.

All those problems induced diagnostic procedures on the written code which needed much time. Since the run time for such programmes took several hours (e.g. the above described loop needs around 20 hours) the search for any errors took a long time, just to localise the error and to find solutions to fix it.

The signals collected and stored were chosen in a first "brain storming" session by people of the production unit. Since the technical/physical interrelationships to the occurring tail breaks were not known, signals were selected if they can have possibly a relation to the occurring defect.

To reduce the number of signals to focus on promising subsets of them people of the production were asked about their personal "theory" about the formation of tail breaks to identify relevant process parameters. Another approach was the application of data mining techniques to produce a type of ranking lists of the relevance of signals or derived features into the occurrence of tail breaks.

The experts' questioning lead to list of signals mainly describing forces and positional information about working and burden rolls. At that point a new variable which describes the strength class of a steel grade was added to the data list. Additionally, experts advised to limit the investigations to coils with a final set thickness up to 2.5 mm because at higher thicknesses nearly no tail breaks occur.

The data mining approach was executed with a software tool existing at BFI (coded in Matlab) which offers three different methods to generate ranking lists of the signals' influence into the target value.

## **Decision trees**

Decision tree algorithms like OC 1 and C 4.5 separate the input space by means of hyper planes to divide the input data sets into the different classes. It is possible to interpret the structure of such a tree as a sequence of premises concerning the values of single variables. Decision trees are normally used as a classification tool, but it's also possible to make use of decision trees to get a hierarchically sorted list of the input parameters by interpreting the produced tree or by means of cross-validation tests with varying data samples used for the training of the trees.

## **Categorized histograms**

Categorized histograms are a simple statistic method to detect one-dimensional dependencies between input parameters and a class information. By dividing the input data into groups by means of the class information (category), and the following comparison of the frequency distributions inside these groups one will derive information about the relationship between the selected input and the class parameter.

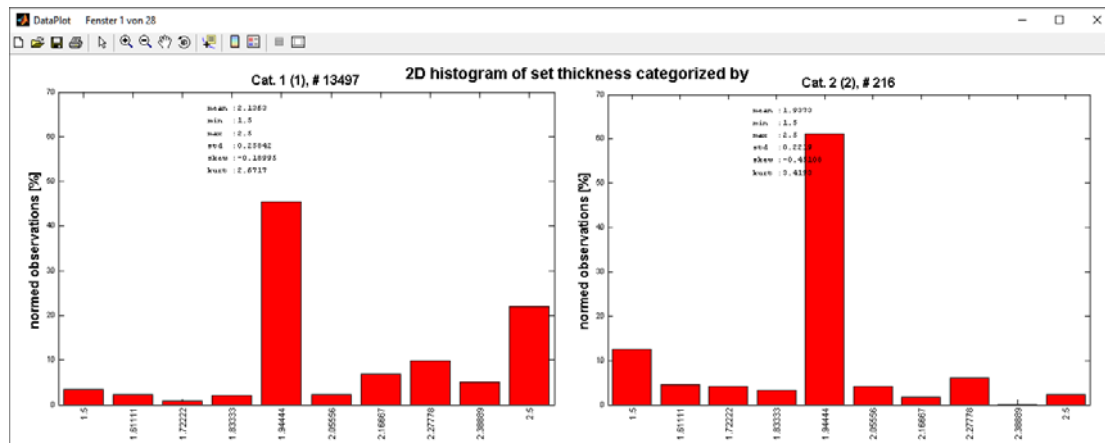


Figure 29: Example for a one signal's categorised histogram

If the frequency distributions of the categories are similar, the examined input parameter is seen to have no or minor relation to the class parameter. But if there are obvious differences between the frequency distributions, this input has a relation to the class parameter. By calculation of a similarity index, or dissimilarity respectively, a list of the input parameters can be generated showing the "level of relationship" between this input and the class parameter.

### Self-Organizing-Map (SOM)

SOM is special type of neural network which represents a non-linear transformation method from a multi-dimensional input space to a 2-dimensional surface. This transformation takes neighbourhood relationships of data points in the input space into consideration during generation of the SOM. The initial use of the SOM is for detection of clusters in multi-dimensional space, but exploiting the component plane representation of a trained SOM supplies the possibility to define a kind of non-linear correlation coefficient between the used input parameters to the defined output (see Self-Organizing Maps and Their Applications to Data Analysis; R. Ponmalai, C. Kamath; 2019, Lawrence Livermore National Laboratory; at <https://www.osti.gov/servlets/purl/1566795>). So the SOM can be used for the detection of relationships between the parameters and thus to find those inputs.

Each of these methods delivers a ranking list for the input parameters:

- for SOM and Categorised Histograms basing on a kind of "correlation" coefficient,
- for Decision Trees by the position in the list.

The following figure shows an example of such a list.

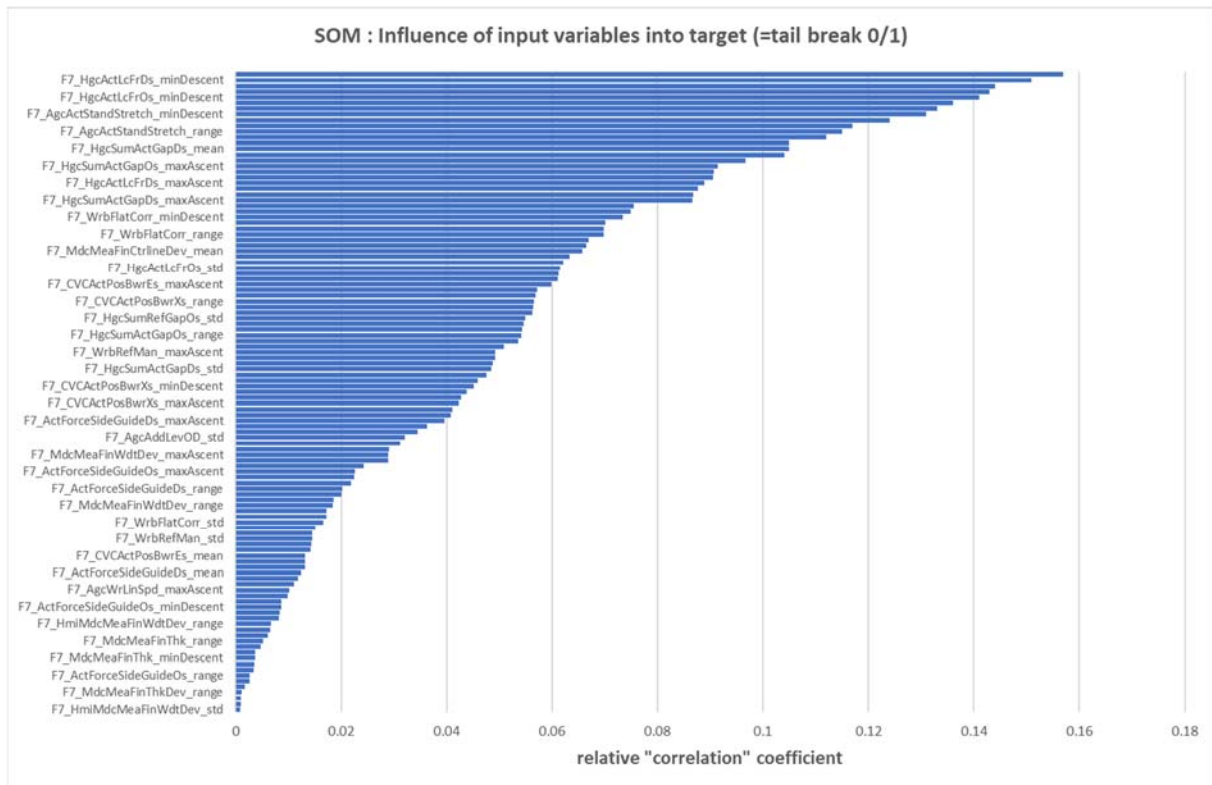


Figure 30: Ranking list by SOM, selected inputs from stand F7, statistical features

The data mining tool is limited in the memory use of loaded data. Therefore data samples were generated that includes subsets of the available signal features and coil meta data (geometry and strength class) for data mining analyses.

### Generation of data samples

One group of samples focussing on the behaviour at single stands includes the

- signal features and meta data for each strand separately  
(F1: 98, F2: 98, F3: 103, F4: 103, F5: 103, F6: 103, F7: 113 features)

Another group of samples, trying to integrate cross-cutting relations between stands, includes

- features of selected signals and meta data combined from stand F5 – F7 (58 features)
- features of selected signals and meta data combined from stand F1 – F4 (63 features)
- features of selected signals and meta data combined from stand F1 – F7 (118 features)

Tests of one sample with the different methods were summed-up by compiling the upper ten of each list. This resulting list for one sample was ordered by the number of appearances of an input. These summed-up lists for each sample were composed again in a new list, again with ordering by the number of occurrences of an input. By this procedure a resulting total list was composed.

The following figures show examples of this procedure. Figure 31 depicts the procedure for one sample/three methods, Figure 32 an example for 3 samples. For reasons of space the examples show only an excerpt of the complete document

subset of signal features selected by personnel decision							Twice under top ten
							3 and more times under top ten
SOM 15 x 15		C4.5 Decision Tree		categorised Histograms			summed list for F5, F6, F6
Variable	relative	Variable	level	Variable	relative		
F6_HgcSumActGapOs_mean	0.141	F6_HgcSumActGapOs_mean	1	F6_HgcSumActGapDs_mean	1		F5_CVCActPosBwrEs_mean
F6_HgcSumActGapDs_mean	0.139	F5_CVCActPosBwrEs_mean	2	F6_HgcSumActGapOs_mean	0.963		F5_HgcActPrFrDs_range
F5_HgcActPrFrDs_range	0.134	F6_LooperActPos_mean	3	Set Thickness	0.944		F5_HgcSumActGapDs_mean
F6_HgcActPrFrDs_range	0.129	Set Thickness	3	F7_MdcMeaFinThk_mean	0.799		F5_HgcSumActGapOs_mean
F6_ActForceSideGuideOs_mean	0.115	F6_HgcActPrFrDs_mean	4	F5_HgcSumActGapDs_mean	0.799		F6_ActForceSideGuideOs_mean
F6_AgcActStandStretch_mean	0.113	F6_HgcActPrFrDs_range	4	F6_HgcActPrFrDs_range	0.753		F6_AgcActStandStretch_mean
F5_HgcSumActGapDs_mean	0.111	F6_CVCActPosTwrEs_mean	5	Strength class	0.73		F6_AgcAddLevOD_mean
F6_AgcAddLevOD_mean	0.109	F5_HgcSumActGapOs_mean	5	F5_HgcSumActGapOs_mean	0.712		F6_CVCActPosTwrEs_mean
F6_HgcActPrFrDs_mean	0.107	F7_MdcMeaFinThk_mean	5	F6_AgcActStandStretch_mean	0.66		F6_HgcActPrFrDs_mean
F7_HgcSumActGapDs_mean	0.107	F7_MdcMeaFinThkDev_mean	5	F7_HgcSumActGapOs_mean	0.647		F6_HgcActPrFrDs_mean
F5_HgcSumActGapOs_mean	0.104	F7_WrbFlatCorr_mean	6	F7_HgcSumActGapDs_mean	0.615		F6_HgcSumActGapDs_mean
F7_HgcSumActGapOs_mean	0.1	F6_AgcWrlinSpd_mean	6	F7_AgcWrlinSpd_mean	0.591		F6_HgcSumActGapOs_mean
F7_HgcActPrFrDs_range	0.09784	F5_LooperActPos_mean	6	F6_HgcActPrFrDs_mean	0.59		F6_LooperActPos_mean
Set Thickness	0.09767	F7_MdcMeaFinThkDev_range	6	F6_AgcAddLevOD_mean	0.536		F7_HgcSumActGapDs_mean
Strength class	0.09683	F7_CVCActPosTwrEs_mean	7	F5_HgcActPrFrDs_range	0.514		F7_HgcSumActGapOs_mean
F7_MdcMeaFinThkDev_minDescent	0.09376	F5_WrbRefMan_mean	7	F7_AgcActStandStretch_mean	0.462		F7_MdcMeaFinThk_mean
F7_MdcMeaFinThk_mean	0.09232	F5_AgcActStandStretch_mean	7	F7_HgcActPrFrDs_range	0.454		F7_MdcMeaFinThkDev_mean
F5_ActForceSideGuideOs_mean	0.0914	F7_AgcAddLevOD_mean	7	F6_ActForceSideGuideOs_mean	0.421		F7_MdcMeaFinThkDev_minDescent
F6_CVCActPosTwrEs_mean	0.07921	F6_ActForceSideGuideOs_mean	7	F7_HgcActPrFrDs_mean	0.412		Strength class
F6_CVCActPosBwrEs_mean	0.07921	F5_ActForceSideGuideDs_mean	7	F6_AgcWrlinSpd_mean	0.41		Set thickness

Figure 31: Example for summarising results at one sample for three data mining methods

summed list F5 - F7	summed list F1 - F4	summed list F1 - F7	summed	times under top 10
F5_CVCActPosBwrEs_mean	F1_ActForceSideGuideDs_mean	F1_ActForceSideGuideDs_mean	set thickness	3
F5_HgcActPrFrDs_range	F1_AgcActStandStretch_mean	F1_HgcRefODMan_mean	F1_ActForceSideGuideDs_mean	2
F5_HgcSumActGapDs_mean	F1_AgcWrlinSpd_mean	F2_HgcActPrFrDs_mean	F1_HgcRefODMan_mean	2
F5_HgcSumActGapOs_mean	F1_AgcWrlinSpd_std	F2_HgcSumActGapDs_mean	F2_HgcSumActGapDs_mean	2
F6_ActForceSideGuideOs_mean	F1_HgcActPrFrDs_minDescent	F2_HgcSumActGapOs_mean	F2_HgcSumActGapOs_mean	2
F6_AgcActStandStretch_mean	F1_HgcRefODMan_mean	F2_LooperActPos_mean	F3_AgcWrlinSpd_std	2
F6_AgcAddLevOD_mean	F2_AgcWrlinSpd_std	F3_AgcActStandStretch_mean	F3_HgcSumActGapDs_mean	2
F6_CVCActPosTwrEs_mean	F2_HgcActPrFrDs_mean	F3_AgcWrlinSpd_std	F3_HgcSumActGapOs_mean	2
F6_HgcActPrFrDs_mean	F2_HgcSumActGapDs_mean	F3_HgcActPrFrDs_minDescent	F4_AgcActStandStretch_mean	2
F6_HgcActPrFrDs_range	F2_HgcSumActGapOs_mean	F3_HgcSumActGapDs_mean	F4_HgcActPrFrDs_mean	2
F6_HgcSumActGapDs_mean	F3_AgcWrlinSpd_std	F3_HgcSumActGapOs_mean	F4_HgcActPrFrDs_minDescent	2
F6_HgcSumActGapOs_mean	F3_CVCActPosTwrEs_mean	F4_AgcActStandStretch_mean	F4_HgcSumRefGapDs_mean	2
F6_LooperActPos_mean	F3_HgcActPrFrDs_range	F4_AgcWrlinSpd_std	F5_HgcActPrFrDs_range	2
F7_HgcSumActGapDs_mean	F3_HgcSumActGapDs_mean	F4_HgcActPrFrDs_mean	F5_HgcSumActGapDs_mean	2
F7_HgcSumActGapOs_mean	F3_HgcSumActGapOs_mean	F4_HgcActPrFrDs_minDescent	F6_HgcActPrFrDs_range	2
F7_MdcMeaFinThk_mean	F4_AgcActStandStretch_mean	F4_HgcSumRefGapDs_mean	F6_HgcSumActGapDs_mean	2
F7_MdcMeaFinThkDev_mean	F4_HgcActPrFrDs_mean	F5_HgcActPrFrDs_minDescent	F6_HgcSumActGapOs_mean	2
F7_MdcMeaFinThkDev_minDescent	F4_HgcActPrFrDs_minDescent	F5_HgcActPrFrDs_range	F7_MdcMeaFinThkDev_mean	2
festigkeit	F4_HgcSumRefGapDs_mean	F5_HgcSumActGapDs_mean		
Soll_Dicke	F4_LooperActPos_std	F5_LooperActPos_mean		
	Soll_Dicke	F6_HgcActPrFrDs_range		
		F6_HgcSumActGapDs_mean		
		F6_HgcSumActGapOs_mean		
		F7_MdcMeaFinThkDev_mean		
		F7_MdcMeaFinThkDev_range		
		Soll_Dicke		

Figure 32: Example for summarising results of 3 samples

The different compositions of selected features were tested by several machine learning technologies, because first tests by comparison of univariate distributions or standard deviations showed no significant differences between features/signals belonging to class without or with tail breaks.

Because of the imbalance between coils with the occurring tail break defect and those without, unsupervised and semi-supervised modelling techniques were in the focus of the methodology search. But those methods should be able to used as an anomaly detector.

Supervised techniques were also been selected if they are able (in theory) to handle the low number of defect observations with respect to the "good" ones.

### Autoencoder (various architectures)

An autoencoder is a type of artificial neural network that is used to learn efficient data encodings so that it learns to map its input to its output. It does not require a target variable like the traditional Y, which is why it is classified as an semi-supervised learning. A typical autoencoder consists of three parts. An encoder that reduces the dimensionality of a high dimensional dataset to a lower dimensionality. A hidden layer called the bottleneck connects the encoder and the decoder and contains the reduced representation of the input that is being fed into the decoder. The decoding



mirrors the encoding part in the number of hidden layers and neurons. The decoder then reconstructs the data into the original input which is used as the output. The main idea of this network is to minimize the reconstruction error, which is basically the difference between the original input and the reconstructed output. The autoencoder is trained to reconstruct normal data, and the network adjusts its weights for that data. When abnormal data is fed to the network, the reconstruction error between the output and input data should be relatively higher than for normal data. If the reconstruction error is higher than a set threshold, we can classify the input as an anomaly. Tested variations of the Deep Autoencoder Neural Network (AE) are

- **Convolutional AE (CNN-AE)**  
1D Convolutional Neural Networks are mainly designed for text and 1D signals. Convolution is a mathematical way of combining two sets of information. In the case of CNN, a convolutional layer contains a set of filters (or kernels) which are applied to the input data to filter the information and produce a feature map. To perform convolution the filter slid across the input, doing dot product between every element of the filter and the input. The result for the area where convolution takes place, is written down in the feature map.  
Convolutional AutoEncoders (CNNAEs) are trained to learn optimal filters that minimize the reconstruction error between input and output. Once these filters have been learned, they can be applied to any input in order to extract features.  
See <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>
- **Fully connected and convolutional AE (FCC-AE)**  
Fully connected Autoencoders are those where every node on the previous layer is connected to every node on the current layer.
- **Long-Short term memory AE (LSTM-AE)**  
Recurrent neural networks such as the LSTM or Long Short-Term Memory network are used to support the sequential or time series data. The LSTM Autoencoder is an implementation of an autoencoder with LSTM architecture. It allows us to understand the pattern of sequential data with LSTM then extract the features with Autoencoders to recreate the input sequence.  
See <https://bobrupakroy.medium.com/lstm-autoencoders-a45a04667346>

### Support Vector Machines

SVM can be used in an unsupervised environment for anomaly detection. The basic idea of single-class SVM is that the training data in the input space are mapped into the feature space via a kernel function (generally transforms the training data set so that a non-linear decision surface can be transformed into a linear equation in a higher number of dimension spaces), and a hyperplane with a maximum margin is found in the feature space to separate the mapped data from the origin. The scikit-learn library provides an implementation of the SVM. The sensitivity of the model is controlled by a hyperparameter "nu", which should be tuned to the approximate ratio of anomalies in the data.

### Random forest

Random forest is a supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. Random forest is an ensemble learning method that combines multiple deep decision trees, trained on different parts of the same training set, see <https://www.ibm.com/cloud/learn/random-forest>.

### Isolation forest

Isolation Forests are designed using decision trees and implemented in an unsupervised manner. In an Isolation Forest, randomly selected data is processed in a tree structure based on randomly selected characteristics. Samples that reach further into the tree and require more cuts to isolate are less likely to be anomalies. Likewise, samples located on the shorter branches of the tree are more likely to be anomalies because it was easier for the tree to separate them from other observations. The scikit-learn library provides an implementation of Isolation Forest in the IsolationForest class (Placeholder2).

The key hyperparameters of the model are the "n\_estimators" parameter, which sets the number of trees to be created, and the "contamination" parameter, which is used to determine the number of anomalies in the dataset. As with the one-class SVM, the model is trained on the normal class, and the output is +1 for the normal class and -1 for the abnormal class, so the predicted labels of the test set must be changed before the predictions are evaluated.

### K-Nearest Neighbour

K-Nearest Neighbour is a non-parametric and supervised learning technique. The KNN algorithm calculates the distance between the test data and all the training points, and then selects the K closest data points to the test data. Finally, it sorts the data point to the class to which most of the K data points correspond. (see <https://scikit-learn.org/stable/modules/neighbors.html>)

For the **evaluation and comparison of modelling results** the confusion matrix and derived measures was used.

To quantify achieved results of modelling approaches, the so-called confusion matrix offers a simple but informative representation of achieved results and the derivation of related KPIs.

The confusion matrix below shows in general the results of modelling in a 2-classes case.

		Predicted Values	
Observed Values	TN True Negative	FP False Positive	
	FN False Negative	TP True Positive	

There exist a negative class and a positive class. After modelling the number of examples are counted which belong to one of the quarters: observed "negatives" which are modelled as "negatives", observed "negatives" which are modelled as "positives", and vice versa.

Figure 33: Schematic confusion matrix

From this matrix several KPIs can be derived:

**Accuracy:**  $\frac{TN + TP}{Total}$  Number of correctly classified data over the total number of data.

**Recall:**  $\frac{TP}{(TP + FN)}$  What proportion of true anomalies was identified ?

**Precision:**  $\frac{TP}{(TP + FP)}$  What proportion of identified anomalies are true anomalies?

**F1 Score:**  $2 \times \frac{Precision \times Recall}{(Precision + Recall)}$  Combining both Recall and Precision (harmonic mean)

Examples of achieved results with calculated KPIs can be found in the following table.

Table 2 : Exemplary results of modelling tests

Model / Sample group	confusion matrix	KPIs Train/Test	Remark
AE features, separate per stand F1 – F7	No confusion matrix for training, because AE delivers no classification results		The classification in test phase is done by comparing the summed recon- struction error for all input/output nodes with a pre-defined threshold. Result not satisfying for tail break class
	<b>Test</b>	<b>model</b>	
	obs	2395      1	
		35      5	
		Acc: -----/0.886 Prec: -----/0.205 Recall: -----/0.142 F1: -----/ <b>0.484</b>	

<b>kNN</b> features, separate per stand F1 – F7	<b>Train</b>	<i>model</i>		Acc: 0.985/0.981 Prec: 0.963/0.385 Recall: 0.152/0.119 F1: 0.252/ <b>0.182</b>	useless, because tail breaks are classified not sufficiently in training and test
	<i>obs</i>	9608	1		
		144	26		
	<b>Test</b>	<i>model</i>			
	<i>obs</i>	2395	1		
35		5			
<b>SVM</b> features, separate per stand F1 – F7	<b>Train</b>	<i>model</i>		Acc: 0.796/0.800 Prec: 0.035/0.043 Recall: 0.400/0.500 F1: 0.140/ <b>0.079</b>	useless, because tail breaks are classified not sufficiently in training and test
	<i>obs</i>	7720	1889		
		102	68		
	<b>Test</b>	<i>model</i>			
	<i>obs</i>	1935	468		
21		21			
<b>k Means</b> features, separate per stand F1 – F7	<b>Train</b>	<i>model</i>		Acc: 0.395/0.390 Prec: 0.026/0.026 Recall: 0.912/0.929 F1: 0.051/ <b>0.051</b>	useless, because class "OK" is classified not sufficiently in training and test
	<i>obs</i>	3705	5904		
		15	155		
	<b>Test</b>	<i>model</i>			
	<i>obs</i>	915	1488		
3		39			
<b>Isolation Forrest</b> features, separate per stand F1 – F7	<b>Train</b>	<i>model</i>		Acc: 0.799/0.801 Prec: 0.040/0.043 Recall: 0.459/0.500 F1: 0.074/ <b>0.079</b>	useless, because both classes are classified not sufficiently in training and test
	<i>obs</i>	7731	1878		
		92	78		
	<b>Test</b>	<i>model</i>			
	<i>obs</i>	1938	465		
21		21			
<b>Random Forrest</b> features, separate per stand F1 – F7	<b>Train</b>	<i>model</i>		Acc: 0.983/0.982 Prec: 0 /0 Recall: 0 /0 F1: 0 / <b>0</b>	useless, because Random Forrest model cannot handle such unbalanced classes
	<i>obs</i>	9609	0		
		170	0		
	<b>Test</b>	<i>model</i>			
	<i>obs</i>	2403	0		
42		0			

Other tests were executed with samples of the other feature based sample group and samples consisting of interpolated signals. For this only smaller subsets of signals were used, because each input vector consisted of 500 values per applied signal which produces a bad relation between the number of input values and the number of tail break examples.

Additional tests were executed to find the best parameter values for investigated model types, e.g. Autoencoder: number of layers and nodes, k-Nearest-Neighbours: number of neighbours . For other methods parameter values were used from literature.

The failure in finding a model of sufficient accuracy may have several reasons:

- the necessary information to identify the generation of tail breaks is not included in the data
- the signals or derived features mostly related to the generation of tail breaks were not identified correctly. The integration of signals/features with no relation to the defect makes a proper model generation difficult.
- the bad ratio between coils with no tail break and those with an occurring tail break enhances the risk of failures in modeling
- the applied model types can not detect the patterns in the signals to identify tail breaks.

On the following pages some screen shots show monitoring information about the running system:

Figure 34: Excerpt from MariaBD with rolling window (aggeegated per 10 seconds) data from Spark Streaming

Figure 35: Web based Spark monitoring: Running and completed jobs

Figure 36: Web based Spark monitoring: Streaming query statistics

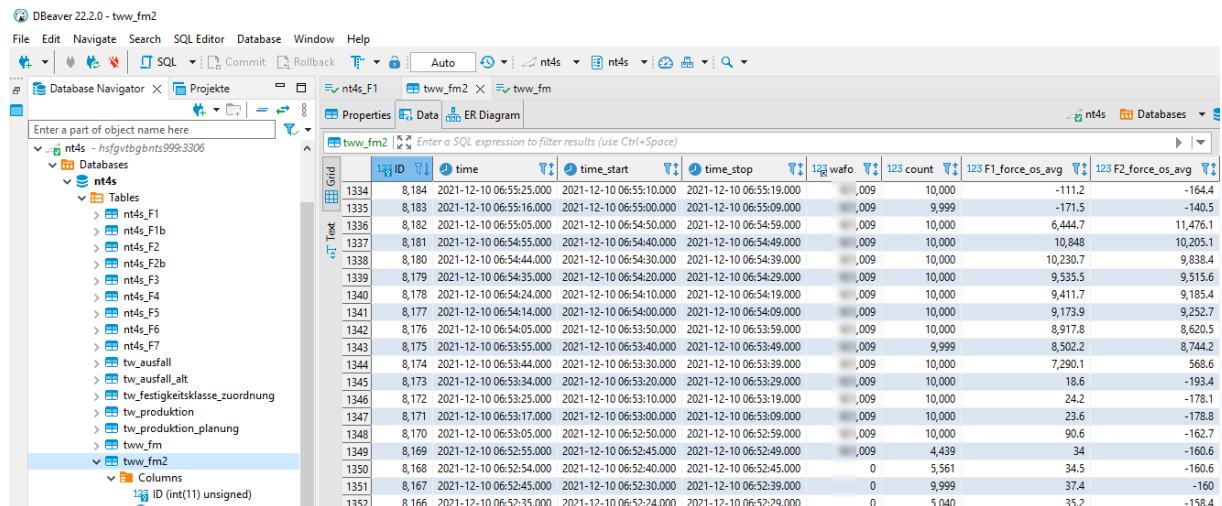


Figure 34: Example of MariaDB, results of rolling window (10 min length) with aggregation – screen shot

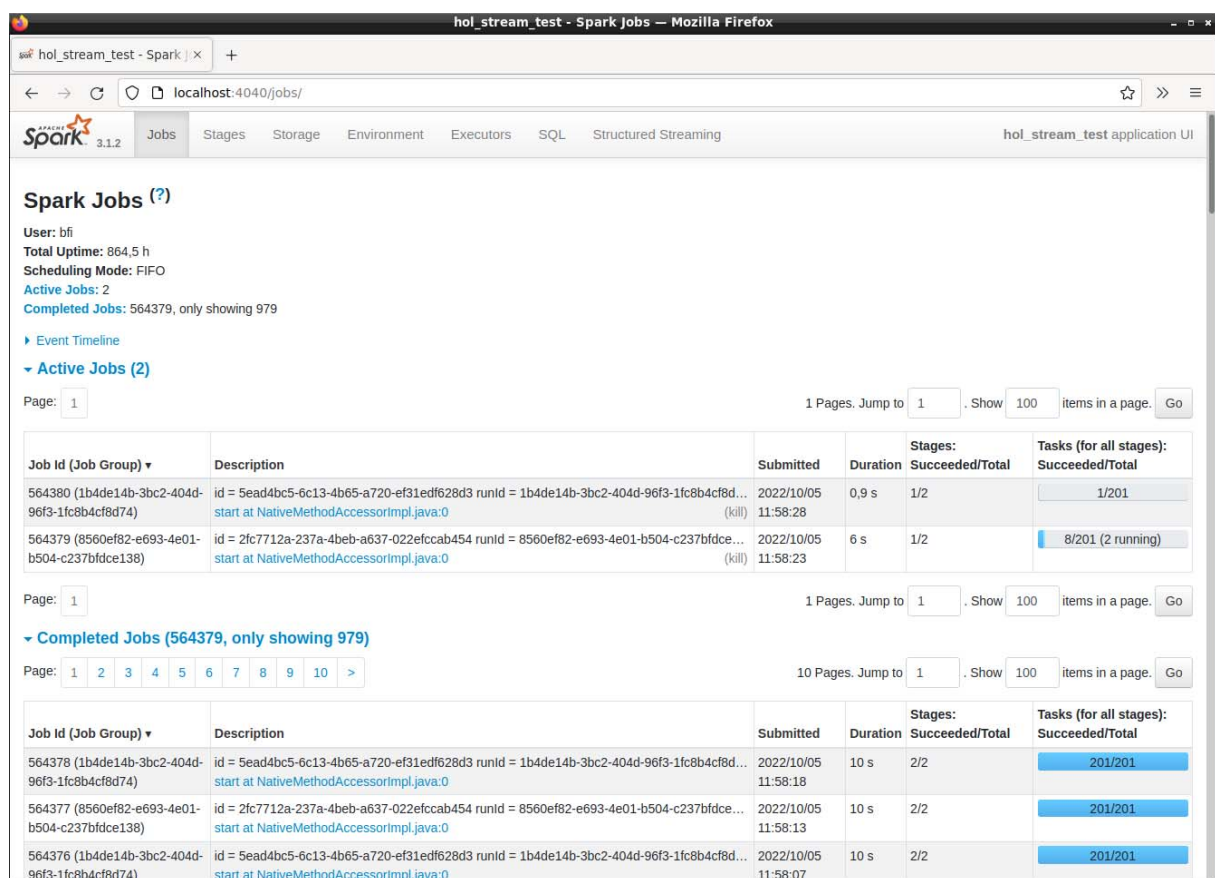


Figure 35: Example of Spark Job Monitoring - screen shot

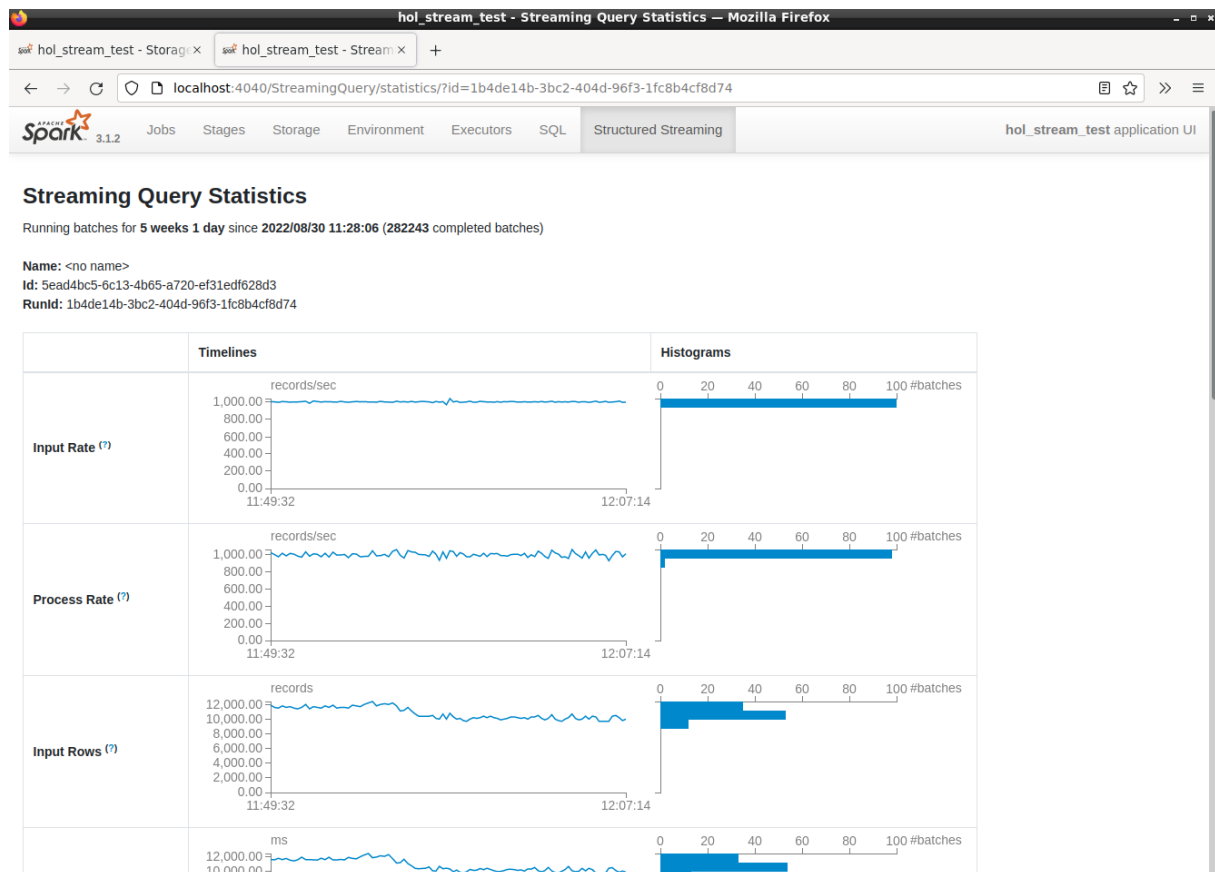


Figure 36: Example of Spark Streaming Monitoring - screen shot

#### 4.2.3. Prospects for the hot rolling use case

To present some ideas for what to do in the next future, here some thoughts and ideas are listed.

The analysis of hot rolling data with semi supervised Autoencoders did not achieve results that were production ready as it was expected. The methodology however of using an Autoencoder to represent the normal production state of a plant and to find unwanted states or critical plant status as anomaly detection should be considered as a valid use case in future work.

Since the other investigated techniques achieved also no results of sufficient accuracy first thoughts and investigations were made at the end of the project's duration to find other promising techniques. For example, at the use case of strip tearing in hot rolling the methodology of matrix profiles which was not investigated in the scope of NT4S promises a better suitability. The matrix profile is a vector that stores the z-normalized Euclidean distance between any subsequence within a time series and its nearest neighbour. In short this provides an elaborated method to compute and match patterns in time series. The Python library Stumpy is an effective implementation for offline and online analysis of the matrix profile. (see <https://stumpy.readthedocs.io/en/latest/index.html>).

Other more data related thoughts, are to think about the possibilities to extend the information about the tail breaks. At the moment there is no information about the stand where a tail break happens. Here a camera observation at the stand's exits could be (perhaps) a way to identify the related aggregate and thus helps to narrow down the necessary set of signals for investigations.

#### 4.2.4. Efforts of system's implementation

The efforts for the systems implementation are basing on activities executed by BFI, SZFG and SZMF. The given figures are an estimation because partially described tasks are overlapping, so a clear allocation is not possible.

##### BFI PM 9.5:

- |  |     |
|--|-----|
| • incorporation into new architectures and technologies (tools)          | 1.0 |
| • programming code for data handling and compilation of data samples     | 4.0 |
| • programming analyses and modelling functions in Python                 | 1.5 |
| • execution of tests for parametrisation of models and model development | 1.5 |
| • parametrisation Kafka streaming tool                                   | 0.5 |

##### SZFG/SZMF PM 9 / 4

- |  |            |
|--|------------|
| • preparation and installation of the hardware for the system including related software tools, interconnection of different systems, incl. test of Hadoop cluster + change to Samba | 5.5 / 0.5  |
| • programming code / software parametrisation for data handling  | 1.0 / 0.5  |
| • programming analyses and modelling functions in Python   | -----/ 1.5 |
| • trouble shooting and maintenance of the systems (hardware, Samba, Linux machine for IDEs, Kafka, interconnections)   | 2.5 / 1    |

Equipment:	computer hardware	11.000 Euro
	related software licenses	3000 Euro

### 4.3. Cold rolling use case

#### 4.3.1. Experiences: Implemented systems

The new system based on Big Data architecture installed at Marcegaglia site consists of the following components:

- ibaPDA-Data-Store-Kafka process for the ingestion of process parameters from ibaPDA system by streaming data directly to Apache Kafka.

ibaPDA-Data-Store-Kafka is a powerful datastore that requires basic knowledge about ibaPDA and the target cloud or cluster storage technology although the performances are very high. The implemented system has benefited greatly from using it by speeding up and simplifying the streaming acquisition process.

- Apache NIFI flow for the ingestion, transformation and transmission of the involved data from different equipment system into the processing platform.

It works as the main data supplier for what concern the cold path implementation because it is involved in the fusion, preparation of data and activation of the executables necessary to run the processing. In the meantime, it is also responsible for the historicization of the acquired data to fill the HDFS datalake with the row files. Apache Nifi has been an efficient choice for data management because it facilitated an easy-to-use, powerful, and reliable way to distribute and process data over numerous resources providing high throughput and data buffering despite fluctuations in processing and flow rates.

- Apache Kafka is used as a message broker connecting data producers and consumers in the hot path.

Because it relies on OS kernels it moves the data more quickly and works on the principle of zero-copy giving high throughput.

But it lacks a full set of management and monitoring tools, then, for the stream ingestion through Kafka broker, the test has been carried out by means of Conductor in addition to some specific Kafka distributed scripts.

Conductor is a powerful tool to manage the cluster Kafka and verify the configuration related to the topic and the receiving messages. The predefined Kafka consumer script were used instead to directly monitor the arrival of messages and validate their contents.

- HDFS is used as a data lake of row files acquired from different equipment systems by means of shared folders.

Despite the effort to configure and manage Hadoop is high and requires administration skills, it has been kept within the platform because it represents a good source for subsequent further data processing. The configuration used is the single node for Windows system then the full features such as fault tolerance, scalability and distribution of data has not been fully tested but it was a good starting point to become familiar with the integration and use of innovative and performing tools of Big Data. For the data ingestion testing, the verification that the data from different source systems were adequately extracted and correctly loaded into HDFS, apart the Command Line Interface, we used the provides Web User Interface to browse through HDFS file system and view list of directories and file contents.

- Apache Spark Structured Streaming to process online data streams coming from ibaPDA via Apache Kafka

Apache Spark Structured Streaming is the stream processing framework built on top of Apache Spark SQL engine, with the main goal to make easier to build end-to-end streaming applications, which integrate with storage, serving systems, and batch jobs in a consistent and fault-tolerant way. In our application, by means PySpark library, it is able to natively join a Kafka topic, trigger the start of a user defined routines on a micro-batch of streaming data, integrate static data and call the model producing prediction on target variables. The final results through Sink are then stored on relational database with good performance and latencies.

- MySQL is the integrated database into the system acting as a source of coil related metadata as well as a storage for results from online modelling.

MySQL is a free relational database easily installed and configurable via XAMPP, a completely free, easy to install Apache distribution containing MySQL, PHP, and Perl. Infact, the XAMPP open-source package has been set up by Apache Friends to be incredibly easy to install and to use

- A dedicated web-based application leveraging on HTTP/php for the interactive exploration of batch computation and online visualization of near real-time results of hot path.

The components for the server-side execution are part of XAMPP distribution that produces HTML pages visualized in standard browser. For online visualisation the application shows the current coil under processing and the predicted results are refreshed each 0.5 secs. The cold path produced results instead, can be explored interactively because it produces classification of coil quality about every 10 minutes.

The new installed Big Data Platform at the industrial site related to the scope of the examined problem involves two types of computing path, the cold and hot. The cold path is not critical in terms of latency and as from the assessment, the results of the preliminary coil-length prediction are available for the next step after 12-15 seconds from the reception of all involved data. The Classification procedure that takes these last results and compute the global and section-related criticality is a bit time-consuming because it takes from 2 to 4 minutes for the calculation on a single coil and for the writing of results on database.

For the other path, given that Big Data application involve the processing of significant data in a very short interval of time with vast computing resources, architecture plays an important role so, between necessary performance testing, to avoid bottlenecks, the majorly focused were on data loading and throughput. In this area, the rate at which the data consumed from streaming source and the rate at which the data created in the output database were observed. For such scope, the performance of the individual components (Kafka, Spark Structured Streaming) was tested to identify possible bottlenecks.

The evaluation was done with an IBA acquisition system providing 16 signals with a rate of 1 ms into a Kafka topic divided in 3 partitions and the consumer application based on Spark Structured Streaming natively connected to the broker running the inference model. An evaluation of times needed for data preparation of model input data and inference of models for the 1 second micro-batch of streamed data has been done and resumed in the following sentence: the average value starting from the trigger expiration (each second) to the availability of results, is less than 1 seconds then the results of the end-to-end pipeline from IBA system to the HMI real-time visualization is almost 2-2.5 seconds.

In terms of accuracy, it has been verified that all coils classified, declared to have subsequent problems in galvanization by the models, presented concrete anomalies during the real production process in about 80% of the verified cases.

Below is a summary of the main cases that emerged during the phase of verifying the forecasts of the implemented ANN.

#### Accurate cases:

Accurate cases are the cases in which the performance of the network forecast can be considered satisfactory, both as regards the accuracy and the response trend, which correctly replicates what is detected on the plant in correspondence with the input. This can be seen in the following pictures for two different coils:



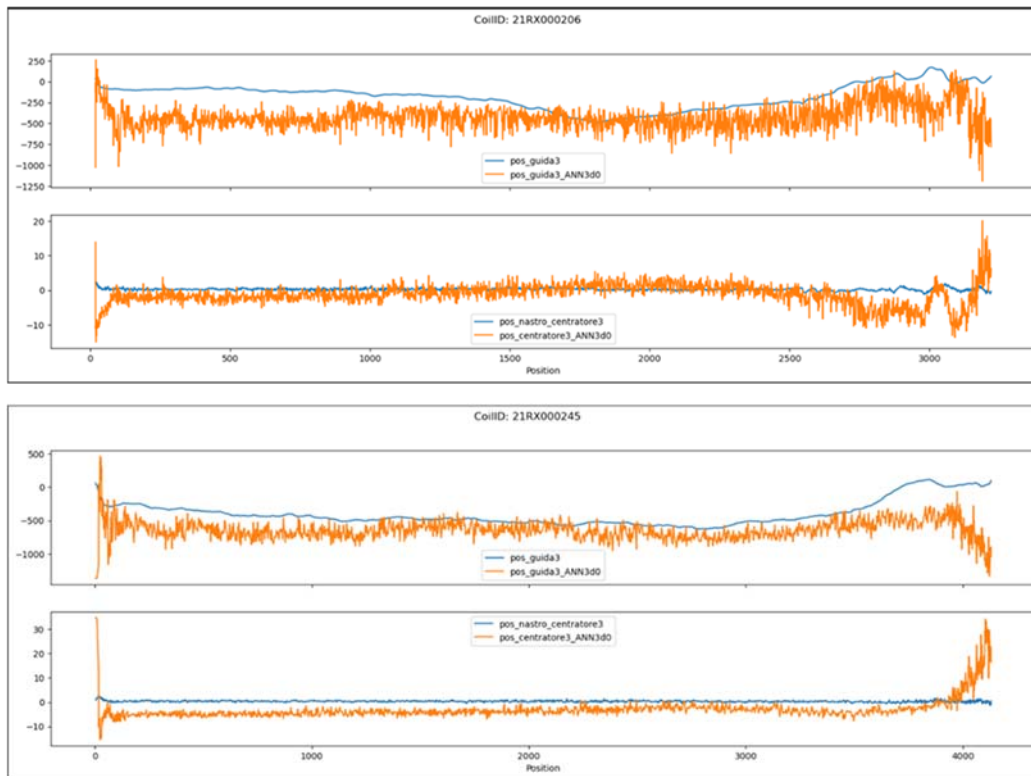


Figure 37: Accurate cases

#### Shifted cases:

Shifted cases, are the cases in which the network forecast, although satisfactory as regards the output trend, is shifted probably due to a considerable and systematic displacement with respect to the average of one or more quantities input of greatest influence. This can be seen in the following pictures for two different coils:

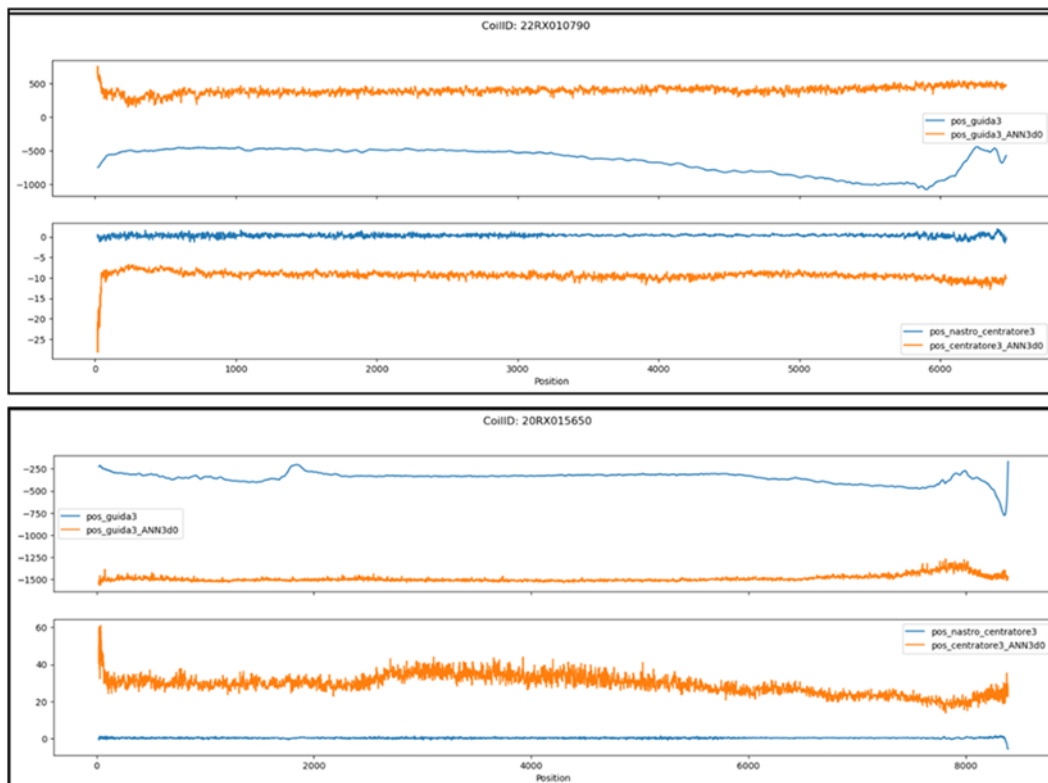


Figure 38: Shifted cases

### Inverted cases:

Inverted cases, are the cases in which the forecast of the network, especially as regards the ends of the coil (head and tail), outside of a general satisfactory accuracy, presents areas in which the trend manifests an opposite trend compared to what is shown in the measured data (zone with increasing trend which is expected to decrease trend, or vice versa), probably due to a continuous regulation applied on the line carried out by modifying one or more control and input parameters of the network according to a opposite criterion compared to what was found in the average of the cases. This can be seen in the following pictures for two different coils:

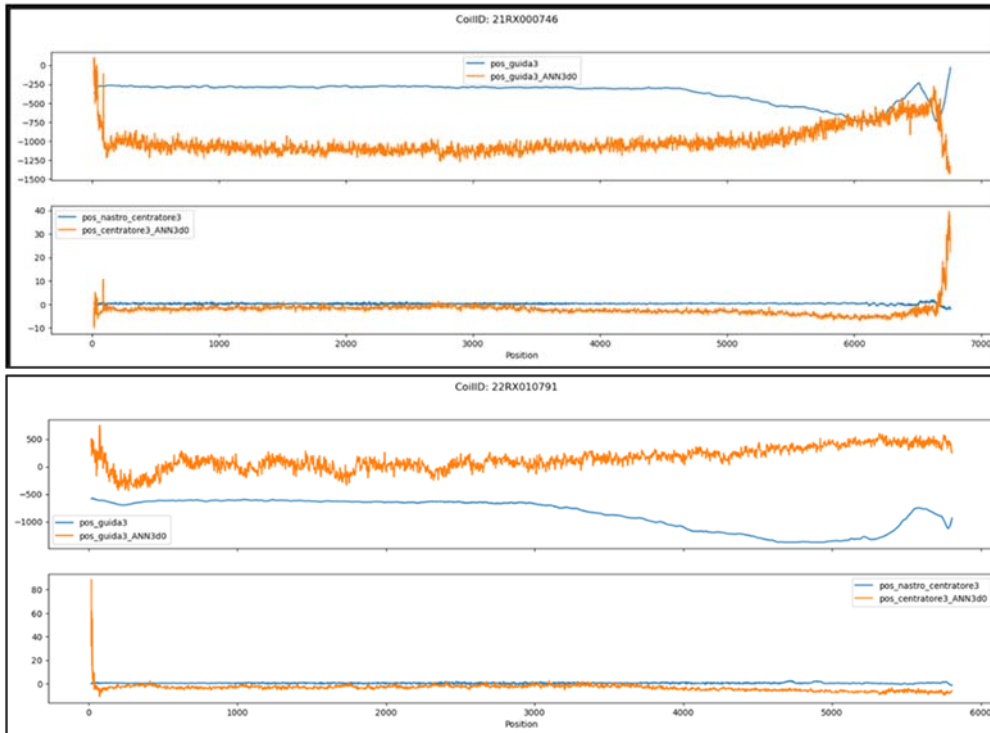


Figure 39: Inverted cases

### Cases with special causes of variation:

Cases with special causes, are the cases in which in some limited areas of the strip there is a local alteration of one or more of the most influential input parameters: the network foresees this alteration, which however, due to the sporadicity, does not necessarily involved feedback from the control system in the practical case of the exercise. This can be seen in the following pictures for two different coils:



Figure 40: Cases with special causes of variation

#### 4.3.2. Prospects for the cold rolling use case

The results of the implementation of the ML model, in terms of productivity optimization, are quite promising and interesting even if still preliminary because the time period of testing was quite short. It is also noteworthy that the most important advantage of this project is related to the possibility to prevent, *ab initio*, the occurrence of the flatness features in cold rolling process. Nevertheless, this requires the understanding and identification of all the cold rolled strip features causing the strip guidance issues in the HDG line. This activity represents the future development of the concluded project.

#### 4.3.3. Efforts for implementation

In order to leverage Big Data in steel process for revenues and profits, in our experience, it was first necessary to fully understand the use case business objective before exploring Big Data technologies and solution. After that we passed to identify the infrastructure challenges to effectively leverage data. The industrial site for this scope needed to adapt the IT infrastructure without lose the control on critical data.

Another critical question was the competencies, find or train human resources to use big data strategy. For example, our use case needed the real-time processing capabilities offered by Spark then also expertise on it.

Definitely, Big Data costs can be considered into the main categories: infrastructure, human resources and maintenance. The infrastructure costs are related mainly to networking, data preservation, data processing but to alleviate the big cost of big data one solution is the leveraging open source and managed big data platforms. Indeed, the consequences of the popularity of cloud-based software led to the proliferation of managed proprietary and open-source software that is opposed to licensed and on-premises solutions. All the software is hosted and runs on vendor hardware, and all the

maintenance work is away from the end-user. In our solution, issues related to security and performance made important to cope with ever-increasing amounts of data with an on-premise solution that involved a lot of effort for the installation, management and maintenance of the solution.

**Estimation of costs:**

The estimated cost of implementation of the complete system could depend significantly on both the brown field level of plants instrumentation, data organizations and presence of specialized employers. In additions to this it is worthy to note the importance of eventual software license cost (for example of Tensil-Pro). Anyway, the cost for complete implementation could be estimated at around 300K€ (not including Tensil-Pro and Self-Training license and development costs).