

EnerMan

Energy Efficient Manufacturing System Management

D3.1 - Big Data Collection and Analytics Platform and Analytics Report

Date : 30/06/2022

Deliverable No : D3.1

Responsible Partner : ITMLCY

Dissemination Level : Public



HORIZON 2020

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No **958478**



Short Description	
This deliverable offers comprehensive state-of-the-art research on data pre-processing and analytics techniques that took place during the design phase of the project and led to the selected techniques relevant to the EnerMan project. Additionally, it offers a complete and thorough description of the EnerMan Big data analytics engine in terms of both architecture and functionalities.	

Project Information	
Project Acronym:	EnerMan
Project Title:	ENERgy-efficient manufacturing system MANagement
Project Coordinator:	Dr. Ing. Giuseppe D'Angelo CRF giuseppe.dangelo@crf.it
Duration:	36 months

Document Information & Version Management			
Document Title:		D3.1 - Big Data Collection and Analytics platform and Analytics report	
Document Type:		Report	
Main Author(s):		ITML	
Contributor(s):		FHOOE, UNINA, STS, TSI, SIMPLAN, ITMLCY, UOP	
Reviewed by:		SUPM, AEGIS	
Approved by:		Dr Ing. Giuseppe D'Angelo (CRF)	
Version	Date	Modified by	Comments
V0.1	14/02/2022	Panagiotis Rodosthenous (ITML)	Initial version
V0.2	13/06/2022	Andrea Rega (UNINA)	Draft input to section 2.3.1
V0.3	15/06/2022	Panagiotis Rodosthenous (ITML)	Draft input to section 2.2
V0.4	15/06/2022	Konstantinos Bouklas (ITML)	Draft input to section 2.3
V0.5	15/06/2022	Mina Marmpena (ITML)	Draft input to Chapter 3
V0.6	15/06/2022	Andreas Miaoudakis (STS)	Draft input to section 2.3.3
V0.7	17/06/2022	Mina Marmpena (ITML)	Draft ready for internal review
V0.8	29/06/2022	Mina Marmpena (ITML), Panagiotis Rodosthenous (ITML), Konstantinos Bouklas (ITML)	Peer review comments addressed and ready for submission
V0.9	29/06/2022	Kubra Yurduseven (INTRACT)	Format review
V1.0	30/06/2022	Ing. Giuseppe D'Angelo (CRF)	Submitted version

Disclaimer

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation, or both. The publication reflects the author's views. The European Commission is not liable for any use that may be made of the information contained therein.

TABLE OF CONTENT

EXECUTIVE SUMMARY 8

1. INTRODUCTION 9

2. STATE-OF-THE-ART 10

2.1. Data pre-processing 10

2.1.1. Data cleaning..... 10

2.1.2. Data encoding 11

2.1.3. Data scaling and transformation..... 11

2.1.4. Data resampling 12

2.1.5. Feature engineering..... 12

2.1.6. Feature selection..... 13

2.2. Data analytics..... 13

2.2.1. Descriptive data analytics, exploratory data analysis and machine learning 13

2.2.2. Unsupervised learning – Clustering 14

2.2.3. Unsupervised learning – Anomaly detection..... 15

2.2.4. Supervised learning - Models..... 16

2.2.5. Supervised learning – ML model performance estimation metrics..... 18

3. BIG DATA ANALYTICS ENGINE 20

3.1. Data and artifacts storage infrastructure 20

3.1.1. Time series database..... 21

3.1.2. NoSQL storage..... 22

3.1.3. Harmonized CSV cloud storage data and TSDB operations 22

3.1.4. Model registry..... 23

3.2. EnerML: Data pre-processing and analytics module 25

3.2.1. Data pre-processing 25

3.2.2. Model training and registration 27

3.2.3. Model inference and serving 32

3.2.4. Exploratory data analysis and statistics 33

3.3. Big Data Analytics Engine API..... 37

3.3.1. FastAPI, tools and standards..... 37

3.3.2. BDAE API interfaces 39

3.4. Security 48

3.5. Deployment..... 48

3.6. Communication with other EnerMan components..... 48

3.6.1. EnerMan Intelligent Node..... 48



3.6.2.	Industrial Management Visualization System.....	48
3.6.3.	Prediction Engine and Simulation Engine	49
3.6.4.	Sphynx Machine Learning and Analytics Platform.....	49
3.7.	Current state of the implementation and future steps	50
4.	CONCLUSION	52
5.	REFERENCES	53

TABLE OF FIGURES

Figure 1: The EnerMan Big Data Analytics Engine Architecture.	20
Figure 2: Machine learning experiment tracking in MLflow UI.	23
Figure 3: Model registry in MLflow UI.	24
Figure 4: MLflow flavours and serving integrations.	24
Figure 5: MLflow architecture with remote Tracking Server, backend and artifact stores.	25
Figure 6: The EnerML Data Pre-processing module.	26
Figure 7: The pre-processing section of the configuration file defining the pre-processing settings. .	27
Figure 8: The EnerML model training and registration module.	28
Figure 9: An example of a complete configuration file for the EnerML training module.	28
Figure 10: Skeletons of the BaseModel and the Anomaly Detection class that inherits from it.	30
Figure 11: The EnerML model inference and serving.	32
Figure 12: EDA overview. The data used in this example are drawn from the EnerMan YIOTIS pilot.	34
Figure 13: EDA interactions and correlations profiling. The data used in this example are drawn from the EnerMan YIOTIS pilot.	35
Figure 14: EDA missing values and sample section. The data used in this example are drawn from the EnerMan YIOTIS pilot.	36
Figure 15: Timeseries statistics. The data used in this example are drawn from the EnerMan YIOTIS pilot.	37
Figure 16: User interface for the YIOTIS pilot BDAE API. Six main categories of endpoints are provided.	40
Figure 17: User interface for the CRF pilot BDAE API. Endpoint to request a data set from the TSDB.	41
Figure 18: User interface for the CRF pilot BDAE API. The schema of the response defined with Pydantic models.	41
Figure 19: User interface for the CRF pilot BDAE API. Example value for a response based on predefined Pydantic models.	42
Figure 20: User interface for the YIOTIS pilot BDAE API. Swagger UI response includes a Curl request, the Request URL and the response body.	42
Figure 21: User interface for the YIOTIS pilot BDAE API. Pandas-profiling EDA endpoint.	43
Figure 22: User interface for the YIOTIS pilot BDAE API. Timeseries statistical analyses offered in the EDA section.	43
Figure 23: User interface for the CRF pilot BDAE API. A GET endpoint to retrieve a pilot's Data Model JSON file from the NoSQL storage.	44
Figure 24: User interface for the YIOTIS pilot BDAE API. Model training endpoint.	45
Figure 25: User interface for the YIOTIS pilot BDAE API. An endpoint in the Predictions section for the anomaly detection machine learning task.	45
Figure 26: User interface for the YIOTIS pilot BDAE API. A response with predictions for the anomaly detection task on batch data.	46
Figure 27: User interface for the YIOTIS pilot BDAE API. Feature importance estimation to detect influential variables.	47
Figure 28: User interface for the YIOTIS pilot BDAE API. Code-generated API documentation.	47
Figure 29: Integration with the EnerMan System Analysis and Prediction.	51

LIST OF TABLES

Table 1: Training configuration file structure. 29
 Table 2: Machine learning models and algorithms that can be supported by the BDAE. 30

ABBREVIATIONS TABLE

AI	artificial analysis
API	application programming interface
ASGI	asynchronous server gateway interface
BDAE	big data analytics engine
COF	connectivity outlier factor
CPU	central processing unit
CRUD	create, read, update and delete
CSV	comma-separated values
DBSCAN	density based clustering
EDA	exploratory data analysis
GBM	gradient boosting model
GMM	gaussian mixture model
GPU	graphics processing unit
GUI	graphical user interface
HTML	hypertext markup language
IETF	internet engineering taskforce
JSON	JavaScript Object Notation
KNN	k-nearest neighbors
LOF	local outlier factor
MAD	median absolute deviation
ML	machine learning
MLP	multilayer perceptron
NaN	not a number
ORM	object relational mapping
PCA	principle component analysis
SFTP	secure file transfer protocol
SOTA	state-of-the-art
SQL	structured query language
SSH	secure shell protocol
SVM	support vector machines
TFM	temporal fusion transformer
TSDB	time series data base
URL	uniform resource locator
UTC	universal time coordinated
YAML	yet another markup language

EXECUTIVE SUMMARY

Big data applications are steadily making their way in industrial enterprises, promising to transform their core processes, introduce new business models and create value. Organizations that embrace such technologies early on will benefit and maintain a competitive edge. However, the successful implementation of such applications can be a highly complex and daunting task. Big data are in most cases unstructured, imperfect, and challenging to manage in terms of infrastructure due the high volumes or velocity of their generation and accumulation. Additionally, efficient predictive algorithms can be time-consuming and capital-intensive to build, test, deploy and maintain, and most manufacturers lack such in-house expertise.

This deliverable presents the EnerMan Big Data Analytics Engine (BDAE), an all-encompassing solution for data management, processing and analytics that aims to abstract users from the low-level complexities of these functionalities.

In the first part of this deliverable, we provide an extended literature review on state-of-the-art methods for data pre-processing-processing. Pre-processing-processing is a critical operation in big data applications because data quality has a direct impact on the performance of the predictive algorithms. The review proceeds with a thorough presentation of state-of-the-art machine learning algorithms and analytics, focusing on the methods that can be leveraged to facilitate the development and operation of energy sustainable industrial systems, e.g., regression models, anomaly-detection, and clustering.

In the second part, we provide an analytical overview of the Big Data Analytics Engine in terms of architecture, infrastructure, and implementation. The infrastructure comprises of several data storages to accommodate not only the extensive requirements for data management, but also the demands for efficient organization of the metadata used from the analytics operations, as well as the products of these operations which are the serialized predictive models. The section also presents the core pre-processing and analytics module, EnerML, which was designed and implemented based on the state-of-the-art methods presented in the first part to provide training and inference services. The last part of the implementation is the Big Data Analytics Engine Application Programming Interface (API), an interface that can serve all the above offerings in an efficient and user-friendly way. The second part concludes by highlighting aspects such as security and deployment, the connection with other EnerMan components, the current state of the implementation and the future steps.

1. INTRODUCTION

Big data applications have drawn the interest of many fields where data management and processing are of critical importance. For example, the governmental sector [3] and supply-chain management systems are utilizing big data techniques and benefit from the positive outcomes[4]. Interestingly, big data applications are also adopted in the medical sector, as they provide solutions for data storage bottlenecks and enhance heuristic methods in search of efficient treatment processes for patients [5]. Most importantly, big data are used in the industrial sector, considering Industry 4.0, where data from IoT sources are managed and analysed in an effort to make the industrial processes more efficient, cost-effective and faster [6].

However, along with the opportunities arising from operationalizing big data, there are significant challenges that need to be addressed. Such challenges are related predominantly to the heterogeneity of the data [7], the various sampling frequencies [8], and the different formats [9]. Additionally, some data could be not only unsynchronized [10] but also structured or unstructured [11], which leads to different database storage requirements and handling/pre-processing of the data. All these parameters should be taken into consideration when implementing a big data analytics mechanism.

Early in the life of the project and through constant communication with the end-users, the EnerMan consortium aspired to understand exactly what techniques and methodologies would be required in order to transform the collected data into a usable cohesive dataset that would be suitable to extrapolate a conclusion on the relevant operations and to try improving the processes.

The consortium in the context of Task 3.1, followed comprehensive state-of-the-art research on current techniques for big data pre-processing and analytics in order to set the stage and be able to select optimal techniques to cover the objectives of the project. With that in mind, this document presents the techniques that were researched and taken into consideration.

In the EnerMan project, sensor measurements from in-field deployed sensors in the manufacturing space are collected and, after the first layer of harmonization, are stored in the Big Data Analytics Engine. These measurements are combined with historical data and data from external sources to create a heterogeneous data set suitable for both simulation of the manufacturing space or/and processes. Furthermore, the aforementioned data set is suitable for future predictions on aspects such as energy consumption to allow the user a holistic view spanning from past to current and finally future. Finally, this data set and accompanied extracted information (e.g., anomalies) can also be presented to the user.

The EnerMan Big Data Analytics Engine has been designed and developed with the aim to provide an all-encompassing solution to manage the EnerMan data flows and support all the stages of the machine learning lifecycle, from data pre-processing to model training and serving, to enable the end-users to create value from their data.

2. STATE-OF-THE-ART

In this section, a SOTA analysis will be presented showcasing the data pre-processing and data analytic techniques that are being widely used, extracting valuable information for supporting and enriching the techniques and methods that were developed in the EnerMan project.

Starting from the former, it constitutes an important first step toward the efficient use of data and the proper application of machine learning mechanisms. Data pre-processing is used to transform the raw data into an adapted or relevant format for the intended exploitation of the data.

On the other hand, data analytics aims at helping in making sense of data. This can be achieved by analysing raw data for trends or other insights. The data analysis can be descriptive, diagnostic, prescriptive, and predictive [1] [2]. Thus, it is also an important step for extracting information from raw data that can be of great value for upcoming post-processing procedures. In the following sections, an overview of the methods related to data pre-processing and analytics are presented.

2.1. Data pre-processing

Data pre-processing is an essential step when it comes to the utilization of data coming from various sources. The data pre-processing is required to take place before consuming the data for the actual processing/analysis [12]. This technique involves the transformation of raw data, originating from different sources, into an understandable, simplified, and uniform format. Data pre-processing could be applied particularly to situations where raw data are not readily operational, e.g., data mining [13] [14], and in machine learning or other science tasks which cannot take place unless the data are processed in a suitable way before application. Once the data are gone through this pre-processing preparation and the quality of the resulted dataset is validated [15], the machine learning methods can be applied [16]. This validation can be achieved by utilizing neural network models [17]. During the data pre-processing a check for missing values, noisy data, and other inconsistencies is carried out [18]. In this way, the datasets are cleaned and formatted properly before being used in the machine learning model [19].

2.1.1. Data cleaning

Missing Values

Missing values or empty records in a dataset (also known as blanks or NaN) limit the proper functionality and application of machine learning algorithms, as valuable information or patterns might be lost. The simplest way to tackle the problem is by completely removing the data points or the features that contain missing values. This method ensures that the information remaining in the data is not contaminated with null values, but it can also result in significant loss of useful information. Another way of dealing with missing values is the imputation of these specific cells. In particular, numeric and categorical imputation parameters can be used in this regard [12] [20]. Common imputation methods are replacement with the column's mean value for numeric features or a constant for categorical features.

Data Types

The features in a dataset can be associated with different data types, for example numeric, categorical, or Datetime. It is important that data types are correct since several processes highly rely on the data types of the features. For example, missing values in a dataset can be imputed incorrectly. These features can be overwritten or ignored during the model training [12].

Outliers

An outlier in a set of data could be any value that is considered abnormal compared to the rest of the values in that dataset. In some cases, the existence of outliers is an indication of bad/incorrect data that originate from cascaded errors and these data must be removed from analysis to come. In other cases, outliers can indicate valid deviations in the dataset (e.g., in scientific measurements) that mean a peculiar behaviour/trend. Before training any machine learning model, the outliers in the dataset need to be identified and removed if needed. The identification can be achieved via the PCA linear dimensionality reduction using the Singular Value Decomposition technique. The proportion of outliers can be controlled by using a threshold value [21]. Another method for identifying outliers is the Mean and Standard Deviation Method, where the mean and standard deviation of the residual values are calculated and compared. The outlier in this case identified by observing how many standard deviations away a value is from the mean value (the number of standard deviations can be controlled using a threshold value). The Median and Median Absolute Deviation Method is another widely used method for outlier identification that involves the calculation of the median of the residuals. Then, the absolute value is calculated as the difference between each historical value and the median. Thus, a new median is formed which is then multiplied by an empirically derived constant yielding the median absolute deviation (MAD). As a result, an outlier is identified in the case of a value being a specific number of MAD away from the median of the residuals. A threshold is again used to set the minimum number of MAD [22].

2.1.2. Data encoding

One-Hot Encoding

Features with categorical type in a dataset contain ordinal or nominal label values instead of continuous numbers, which make the machine learning algorithms incapable of directly dealing with such features and thus a transformation into numeric values is required before the model training. One-Hot encoding (dummy encoding) is considered the most common type of categorical encoding. This involves the transformation of each categorical level into a separate feature in the dataset (with binary values 0 or 1). One-Hot encoding could be ideal for features with nominal categorical data (which cannot be ordered). This can be considered a mandatory step for ML application.

Ordinal Encoding

In the case of ordinal data (with intrinsic levels), Ordinal encoding is applicable. In this way, a dictionary is accepted, containing the feature names and the levels (in ascending order).

2.1.3. Data scaling and transformation

Normalization

As part of the data preparation that needs to take place before applying a machine learning model, the normalization can turn out to be essential. This technique involves the rescaling of the values of numeric columns in the dataset while avoiding (a) any distorting differences in the ranges of values and (b) any loss of information [12]. The most popular method of normalization includes the scaling and translation of each feature individually within the range 0 – 1, the scaling and translation of each feature individually with a maximal absolute value of each feature equal to 1.0. This method does not shift/centre any data and thus the sparsity is maintained. Another method is the scaling and translation of each feature individually according to the Interquartile range.

Feature Transform

Feature transformation is considered a more radical technique than the normalization methods described above. Specifically, the transformation of the features changes the shape of the distribution in a way that the transformed data can be represented by a normal or approximate normal distribution

[23]. The feature transformation may include the application of a power transformer that makes the data more normal/Gaussian-like. This particular method is beneficial for addressing modelling issues relevant to heteroscedasticity or situations where normality is required. Another method is the quantile transformation [24]. This is a non-linear transformation that may distort the linear correlations between variables measured at the same scale.

Target Transform

This kind of transformation is similar to the feature transformation mentioned above, however, following this method will change the shape of the distribution of the target variable instead of the feature. This method is applied separately from the feature transformations. One of the characteristics of this method is that it requires input data to be strictly positive when supporting both positive and negative data. In the case where the variable contains negative values, the method is internally forced to avoid any exceptions [12].

2.1.4. Data resampling

Data resampling technique refers to the repeated drawing of samples from a dataset and the utilization of these samples to carry out a statistical analysis or to address specific problems in a predictive model (e.g., imbalanced data). The four main resampling methods are the randomization, Monte Carlo, bootstrap, and jack-knife [25]. When referring to sampling or resampling, the sampling rates are also an important parameter to take into consideration.

Up sampling

Up sampling is a technique of increasing the data sampling rates by inserting zero-valued samples between the actual samples. This is also known as interpolation and it can increase the resolution, enhances anti-aliasing and reduces noise [26].

Down sampling

On the contrary, down sampling refers to the reduction of the sampling rates by removing samples from the original set of data. It is important, however, that the length of the signal must be maintained. This technique is similar to compression of data, as the data are compressed in lower bandwidths and sample rates [27].

2.1.5. Feature engineering

Feature engineering is known as the process of transforming raw sets of data into features, aiming at utilizing them later on in predictive models based on machine learning on any form of statistical modelling. Among the wide list to techniques, in ENERMAN project we consider methods such the polynomial features, the group features and the creation of clusters.

Polynomial Features

When it comes to the dependent and independent variables in machine learning applications, the relationship between the two is not always linear. The creation of new polynomial features can handle this complexity by capturing that non-linearity in the first place [16]. This could be implemented by creating the features based on all polynomial combinations existing within the numeric features in a dataset, and up to a pre-defined degree. Additionally, the degree of polynomial features must be defined, e.g., in the case of a two-dimensional input of the form of $[a, b]$, the degree equals 2 and can be expressed as $[1, a, b, a^2, b, ab, b^2]$. Lastly, a sparse matrix of polynomial and trigonometric features needs to be compressed. The remaining features are left behind.

Group Features

Considering a dataset with related features (piece of data recorded at fixed time intervals), new statistical features for such a feature group can be generated. The new features can be characterized

by the following types: mean, median, variance, and standard deviation [28]. The methods for feature grouping include features in a dataset with similar characteristics that are used for statistical feature creation. In the case of related numeric features, the column names' list is passed under the new group in order to extract statistical information. Then, the name of the new group can be passed into a parameter that holds the names of the groups.

Create Clusters

The existing features from a dataset can be used with clustering analysis to craft new features. Clustering is an unsupervised learning technique which is used on its own to offer insights to the data (more details in section 2.2.2 Unsupervised learning – Clustering), but it can also be used to generate values for a new feature. This is achieved by using the cluster label assigned to each datapoint as a value for a new categorical feature. [29].

2.1.6. Feature selection

In the process of developing a predictive model, it is desirable to have as few variables as possible to deal with, as this can significantly reduce the computational requirements and elapsed time. Feature selection process facilitates exactly that by reducing the input variables. A few measures for feature selection might be the Bayesian error rate, the Laplacian score and the Fisher score. Additionally, supervised feature selection is also used [30]. In the context of ENERMAN, we consider the Principal Component Analysis (PCA) which is an unsupervised machine learning technique used for data dimensionality reduction [21]. This can be achieved by compressing the feature space by (a) identifying the subspace that captures most of the information and (b) projecting the original feature space into lower dimensionality [31]. It is possible to decompose all datasets efficiently using the linear PCA technique. The PCA, however, can cause information loss [32]. Additionally, the linear PCA involves the reduction using Singular Value Decomposition. Kernel and incremental methods are also available [33]. Finally, the number of components needed must be defined.

2.2. Data analytics

In its broad sense, data analytics is defined as the process of analysing raw data to derive conclusion about the given information. It utilizes a variety of techniques and processes that have been automated into mechanical processes and algorithms that work over the raw data for human consumption and understanding. As is the case of EnerMan, a successful data analytics initiative will provide a clear picture of where we are (present/live data), where have we been (historical data) and a project of where we are going or where we should go (prediction models).

2.2.1. Descriptive data analytics, exploratory data analysis and machine learning

Among the tools for the description and exploration of energy consumption data, the recent statistical literature on functional data analysis provides methods for the analysis of complex data such as images and curves, which more and more often arise in the modern Industry 4.0 framework. Statistical models can be built on this type of data, with the aim to forecast the energy consumption daily profiles or to build control chart to detect unusual conditions in the manufacturing process when special causes of variation act on the energy consumption. Before fitting models, descriptive data analytics of functional data can be performed by calculating mean functions, correlation surfaces, and visualization tools such as spaghetti plot of the individual functional data are useful to help the statistical analysis.

Machine learning is a branch of AI and computer science that its well established and widely used nowadays. It focuses on the use of data and algorithms to imitate the same way a human would learn and gradually adapt leading to a gradually improved accuracy. It is an important aspect of data science

which through statistical methods, trains algorithms in order to make classifications or predictions uncovering and locating key insights on the given data set.

2.2.2. Unsupervised learning – Clustering

A subcategory of machine learning is unsupervised learning and is used to analyse and cluster unlabelled datasets. These algorithms are designed to discover hidden patterns or data groupings within the data set without the need of human intervention. The ability to discover similarities or differences in information make it the ideal solution for exploratory analysis which is the main use of the EnerMan project. Clustering and anomaly detection are of the most widely used unsupervised learning methods.

Clustering can be defined as the process of analysing a data set and then separating the data points into groups that share common traits and assign them into clusters. It is one of the most common and popular data science techniques and can be further divided into 2 subgroups, firstly, the hard clustering where each data point either belongs to a cluster or it does not, and secondly, the soft clustering where the data points are assigned a probability to belong in a number of clusters.

Following we will provide examples of unsupervised learning clustering methods that were deemed the most relevant to EnerMan.

k-means clustering

This is a common example of a hard clustering method where data points are assigned into K groups with K representing the number of clusters based on the distance of each group's centroid. [34] The data points closest to a given centroid will be clustered under the same category. A larger K value will be indicative of smaller groupings with more granularity whereas a smaller K value will have larger groupings and less granularity.

Meanshift clustering

Another popular clustering method for unsupervised learning is meanshift clustering. It is widely used in real-world data analysis with its main advantage being that it is nonparametric and so it does not require a predefined shape of the clusters in the feature space [35]. Simply speaking, "mean shift" is equal to "shifting to the mean" in an iterative way. In the algorithm, every data point is shifting to the "regional mean" step by step and the location of the final destination of each point represents the cluster it belongs to. In order to locate the mean point of the entire data set we calculate the mean point of the features included in the data set. For example, if a data set has 2 features the mean point would be calculated by the arithmetic mean of feature 1 and feature 2.

Hierarchical clustering

Hierarchical clustering methods are classified into divisive (top-down) and agglomerative (bottom-up), depending on whether the hierarchical decomposition is formed in a bottom-up or top-down fashion [36]. An agglomerative clustering starts with a singleton (one object) cluster and then successively merges pairs of clusters until all clusters have been merged into one big cluster containing all objects. Divisive clustering is a reverse approach of agglomerative clustering; it starts with one cluster of the data and then partitions the appropriate cluster. Although hierarchical clustering is easy to implement and applicable to any attribute type, they are very sensitive to outliers and do not work with missing data. Moreover, initial seeds have a strong impact on the results (involving lots of arbitrary decisions).

Density-based spatial clustering

Density-based clustering (DBSCAN) is the method of identifying distinctive groups or clusters in a dataset relied on the notion that a cluster is a dense contiguous region in the total data space, which is separated from other clusters by adjacent areas of relatively lower data density. The data points having a comparatively lower object density in the separating regions are typically labelled as noise or outliers. DBSCAN [37] is deemed as one of the most powerful and most cited density-based clustering algorithms which can identify with significant accuracy the clusters of random shape and size in large databases corrupted with noise. One of the main advantages of the DBSCAN algorithm is that predetermination of the number of clusters is not required on datasets. As the DBSCAN algorithm can handle the noise points correctly and effectively, it is more applicable to find a group surrounded by noise as well as different other group [38].

Gaussian mixture model clustering

Gaussian mixture models are a probabilistic model for representing normally distributed subpopulations within an overall population. Mixture models in general don't require knowing which subpopulation a data point belongs to, allowing the model to learn the subpopulations automatically. Since subpopulation assignment is not known, this constitutes a form of unsupervised learning. GMMs have been used for feature extraction from data sets and have also been used extensively in object tracking of multiple objects, where the number of mixture components and their means predict object locations. Gaussian mixture model is parameterized by two types of values, the mixture **component weights**, and the component **means** and **variances/covariances**. If the component weights aren't learned, they can be viewed as an a-priori distribution over components. If they are instead learned, they are the a-posteriori estimates of the component probabilities given the data [39].

2.2.3. Unsupervised learning – Anomaly detection

Anomaly detection in machine learning can be defined as identifying data points withing a data set that do not fit normal patterns or behaviour. It can be used in many cases such as identifying outliers that can have adverse effects on the normal operation of a system or to exclude instances that might indicate abnormal behaviour uncharacteristic of the system in question. The following common anomaly detection methods, automate detection and moreover make it more effective especially considering very large data sets as the one expected to be analysed in EnerMan.

The rest of this section focuses on some of the most widely used methods for the unsupervised anomaly detection task.

Clustering based local outlier

Clustering-based outlier detection methods, in general, make the assumption that data objects belong to large and dense clusters. On the other hand, outliers belong to small or sparse clusters, or they do not belong to clusters all together meaning that outliers are identified by the relationship between objects and clusters (whether they belong to a cluster or not). If an object does not belong to any cluster, then its identified as an outlier. Additionally, if there is a large distance between the object and the cluster to which it is closest, then it is an outlier. Finally, if the object part of a small or sparse cluster, then all the objects in that cluster are outliers.

Local outlier factor

In anomaly detection, the local outlier factor (LOF) is an algorithm proposed by Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng and Jörg Sander in 2000 for finding anomalous data points by measuring the local deviation of a given data point with respect to its neighbours [40]. The local outlier

factor is based on the idea of a local density, where locality is given by k nearest neighbours, whose distance is used to estimate the density. By comparing the local density of an object to the local densities of its neighbours, one can identify regions of similar density, and points that have a substantially lower density than their neighbours. The later points are considered to be outliers. The local density is estimated by the typical distance at which a point can be "reached" from its neighbours. The definition of "reachability distance" used in LOF is an additional measure to produce more stable results within clusters [41].

Connectivity based outlier factor

Identifying abnormality in any industrial data set is considered as one of the major challenges for data scientists [40]. One of the many techniques for outlier identification is connectivity-based outlier factor and it is an improved version of LOF (local outlier factor) technique. The COF algorithm is similar to LOF but consider the density estimation on a different way. COF estimates the local density of the neighbourhood with an approached called the chaining distance. It assumes the data points to follow a linear distribution and so the chaining distances are the minimum of the total sum of the distances linking all neighbours. The above makes this method suitable and with good accuracy when used for data sets that display linear correlation.

K-nearest neighbours detector

kNN is a supervised ML algorithm frequently used for classification problems (sometimes regression problems as well) in data science [42]. It is one of the simplest yet widely used algorithms with good use cases such as building recommender systems, face detection applications etc. This is a non-parametric algorithm in which the relation of data point to a group is determined by the set of the k nearest neighbours, commonly by calculating the Euclidean distance in an n -dimensional feature space. The data point under evaluation is classified as part of the group with the most common neighbours among its k nearest ones. kNN is a supervised ML algorithm but when used for anomaly detection it takes an unsupervised approach. There is no actual learning involved in the process of this algorithm or labelling of outliers or non-outliers in the dataset rather just identifying outliers based on threshold values. Data scientists using the algorithm, decide the cut-off values and thus allowing the algorithm to identify outliers.

One class SVM detector

One class SVMs attempt to learn the boundary that achieves the maximum separation between the points and the origin [43]. A one class SVM uses an implicit transformation function $\Phi(\cdot)$ Defined by the kernel to project the data into a higher dimensional space. The algorithm then learns the decision boundary that separates the majority of the data from the origin and only a small fraction of data points are allowed to lie on the other side of the decision boundary and these are considered outliers.

2.2.4. Supervised learning - Models

Supervised machine learning is the machine learning approach that translates an input to an output based on known input-output training pairs. It uses labelled training data and a series of training examples to infer a function. Each method in supervised learning is made up of an input object (usually a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm examines the training data and generates an inferred function that can be applied to fresh cases.

Supervised ML models

A supervised ML algorithm will be able to accurately determine the class labels for unseen cases in the best-case scenario. Such models include:

Linear Models

Linear models tend to be the simplest class of algorithms, and work by generating a line of best fit for the training data. They're not always as accurate as newer algorithm classes, but are still used quite a bit, mostly because they're fast to train and straightforward to interpret.

Decision trees

Decision Trees [44] are a type of Supervised Machine Learning based on the splitting of the training dataset according to a specific parameter in a recursive way. This way a tree graph is generated having nodes as the points where the dataset was split (decision nodes) and leaves are the decisions taken in the node split. A great advantage for the Decision tree models is that they offer great explainability as the model's results are easy to visualise and can be understood without requiring statistical knowledge.

Random Forest

Random Forest [45] is an algorithm that can be used for classification and regression tasks. It combines the results of many different decision trees to make the best possible decisions. The learning algorithm is one of the methods of supervised learning and can be used in machine learning. The procedure is relatively simple and offers short training times. It is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. The Random Forest algorithm provides rules on how to generate the many different decision trees and then combines them using a special ensemble method to achieve an overall result. Which properties and decision criteria the individual decision trees use to reach their results is based on a random principle and differs from decision tree to decision tree. A "forest" of random, slightly different decision trees is created. As a small independent model, each decision tree contributes to the overall decision. The random variance of the trees increases the result and prediction accuracy of the random forest algorithm. The way in which the decision trees are to be constructed and how the random forest is trained is variable. For example, the structure or the maximum depth of the decision trees can be selected.

Extra trees

Extra Trees [46] is an ensemble machine learning algorithm that combines the predictions from many decision trees. It is related to the random forest algorithm, with some differences during the training phase. It consists of randomizing strongly both attribute and cut point when choosing data while splitting a tree node. In the extreme case, it builds totally randomized trees whose structures are independent of the output values of the learning sample. The strength of the randomization can be tuned to problem specifics by the appropriate choice of a parameter.

LightGBM

LightGBM [47] is a gradient boosting framework that uses tree-based learning algorithms. It is to be distributed and efficient with the following advantages: faster training speed and higher efficiency; lower memory usage; better accuracy; support of parallel, distributed, and GPU learning; capable of handling large-scale data. Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. LightGBM is called "Light" because of its computation power and giving results faster. It takes less memory to run and is able to deal with large amounts of data. Several parameters of LightGBM can be tuned to speed up, increase accuracy or deal with the produced model overfitting.

CatBoost

CatBoost [48] is based on gradient boosted decision trees. It is a new open-sourced gradient boosting library that successfully handles categorical features and outperforms existing publicly available implementations of gradient boosting in terms of quality on a set of popular publicly available datasets. The library has a GPU implementation of learning algorithm and a CPU implementation of scoring algorithm, which are significantly faster than other gradient boosting libraries on ensembles of similar sizes. During training, a set of decision trees is built consecutively. Each successive tree is built with reduced loss compared to the previous trees.

2.2.5. Supervised learning – ML model performance estimation metrics

Binary classification (f1-score)

The F1 score is a machine learning metric that combines the precision (the proportion of positive identifications that was correct) and recall (the proportion of actual positives that were identified correctly) of a classifier into a single metric by taking their harmonic mean [49].

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{F1 Score} = \frac{\text{Precision} * \text{Recal}}{\text{Precision} + \text{Recal}}$$

Multiclass classification (f1-score, micro-averaged)

By micro-averaging the f1-score, we aggregate the contributions of all classes to compute the average metric and compensate for possible class imbalance, i.e., you may have many more examples of one class than of other classes. In such a case the micro-averages F1-Score is calculated on the prediction of all classes as [50]

$$\text{micro averaged F1 Score} = \frac{\text{True Positives}}{\text{True Positives} + \frac{1}{2}(\text{False Positives} + \text{False Negatives})}$$

Regression (R²)

R-squared is a statistical metric that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. R² provided an insight for the accuracy of fit of a model and can be calculated as [51]:

$$R^2 = \frac{\text{Sum of squared regression}}{\text{sum of squares total}} = 1 - \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

where \hat{y} represents the prediction or a point on the regression line, \bar{y} represents the mean of all the values and y_i represents the actual values or the points. Whereas correlation explains the strength of the relationship between an independent and dependent variable, R-squared explains to what extent the variance of one variable explains the variance of the second variable. Therefore, if the R² value of a model is calculated as 0.50, that means that approximately half of the observed

prediction variation can be explained by the model input vectors. In simple words, the closer the value of R^2 is near to 1, the better is the model.

Balanced accuracy

Balanced accuracy is a metric we can use to assess the performance of a classification model.

It is calculated as:

$$\text{Balanced accuracy} = (\text{Sensitivity} + \text{Specificity}) \div 2$$

where:

Sensitivity: The “true positive rate” – the percentage of positive cases the model can detect.

Specificity: The “true negative rate” – the percentage of negative cases the model can detect.

This metric is particularly useful when the two classes are imbalanced – that is, one class appears much more than the other.

Explained variance

In statistics, explained variance or explained variation is a metric for the exploitability of a ML model and is calculated by the difference of the expected and predicted values of a model. Therefore, the explained variance measures the percentage of the variability of the predictions of a ML model. In other words, it is the portion of the model's total variation that can be accounted for by actually present factors in the prediction (input vector) rather than model's error variance. The complementary part of the total variation is called unexplained or residual variation.

Cross validation

Cross-validation is an approach that combines (averages) the fitness measures of prediction of an ML predictive model in order to obtain a more accurate estimate of model's performance. The initial step in cross validation includes the split of the train dataset into two sub-datasets where one is used for the initial model training (training set) and the other for the validation analysis (testing set) of the trained model towards its performance evaluation. This process is repeated with different splits on the initial datasets (i.e., using different data for training and testing each time) and the performance validations (based on each test set) are merged (averaged) to produce a more accurate estimate of the predictive performance of the model

3. BIG DATA ANALYTICS ENGINE

The main task of the Big Data Analytics Engine (BDAE) is to contribute to the creation of value from the EnerMan data. This is achieved by providing robust data management operations including data ingestion, annotation and retrieval, but also a framework to support state-of-the-art methods for data pre-processing and analytics. The latter is the EnerML module which lies in the core of the BDAE. It supports not only the state-of-the-art methods presented in the previous section, but potentially many more since it is designed to be flexible, extensible and modular. The module is presented in detail in section 3.2 EnerML: Data pre-processing and analytics module. The other prominent entities supporting the BDAE are the databases. Depending on the data structure, representation and purpose, different databases are employed to facilitate the required operations. The storage infrastructure is presented at length in the following section (3.1 Data and artifacts storage infrastructure).

From an overall point of view, the BDAE architecture is introduced in Figure 1, with a diagram that abstracts the outline of the system, its relationships with other EnerMan components, dataflows and boundaries. With respect to the relationships, the diagram only depicts the communications and data flow with the EnerMan components that have been integrated in the current state of implementation. However, more connections are expected as the integration progresses (for more details see section 3.6 Communication with other EnerMan components). In the next sections, each internal component, as well as the relationships with the external components, will be presented from several architectural viewpoints.

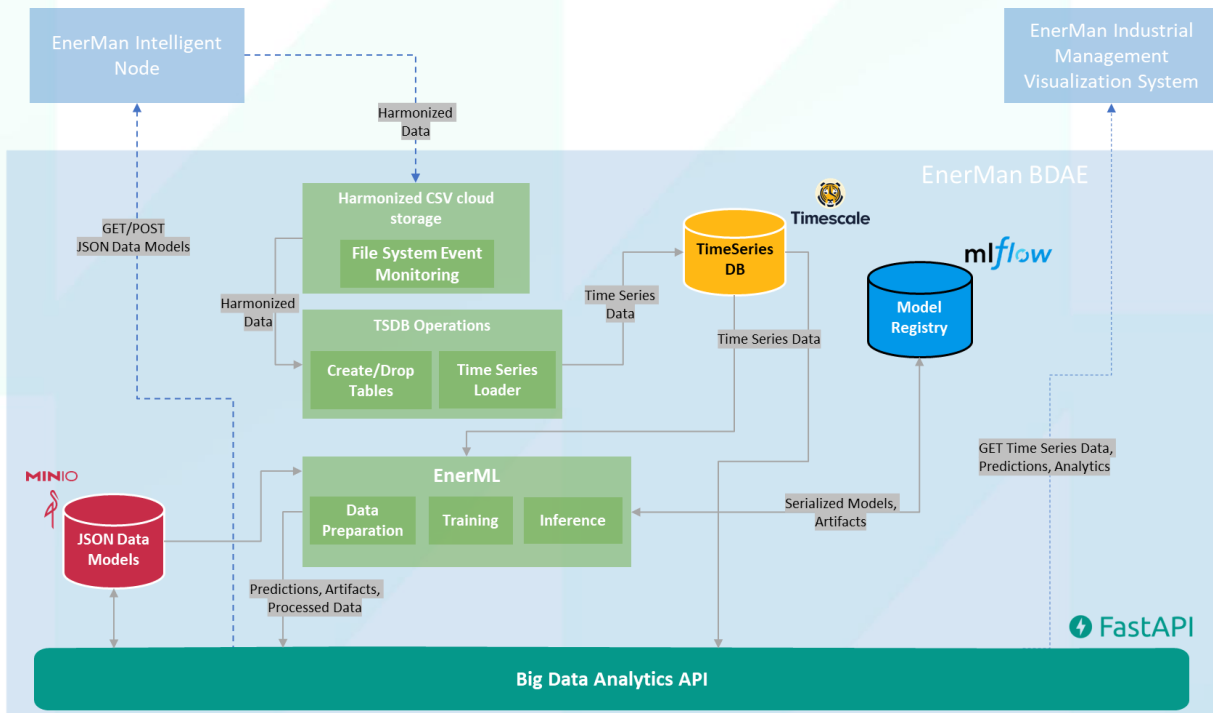


Figure 1: The EnerMan Big Data Analytics Engine Architecture.

3.1. Data and artifacts storage infrastructure

Big data applications introduce significant data management challenges which cannot be addressed with traditional tools. The EnerMan platform has to support complex and high-volume data flows and persist or retrieve data in efficient and safe ways. Some of the prominent challenges are related to the

data representation (structured, unstructured), formats, volume, velocity, and heterogeneity of the data flow. The data storage infrastructure for BDAE was designed taking into account these challenges in the context of the EnerMan solution. Our goal is to incorporate robust cutting-edge solutions which are scalable, secure, and integrate effortlessly. In the following sections, we present the main storage components involved in the BDAE architecture.

3.1.1. Time series database

Time series databases (TSDBs) are optimized to store and retrieve time series data, i.e., sets of data points associated with a timestamp. Time series data are sequential, collected over time intervals, and allow us to track changes over time, from milliseconds, to days, or even years. TSDBs are the fastest-growing category of databases today since they offer scalability and usability. Since time series can accumulate rapidly, TSDBs can offer out of the box functionalities to improve scalability, ingest rate, query latency, and data compression. In terms of usability, TSDBs offer built-in functions for flexible retention policies, time aggregations and other operations that make time series analysis easier.

EnerMan data are primarily time series, collected from multiple sensors and IoT devices. In this setting, the value of a TSDB is enormous. The BDAE cloud infrastructure includes a TimescaleDB¹, a time series database which is used to store and retrieve pilots' time-stamped data. TimescaleDB is an open-source relational database, implemented as an extension to PostgreSQL². This means that the functionality and reliability of SQL is available, while extra TSDB are introduced on top of it using abstractions such as *hypertables* and *chunks*. TimescaleDB also enables a distributed and parallelized setup, with fast queries, ingestion, and compression capacities.

Other alternatives that have been ranking high³ in the TSDBs landscape include InfluxDB⁴, Graphite⁵, Prometheus⁶ and Kdb+⁷. We have considered these solutions and compared their features and characteristics in the light of the EnerMan solution requirements. We initially excluded Kdb+ since it is not an open-source solution. Our next criterion was the partitioning features, i.e., the ability to store data on different nodes, since in the EnerMan context, data volumes are anticipated to be high, and a distributed database can be of critical importance. This requirement excluded Graphite. The final choice was decided based on the compatibility with PostgreSQL, an open-source relational database management system with a long history of reliability, extensibility, scalability, security, and SQL compliance. TimescaleDB, is developed as an extension on top of PostgreSQL, thus it is fully compatible with SQL and offers fine grained access rights. InfluxDB or Prometheus, each has its own query language, and this would potentially require more advanced training for the maintainers of the EnerMan solution. In the BDAE architecture, TimescaleDB is deployed in the cloud and used as a sink for the data ingested from the edge nodes. TSDB data are retrieved with GET requests via the BDAE API. Create, read, update, and delete (CRUD) operations are implemented mainly with *object-relational mapping* (ORM) methods in Python using the SQLAlchemy⁸ toolkit (other alternatives considered included Django and `ORM[OBJ]` and `Peewee[OBJ]`, nonetheless SQLAlchemy was preferred for

¹ TimescaleDB: <https://www.timescale.com/>

² PostgreSQL: <https://www.postgresql.org/>

³ DB-Engines Ranking of Time Series DBMS: <https://db-engines.com/en/ranking/time+series+dbms>

⁴ InfluxDB: www.influxdata.com/-products/-influxdb-overview

⁵ Graphite: <https://github.com/graphite-project/graphite-web>

⁶ Prometheus: <https://prometheus.io/>

⁷ Kdb+: <https://kx.com>

⁸ SQLAlchemy: <https://sqlalchemy.org>

its compatibility with the API framework we adopted (i.e., FastAPI⁹ section 3.3.1), and secondly using the Psycopg¹⁰, the de-facto PostgreSQL Python¹⁰ TimescaleDB powered operations also include hyperfunctions for gap filling and interpolation, time buckets extraction, resampling, and frequency analysis. Furthermore, we use two tools to directly monitor and interact with the database server: psql, a command line tool built in PostgreSQL and pgAdmin¹¹, a database administration tool.

3.1.2. NoSQL storage

A NoSQL storage is useful to store and retrieve unstructured data (non-tabular data). EnerMan data are predominately tabular but their processing at several stages requires additional information which is available in JSON format. More specifically, each type of dataset is related to a Data Model JSON file which contains all the necessary information for the processing steps. These files need to be accessible for reading and writing operations from both the EnerMan Intelligent nodes (edge nodes) and the other cloud components (e.g., the BDAE API presented later, in section 3.3). The NoSQL database incorporated in the BDAE is a MinIO¹² object storage deployed in the cloud infrastructure. MinIO is an open-source, multi-cloud object storage which offers encryption, high-performance, compatibility with AWS S3¹³ and is natively available in the Kubernetes¹⁴ orchestration platform. It offers graphical user interfaces (GUI), command-line interfaces (CLI) and application programming interfaces (API) to enable monitoring and observability. Another important feature of this solution is the bucket and object versioning.

In the context of the EnerMan platform, the edge nodes (EnerMan Intelligent Nodes) read and write in the MinIO storage (via the BDAE API) in order to retrieve metadata information and update it. Besides the Data Model files, a second bucket in the MinIO storage is also used as a backend for the model registry discussed in the following subsections.

3.1.3. Harmonized CSV cloud storage data and TSDB operations

The *data-harmonization module* (D2.1, D2.2) which is deployed at the edge nodes (EnerMan Intelligent Nodes) uses the Data Model JSON files residing in the NoSQL storage of the BDAE server to harmonize the raw data. The harmonization workflow concludes with the harmonized CSV file transfer to the cloud server. This operation uses SFTP¹⁵ (SSH File Transfer Protocol), a secure file transfer protocol which runs over the SSH protocol and is equally secure and functional. SFTP protects against password sniffing and man-in-the-middle attacks. It protects the integrity of the data using encryption and cryptographic hash functions and authenticates both the server and the user.

On the cloud side, a filesystem event monitoring service has been implemented with *Watchdog*¹⁶. The service monitors the assigned directory for any new file events. Watchdog uses an `Observer` and a `PatternMatchingEventHandler` to detect each new CSVs transferred to the cloud server. In such event, the service processes the CSV with the Python TSDB utils for creating a table (in case it does

⁹ Psycopg: <https://www.psycopg.org>

¹¹ pgAdmin: <https://www.pgadmin.org/>

¹² MinIO: <https://min.io/>

¹³ AWS S3: <https://aws.amazon.com/s3/>

¹⁴ Kubernetes: <https://kubernetes.io>

¹⁵ SFTP: <https://www.ssh.com/academy/ssh/sftp>

¹⁶ Watchdog: <https://python-watchdog.readthedocs.io>

not already exist) and append the new data in the TSDB. The processed CSV is then transferred in another cloud directory where the historical CSVs are stored. The service is running in a Docker¹⁷ container.

3.1.4. Model registry

A model registry is a repository that facilitates tracking, versioning and storing trained ML models. The data analytics offered by the BDAE are produced with multiple types of models applied on the different datasets for each pilot organization’s use cases. This setup makes tracking and versioning the trained models a critical component of the BDAE. To address these requirements, we have integrated the BDAE cloud infrastructure with an MLflow¹⁸ instance which is an open-source platform for managing the end-to-end machine learning lifecycle, using a centralized model repository, a UI and set of APIs. More specifically, the BDAE uses three MLflow functionalities: Tracking, Model Registry and Models Managing.

MLflow Tracking

Experiment tracking API (Figure 2) enables recording each separate training run along with the related tags, parameters, evaluation metrics and useful artifacts. Tracking facilitates the comparison of model configurations and performance results to select the best model. Additionally, we can attach any other artifact, e.g., configuration files, plots, pre-processing pipeline objects to each run, to retrieve them effortlessly later if the model is selected.

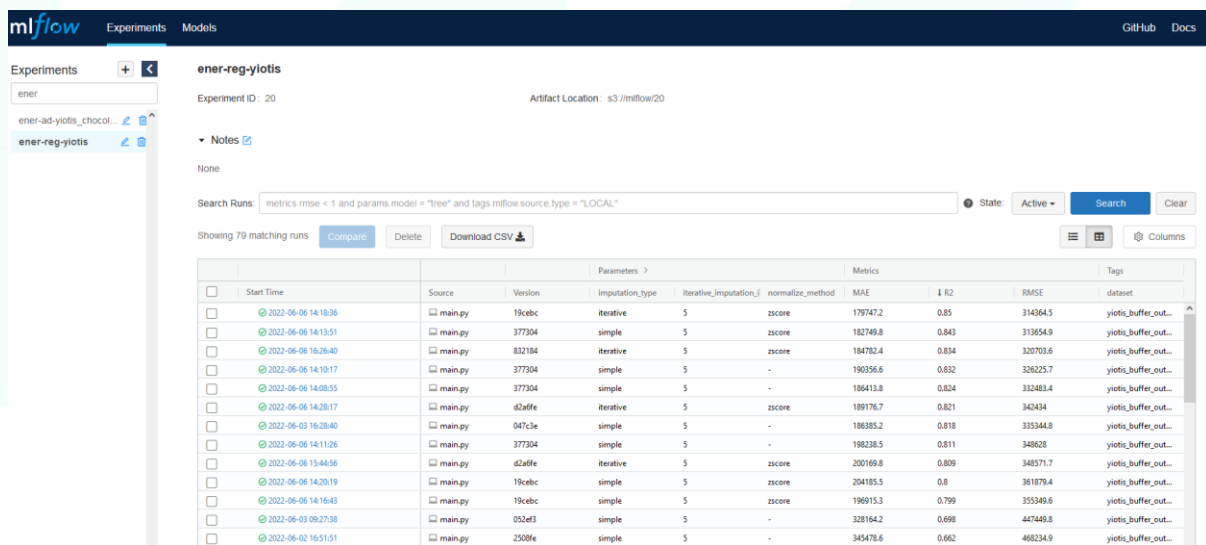


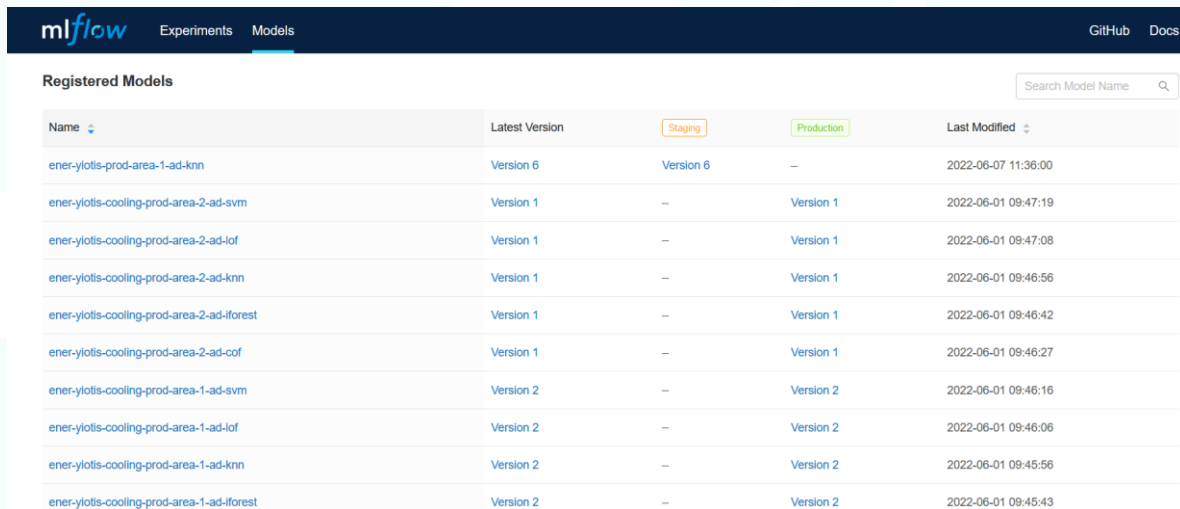
Figure 2: Machine learning experiment tracking in MLflow UI.

MLflow Model Registry

The model registry (Figure 3) provides a central model store for collaborative usage. The API offers useful functionalities for chronological model lineage (i.e., which MLflow experiment and run produced the model at a given time), model serving and versioning, as well as stage transitions (for example, from staging to production or archived).

¹⁷ Docker: <https://www.docker.com/>

¹⁸ MLflow: <https://www.mlflow.org>



Name	Latest Version	Staging	Production	Last Modified
ener-yjolts-prod-area-1-ad-knn	Version 6	Version 6	–	2022-06-07 11:36:00
ener-yjolts-cooling-prod-area-2-ad-svm	Version 1	–	Version 1	2022-06-01 09:47:19
ener-yjolts-cooling-prod-area-2-ad-lof	Version 1	–	Version 1	2022-06-01 09:47:08
ener-yjolts-cooling-prod-area-2-ad-knn	Version 1	–	Version 1	2022-06-01 09:46:56
ener-yjolts-cooling-prod-area-2-ad-forest	Version 1	–	Version 1	2022-06-01 09:46:42
ener-yjolts-cooling-prod-area-2-ad-cof	Version 1	–	Version 1	2022-06-01 09:46:27
ener-yjolts-cooling-prod-area-1-ad-svm	Version 2	–	Version 2	2022-06-01 09:46:16
ener-yjolts-cooling-prod-area-1-ad-lof	Version 2	–	Version 2	2022-06-01 09:46:06
ener-yjolts-cooling-prod-area-1-ad-knn	Version 2	–	Version 2	2022-06-01 09:45:56
ener-yjolts-cooling-prod-area-1-ad-forest	Version 2	–	Version 2	2022-06-01 09:45:43

Figure 3: Model registry in MLflow UI.

MLflow Models Managing

MLflow API for model managing (Figure 4) can be used for storing, annotating, discovering, and managing models in a central repository. Model deployment is enabled from a variety of ML libraries to a variety of model serving and inference platforms. In particular, MLflow provides deployment tools to support utilities for saving, loading, and serving in several standard flavours, e.g., scikit-learn, PyTorch, TensorFlow, Keras as well as a custom flavour using pyfunc for a generic model format for Python models¹⁹.

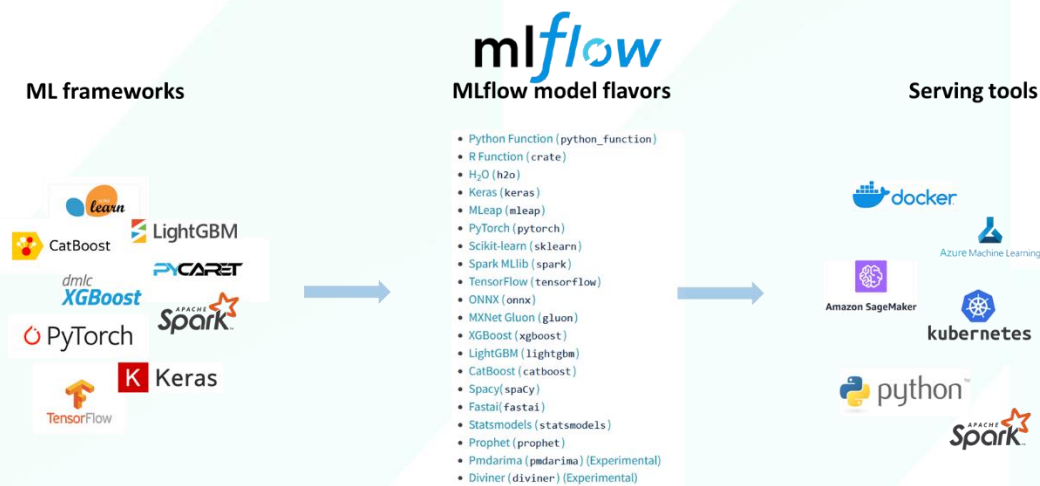


Figure 4: MLflow flavours and serving integrations.

In the BDAE context, the EnerML module (presented in 3.2) is using the MLflow experiment tracking API to record the training runs it executes offline. The trained models are also registered offline through the MLflow interface. However, the model serving is automated, and the output predictions can be requested through the BDAE API. Furthermore, the model registry can be also accessed via the BDAE API and enable users to upload their own models.

MLflow can be supported either in a local machine or in a distributed architecture where the tracking server, backend store, and artifact store reside on remote hosts. For the integration with BDAE in the

¹⁹ MLflow built-in model flavors: <https://www.mlflow.org/docs/latest/models.html#built-in-model-flavors>

cloud infrastructure we used a remote MLflow Tracking Server, a Postgres database for backend entity storage, and an S3 bucket for artifact storage (Figure 5). The S3 bucket in our case is implemented with a MinIO bucket, as fully S3 compatible solution (more information on the MinIO in subsection 3.1.1).

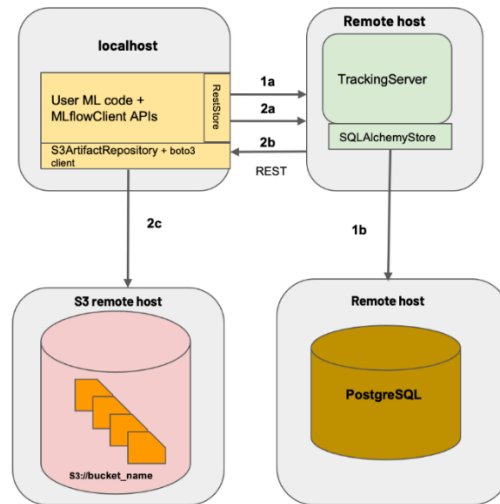


Figure 5: MLflow architecture with remote Tracking Server, backend and artifact stores²⁰.

Another open-source alternative considered for the model registry functionality was Kubeflow²¹. Both technologies provide a collaborative environment for model development, are scalable, portable, and customizable. However, Kubeflow, is essentially a container orchestration system based on Kubernetes²², while MLflow is a Python library that only solves experiment tracking and model versioning as a single service, without the extra cost and complexity of setting up and maintaining an infrastructure orchestrator.

3.2. EnerML: Data pre-processing and analytics module

Data pre-processing and analytics in the BDAE API are powered by the EnerML Python module which lies in the core of the overall architecture (Figure 1). In essence, the module integrates multiple machine learning libraries and frameworks to provide an extended inventory of pre-processing and analytics methods and algorithms. The offered functionalities are the following: data pre-processing, model training and registration, model inference and serving, and exploratory data analysis and statistics. We present each of these aspects at length in the following sections.

3.2.1. Data pre-processing

EnerML data pre-processing is essentially the second pre-processing layer applied to the EnerMan datasets. The first layer takes place directly at the edge nodes during the collection of the raw data

²⁰ Image adapted from <https://www.mlflow.org>

²¹ Kubeflow: <https://www.kubeflow.org/>

²² Kubernetes: <https://kubernetes.io>

and is called harmonization. It is implemented as a Python module (described in D2.1, Section 3.1, updated in D2.2, Section 3.2). Data harmonization at the edge ensures that the raw data can be transferred to the cloud infrastructure in a unified and standardized form with regards to the file formats, column names, timestamp format, missing data indicators and so on. On the other hand, data pre-processing in the cloud is responsible for more sophisticated processes of data preparation to increase the value of the data and make them suitable for machine learning.

The EnerML data pre-processing module provides functions for a variety of state-of-the-art methods to handle data cleaning, encoding, scaling, transformation, feature engineering and feature selection (Figure 6). Those functions are integrated from several open-source machine learning libraries built on top of the Python programming language, such as pandas²³, scikit-learn²⁴, Darts²⁵, PyCaret²⁶. Furthermore, Dask²⁷ is used to enable scaling these processes in multi-core and distributed setups.

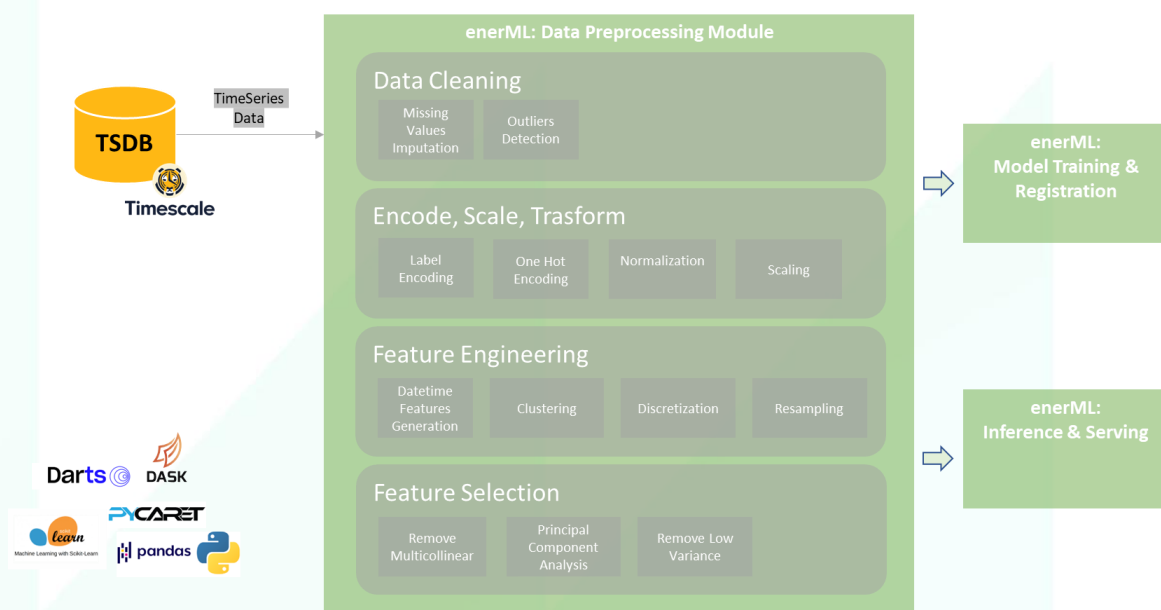


Figure 6: The EnerML Data Pre-processing module.

Data pre-processing pipelines in the cloud are guided by configuration files which are drafted for each type of machine learning and analytics model (Figure 7). Essentially, the configuration files are JSON files that handle all configurable aspects of the training pipeline besides pre-processing (see section 3.2.2). Pre-processing in the cloud mainly takes place to improve a model's performance and speedup algorithmic convergence. For example, in Figure 7, the configuration shown is defined for an unsupervised anomaly detection model, thus some supervised machine learning concepts for pre-processing are missing (e.g., what is the target variable, how to split, shuffle or stratify the data, cross validation parameters are omitted).

²³ pandas: <https://pandas.pydata.org/>

²⁴ scikit-learn: <https://scikit-learn.org/stable/>

²⁵ Darts: <https://unit8co.github.io/darts/>

²⁶ PyCaret: <https://pycaret.gitbook.io/docs/>

²⁷ Dask: <https://www.dask.org/>

```
preprocessing : {
  "normalize_method": "minmax",
  "imputation_type": "simple",
  "iterative_imputation_iters": 5,
  "categorical_imputation": "mode",
  "categorical_iterative_imputer": "lightgbm",
  "high_cardinality_method": "frequency",
  "numeric_imputation": "mean",
  "numeric_iterative_imputer": "lightgbm",
  "date_features": ["time"],
  "transformation_method": "yeo-johnson",
  "handle_unknown_categorical": True,
  "unknown_categorical_method": "least_frequent",
  "pca_method": "linear",
  "pca_components": 4,
  "ignore_low_variance": False,
  "rare_level_threshold": 0.1,
  "remove_multicollinearity": False,
  "multicollinearity_threshold": 0.9,
  "remove_perfect_collinearity": True,
}
```

Figure 7: The pre-processing section of the configuration file defining the pre-processing settings.

Some pre-processing operation can also take place using the timeseries database functions directly when we query the data. The BDAE implements some of the resampling operations using those functions; given a time window, a time bucket operation applies an aggregation function to each bucket's row to reduce or increase the sampling rate of the data. However, Python libraries offer more flexibility to achieve complicated operations, and, in many cases, we use it in a complementary way, e.g., imputation of missing values with interpolation.

Overall, although we have configured the pre-processing pipelines taking into account the datasets and model specifics, we also aimed to provide a more generic and configurable pre-processing module for BDAE. In fact, the pre-processing functionality is available either integrated in the model training functionality or as a separate process, to enable other EnerMan modelling activities to readily use the pre-processed data.

3.2.2. Model training and registration

The EnerML training module (Figure 8) facilitates the offline training of the models offered by BDAE to provide predictions, analytics, and data insights. The module is comprised of a dedicated class for each type of machine learning operation, e.g., clustering, anomaly detection, regression. The design aimed to ensure easy configuration from a centralized JSON file, as well as modularity and extensibility, as more algorithms might appear relevant for the use-cases as the project progresses. It uses a DataLoader class to get data from the TSDB and pull the respective data models from the NoSQL database. Several Python libraries are used for the machine learning algorithms' implementation (pandas, PyCaret, Darts, scikit-learn, statsmodels and Dask). Finally, the module logs all training relevant information, models and artifacts to the MLflow platform.

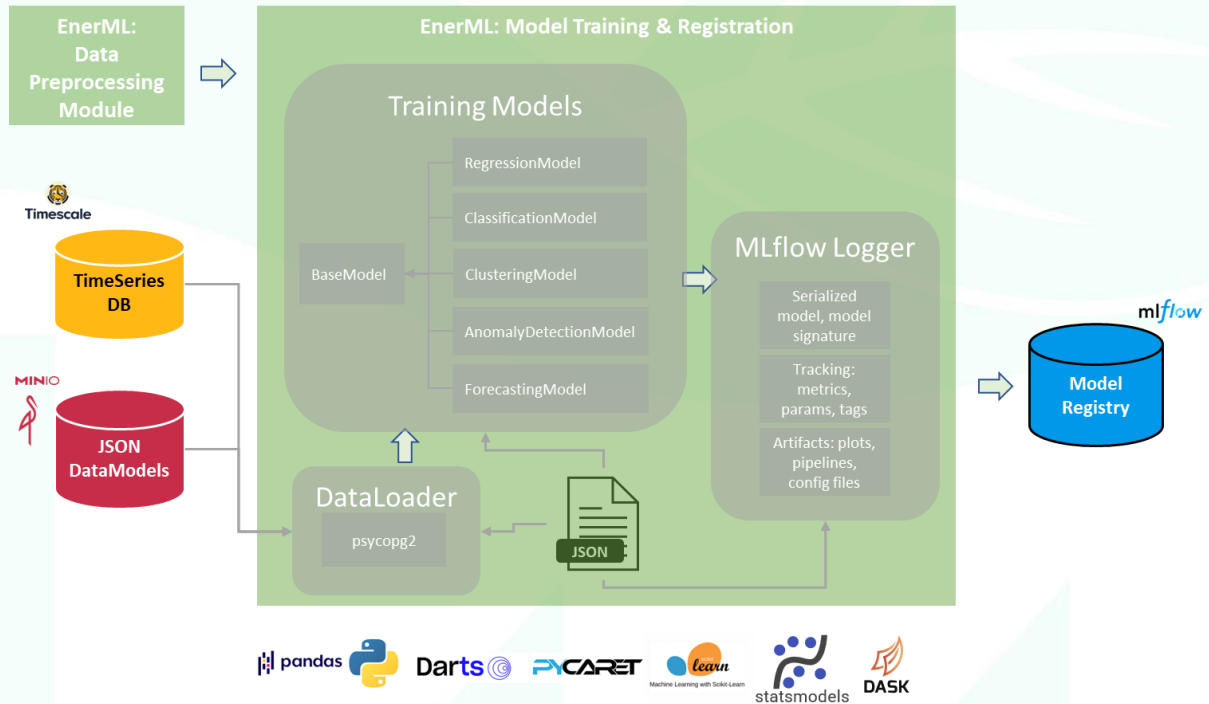


Figure 8: The EnerML model training and registration module.

The JSON configuration file defines all the necessary information for loading the data, pre-processing, training, evaluating, and saving a model. pre-processing in Figure 9, an example of such a file is presented, while Table 1 provides an overview of the generic structure of these files. In particular, it contains the information to run the appropriate database query to obtain the dataset, the model configuration (e.g., specific algorithm to be used or enabling the AutoML model selection process, whether to use hyperparameter tuning), as well as some parameters required to guide MLflow logging.

```
CFG = {
  "data": {
    "table_name": "yiotis_buffer_outdoor",
    "columns": ['time', 'outdoor_temperature', 'outdoor_humidity', 's_1_production_area_1_temperature_c',
               's_1_production_area_1_relative_humidity_prc', 's_5_l_11_reactive_energy_delivered_v_arh'],
    "target": "s_5_l_11_reactive_energy_delivered_v_arh",
    "plot_type": ['feature', 'residuals']
  },
  "preprocessing": {...},
  "model": {
    "model": "catboost",
    "complete_pipeline_prefix": "ener-yiotis-mixing-reg-",
    "tuning": True,
    "mode": "model-selection"
  },
  "mlflow_config": {
    "enabled": True,
    "experiment_name": "ener-reg-yiotis",
    "tags": {"train_data": "normal"},
  }
}
```

Figure 9: An example of a complete configuration file for the EnerML training module.

Table 1: Training configuration file structure.

Top level section	Nested subsection
Data	Table name Columns to be used in training Target variable (for supervised learning only) Plot type (specific to the model)
Pre-processing	Settings for data cleaning, encoding, scaling, transformation, feature engineering and selection (specific to the model)
Model	Algorithm id Naming convention Tuning Training mode (if True all algorithms are tested)
MLflow configuration	Enabled (when True, results are logged) Experiment name to log results Tags

Figure 10 presents the BaseModel and one of the specific machine learning classes that inherits from it, the Anomaly Detection model. The BaseModel expects the JSON configuration file to enable the model initialization. Furthermore, it declares some abstract methods which are required to be implemented in the child model class. An Mlflow connection is created immediately after the initialization and logging all the important entities is enforced with an abstract method in the BaseClass (`_write_logs`), the implementation of which handles all the logging activity to Mlflow. Model saving is implemented with another method (`save_model`) which ensures that the model will be saved along with its signature, i.e., the schema of the features used during training, as well as the pre-processing pipeline. These two artifacts are necessary to ensure that at inference time, data pre-processing will be executed with the same process, and the same data schema will be fed to the trained model.

```

class BaseModel(ABC):
    """Abstract Model class that is inherited to all models"""
    def __init__(self, cfg):
        self.config = JSONConfig.from_json(cfg)

    @abstractmethod
    def load_data(self):
        pass

    @abstractmethod
    def train(self):
        pass

    @abstractmethod
    def eval(self):
        pass

    @abstractmethod
    def save_model(self):
        pass

    @abstractmethod
    def _write_logs(self):
        pass

class ADModel(BaseModel):
    """AD model"""
    def __init__(self, config):...
    def load_data(self):...
    def _get_pipeline(self):...
    def _preprocess_data(self):...
    def _tuning(self, space):...
    def model_selection(self, space):...
    def train(self):...
    def eval(self):...
    def save_model(self):...
    def visualize_model(self):...
    def _write_logs(self):...
    
```

Figure 10: Skeletons of the BaseModel and the Anomaly Detection class that inherits from it.

Furthermore, the module uses a DataLoader class that handles all data loading processes. It is currently using a generic SQL query and a psycopg2 connection to bring the data from the TSDB, which indicates that any dataset used for training should already reside in the TSDB as a single table. This is a fact for all the original datasets provided by the pilots, however derivative tables obtained by merged tables will have to be created manually and proactively, using independent Python scripts or via pgAdmin and psql. The DataLoader also connects with the JSON DataModel NoSQL DB, to pull the data models and extract relevant information for processing the data, e.g., the original data time zone, since everything is stored in UTC in the TSDB. The DataLoader class is also used by the inference module described in the following section.

Overall, the current implementation can support the machine learning models presented in Table 2.

Table 2 Machine learning models and algorithms that can be supported by the BDAE.

Machine learning operation	Algorithms
Regression	Linear Regression Lasso Regression Ridge Regression Elastic Net Lasso Least Angle Regression Bayesian Ridge Support Vector Regression K Neighbors Regressor Decision Tree Regressor Random Forest Regressor Extra Trees Regressor AdaBoost Regressor Gradient Boosting Regressor MLP Regressor

	Extreme Gradient Boosting Light Gradient Boosting Machine CatBoost Regressor
Classification	Logistic Regression K Neighbors Classifier Naive Bayes Decision Tree Classifier SVM - Linear Kernel SVM - Radial Kernel Gaussian Process Classifier MLP Classifier Ridge Classifier Random Forest Classifier Quadratic Discriminant Analysis Ada Boost Classifier Gradient Boosting Classifier Linear Discriminant Analysis Extra Trees Classifier Extreme Gradient Boosting Light Gradient Boosting Machine CatBoost Classifier
Clustering	K-Means Clustering Affinity Propagation Mean shift Clustering Spectral Clustering Agglomerative Clustering Density-Based Spatial Clustering OPTICS Clustering Birch Clustering K-Modes Clustering
Anomaly Detection	Angle-base Outlier Detection Clustering-Based Local Outlier Connectivity-Based Outlier Factor Histogram-based Outlier Detection Isolation Forest k-Nearest Neighbours Detector Local Outlier Factor One-class SVM detector Principal Component Analysis Minimum Covariance Determinant Subspace Outlier Detection Stochastic Outlier Selection
Forecasting	Baseline Models Block Recurrent Neural Networks Croston method Exponential Smoothing Fast Fourier Transform LightGBM Model Kalman Filter Forecaster Linear Regression model N-BEATS N-HiTS Facebook Prophet Random Forest Regression ensemble model Regression Model

Recurrent Neural Networks
 StatsForecastAutoARIMA
 BATS and TBATS
 Temporal Convolutional Network
 Temporal Fusion Transformer (TFT)
 Theta Method
 Transformer Model
 VARIMA

3.2.3. Model inference and serving

The EnerML inference module (Figure 11) contains the Inferer class with methods to accommodate the model serving utilities. It uses the same DataLoader class with the training module, and its main functionality is to fetch a trained model from the Model Registry to produce predictions which are forwarded to the BDAE API. The whole process is triggered with a GET request sent by the BDAE API, defining the table with the data to be used for prediction and the ML operation/algorithm as a query parameter. Then, the inference workflow begins with loading the data from the TSDB using the JSON data model for localization instructions. Subsequently, the Inferer fetches the appropriate model from the MLflow Model Registry and provides the model signature (schema of the training data) to the pre-processing method which is responsible to match the signature with the prediction data and apply the related pre-processing pipeline. The next step is to call the prediction method passing the preprocessed data and return the predictions to the BDAE API. Additional optional functionalities include methods to fetch and return other artifacts from MLflow, e.g., plots created at training time, a feature importance estimation (currently for regression models), append the predictions with explainable datetime features, e.g., day_of_week. Finally, the Inferer can provide the pre-processing service independently, which can be useful if a user only wants the preprocessed dataset to use it with their own models.

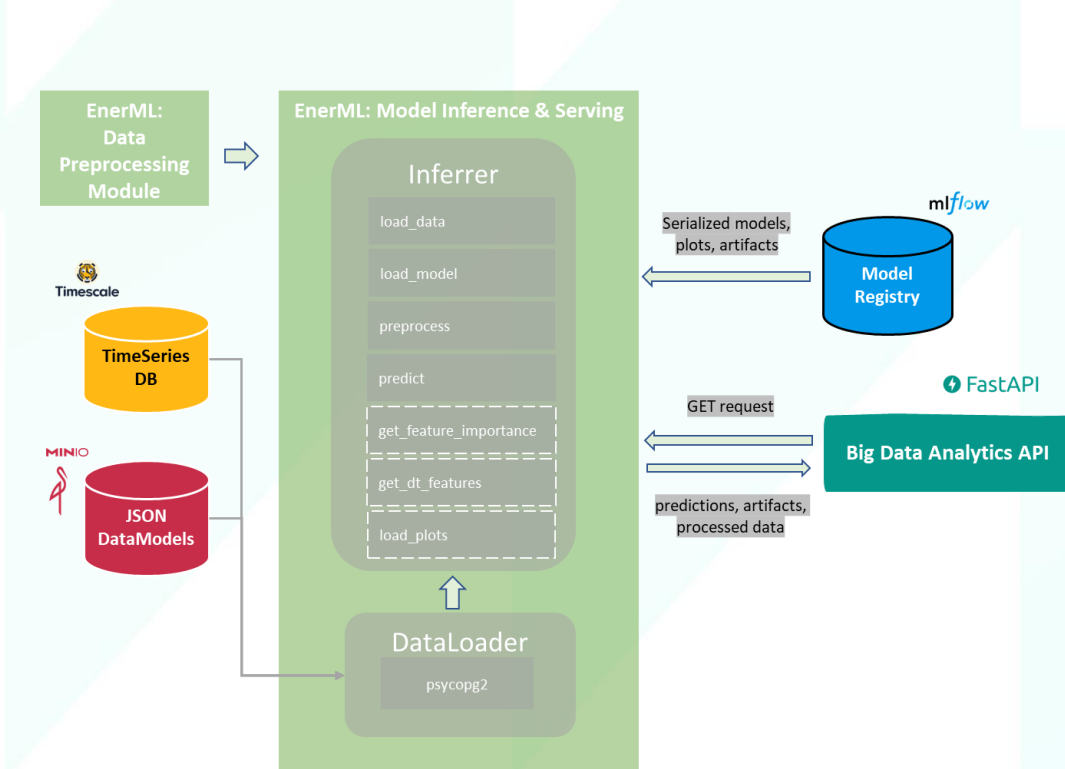


Figure 11: The EnerML model inference and serving.

3.2.4. Exploratory data analysis and statistics

Besides the machine learning predictions, other data analytics are offered by BDAE. Exploratory data analysis (EDA) is provided as an overview data profile to inform the user on a dataset's characteristics. This is particularly helpful to find suitable machine learning strategies regarding pre-processing and training. The timeseries statistical analysis is another form of EDA but more focused on discovering patterns and characteristics related to the temporal nature of the data.

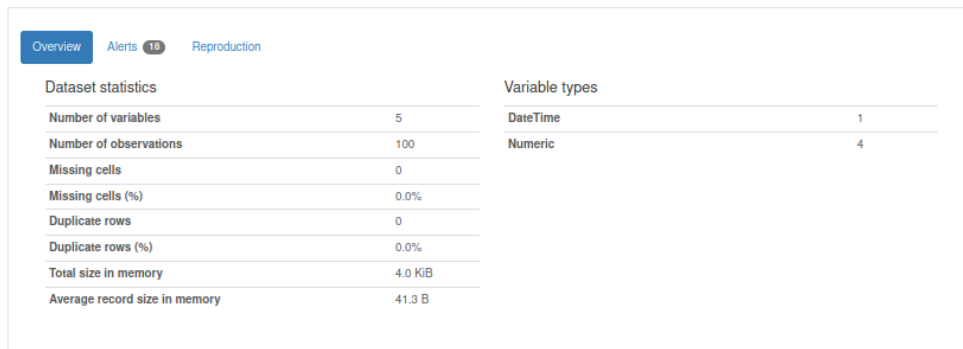
Exploratory data analysis

Exploratory data analysis (EDA) is provided with *pandas-profiling*²⁸, an open-source Python module for generating interactive HTML reports in web format. The reports are generated automatically from pandas dataframes in a standardised format. Each data profile report includes several sections: Overview, Variables, Interactions, Correlations, Missing values, and Samples.

The *Overview* presents global details about the dataset, such as variable types, i.e., whether they are numerical or categorical, number of observations, missing cells count, duplicates count, memory footprint. Furthermore, for each variable, the values' data types, minimum/maximum values, distinct and missing counts, as well as size in memory are provided along with a histogram of the variable's distribution (Figure 12 Figure 1). The user can use a "Toggle details" button to view quantile statistics (minimum value, Q1, median, Q3, maximum, range, interquartile range), descriptive statistics (mean, mode, standard deviation, sum, median absolute deviation, coefficient of variation, kurtosis, skewness), common and extreme values. Furthermore, an Alerts interface is provided to overview potential data quality issues (high correlation, skewness, uniformity, zeros, missing values, constant values, between others).

²⁸ pandas-profiling: <https://pandas-profiling.ydata.ai/docs/master/>

Overview



Variables

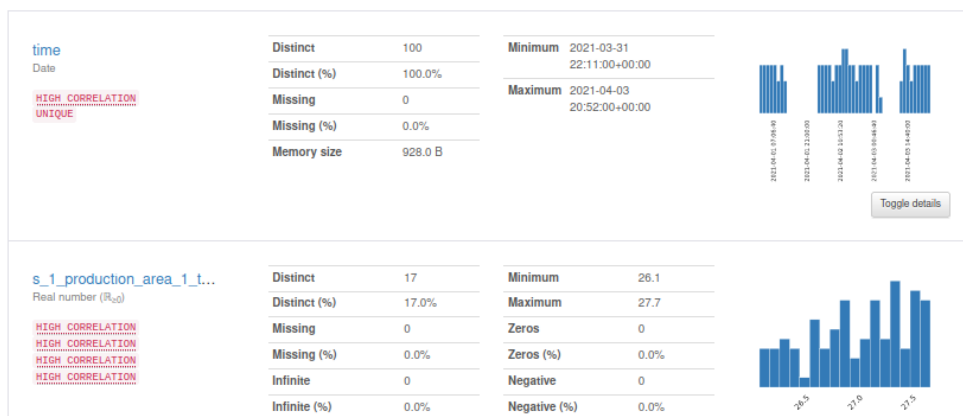
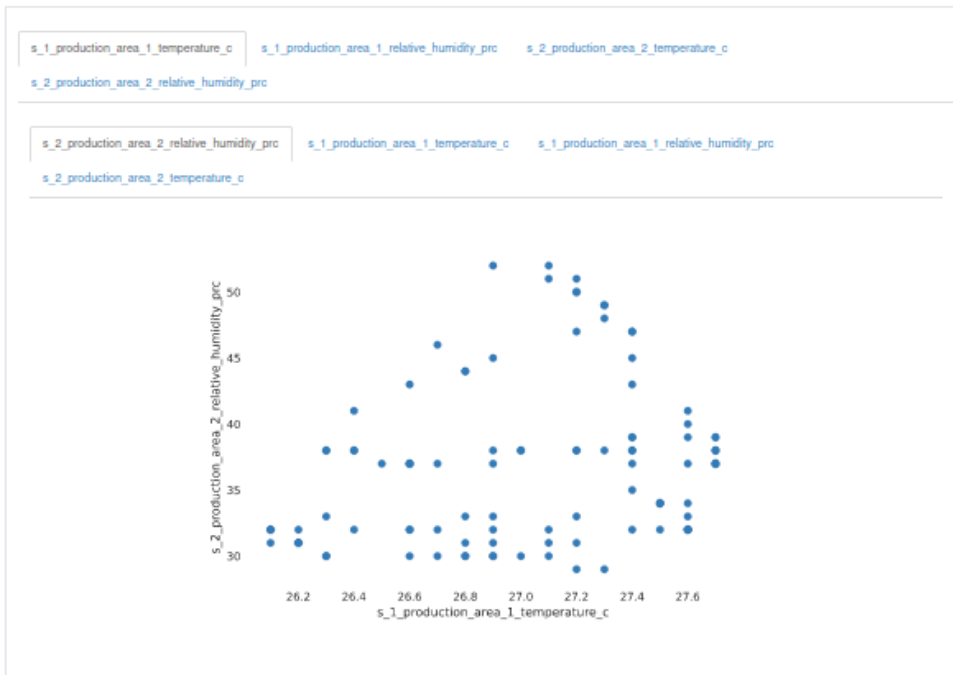


Figure 12: EDA overview. The data used in this example are drawn from the EnerMan YIOTIS pilot.

The Interactions and Correlations sections (Figure 13) present relationships between two variables as a scatterplot and using the correlation coefficients, respectively. To estimate the correlation coefficients there several choices, such as Spearman, Pearson, Kendall, Phik coefficients, and for each choice a description is available.

Interactions



Correlations

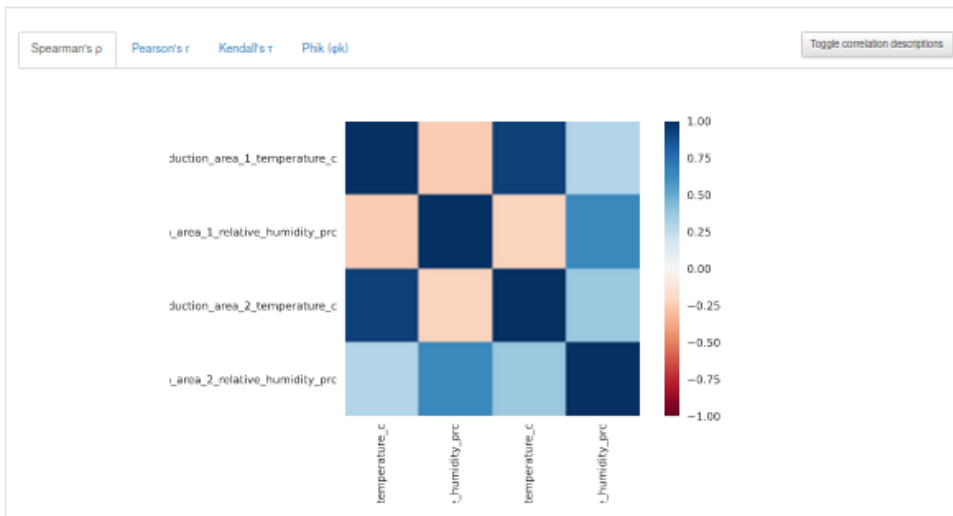
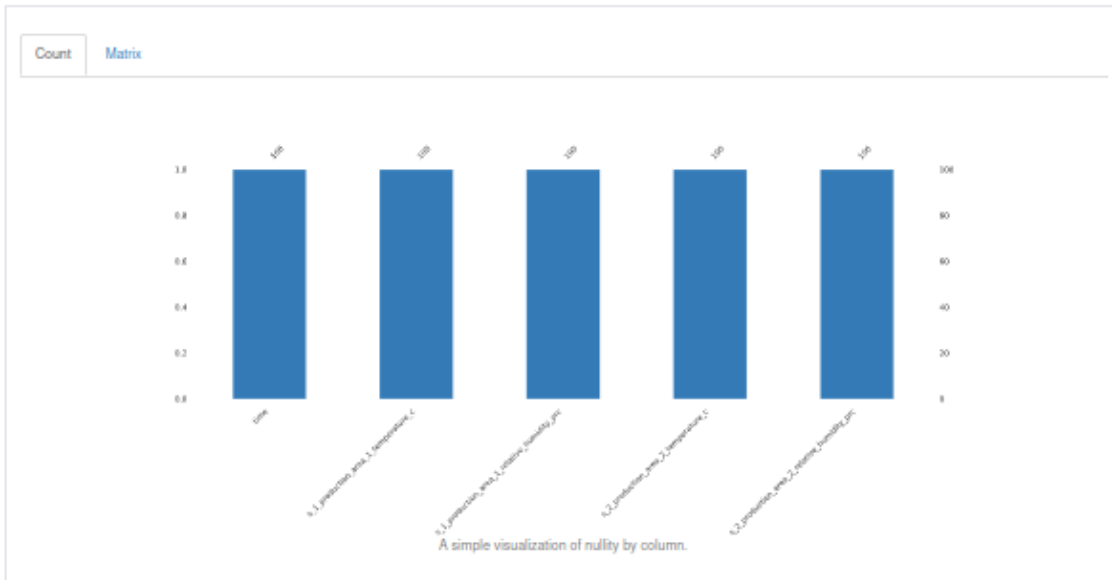


Figure 13: EDA interactions and correlations profiling. The data used in this example are drawn from the EnerMan YIOTIS pilot.

Finally, a Missing values section offers a visualization of NULL values per column, and in the Sample section, the user can see the first and the last rows of the dataframe (Figure 14).

Missing values



Sample

First rows

	time	s_1_production_area_1_temperature_c	s_1_production_area_1_relative_humidity_prc	s_2_production_area_2_temperature_c	s_2_production_area_2_relative_humidity_prc
0	2021-03-31 22:11:00+00:00	26.6	20.0	21.5	31.0
1	2021-03-31 22:41:00+00:00	26.7	20.0	21.7	31.0
2	2021-03-31 23:11:00+00:00	26.9	21.0	21.9	31.0
3	2021-03-31 23:41:00+00:00	26.9	22.0	22.1	31.0
4	2021-04-01 00:11:00+00:00	26.9	21.0	22.2	31.0
5	2021-04-01 00:41:00+00:00	26.8	22.0	22.2	31.0
6	2021-04-01 01:11:00+00:00	27.1	21.0	22.3	31.0
7	2021-04-01 01:42:00+00:00	27.2	21.0	22.5	31.0
8	2021-04-01 02:12:00+00:00	27.4	21.0	22.7	31.0
9	2021-04-01 02:42:00+00:00	27.5	21.0	22.8	31.0

Figure 14: EDA missing values and sample section. The data used in this example are drawn from the EnerMan YIOTIS pilot.

Timeseries statistical analyses

The BDAE provides additional analytics focused on timeseries statistics. This functionality is powered by Darts²⁹ with statsmodels³⁰ as a backend. Darts is a Python module that supports both univariate and multivariate time series and models. In the EDA, we use the `darts.utils.statistics` module to obtain insights in a timeseries’ seasonality, trend and other statistic patterns (Figure 15).

²⁹ Darts: <https://unit8co.github.io/darts/>

³⁰ Statsmodels: <https://www.statsmodels.org/stable/index.html>

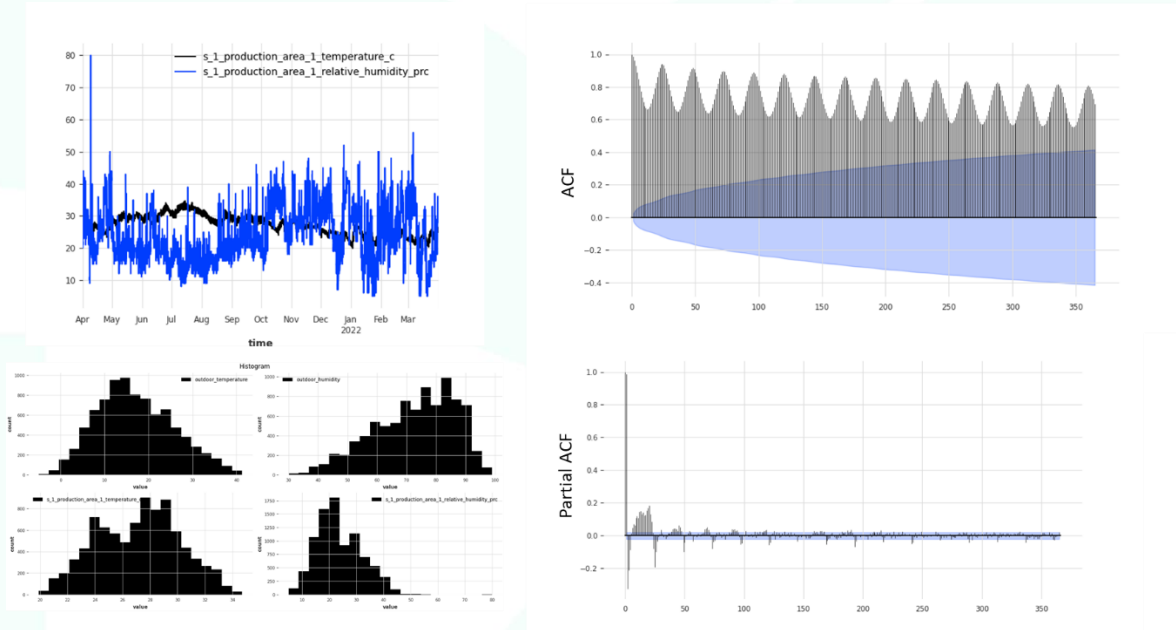


Figure 15: Timeseries statistics. The data used in this example are drawn from the EnerMan YIOTIS pilot.

3.3. Big Data Analytics Engine API

The EnerMan BDAE API serves as the starting point to access the historical data per use-case and process, request predictions and analytics on selected data, download preprocessed data to use for downstream tasks, and read or update the data models. Each pilot has its own application protected with security protocol. Each application is connected to TimescaleDB to read historical data and other derivative data tables, MLflow to fetch the appropriate models and use them for predictions, and MinIO to retrieve the data models. In the following sections, we will present the tools we used for building the API, the interfaces it provides and the operations they cover.

3.3.1. FastAPI, tools and standards

The BDAE API is implemented with FastAPI³¹, a high-performance web framework to develop RESTful APIs in Python. Furthermore, it is designed around and fully compatible with the open standards, i.e., OpenAPI³² and JSON Schema³³, and provides by default Swagger UI³⁴ and Redoc³⁵ code generated documentation interfaces. The framework depends on Pydantic³⁶ as well as the standard Python type hints to validate, serialize and deserialize data. Furthermore, FastAPI fully supports asynchronous programming using the Uvicorn³⁷ server and the Starlette³⁸ framework/toolkit for Asynchronous Server Gateway Interface (ASGI)³⁹. A quick overview of these tools and their functionalities follows:

³¹ FastAPI: <https://fastapi.tiangolo.com/>
³² OpenAPI: <https://www.openapis.org/>
³³ JSON Schema: <https://json-schema.org/>
³⁴ Swagger UI: <https://swagger.io>
³⁵ Redoc: <https://redocly.com/>
³⁶ Pydantic: <https://pydantic-docs.helpmanual.io/>
³⁷ Uvicorn: <https://www.uvicorn.org/>
³⁸ Starlette: <https://www.starlette.io/>
³⁹ ASGI: <https://asgi.readthedocs.io>

OpenAPI

The OpenAPI Specification defines a standard, programming language-agnostic interface description for HTTP APIs. An OpenAPI document describes API services, either be in YAML or JSON format. The document can be produced statically, or it can be generated dynamically from an application, which makes it both human- and machine-readable. The OpenAPI documents enable code generated interactive documentation. The current implementation of BDAE API uses OpenAPI version 3.0.2.

JSON Schema

This is an IETF⁴⁰ standard that provides a format to describe the structure and the validation constraints of a JSON document. Its application enforces consistency and data validity across similar JSON data. More specifically, JSON Schema uses a vocabulary to describe existing data formats, with a clear human- and machine- readable documentation and validates data to enable automated testing and ensure the quality of the submitted data. The OpenAPI specification has incorporated JSON Schema.

Redoc

Redoc is an open-source tool for generating API documentation using an OpenAPI Specification. The generated documentation is clean and customizable with a three-panel, responsive layout: the left panel contains a search bar and navigation menu, the central panel contains the documentation, and the right panel contains request and response examples. Some of the benefits of this solution, besides being open source, is the attractive design, customizability, and compatibility. FastAPI provides as the default documentation solution.

Swagger UI

Swagger UI enables client side to visualize and interact with an API's resources without any implementation logic, since the interface is automatically generated by the OpenAPI Specification. The interface is user friendly and easy to understand, and it enables developers of API downstream tasks to quickly try endpoints execution and monitor their API requests and the responses they receive. Swagger UI is enabled by default in FastAPI.

Pydantic

Pydantic is a library that uses Python type annotations to facilitate data validation. It enforces type hints at runtime and provides user friendly errors in case of invalid data. Using the typing module, Pydantic schemas can also be used recursively, allowing to define very complex hierarchical objects. Furthermore, Pydantic can be used for settings managements. Fast API is fully compatible with Pydantic.

Uvicorn

Uvicorn is an ASGI web server implementation for Python. It provides a minimal low-level server/application interface for async frameworks, which supports HTTP/1.1 and WebSockets.

Starlette

Starlette is a lightweight ASGI framework/toolkit, which uses the Uvicorn server. It can run completely asynchronously and is one of the fastest Python frameworks available. It allows multiple incoming events and outgoing events for each application, as well background coroutines. This is ideal for machine learning application, where time-consuming training events should not block other API requests. FastAPI is a sub-class of Starlette.

⁴⁰ IETF: <https://www.ietf.org/>

Other web frameworks that have been examined as alternatives for the EnerMan BDAE API, are Django REST framework⁴¹ and Flask⁴². FastAPI, is speed-oriented by design which makes it the top performer among the three, and this was our basic criterion. Additional advantages included the built-in documentation and OpenAPI and JSON Schema compliance, as well as the ease-of-use code-wise (code autocompletion everywhere) and datatype validation out-of-the-box.

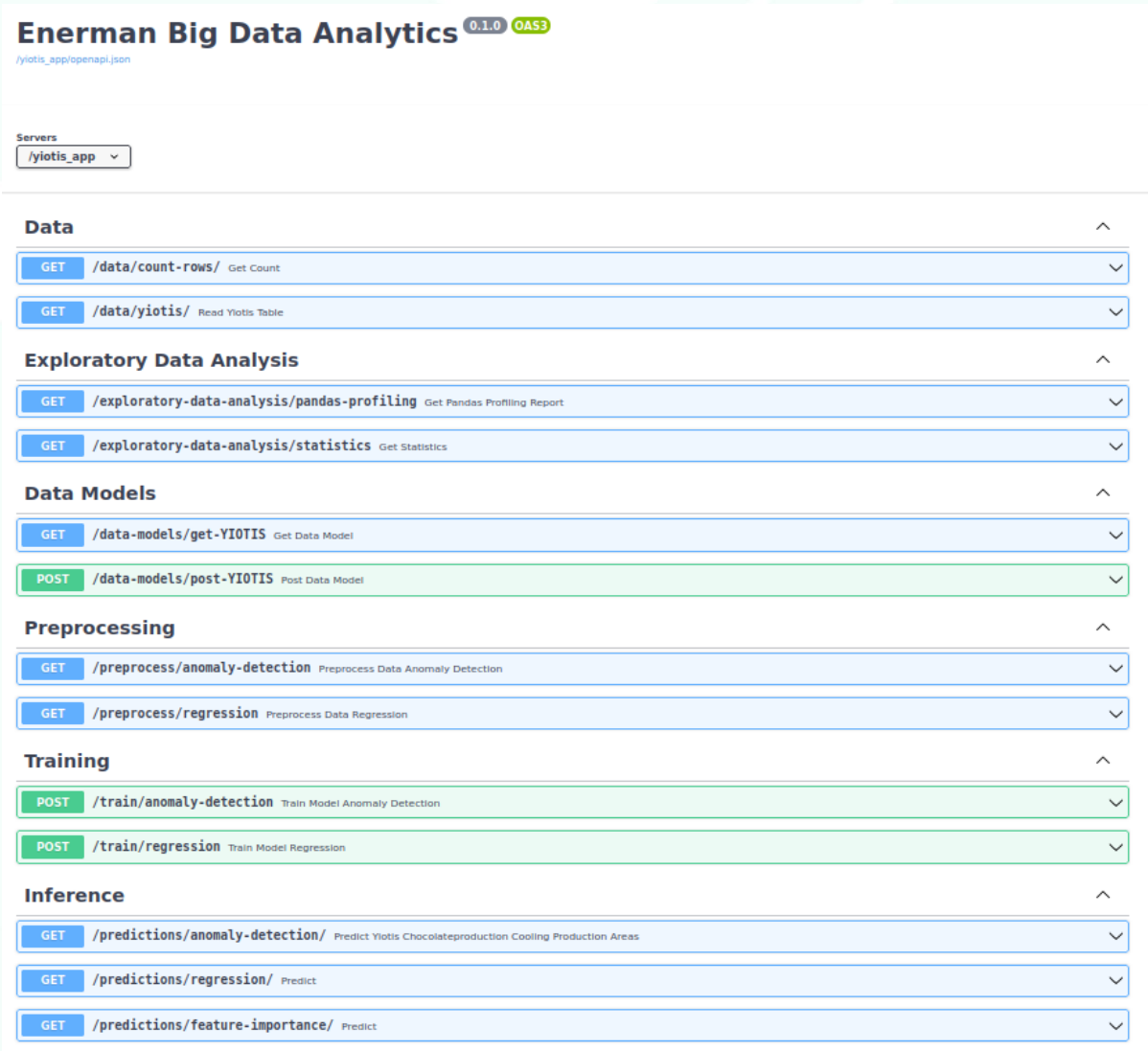
3.3.2. BDAE API interfaces

In this section we will present several interfaces of the BDAE API to offer some insights on its functionalities to the user, whether this is another EnerMan component, e.g., the Visualization Framework, or a human user. Initially, we will look at the `/docs` interface powered by Swagger UI. For each pilot API, the Swagger UI provides endpoints in six different categories, but more categories and endpoints can be easily added in the future, since modularity and extensibility are central to the API design.

The main view of the interface (Figure 16) is split into the six main sections of endpoints: Data, Exploratory Data Analysis, Data Models, Pre-processing, Training, and Predictions.

⁴¹ Django REST framework: <https://www.django-rest-framework.org/>

⁴² <https://flask.palletsprojects.com/en/2.1.x/>



Enerman Big Data Analytics 0.1.0 OAS3
/yiotis_app/openapi.json

Servers
/yiotis_app

Data

- GET /data/count-rows/ Get Count
- GET /data/yiotis/ Read Yiotis Table

Exploratory Data Analysis

- GET /exploratory-data-analysis/pandas-profiling Get Pandas Profiling Report
- GET /exploratory-data-analysis/statistics Get Statistics

Data Models

- GET /data-models/get-YIOTIS Get Data Model
- POST /data-models/post-YIOTIS Post Data Model

Preprocessing

- GET /preprocess/anomaly-detection Preprocess Data Anomaly Detection
- GET /preprocess/regression Preprocess Data Regression

Training

- POST /train/anomaly-detection Train Model Anomaly Detection
- POST /train/regression Train Model Regression

Inference

- GET /predictions/anomaly-detection/ Predict Yiotis Chocolateproduction Cooling Production Areas
- GET /predictions/regression/ Predict
- GET /predictions/feature-importance/ Predict

Figure 16: User interface for the YIOTIS pilot BDAE API. Six main categories of endpoints are provided.

Endpoints in the *Data* section are related to TSDb operations, such as count datapoints in a table or return the table itself. The `data/<pilot_org>` endpoint (Figure 17) in specific, accepts several query parameters to control the response. The data format can either be JSON or CSV, and a dropdown menu shows the tables which are available to select. Data can be selected in specific time range and resampling can be applied given a bucket width size. Finally, each table can be selected using an offset and a limit.

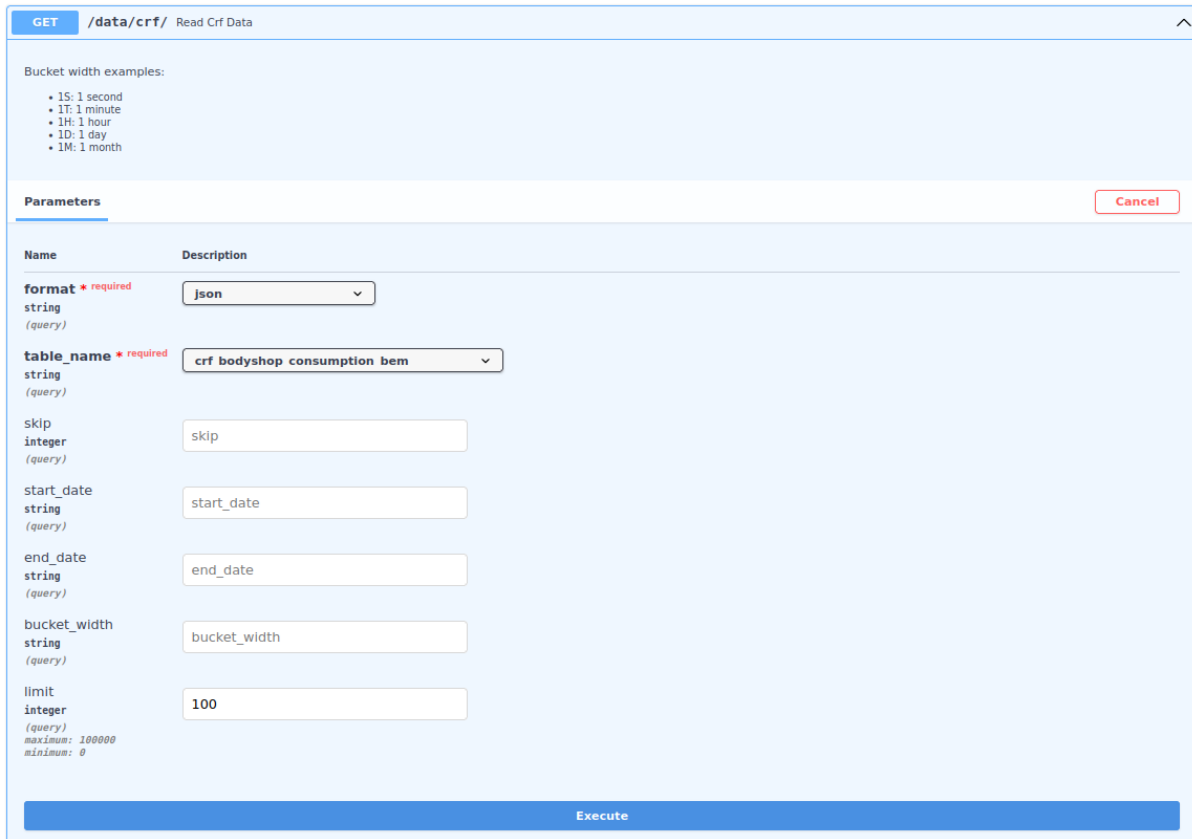


Figure 17: User interface for the CRF pilot BDAE API. Endpoint to request a data set from the TSDB.

The Swagger UI provides the description of the response as it was defined with the respective Pydantic schema, to enable data validation (Figure 18), as well as an example value (Figure 19).

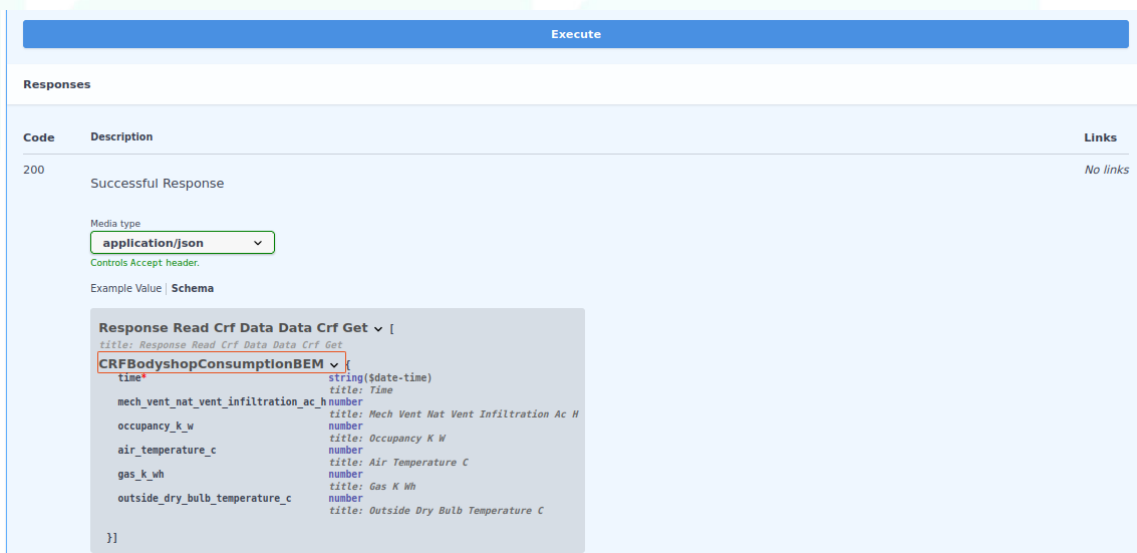


Figure 18: User interface for the CRF pilot BDAE API. The schema of the response defined with Pydantic models.

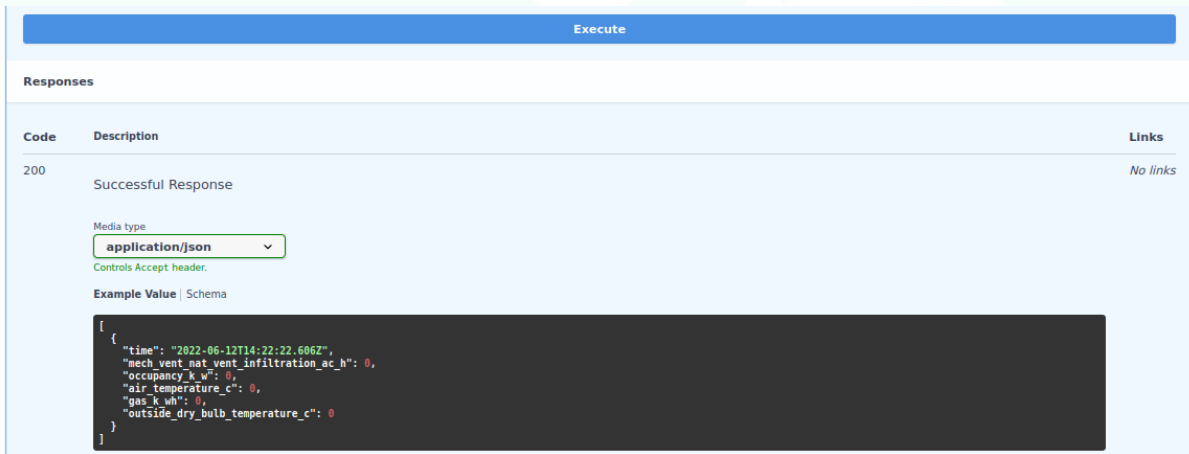


Figure 19: User interface for the CRF pilot BDAE API. Example value for a response based on predefined Pydantic models.

When an endpoint is executed, a Curl command and a Request URL with the query parameters are presented in the Responses section of the Swagger UI. The response in Figure 20 was queried with format=json, table_name=yiotis_cooling_outdoor, skip=0, start_date=2015-01-01%2000:00:00Z, end_date=2022-12-31%2023:59:59Z, bucket_width=1 and limit=100.

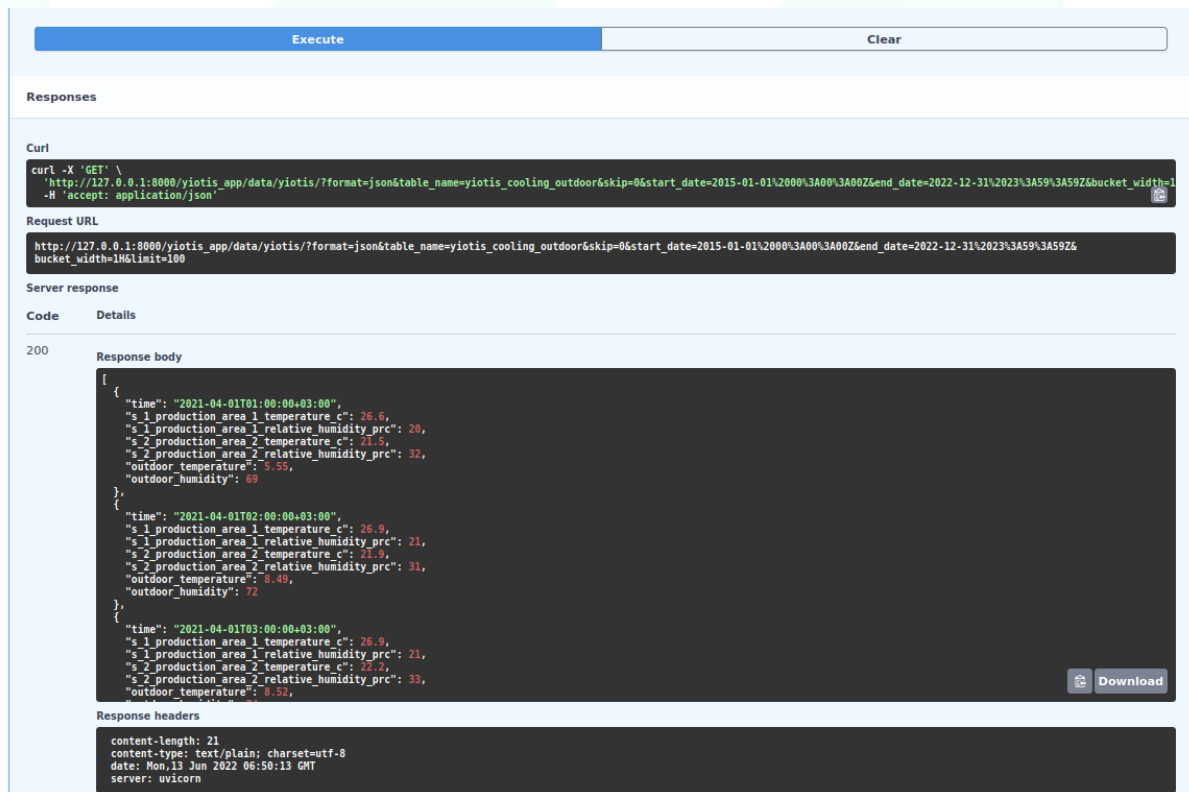


Figure 20: User interface for the YIOTIS pilot BDAE API. Swagger UI response includes a Curl request, the Request URL and the response body.

In the Exploratory Data Analysis (EDA) section, the endpoint available triggers the *pandas-profiling* process described in section Exploratory data analysis and 3.2.4. Additional EDA tools can be added here such as Dtale⁴³ or Sweetviz⁴⁴. The query parameters for this EDA endpoint (Figure 21) are the table name of interest and the limit of the rows selected. Data profiling tools are relatively time consuming and is better to be requested directly with the Request URL, instead of executed through the Swagger UI. Another endpoint in the EDA section provides data insights based on several time series statistical analyses using Darts (Figure 22).

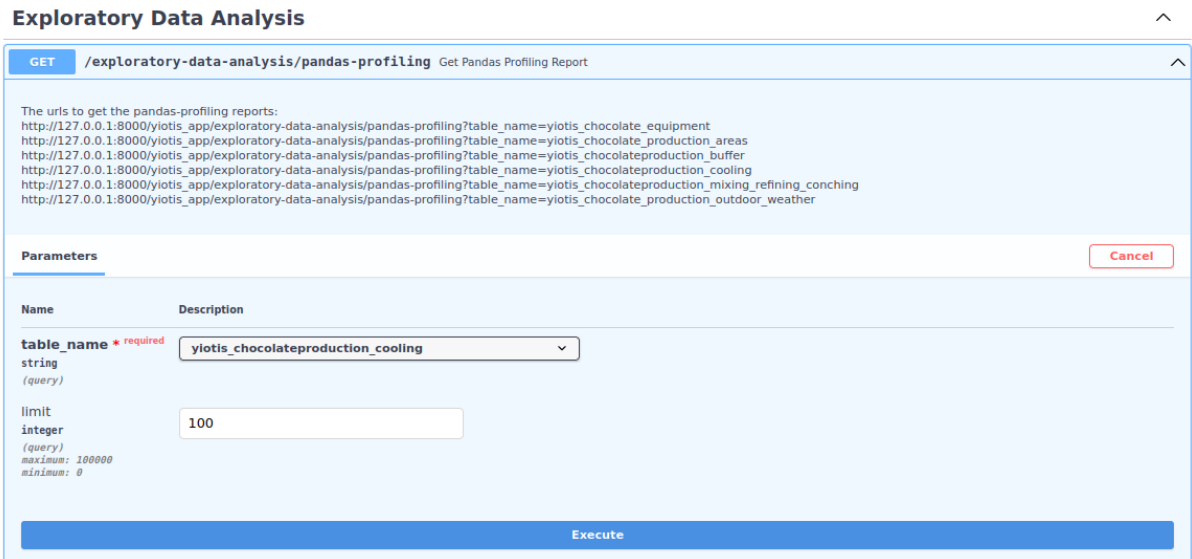


Figure 21: User interface for the YIOTIS pilot BDAE API. Pandas-profiling EDA endpoint.

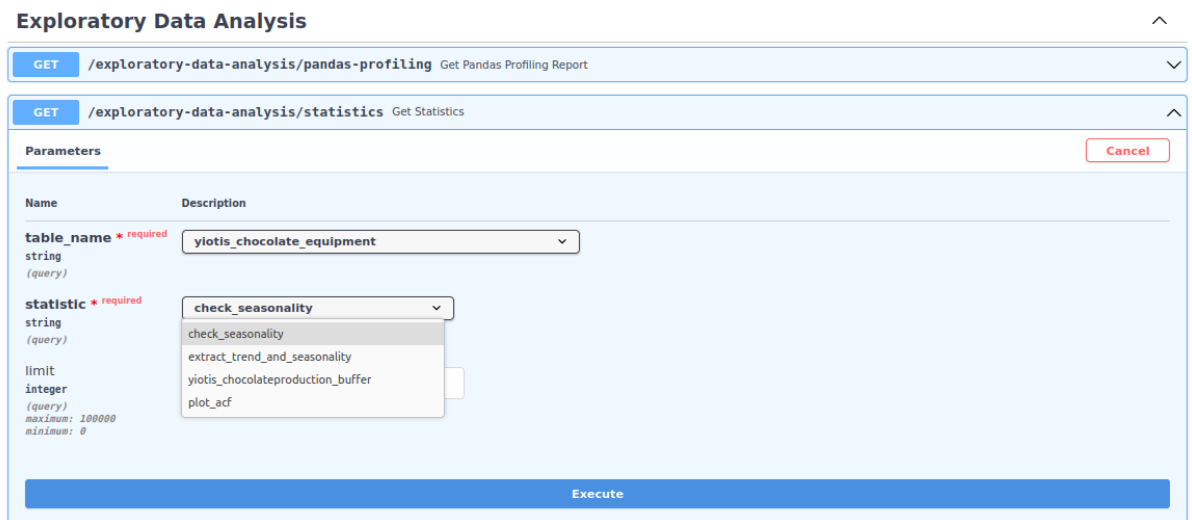
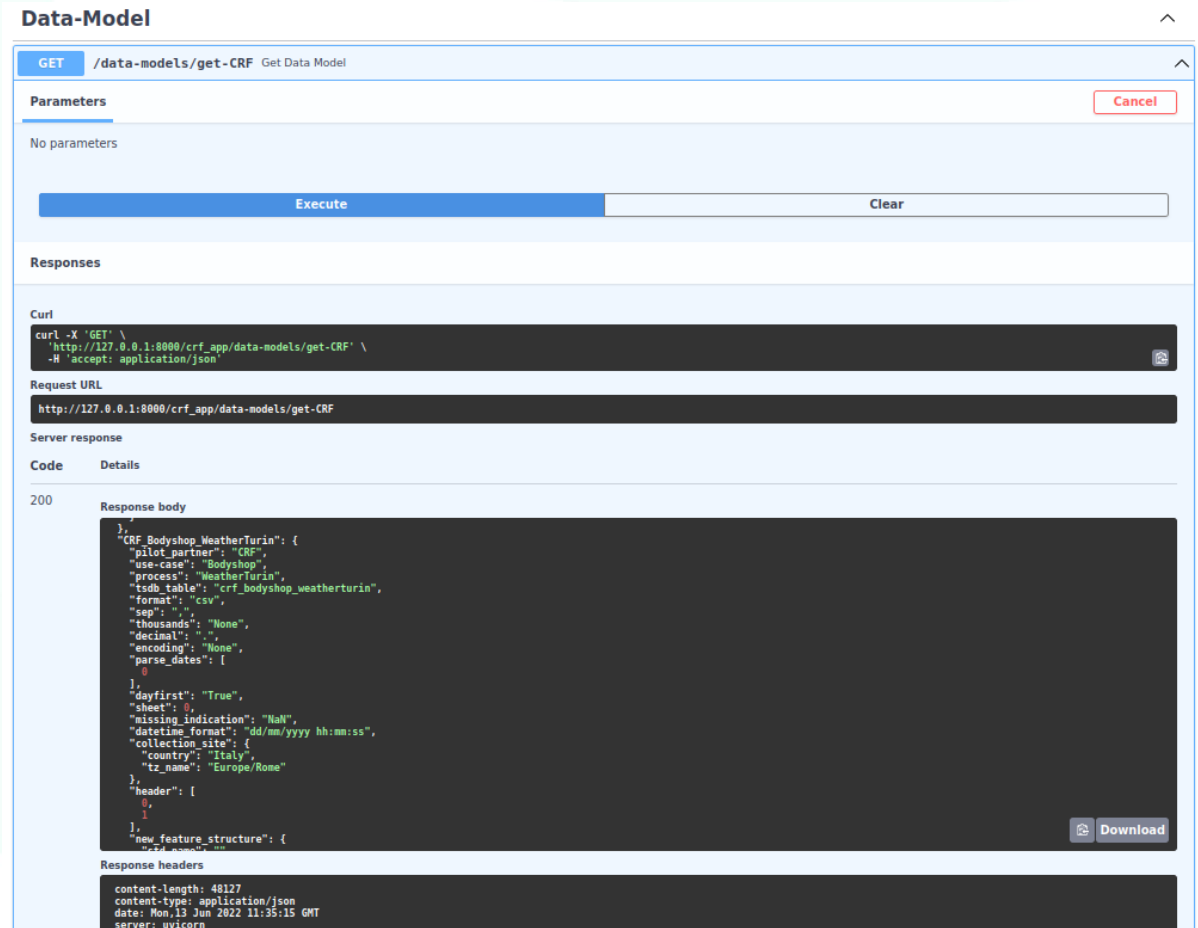


Figure 22: User interface for the YIOTIS pilot BDAE API. Timeseries statistical analyses offered in the EDA section.

⁴³ Dtale: <https://github.com/man-group/dtale>

⁴⁴ Sweetviz: <https://github.com/fbdesignpro/sweetviz>

In the Data Models section, there are both GET and POST endpoints to retrieve and update the Data Models JSON files used by the edge nodes to harmonize the data. An edge node can use the GET method to retrieve a pilot's Data Model file and use the metadata information to run the suitable harmonization operations. Subsequently, if any changes have been introduced in the data schema, the edge node harmonizer will update the Data Model file accordingly and use the POST method to apply the update in the NoSQL database (MinIO, see also section 3.1.1).



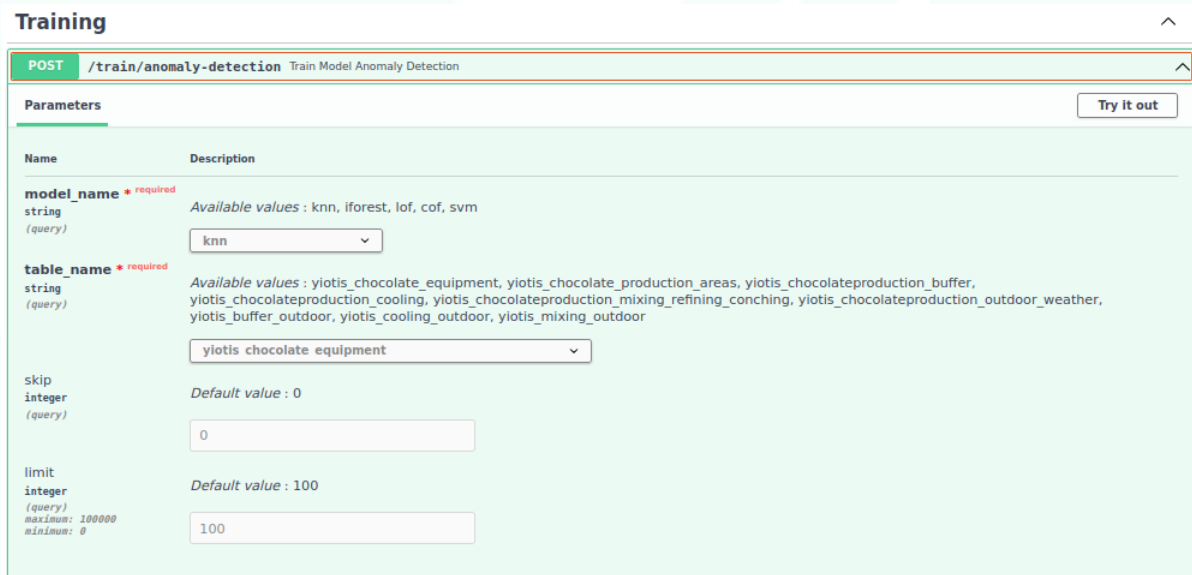
The screenshot displays the 'Data-Model' API interface. At the top, it shows the endpoint 'GET /data-models/get-CRF' with the description 'Get Data Model'. Below this, there is a 'Parameters' section which is currently empty, with a 'Cancel' button. An 'Execute' button is visible, followed by a 'Clear' button. The 'Responses' section is expanded, showing the following details:

- Code:** 200
- Response body:** A JSON object representing a Data Model for 'CRF_Bodyshop_WeatherTurin'. The JSON includes fields for 'pilot_partner', 'use-case', 'process', 'isdb table', 'format', 'sep', 'thousands', 'decimal', 'encoding', 'parse_dates', 'dayfirst', 'sheet', 'missing_indication', 'datetime format', 'collection_site' (with country and tz_name), 'header', and 'new_feature_structure'.
- Response headers:** content-length: 48127, content-type: application/json, date: Mon, 13 Jun 2022 11:35:15 GMT, server: uvicorn.

Figure 23: User interface for the CRF pilot BDAE API. A GET endpoint to retrieve a pilot's Data Model JSON file from the NoSQL storage.

The Pre-processing section provides endpoints to request a table's data after they have been prepared and they are ready to use as input to a machine learning training model, such as anomaly detection or regression. The same pre-processing pipelines is used for training via the BDAE API, however, providing just the pre-processed data can be useful if end-users want to train their own architectures.

In the Training section (Figure 24), there are dedicated endpoints for each machine learning class, e.g., anomaly detection, regression. Each endpoint takes the table name of interest and the algorithm name as query parameters. Furthermore, for supervised learning, such as regression, the target variable name is also required.



Training

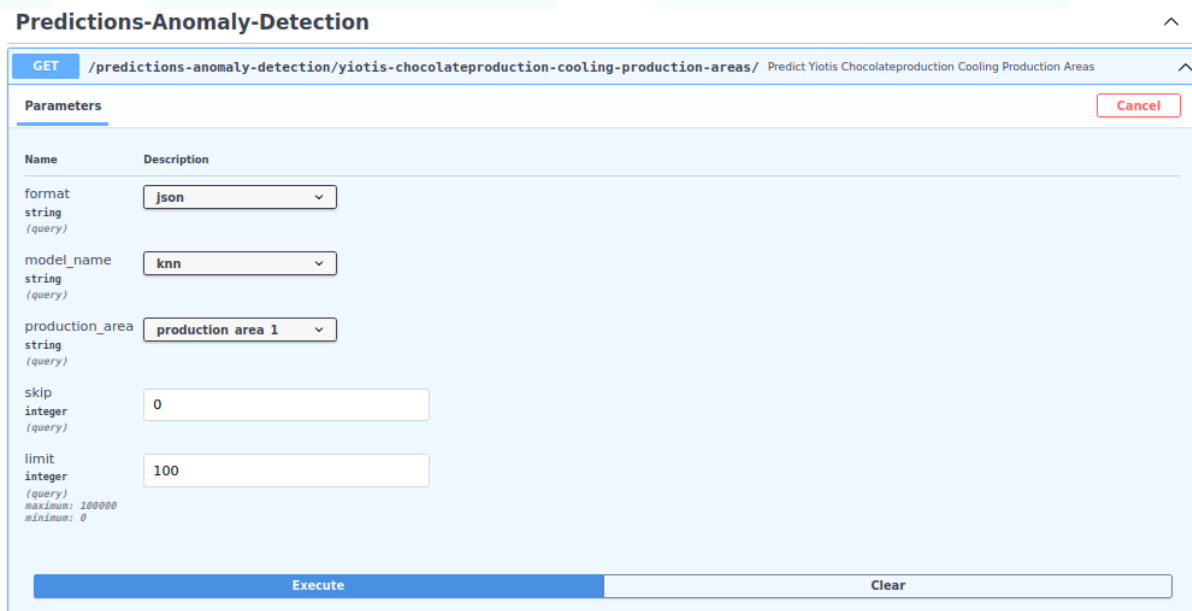
POST /train/anomaly-detection Train Model Anomaly Detection

Parameters Try it out

Name	Description
model_name * required string (query)	Available values : knn, iforest, lof, cof, svm knn
table_name * required string (query)	Available values : yiotis_chocolate_equipment, yiotis_chocolate_production_areas, yiotis_chocolateproduction_buffer, yiotis_chocolateproduction_cooling, yiotis_chocolateproduction_mixing_refining_conching, yiotis_chocolateproduction_outdoor_weather, yiotis_buffer_outdoor, yiotis_cooling_outdoor, yiotis_mixing_outdoor yiotis chocolate equipment
skip integer (query)	Default value : 0 0
limit integer (query) maximum: 100000 minimum: 0	Default value : 100 100

Figure 24: User interface for the YIOTIS pilot BDAE API. Model training endpoint.

In the Inference section, an endpoint per machine learning task is provided for prediction requests. Query parameters are given as input to define the specific algorithm that is available, and the data schema (model signature) that was used to train the data (Figure 25). The results are returned either in JSON or CSV format, and they contain the values of the initial features plus the prediction result (Figure 26), i.e., a 0 or 1 label and an anomaly score for anomaly detection tasks, the value of the target variable in supervised tasks such as regression, or a cluster id for clustering tasks.



Predictions-Anomaly-Detection

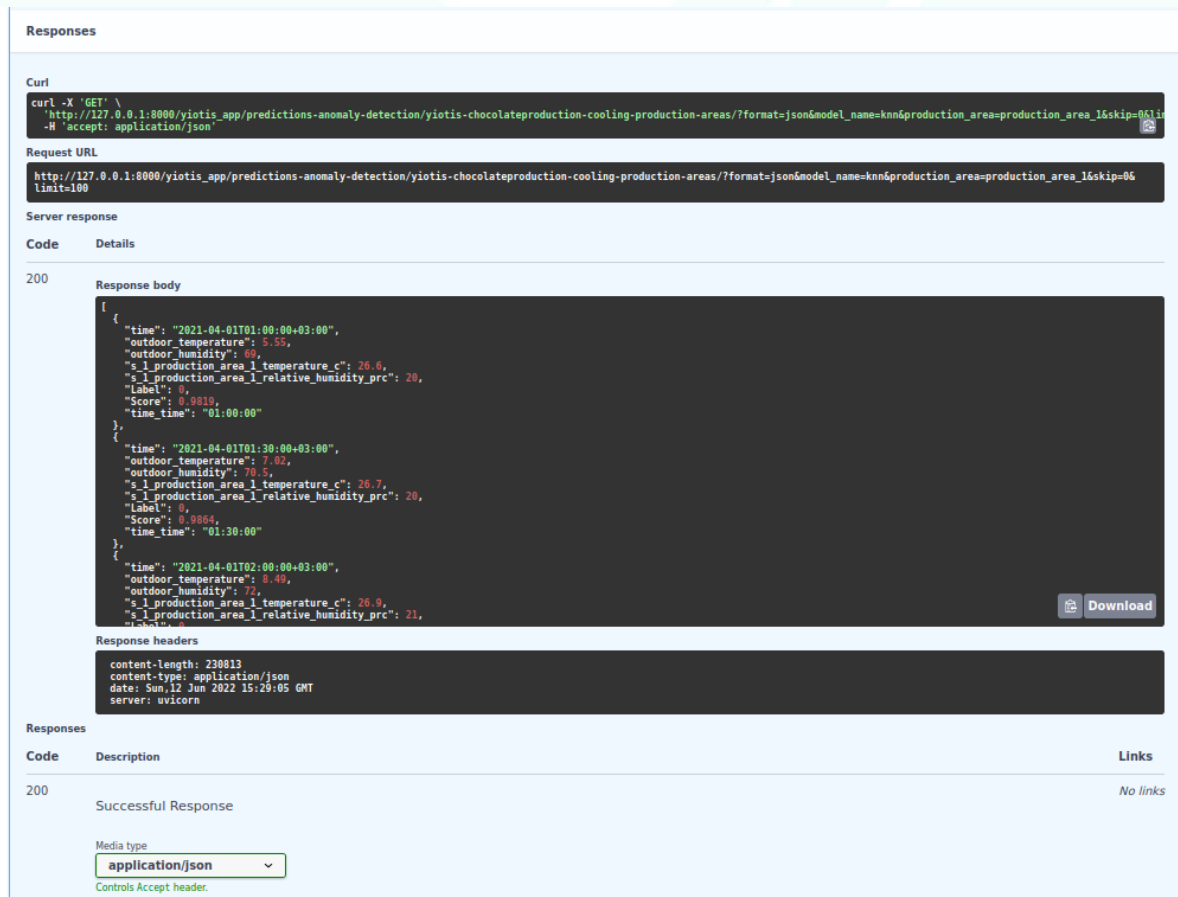
GET /predictions-anomaly-detection/yiotis-chocolateproduction-cooling-production-areas/ Predict Yiotis Chocolateproduction Cooling Production Areas

Parameters Cancel

Name	Description
format string (query)	json
model_name string (query)	knn
production_area string (query)	production area 1
skip integer (query)	0
limit integer (query) maximum: 100000 minimum: 0	100

Execute Clear

Figure 25: User interface for the YIOTIS pilot BDAE API. An endpoint in the Predictions section for the anomaly detection machine learning task.



Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/yiotis_app/predictions-anomaly-detection/yiotis-chocolateproduction-cooling-production-areas/?format=json&model_name=knn&production_area=production_area_1&skip=0&limit=100' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/yiotis_app/predictions-anomaly-detection/yiotis-chocolateproduction-cooling-production-areas/?format=json&model_name=knn&production_area=production_area_1&skip=0&limit=100
```

Server response

Code	Details
200	<p>Response body</p> <pre>[{ "time": "2021-04-01T01:00:00+03:00", "outdoor_temperature": 5.55, "outdoor_humidity": 69, "s_1_production_area_1_temperature_c": 26.6, "s_1_production_area_1_relative_humidity_prc": 20, "Label": 0, "Score": 0.9819, "time_time": "01:00:00" }, { "time": "2021-04-01T01:30:00+03:00", "outdoor_temperature": 7.02, "outdoor_humidity": 70.5, "s_1_production_area_1_temperature_c": 26.7, "s_1_production_area_1_relative_humidity_prc": 20, "Label": 0, "Score": 0.9864, "time_time": "01:30:00" }, { "time": "2021-04-01T02:00:00+03:00", "outdoor_temperature": 8.49, "outdoor_humidity": 72, "s_1_production_area_1_temperature_c": 26.9, "s_1_production_area_1_relative_humidity_prc": 21, "Label": 0, "Score": 0.9864, "time_time": "02:00:00" }]</pre> <p>Response headers</p> <pre>content-length: 230813 content-type: application/json date: Sun, 12 Jun 2022 15:29:05 GMT server: uvicorn</pre>

Responses

Code	Description	Links
200	Successful Response	No links

Media type:
Controls Accept header.

Figure 26: User interface for the YIOTIS pilot BDAE API. A response with predictions for the anomaly detection task on batch data.

In addition to the predictions, the section also provides an endpoint for a feature importance estimation (Figure 27). This analysis is based on the computation of the regression coefficients, and it provides an idea on how influential each independent variable is. For example, Figure 27, shows the results for a YIOTIS pilot use case, where the production area temperature is the most significant feature to impact power consumption.

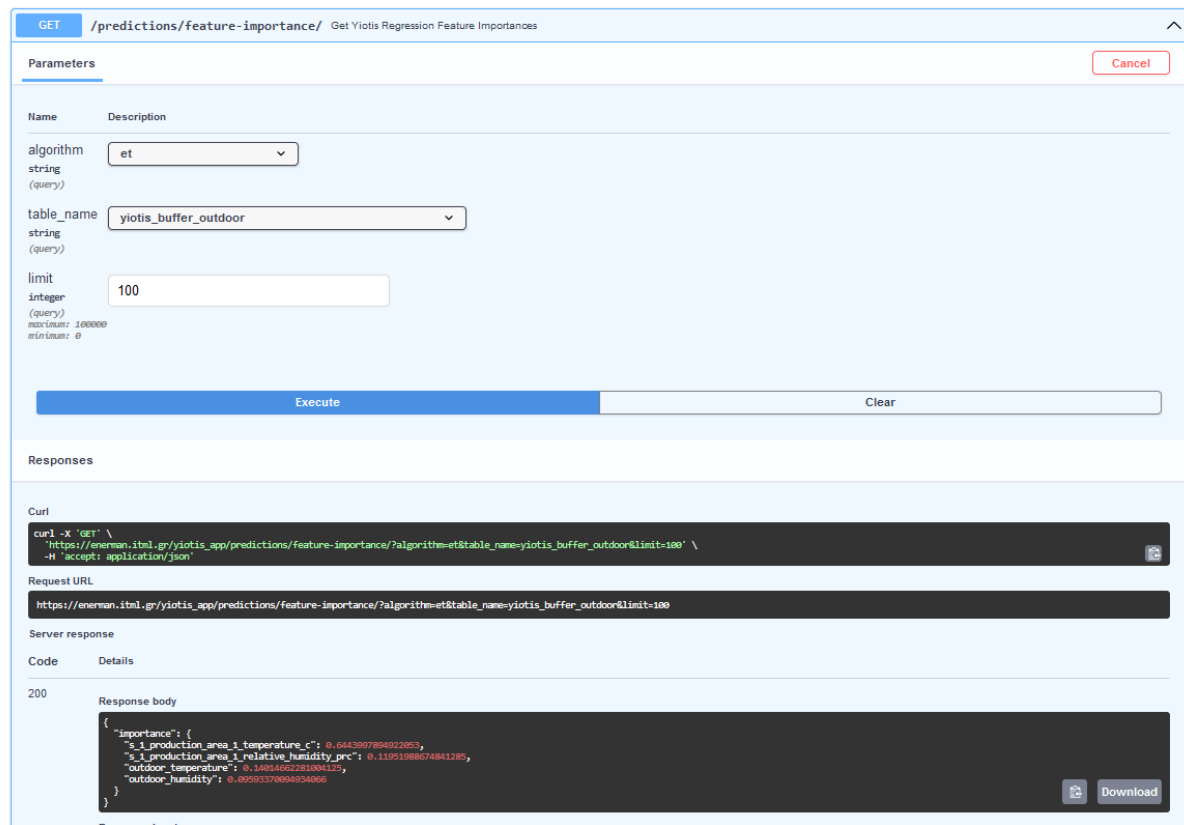


Figure 27: User interface for the YIOTIS pilot BDAE API. Feature importance estimation to detect influential variables.

Finally, the API provides automatically generated documentation powered by Redoc (Figure 28). The documentation page has three panels: the left panel contains a search bar and a navigation menu to the different sections of endpoints, the central panel contains the query parameters and types, and the right panel contains request and response examples.

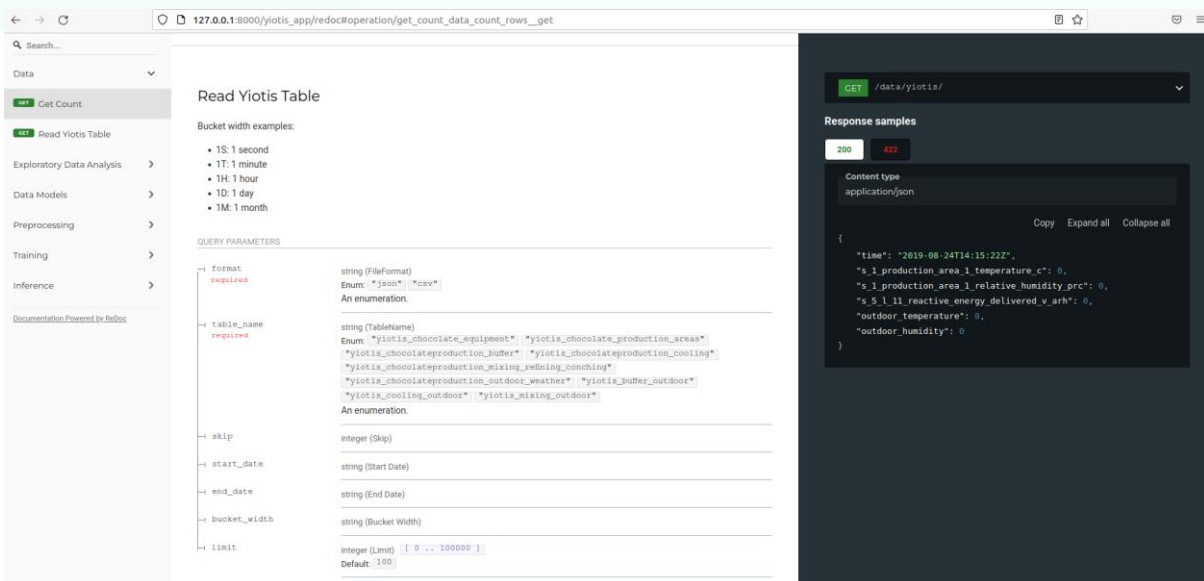


Figure 28: User interface for the YIOTIS pilot BDAE API. Code-generated API documentation.

3.4. Security

The BDAE API is protected with HTTP Basic Auth, a general framework for access control and authentication defined in RFC 7235⁴⁵. With this method, the cloud server responds to a client request with an HTTP 401 "Unauthorized" error and asks for a username and a password with a WWW-Authenticate response header. The client then presents the integrated prompt for a username and password to the user and issues a request that includes the correct Authorization header. Furthermore, each of the databases deployed in the BDAE architecture has each own user authentication and access control, content encryption scheme, and client connections are restricted to specific IPs. Loading to TimescaleDB is restricted to internal service after checking the file format and content adheres to certain rules. Loading a model directly to MLflow is available to any user with password, but again, the file needs to fulfil certain format and content requirements. Finally, API Post requests to MinIO that intend to change the JSON content are protected since there is no dynamic code evaluation taking place and the JSON schema is well defined with Pydantic models.

3.5. Deployment

The current deployment of the BDAE is fully containerized integrated with Docker images of TimeScaleDB, MLflow, MinIO. Docker compose is used to run it as a multi-container application.

3.6. Communication with other EnerMan components

BDAE communicates with several other components of the EnerMan system to facilitate data storage, processing and retrieval needs, as well as to provide predictions and analytics. Following, we highlight these connections and their characteristics.

3.6.1. *EnerMan Intelligent Node*

The EnerMan Intelligent nodes deployed in the edge units incorporate a harmonization module that communicates with the BDAE and executes a workflow that applies the first layer of pre-processing to the raw data. The edge nodes use a GET request to read the data models JSON files from the BDAE API (see also section 3.1.2) and a POST request to modify their content whenever the data schema changes at the client side. The edge node uses SFTP to transfer the harmonized CSVs to the BDAE server.

More details on the harmonization module are provided in D2.1 (Preliminary version of EnerMan Data Collection and Management Components, M12) and D2.2 (Final Version of EnerMan Data Collection and Management Components, M18).

3.6.2. *Industrial Management Visualization System*

BDAE serves data, predictions and analytics to the visualization system, which accesses these outputs using GET requests to the BDAE API. The output is provided in JSON responses, which is the

⁴⁵ RFC 7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication:
<https://datatracker.ietf.org/doc/html/rfc7235>

appropriate format to be consumed by the visualization system interface. Endpoints in the BDAE API are continuously developed to accommodate the visualization system needs.

More details on the Industrial Management Visualisation System module are provided in D3.2 (EnerMan Visualization and Management Framework Design, M18).

3.6.3. Prediction Engine and Simulation Engine

The Prediction Engine as well as the Simulation Engine can communicate with the BDAE API using its endpoints to request pilots' data, either in their harmonized format or fully preprocessed and ready for downstream modelling task. Furthermore, since these two components produce their own data, they can use the SFTP transfer service along with the endpoint returning the Data Models to create new tables and load data in TSDB, similarly to the process followed with the pilot's data by the edge nodes system (described in 3.6.1). Finally, the BDAE API design seeks to accommodate potential requirements to host models trained offline and enable their serving through its endpoints.

More details on the prediction engine are provided in D4.3 (Computational prediction engine report, M24) and for the simulation engine in D4.2 (Simulation approach/mechanism for providing energy related indicators, M18).

3.6.4. Sphynx Machine Learning and Analytics Platform

Sphynx Machine Learning and Analytics (SphynxMLA) is a system developed for the STS platform, used to create custom ML workflows. SphynxMLA supports training and deployment of ML models for predictions and predictive performance evaluation. It is based on Alteryx' EvalML⁴⁶ AutoML library to perform the AutoML pipeline generation and currently supports supervised machine learning tasks. The SphynxMLA supports a variety of ML predictive models that are able to be included in the pipeline optimization and select the one with the best performance. Additionally, SphynxMLA supports ensembling, i.e., whether to use a combination of the best models to make predictions, instead of just the best one performance-wise.

SphynxMLA supports both single-sample and batch predictions on new unseen data, outputting multiple performance metrics:

- For classification: f1 micro, balanced accuracy, precision micro, recall micro
- For regression: negative mean squared error, R^2 , negative mean absolute percentage error, explained variance.

Through its GUI, the user can define specific hyperparameters of EvalML such as the model categories to include, the time and unit budget as well as the enabling of ensembling.

In the BDAE context, SphynxMLA can be used as an independent platform for supervised model training, and then use the BDAE API to load and register the serialized models with a POST Request. From that stage, the model serving can be conducted similarly to the natively trained models (section 3.2.3 Model inference and serving).

⁴⁶ EvalML: <https://evalml.alteryx.com/en/stable/>

3.7. Current state of the implementation and future steps

Currently, the first release of the BDAE implementation contains three main APIs related to three pilots, one for each pilot category:

- Energy Sustainable Efficient Food and Beverages Pilot: YIOTIS API
- Energy Sustainable Efficient Metal Processing Pilot: 3DNT API
- Energy Sustainable Automotive manufacturing, industrial equipment manufacturing Pilot: CRF API

As the integration progresses, more APIs can be replicated from the ones already developed. Regarding the specific functionalities, all the datasets from the three pilots have been harmonized, uploaded to the TSDB and are available to be requested in JSON or CSV format from the API as described in section 3.3.2 BDAE API interfaces, Figure 17).

Regarding the prediction services, several models have been trained and registered, and their predictions or artifacts can be requested from the pilots' APIs. Models are available for the following demonstrators:

- Anomaly detection for YIOTIS cooling stage of chocolate production (five algorithms: Connectivity-Based Outlier Factor, Isolation Forest, k-Nearest Neighbours Detector, Local Outlier Factor, One-class SVM detector)
- Regression models for YIOTIS buffer stage of chocolate production (two algorithms: Random Forest, Extra Trees)
- Clustering models for 3DN additive manufacturing (two algorithms: K-Means, Mean shift)

Although, the EnerML module in conjunction with the Sphynx MLA platform can support a plethora of models and analytics, and the base classes and operations are already in place, training *meaningful* models involves a good understanding of the pilots' requirements, which is an ongoing process. Thus, more models are expected to be added as the integration process progresses with and domain expert knowledge is injected in the implementation.

Regarding integration with the rest of the EnerMan components, the current release supports integrations with the EnerMan Intelligent Node (see section 3.6.1) and the Industrial Management Visualization System (section 3.6.2). Integration with the Sphynx Machine Learning and Analytics Platform is also supported by loading models trained with it to the BDAE model registry via the API. Next steps will focus on integrating with Prediction Engine and Simulation Engine (section 3.6.3) in a

similar approach that was followed for the Intelligent Node, since prediction engine needs to both read and write data to BDAE (Figure 29).

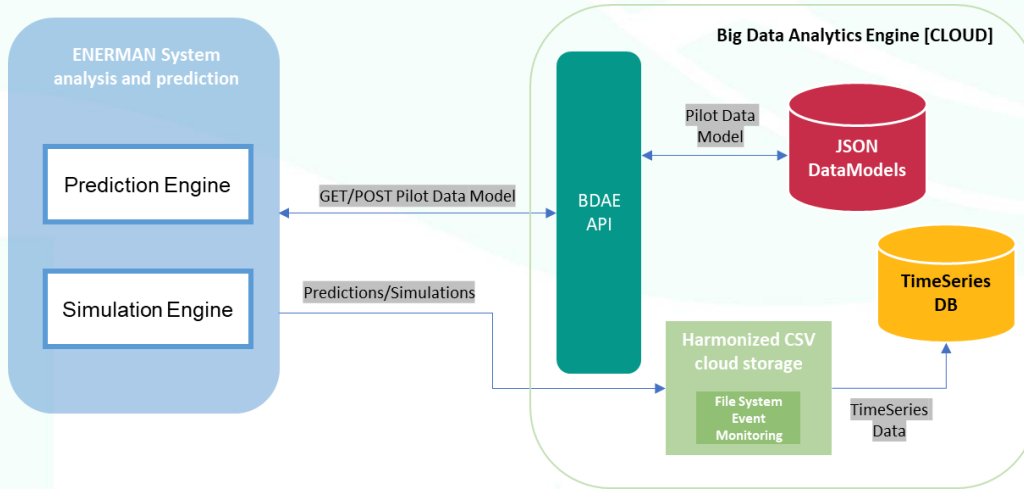


Figure 29: Integration with the EnerMan System Analysis and Prediction.

Finally, regarding model training, as we have discussed in section 3.2.2 (Model training and registration), currently, it is executed offline, using the configuration files and uploading the serialized model. A future plan is to be able to offer endpoints to request different training scenarios.

4. CONCLUSION

Big Data Analytics Engine (BDAE) is a critical component of the EnerMan system since it is expected to communicate with most of the other EnerMan components to facilitate data operations such as extract, read, write, preprocess, store, retrieve and analyse. In the context of Big Data applications, all these operations can be quite challenging, especially when the aim is towards building a generic application to accommodate multiple pilots, organizations, use cases and processes. Considering these underlying complexities, it became imperative to follow a robust design of the BDAE architecture. Our goal was to design a system that is modular, scalable, extensible, portable, and most importantly secure.

Implementation-wise, we incorporated strictly open-source software solutions of high-performance, security and flexibility, aiming to an innovative and cost-efficient solution. Python was the main programming language of the implementation since it offers access to an extremely rich ecosystem of libraries and solutions for data management and analytics. Regarding the storage infrastructure, a critical aspect of the BDAE architecture, we aimed at high-performance, scalable, and distributed solutions. Similarly, the API implementation adopted cutting-edge solutions of high-performance and extensibility, based on and fully compatible with open standards.

The EnerML module of the BDAE was envisioned as a decoupled solution that provides all the necessary pre-processing and analytics functionalities by integrating state-of-the-art ML frameworks. The module design has been influenced by the qualities of modularity and extensibility, since the analytics domain is vast and as the project progresses, it should be straightforward and effortless to incorporate more algorithms and analyses.

At a conceptual level, the BDAE provides methods for detecting anomalies, predictions, feature importance analysis to identify influential variables, cluster analysis, and other analytics to understand the variables of interest and their interactions. In the future, the main action points will be to continue integrating the BDAE with the other EnerMan components and improve the analytics by incorporating more feedback and requirements by the end users.

Overall, the BDAE architecture aims to provide a complete solution to manage the EnerMan data flows and support all the stages of the machine learning lifecycle, from data pre-processing to model training and serving, to assist pilots to extract value from their data.

5. REFERENCES

- [1] K. C. Houtmeyers, A. Jaspers and P. Figueiredo, “Managing the Training Process in Elite Sports: From Descriptive to Prescriptive Data Analytics,” *International Journal of Sports Physiology and Performances*, vol. 16, no. 11, p. 1719–1723, April 2021.
- [2] A. Shi-Nash and D. R. Hardoon, “Chapter 19: DATA ANALYTICS AND PREDICTIVE ANALYTICS IN THE ERA OF BIG DATA,” in *Internet of Things and Data Analytics Handbook*, Wiley Online Library, 2016.
- [3] G.-H. Kim, S. Trimi and J.-H. Chung, “Big-data applications in the government sector,” *Communications of the ACM*, vol. 57, no. 3, p. 78–85, 2014.
- [4] H. R. Addo-Tenkorang and T. Petri, “Big data applications in operations/supply-chain management: A,” *Computers & Industrial Engineering*, vol. 101, pp. 528-543, 2016.
- [5] I. H. Khan and M. Javaid, “Big Data Applications in Medical Field: A Literature Review,” *Journal of Industrial Integration and Management*, vol. 6, no. 1, p. 53–69, 2021.
- [6] M. O. Gökalp, K. Kayabay, M. A. Akyol, P. E. Eren and A. Kocyigit, “Big Data for Industry 4.0: A Conceptual Framework,” in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2016.
- [7] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani and S. U. Khan, “The rise of “big data” on cloud computing: Review and open research issues,” *Information Systems*, vol. 47, pp. 98-115, 2015.
- [8] M. Andreolini, M. Colajanni, M. Pietri and S. Tosi, “Adaptive, scalable and reliable monitoring of big data on clouds,” *Journal of Parallel and Distributed Computing*, Vols. 79-80, pp. 67-79, 2015.
- [9] C. Imen, B. Wadii and F. I. Riadh, “Big Data: Concepts, Challenges and Applications,” in *Computational Collective Intelligence*, 2015.
- [10] D. Libes, S. Shin and J. Woo, “Considerations and recommendations for data availability for data analytics for manufacturing,” in *2015 IEEE International Conference on Big Data (Big Data)*, Santa Clara, CA, USA, 2015.
- [11] A. C. Eberendu, “Unstructured Data: an overview of the data of,” *International Journal of Computer Trends and Technology (IJCTT)*, vol. 38, no. 1, pp. 46-50, 2016.
- [12] J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García and F. Herrera, *Big Data Pre-processing*, Springer, 2020.
- [13] C. C. Aggarwal, *Data Mining: The Textbook*, USA: Springer, 2015.
- [14] S. García, J. Luengo and F. Herrera, *Data Pre-processing in Data Mining.*, Berlin: Springer, 2015.
- [15] Z. Changming and G. Daqi, “Influence of Data Pre-processing,” *Journal of Computing Science and Engineering*, vol. 10, no. 2, pp. 51-57, 2016.

- [16] G. Salvador, R.-G. Sergio, L. Julián, B. J. Manuel and H. Francisco, “Big data pre-processing: methods and prospects,” *Big Data Analytics*, pp. 1-9, 2016.
- [17] H. Woo, J. Kim and W. Lee, “Validation of Text Data Pre-processing Using a Neural Network Model,” *Mathematical Problems in Engineering*, vol. 2020, pp. 1-9, 2020.
- [18] K. A. Khalid, O. Tayo, R. O. Gayla and C. W. Donald, *Computational Learning Approaches to Data Analytics in Biomedical Applications*, Elsevier, 2019.
- [19] C. V. G. Zelaya, “Towards Explaining the Effects of Data Pre-processing on Machine Learning,” in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, China, 2019.
- [20] J. Luengo, S. García and F. Herrera, “On the choice of the best imputation methods for missing values considering threegrups of classification methods.,” *Knowl Inf Syst.*, vol. 1, no. 77, pp. 77-108, 2012.
- [21] J. Tenenbaum, V. Silva and J. Langford, “A global geometric framework for nonlinear dimensionality reduction.,” *Science*, vol. 290, no. 5500, pp. 2319-323, 2000.
- [22] P. R. Jones, “A note on detecting statistical outliers in psychophysical data,” *Atten Percept Psychophys.*, vol. 81, no. 5, p. 1189–1196, 2019.
- [23] Y. Lei, F. Jia, J. Lin, S. Xing and S. X. Ding, “An Intelligent Fault Diagnosis Method Using Unsupervised Feature Learning Towards Mechanical Big Data,” *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 3137 - 3147, 2016.
- [24] R. A. Peterson, “Ordered quantile normalization: a semiparametric transformation built for the cross-validation era,” *Advances in Computational Data Analysis*, vol. 47, no. 13-15, pp. 2312-2327, 2020.
- [25] M. Khushi, K. Shaukat, T. M. Alam, I. A. Hameed, S. Uddin, S. Luo, X. Yang and M. C. Reyes, “A Comparative Performance Analysis of Data,” *IEEE Access*, vol. 9, pp. 109960 - 109975, 2021.
- [26] W. Zhang, H. Jiang, Z. Yang, S. Yamakawa, K. Shimada and L. B. Kara, “Data-driven Upsampling of Point Clouds,” *Computer-Aided Design*, vol. 112, pp. 1-13, 2019.
- [27] Q. Qiu, M. Wanga, J. Guoc, Z. Liud and Q. Wang, “An adaptive down-sampling method of laser scan data for scan-to-BIM,” *Automation in Construction*, vol. 135, p. 104135, 2022.
- [28] K. Yu, X. Wu, W. Ding and J. Pei, “Scalable and Accurate Online Feature Selection for Big Data,” *ACM Transactions on Knowledge Discovery from Data*, vol. 11, no. 2, p. 1–39, 2017.
- [29] G. S. Dean J, “Mapreduce: Simplified data processing on large clusters.,” *OSDI*, pp. 137-50, 2004.
- [30] J. Cai, J. Luo, S. Wang and S. Yang, “Feature selection in machine learning: A new perspective,” *Neurocomputing*, vol. 300, p. 70–79, 2018.
- [31] M. Tan, I. W. Tsang and L. Wang, “Towards Ultrahigh Dimensional Feature Selection,” *Journal of Machine Learning Research*, vol. 15, pp. 1371-1429, 2014.

- [32] F. S. L. Jesse, M. Hajj and R. Sazdanovic, "Big Data Approaches to Knot Theory: Understanding the Structure of the Jones Polynomial," arXiv, 219.
- [33] N. Tsapanos, A. Tefas, N. Nikolaidis and I. Pitas, "A distributed framework for trimmed kernel k-means clustering.," *Pattern Recogn.*, vol. 48, no. 8, pp. 2685-698, 2015.
- [34] Y. Wei, X. Zhang, Y. Shi, L. Xia, S. Pan, J. Wu, M. Han and X. Zhao, "A review of data-driven approaches for prediction and classification of building energy consumption," in *Renewable and Sustainable Energy Reviews*, <https://doi.org/10.1016/j.rser.2017.09.108>., 2018, pp. 1027-1047.
- [35] J. D. de Guia, R. S. Concepcion, H. A. Calinao, S. C. Lauguico, E. P. Dadios and R. R. P. Vicerra, "Application of Ensemble Learning with Mean Shift Clustering for Output Profile Classification and Anomaly Detection in Energy Production of Grid-Tied Photovoltaic System," in *12th International Conference on Information Technology and Electrical Engineering (ICITEE)*, doi: 10.1109/ICITEE49829.2020.9271699, 2020.
- [36] M. Roux, "A comparative study of divisive hierarchical clustering," arxiv, arXiv:1506.08977v2 [cs.DS], 2015.
- [37] J. Sander, "Density-Based Clustering," in *Encyclopedia of Machine Learning and Data Mining*, Springer, Boston, MA., 2017, p. 349–353.
- [38] J. W. a. K. Z. Cai, "Adaptive Density-Based Spatial Clustering for Massive Data Analysis," *IEEE Access*, vol. 8, no. doi: 10.1109/ACCESS.2020.2969440, pp. 23346-23358, 2020.
- [39] J. McGonagle, G. Pilling and A. Dobre, "https://brilliant.org/," 2022. [Online]. Available: <https://brilliant.org/wiki/gaussian-mixture-model/>. [Accessed 6 2022].
- [40] O. Alghushairy, R. Alsini, T. Soule and X. Ma, "A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams.," *Big Data Cognitive Computing*, vol. 5, no. 1, 2021.
- [41] M. B. Markus, H.-P. Kriegel and R. T., "OPTICS-OF: Identifying Local Outliers," in *3rd European Conference on Principles and Practice of Knowledge*, Prague, 2000.
- [42] S. D. Anton, S. Kanoor, D. Fraunholz and H. D. Schotten, "Evaluation of Machine Learning-based Anomaly Detection Algorithms on an Industrial Modbus/TCP Data Set," in *13th International Conference on Availability, Reliability and Security*, <https://doi.org/10.1145/3230833.3232818>, 2018.
- [43] A. Mennatallah, G. Markus and A. Slim, "Enhancing one-class Support Vector Machines for unsupervised anomaly detection," in *ACM SIGKDD Workshop on Outlier Detection and Description*, 10.1145/2500853.2500857, 2013.
- [44] R. Lior and O. Maimon, "Decision trees," in *Data mining and knowledge discovery handbook*, Springer, Boston, MA, 2005, pp. 165-192.
- [45] L. Breiman, "Random Forests," in *Machine Learning 45*, Springer Boston MA, 2001, pp. 5-32.
- [46] P. Geurts, D. Ernst and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, no. 63, pp. 3-42, 2006.

- [47] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.
- [48] A. V. Dorogush, V. Ershov and A. Gulin, "CatBoost: gradient boosting with categorical features support," arXiv:1810.11363v1 [cs.LG], 2018.
- [49] M. Á. Carreira-Perpiñán, "A review of mean-shift algorithms for clustering," in *CRC Handbook of Cluster Analysis*, arXiv, 2015.

Energy Efficient Manufacturing System Management

enerman-H2020.eu



enermanh2020



enermanh2020



enermanh2020



HORIZON 2020

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958478

