Federated decentralized trusted dAta Marketplace for Embedded finance

D5.1 - Trusted and Explainable AI Techniques I

Title	D5.1 - Trusted and Explainable AI Techniques I
Revision Number	1.0
Task reference	T5.1 T5.2
Lead Beneficiary	ATOS
Responsible	Raquel Lazcano
Partners	ENG, IBM, MOH, UNP
Deliverable Type	DEM
Dissemination Level	PU
Due Date	2024-03-31 [Month 15]
Delivered Date	2024-03-25
Internal Reviewers	AL IQB
Quality Assurance	UPRC
Acceptance	Coordinator Accepted
Project Title	FAME - Federated decentralized trusted dAta Marketplace for Embedded finance
Grant Agreement No.	101092639
EC Project Officer	Stefano Bertolo
Programme	HORIZON-CL4-2022-DATA-01-04

This project has received funding from the European Union’s Horizon research and innovation programme under Grant Agreement no 101092639

Revision History

Version	Date	Partners	Description
0.1	2024-01-30	ATOS	ToC created
0.2	2024-02-18	IBM, JSI, UPRC	Contributions for SAX techniques component
0.3	2024-02-18	UPRC, IBM, JSI	Contributions for XAI scoring component
0.4	2024-02-29	ATOS	Contributions for AI/ML component, rest of the content integrated, introduction and conclusions written
0.9	2024-03-07	ATOS	Version for peer review
0.92	2024-03-19	ATOS	version with review from ubi+gft
0.93	2024-03-25	UPRC	Version after internal QA
1.0	2024-03-25	UPRC	Version for submission

Definitions

Acronyms	Definition
AAAI	American Association for Artificial Intelligence
AAI	authentication authorization infrastructure
ACM	Association for Computing Machinery
AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CEP	Center (for) Energy Policy
DL	Deep Learning
ESG	Environmental, social, and corporate governance
FAME	Federated decentralized trusted dAta Marketplace for Embedded finance
FDAC	Federated Data Assets Catalogue
FML	Federated Machine Learning
IBM	International Business Machines
ID	Identity
IEEE	Institute (of) Electrical (and) Electronic Engineers
JSI	Institut Jozef Stefan
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LLM	Large language model
ML	Machine Learning
NLP	Natural language processing
OS	Operating System
RAM	Random Access Memory
SA	Supervisory Authority
SAX	Situation Aware Explainability
UI	User Interface
URL	Uniform Resource Locator
WP	Workpackage
XAI	eXplainable Artificial Intelligence

Executive Summary

This document summarizes the work performed within the FAME project to enrich FAME’s marketplace with analytics, highlighting among them the creation of a library of Artificial Intelligence / Machine Learning (ML) models specifically tailored for the pilots of the project and coupled with eXplainable AI (XAI) and Situation Aware eXplainability (SAX) techniques to increase the trustworthiness of the solutions, hence fostering their adoption among the end users, which in this case are within the Embedded Finance (EmFi) sector.

The FAME project aims to establish a lightweight and accessible marketplace that facilitates the development of Embedded Finance (EmFi) applications and services. This is achieved by harnessing cutting-edge technologies such as Artificial Intelligence (AI), Machine Learning (ML), and eXplainable AI (XAI). Unlike existing marketplaces, FAME overcomes limitations in the adoption of trusted analytics by federating services like XAI techniques such as Situation Aware Explainability (SAX). This approach improves understanding of AI/ML models for end users and fosters confidence and trust in these technologies. FAME offers a standard-based, secure, interoperable, trustable, and federated data and analytics marketplace. This empowers users to exploit specific AI/ML models tailored to their scenarios and provides the means to enhance them with explainability tailored to their specific business processes.

This deliverable reports the progress made in two key components of the FAME architecture: the AI/ML catalogue and XAI techniques. The AI/ML catalogue is intended to contain a collection of models trained with data from FAME pilots, as well as potentially from public or synthetic datasets, specifically tailored to embedded finance applications. XAI techniques are essential to provide insight into these ML processes, not only to explain them, but also to provide tailored answers for each specific use case, which is made possible by incorporating business processes through the SAX subcomponent.

Specifically, this deliverable describes the work conducted during the first year of FAME in two tasks of WP5: Task 5.1 - Catalogue of AI/ML Techniques for EmFi and Task 5.2 - Explainable and Trustworthy Artificial Intelligence. The work was carried out collaboratively with the Consortium beneficiaries participating in these tasks as a continuous and incremental process. It should be noted that the work presented here is constantly evolving and may be subject to change as the project progresses. This deliverable should be considered as the basis for the Trusted and Explainable AI techniques that will be offered within the FAME marketplace, either as assets or services. The final version of this document, D5.3, is expected to be delivered in M33. It is important to note that this deliverable should be read in conjunction with D5.2 as they are closely related. While D5.1 discusses AI/ML/XAI/SAX techniques, D5.2 focuses on how to deploy these models in a smart manner, improving energy efficiency and the required data management infrastructure.

The document describes the specifications and prototype implementations of the components mentioned above, decomposing them into subcomponents, describing their functionalities and interfaces, analysing the state of the art, and highlighting their contribution to FAME. Additionally, it presents individual demonstrations for these components and provides links to demo videos.

Table of Contents

List of Figures

List of Tables

List of Codes

1 Introduction

As already established in D2.2 - Technical Specifications and Platform Architecture I [1], FAME project's ambition is to act as a low-complexity Data Space, offering assets to gain valuable insights in the field of Embedded Finance (EmFi) by leveraging state-of-the-art technologies, such as Artificial Intelligence (AI), Machine Learning (ML) and eXplainable AI (XAI). To do so, FAME's solution will consist of a standard-based, secure, interoperable, trustable, and federated data and analytics marketplace.

Although nowadays there are already many marketplaces up and running, they present some limitations, one of the most important ones being providing a poor adoption of trusted analytics. Most marketplaces exhibit limited utilization of trusted analytics technologies like XAI, which provide the necessary tools to simplify the understanding of AI/ML models for the end users, hence increasing their confidence, trust, and ultimately the adoption of these technologies. To that end, Situation Aware Explainability (SAX) comes in especially handy, since it provides not only this layer of explainability, but also enriches it with unique information coming from the related business process, hence providing distinctive insights for the end users.

As a result, these trusted analytics will provide end users with the ability to deploy and execute specific AI/ML models for their specific scenarios, also offering the possibility to enhance them with explainability considering their specific business processes.

These two components conform to the basis of the AI/ML Analytics Toolbox of FAME project, which, together with the components presented in D5.2 – Energy Efficient Analytics Toolbox I [2], will create the Energy Efficient Analytics Services system of FAME SA, as will be shown in Section 2 of this document.

1.1 Objective of the Deliverable

The objective of this deliverable is to document the progresses that have been made concerning the components of the FAME architecture associated with AI/ML processes: the AI/ML catalogue and the XAI techniques, including their high-level design and the first implementation work towards building the prototypes to be indexed in FAME.

The AI/ML catalogue will consist of a set of AI/ML models already trained both with data coming from FAME pilots and potentially from public/synthetic datasets, hence specifically tailored to embedded finance applications. In the following subsections, a list of potential models to be included in the catalogue will be presented, together with a mapping to their application in FAME pilots.

Whereas XAI techniques will provide much needed insights on these ML processes, not only providing a layer of explainability on top of the models, fostering and increasing trustworthiness in the end users, but also providing key answers tailored to each specific use case, since also the business processes will be considered thanks to the SAX subcomponent, which will be further explained in the document.

1.2 Insights from other Tasks and Deliverables

As already mentioned, the objective of this deliverable is to document the advancements of Task 5.1 - Catalogue of AI/ML Techniques for EmFi and Task 5.2 - Explainable and Trustworthy Artificial Intelligence, covering the first year of the project. These two tasks provide as main outcomes two of the main pillars of the analytics offering within FAME, since they cover the full AI/ML model offering: a catalogue of trained models together with XAI and SAX techniques to provide explainability on top of them. These two tasks are heavily intertwined, but they also rely on the rest of the tasks of WP5: Task 5.3 – Incremental Energy Efficient Analytics provides the necessary incremental data management, and Task 5.4 – Edge Data Management and EdgeAI Optimization and Task 5.5 – FML for Privacy Friendly and Energy Efficient Data Markets provide the smart deployment component needed to put in production the models and tools developed in the context of Task 5.1 and Task 5.2 in an energy efficient manner. Outside ofWP5, the contents of this deliverable are also connected, to some extent, to: 1) WP2, since FAME architecture is coming from it; 2) WP3 and WP4, since, for the results of these components to be indexed in the Federated Data Asset Catalogue (FDAC) coming from WP4, key components coming from WP3 will have to be used (i.e., the Authentication and Authorization Infrastructure (AAI)); and 3) WP6, since the data coming from FAME pilots is crucial to the components being presented in this document, as well as the requirements extracted from them are key to populate the catalogue of AI/ML models and provide the necessary contextual information for performing SAX.

1.3 Structure

This deliverable is structured following the common template defined within FAME for all deliverables detailing one or more components of the FAME SA. Specifically, the structure is the following:

- **Chapter 1 - Introduction:** This section summarizes the main objectives of the deliverable, explaining its connection to other work packages and its role within FAME.
- **Chapter 2 - Positioning into FAME SA:** This section provides a comprehensive view of the present deliverable with respect to FAME SA, highlighting the components that are covered in this deliverable.
- **Chapter 3 - Components Specification:** This section provides an in-depth description of the components of the Energy Efficiency Analytics Services related to ML and XAI, providing a description of their specifications, interfaces, functionalities, and relevant related work, as well as their integration with other components of the architecture. Hence, this section provides key insights to better understand the main components of FAME SA.
- **Chapter 4 - Components Demonstration:** This section aims at highlighting the practical application of the components described in Section 3. For all of them, the same information is provided if necessary: pre-requisites, installation guide, and user manual. The objective of this section is to illustrate how the components presented in this document can be used from FAME perspective.
- **Chapter 5 - Conclusions:** This section closes the deliverable, summarizing the main achievements and technical advancements of the components presented in this document. It also assesses the Key Performance Indicators (KPIs) established for the components described in this document.

2 Positioning into FAME SA

Considering the C4 architecture model of the FAME SA [1] [4], replicated in Figure 1 for the reader's convenience, two of its components are going to be presented in depth in the remaining of the document: the ML/AI Analytics container, which is the main outcome of Task 5.1, and the SAX techniques container, which is produced by Task 5.2, both of them being embedded in the Energy Efficient Analytics Services system. It should be noted that the SAX techniques container has been subdivided into two different containers: SAX techniques and XAI scoring. This division will be maintained throughout the remaining of the document.

The main objective of the ML/AI analytics container is to provide a catalogue of trained AI/ML models ready to be deployed, while the XAI container aims to add a layer of explainability on top of these models. Finally, the goal of the SAX container is to enrich the provided explainability, not only considering the model itself and the data that has been used for its training, but also all the contextual information related to data, its provenance, and its usage. These components are going to be fully detailed in Section 3 (and demonstrated in Section 4), going into the details of their internal structure and subcomponents.

The rest of the components under the mentioned system are presented in D5.2 – Energy Efficient Analytics Toolbox I [2], which is more focused on the deployment of these models and tools, including all data management aspects.

As already mentioned, both the present deliverable and D5.2 present work in progress and are expected to be continued and used as the basis for D5.3 and D5.4, which will be delivered towards the end of the project, and which will present the final version of all the components under the Energy Efficient Analytics Services system.

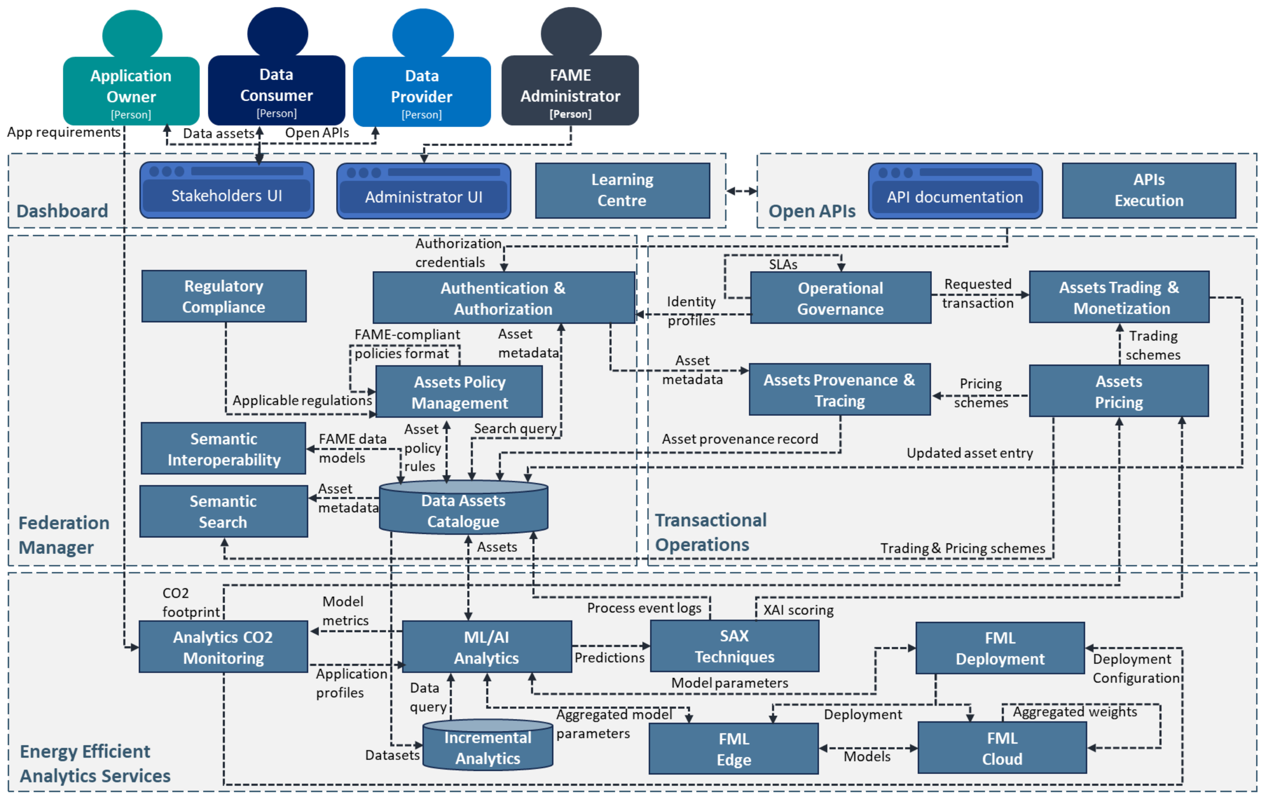


Figure 1 – FAME SA C4 container diagram

3 Components Specification

3.1 ML/AI Analytics

3.1.1 Description

The AI/ML Analytics component brings to FAME the necessary tools to build, train, store, version, track, and deploy different ML models that meet the requirements provided by the pilot partners.

From this component, various deep learning-based models will be indexed on the FAME marketplace to be available to future customers. These models cover a wide range of EmFi applications, leveraging the use cases proposed by the project's pilot partners, as well as providing the necessary functions to let users train and perform inferences with them. Table 1 shows the current progress of the solutions implemented (Green) or in progress (Blue for those that need further refinement with pilot partners and Yellow for those that are already under development) that have been addressed for each of the pilots of this project.

Table 1 - Summary of all the solutions already implemented or under development.

	Pilot 1	Pilot 2	Pilot 3	Pilot 4	Pilot 5	Pilot 6	Pilot 7
ML Recommender	1				1		
ML Profiler	1	1					
ML Risk Assessment	1		1			1	1
ML Fraud Detector			1				
ML Forecasting					1	1	1
ML Sentiment Analysis					3		
ML High-Res Interpolator						1	
ML Anomaly Detector							1
Under Discussion	3	1	2		1	1	3
Under Development						2	
Already Implemented					4		
Total	**3**	**1**	**2**		**5**	**3**	**3**

It is important to note that the proposed solutions are based on the Use Cases defined in D6.1 - Use Cases Specification and Pilot Sites Preparation I [5] and the business requirements provided by the pilot partners (e.g., there are no solutions that address the use cases of Pilot 4 because the need of further refinement in terms of requirements for this pilot.). A brief description of these solutions is given below:

- **ML Recommender Solution:** This is a system that focuses on filtering information, usually based on ML algorithms, that can predict a customer's ratings or preferences for a particular asset or family of assets from a set of client's features.
- **ML Profiler Solution:** Given a set of selected features from a provider, this type of system focuses on providing the most relevant customer profile by using the insights from an ML model trained with several features-profile pair samples.
- **ML Risk Assessment Solution:** Here, systems are being designed that can provide a risk score from a set of characteristics about a process, using novel ML technologies.
- **ML Fraud Detector Solution:** These solutions are designed to decide on the legitimacy of a given input by making use of the knowledge of ML models that have been previously trained for this task.
- **ML Forecasting Solution:** These solutions are based on predicting future data from given past sequences. They are well suited to time series data. Currently, there is one solution already implemented for Pilot 5, which uses a Long-Short Term Memory (LSTM) model to predict future ESG scores from past scores.

- **ML Sentiment Analysis Solution:** These systems rely on process text records to extract a sentiment about them. Typically, these sentiments are positive, neutral, or negative. Three different sentiment analysis systems are currently being implemented in Pilot 5, based on widely used technologies such as VADER, NNLM or BERT encoder models (discussed in detail in Section 4.1).
- **ML High-Res Interpolator Solution:** These are systems that can accurately predict midpoints from a cloud of points using ML techniques. This allows more complex approximations (considering the context of the cloud of points) to be made than other more classical approaches such as linear, cubic or spline interpolation methods.
- **ML Anomaly Detector Solution:** These systems attempt to detect outliers from a stream of data by examining the normal behaviour of the data flow.

3.1.2 Related Work

Over the past few years, AI has become a hot topic in the scientific community in fields such as healthcare, automotive, environment, and security. However, there is little work on AI applied to EmFi use cases. In [6], Goodell et al. present a study in which several proposals are collected under the same heading of financial context, providing a good overview of the current research on AI in this area.

There are several approaches where AI can be applied in EmFi applications to bring financial resources to non-financial users. Dan et al. [7] first explored different AI-based models capable of performing text mining to enable classifying and predictive text analysis in financial applications. In [8], de Prado et al. evaluated that the accuracy of credit risk and bankruptcy assessment tasks can be improved by using a hybrid approach that combines a classical technique such as logistic regression or discriminant analysis with a neural network. Furthermore, a comprehensive analysis of the literature related to financial fraud detection tasks using deep learning techniques was presented by West et al. in [9]. The focus of this study was on the accuracy of different models in the fraud detection problem.

In recent years, research such as that presented by Al-Gasaymeh et al. in [10] shows how AI can improve the experience of non-financial users in EmFi applications through the use of decision-making processes [11, 12], where AI techniques facilitate their implementation in a way that benefits both end users and corporate management.

As the development of Natural Language Processing (NLP) algorithms has accelerated in recent years, the financial industry is demanding novel NLP models capable of analysing vast amounts of text data to gain insights and deliver better products to customers [13].

In this context, a common approach is to implement NLP models to evaluate financial text from different companies with the aim of predicting the positive or negative impact they can have on the assets of a company in question.

To do this, Transformers-based [14] encoders, such as BERT [15], are commonly used to obtain text embeddings. These embeddings are then fed into fully connected layers to predict sentiment scores related to the given financial texts.

Another example is those AI-based models that can forecast the evolution of property prices over time. For this purpose, technologies such as the LSTM [16] model or the Transformers [14] model are used.

3.1.2.1 Advancements beyond the Related Work

The development of a catalogue of AI/ML based analytical tools will enable the FAME project to offer a wide range of AI algorithms to FAME customers, enabling them to perform financial analysis on their problems. These models will cover the majority of EmFi use cases, so there will be many different trained models indexed in the FAME marketplace. Nevertheless, functionalities will be available to fine-tune and customize these models at the customer's premises.

Moreover, interactions with other components will enhance this catalogue as these models should be explainable, energy efficient and easy for customers to deploy. It is worth noting that these enhancements can be used to improve the performance of the models by using these insights in further retraining processes.

3.1.3 Technical Specification

3.1.3.1 Component-level C4 Architecture

The AI/ML analytics component brings to the FAME project the necessary functionalities to enable two main actions: *i)* providers would be able to train their own models and index them in the FDAC component, and *ii)* customers would be able to gather already trained models to perform inference on their data. In addition, these customers could access these AI analytics tools either by purchasing an AI asset that they would download and install on their premises, or by paying a subscription fee for an AI microservice that would run on the provider's premises.

As this new vision of how AI-based analytics is addressed within the FAME platform has been recently defined, it is not accurately represented in section 6.8 of D2.2 and is therefore described again in this section.

Assuming an AI model provider is interested in indexing a new EmFi AI model in the FDAC and making it available to potential customers, the AI/ML Analytics component allows the provider to do this using the trainer module, as shown in Figure 2.

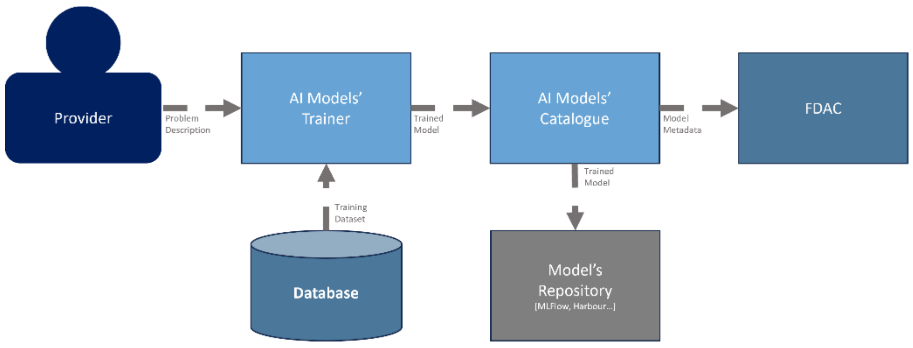


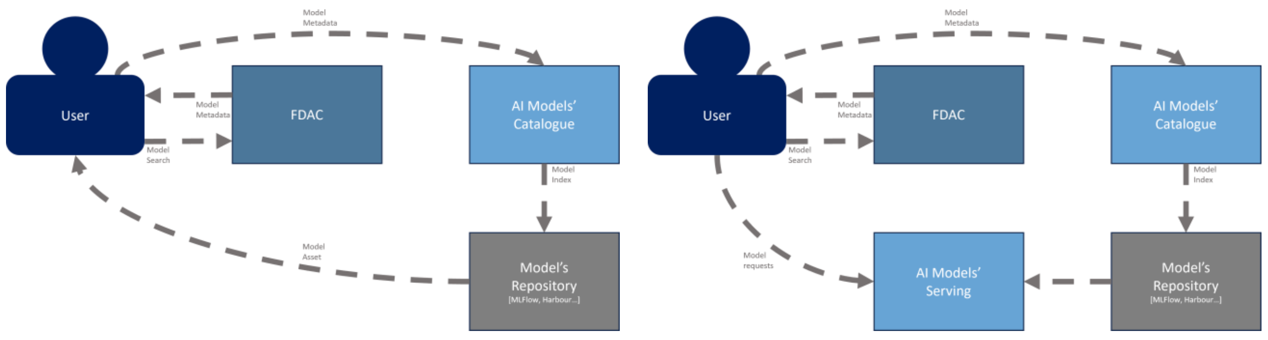
Figure 2 – AI/ML Catalogue C4 component. Model trainer module.

The Trainer Module consists of the following two sub-modules:

- **AI Models' Trainer sub-module:** It implements all the necessary functions to allow the user to train the selected model. These functions include data preparation for the model's input requirements, the model training process and the generation of logs or model evaluation, among others. These actions would be accessible through an API connection to Docker image endpoints or web microservices (this is to be decided on the provider side).
- **AI Models' Catalogue sub-module:** This block contains all the functions that allow providers generate metadata related to generated AI assets and to index it into the FDAC and the model tracking repository (e.g. MLFlow server).

When a provider properly indexes an AI asset in the FDAC, it becomes available for customers to purchase. Once this transaction is complete, there are two approaches to deploying the acquired technology:

1. **Analytics as an Asset**: According to Figure 3.a, providers would allow customers to download the asset from the repository where it resides (GitHub, Docker Hub...) and provide them with the necessary tools to easily install and run it on the customer's premises. The customer only pays for the asset provided.
2. **Analytics as a Service**: Providers would give customers an endpoint to connect to their servers to use the asset, as can be seen in Figure 3.b. It is up to the provider to maintain the AI resource and make it available to users. In this way, customers can be charged for accessing the asset (e.g. subscription fee) and/or for consuming the provider's resources (e.g. computing fee).



(a) Analytics as an Asset (b) Analytics as a Service

Figure 3 - AI/ML Catalogue C4 component. Model serving module.

According to these two approaches, two different components are identified to address the task of allowing users to use models already trained by providers:

- **AI Models' Catalogue sub-module:** This block processes the metadata of the model requested by the customer and obtains its ID within the tracking repository. This module is the core of the component an has presence in both approaches (Analytics as a Service & Analytics as an Asset)
- **AI Models' Serving sub-module:** It provides customers with a web access point to the AI services using technologies such as FastAPI, KServe or MLFlow Serving.

3.1.3.2 Baseline Technologies and Tools

The AI/ML Analytics component relies on the following core technologies to provide the tools necessary to train and deploy AI-based analytics tools.

Table 2 – Baseline Technologies and Tools for the AI/ML Analytics Component

Baseline Technology	**Description**	**Added value to FAME**
MLFlow Tracking Server[1]	The MLFlow Tracking Server is a standalone HTTP server that allows users to store and centrally serve different versions of the same model.	Tracking all the models a provider indexes in FAME allows them to continually improve the service as it is offered.

[1] https://mlflow.org/docs/latest/tracking/server.html

***Docker Engine*[2]**	Docker Engine is an open-source container technology for building and packaging applications.	This technology allows FAME to easily share AI assets or deploy AI-based services, so customers or DevOps teams don't need to have a deep understanding of AI techniques.
***Tensorflow Hub*[3]**	Tensorflow Hub is a repository of trained ML models ready for fine-tuning.	This allows large models (e.g. BERT) to be adapted and used as the backbone of a solution for EmFi use cases, whereas implementing and training these models from scratch would be costly in terms of time and computing resources.
***Tensorflow*[4]**	Tensorflow is a Python framework specifically designed for implementing AI-based solutions.	Using this framework allows AI solutions to be implemented in a way that enables parallel computing when the necessary resources are identified.

3.1.4 Interfaces

The AI/ML Analytics component will provide various API connections that will allow users to access all the functionality mentioned above. However, as some sub-modules are still under development, here is the set of available endpoints for already implemented tools when they are running:

Title	**Root interface**
Endpoint:	{HOST}
HTTP Method	GET
Description:	This interface returns a description of the general use of the AI asset, along with a list of the API interfaces available to users to interact with it.
Body Data:	None
Successful Response:	Text/plain
CURL example:	`curl -request GET "$HOST"`

Title	**Train interface**
Endpoint:	{HOST}/app/train
HTTP Method	POST
Description:	This interface trains the AI asset. To do this, it is necessary to pass an endpoint where the data is hosted (including log data if the site is secured) or a file path if the data is stored locally.
Body Data:	Raw (JSON)
Successful Response:	JSON Object with a summary of the training results

CURL example:	`curl -request POST "$HOST"/app/train \`
	 ``--data-raw `{**valid JSON schema**}` ``

[2] https://docs.docker.com/engine/
[3] https://www.tensorflow.org/hub
[4] https://www.tensorflow.org/

Title	Evaluate interface
Endpoint:	{HOST}/app/evaluate
HTTP Method	POST
Description:	This interface evaluates the AI asset. To do this, it is necessary to pass an endpoint where the data is hosted (including log data if the site is secured) or a file path if the data is stored locally.
Body Data:	Raw (JSON)
Successful Response:	JSON Object with a summary of the evaluation results
CURL example:	`curl -request POST "$HOST"/app/evaluate \` `--data-raw `{**valid JSON schema**}``

Title	Inference interface
Endpoint:	{HOST}/app/infer
HTTP Method	POST
Description:	This interface performs inferences over provided data. To do this, it is necessary to pass data in proper format.
Body Data:	Raw (JSON)
Successful Response:	JSON Object with a summary of the training results
CURL example:	`curl -request POST "$HOST"/app/infer \` `--data-raw `{**valid JSON schema**}``

Title	Indexing interface
Endpoint:	{HOST}/app/index
HTTP Method	POST
Description:	This interface index provided model's metadata into the FDAC component.
Body Data:	Raw (JSON)
Successful Response:	Text/plain
CURL example:	`curl -request POST "$HOST"/app/index \` `--data-raw `{**valid JSON schema**}``

3.2 SAX techniques

3.2.1 Description

The use of machine learning (ML) models has become widespread in recent years [19, 20, 21]. Applications of machine learning include domains that could significantly impact individuals and society, such as medical diagnostics, credit scoring, recidivism prediction, and autonomous driving. However, advancements in the field have led to increasingly complex models that can be difficult to interpret, resulting in 'black box' models that hinder their full adoption. Therefore, we have also witnessed a rise in the necessity to clarify ML models and their predictions. This is driven in part by legislation, as well as incentives from the user's or stakeholder's perspective (e.g. to justify the ML model and gain domain insight), and from the developer's point of view (e.g. to evaluate and enhance the ML model). In order for users and operators to adopt AI-powered augmented systems, they must have confidence in the technology. This trust can be established by providing a clear explanation of the models' operation [22].

While some machine learning models, such as decision trees and linear models, are inherently explainable and their internal logic predictions can be easily understood by humans, more complex models like ensemble models and deep learning models require external explanation frameworks, such as eXplainable AI (XAI), to be human-understandable [23, 24].

eXplainable AI (XAI) refers to the ability to comprehend and interpret how AI systems arrive at decisions or conclusions. These frameworks are primarily designed for post-hoc interpretations of ML models [19, 20]. Contextually, they can be categorized into global, local, and hybrid explanations [20, 25, 26]. Global explanations aim to clarify the internal logic of the ML model, while local explanations aim to explain the ML model's prediction for a single input instance. Hybrid approaches may vary, such as explaining the ML model's internal logic for a subspace of the input space.

Contemporary XAI techniques, such as LIME [27] or SHAP [28], use a surrogate model that is trained using historical process execution logs of the BP. The predicted value for a single instance, which represents the process outcome, serves as input for the XAI explainer to produce an explanation.

Our focus in SAX is on improving the explainability of business processes. In essence, a business process (BP) is a series of tasks performed in a specific order to achieve a particular business objective [25]. However, current XAI techniques are often insufficient for providing accurate and reliable explanations when applied to BPs, as they frequently struggle to:

- Express the business process model constraints (i.e., the semantics of the process model),
- Include the richness of contextual situations that affect process outcomes (additional information that affects the outcome but is usually not modelled),
- Reflect the true causal execution dependencies among the activities in the business process, or
- Make sense and be interpretable to process users (explanations are usually not given in a human-interpretable form that can ease the understanding by humans).

To address the above shortcomings, we introduce ***Situation-aware eXplainability (SAX)*** as a framework for generating explanations.

A 'SAX explanation' is a causally sound explanation that considers the process context in which a condition occurred (the 'explanandum'). The explanation should provide an account of why the condition occurred in a faithful and logical entailment that reflects the genuine chain of business process executions that led to the condition. Context includes knowledge elements that were originally implicit or part of the surrounding system, yet affected the choices made during process execution.

One method of automating explanations is to utilise the capabilities of Large Language Models (LLMs). According to a recent report by Gartner [29], LLMs are being implemented in all industries and business functions, facilitating various use cases such as text summarisation, question-answering, document translation, and more. LLMs can also be enhanced with additional capabilities to create more robust systems and feature a growing ecosystem of tools.

Large language models (LLMs) are deep-learning (DL) models trained on vast amounts of text data to perform natural language processing tasks. They excel at few-shot and zero-shot learning using prompt-based learning [30]. The emergence of LLMs has led to the exploration of 'prompt engineering' as a new research field. This field focuses on designing, refining, and implementing instructions to guide the output of LLMs in various tasks. The goal is to achieve a more concise and effective utilization of LLMs, adapting them to the users' skills and the context of the tasks, such as analysis or reasoning [31]. This may involve various techniques, such as using one-shot or few-shot samples, annotating different parts of a prompt with quotes, employing advanced methodologies like 'Chain of Thought' (CoT) to evolve the interaction methodologically, and configuring LLM-specific settings like temperature and top-p sampling to control output variability.

The SAX framework is implemented through a set of services in the SAX4BPM library that assist with automatically deriving SAX explanations using existing LLMs. The library generates SAX explanations for processes using event logs as the primary input. An event log is a multi-set of traces, where a trace is a digital footprint that depicts a single execution of a process as a concrete sequence of activities [24].

To better adapt to the use case requirements in FAME, the SAX framework has been extended to include text and time series data as input, instead of event logs executions from processes. With regards to text, we extended the original framework to leverage LLMs for the explanations of sentiment analysis as explained next.

Text classification is a common functionality that is needed to drive a variety of contemporary applications such as document topic categorization for archiving, automatic detection of spam emails, determination of customer intents, identification of fake news, and filtering of abusive content. One concrete application of textual classification is for sentiment analysis.

Sentiment Analysis (SA) maps from the underlying input text onto higher-level sentiment lexicons that depict the contextual orientation of the text. Customer level of satisfaction post-acquisition of a product or a service, reviews about social media posts, prediction of stock prices based on public media advertisements, or wellness tracking of individuals, these are all examples where sentiment analysis may provide valuable feedback about the trending collective emotional attitude and public opinion held by any population of interest towards any topic, product, or service. As a field of Natural Language Processing (NLP), a variety of techniques may be employed nowadays to enable SA, ranging from basic rule-based methods to ML and deep learning-based approaches, that typically involve model training based on corpuses of textual narratives that are labelled by sentiment data.

While the fundamental anticipation in SAX is having its data input in a form of a **process log** that records execution of process activities, many real-world scenarios record their surrounding conditions via a different type of data recording, one that is typically referred to as **time series data**. In this type of scenarios, the density of the data stems from the sampling frequency of the sensors or the rate of advertising, while there is no inherent a-priori partitioning of the data into higher level, domain meaningful events. Time series data in its simplest single-variate form captures a single recording value for each data point. In non-numeric context, the data may capture a series of timestamped labels (e.g., sentiment values). A more advanced form is a multivariate time series in which each timestamp captures a set (vector) of recorded values or labels. Given time series data as an input, various techniques could be employed to transform it into a data form that corresponds to the form of a process event log. Such transformations employ different algorithmic approaches for the segmentation of the raw time series data into some higher-level partitioning, marked by a finite set of labels, that would usually correspond to some domain meaningful phases along which the system traverses. Such partitioning could sometimes arise naturally from the timely distribution and clustering of the underlying data, while in certain cases, it may also require some understanding of the source domain. Respectively, our implementation of SAX also includes a service to cope with such data partitioning (see Section 3.2.3). One potential technique to be leveraged to this end is the StreamStory tool developed by partner JSI and described next. The status of our current experiments applying StreamStory to the motor oil use case in the project are reported in Section 4.2. The next version of this deliverable will include further developments in this direction.

StreamStory is an interactive web-based tool developed for the visualization, exploration, and interpretation of large multivariate time series data. It represents data by abstracting the continuous flow of data into discrete states and transitions, effectively summarizing complex temporal dynamics in a graph-based visual format. These states are not mere statistical aggregations but conceptual representations of the underlying phenomena, allowing for the identification of cyclic patterns and recurrent behaviours across different scales. The methodology underlying StreamStory is built on hierarchical Markov chains, where each state represents a cluster of similar data points in the time series, and transitions indicate the likelihood of moving from one state to another over time. This approach reduces clutter and simplifies the dynamics of the data, making it easier for users to discern patterns and relationships. One of the key features of StreamStory is its multiscale exploration capability. It allows users to navigate through different levels of data granularity, from finer to coarser representations, uncovering patterns that are not immediately apparent at a single scale. This is achieved by merging or splitting states based on the scale of interest, facilitating the identification of both high-level trends and detailed behaviours. StreamStory is designed with a user-friendly interface that requires minimal domain-specific knowledge or analytical expertise, making it accessible to a broad audience. It includes several auxiliary tools and visual cues that help map the abstract representations back to domain-specific concepts, enhancing interpretability. The tool supports tasks such as identifying the main states of the system, finding long-term periodic behaviour, and exploring data at multiple scales to uncover interesting phenomena. The system's architecture ensures scalability and efficiency, enabling it to handle datasets with potentially billions of examples without significant performance degradation. StreamStory's visualization and exploration tools leverage statistical and machine learning techniques to provide insights into the data, offering automatically generated suggestions for possible interpretations of the identified patterns.

3.2.2 Related Work

As aforementioned, the SAX framework has been extended in the scope of FAME to include text and time series data as input. Therefore, we relate our work to the state-of-the-art literature in the domain of explainability for sentiment analysis and techniques for time series data. Note, that although we intend to leverage techniques for converting time series data into some form of process event logs, segmentation of time series data into clustering of higher-level concepts may itself be considered as a type of explainability.

With the appearance of models that learn the patterns and structure of their input training data and then generate new data having similar characteristics (Generative AI) and transformer based LLMs, such as BERT [15] and GPT [32], trained on diverse text data for a variety of tasks, LLMs have shown also success in sentiment analysis. In fact, a model such as BERT manages to outperform the norm on a range of general language understanding tasks, including sentiment analysis [33]. With the use of LLMs, this performance comes at the cost of slow training due to model size and the number of weights to update. Similarly, the work in [34] examines the performance of GPT-3.5, which also demonstrates significance superiority of the LLM compared to state-of-the-art methods. Concretely in our work, we are extending this capacity further to employ LLMs for explainability of another ML model that is used for SA. Also, in [34], LLMs are being considered recently as a powerful alternative to using the more conventional post-hoc XAI tools such as SHAP or LIME for black box explainability of complex ML models such as deep learning. A recent attempt also assessed the use of LLM themselves for sentiment elicitation from financial texts, showing 35% enhanced performance in sentiment classification, and a similar increase in correlation with market returns [35].

In our own recent work [36], we demonstrated how LLMs may be employed for the explainability of business process models. While the use of the existing XAI tools may be very tedious to nonexpert users, LLMs can be also employed to mitigate the complexity of their usage and to simplify the output to match user expertise as concluded in [37]. In fact, as articulated in [38], LLMs themselves are not only suitable for a wide range of applications but also present a complex inner architecture that can benefit from being accompanied by LLM empowered explainability to facilitate a better understanding of their results and to increase trust in their usage. A recent survey in [39] suggests a variety of techniques for the explainability of LLM results based on the training paradigms of LLMs, distinguishing between local explanations concerning how a model predicts for a specific input, and global explanations regarding general model behaviour.

Probably the closest to our scheme developed here for methodologically using LLMs to explain sentiment analysis results is the work in [40] where LLMs are being employed for generating counterfactual explanations. It presents a systematic approach to identifying important features in the input text and alters a minimal set of text elements to change classification outcomes.

Following the use of LLMs for explaining SA results, their application can be further extended to ensure the quality of the generated explanations. While the interpretability of explanations for specific users can only be verified through empirical assessment, as pursued in [36], LLMs can also assist in automating the fidelity check of explanations. As indicated in [19], LLMs can contribute to the factuality evaluation of a given summary to assess its faithfulness.

With regards to time series data, techniques for visualizing time series beyond simple graphs often cater to specific domains with general overviews (see e.g., [41] and [42]). Shneiderman et al. [43] introduced TimeSearcher, facilitating the visualization of multiple time series via a tool that employs TimeBoxes for interactive filtering. Despite its comprehensive nature, its tabbed interface may overwhelm users and obscure pattern recognition. Havre et al. [44] and others have explored visualizing thematic variations and discrete events over time, yet these methods struggle with visual clutter and the identification of recurrent behaviours, especially in large datasets.

Peng et al. [45] and Wang et al. [46] presented approaches for high-dimensional and discrete event data visualization, respectively, with a focus on clarity in transition representation and temporal summarization. However, these methods often falter when it comes to revealing the underlying data structure and managing clutter in extensive datasets. Haroz et al. [47] and Bach et al. [48] introduced connected scatterplots and TimeCurves, respectively, for visualizing temporal dynamics, but their effectiveness diminishes with data scale due to clutter.

Recent innovations include static graph-based techniques [49], [50] for temporal data visualization, employing directed graphs and Multidimensional scaling (MDS) for layout. These approaches aim at modelling data trajectories through ambient space, contrasting with earlier methods that often lacked in conveying temporal information or became cluttered with large data volumes. Our work builds on these concepts, using hierarchical Markov chains for a scalable and interpretable visualization of temporal dynamics, enhanced by auxiliary widgets and automatic state labels for user guidance.

3.2.2.1 Advancements beyond the Related Work

Our core innovation in SAX is in the post-hoc methodological approach for leveraging an LLM to identify the least set(s) of K-terms (sufficient) that retains the original classification of the SA model as a novel approach (inspired by explanation sufficiency). As opposed to state-of-the-art work in this direction [51], our approach leverages the inherent capacity of LLMs to infer good candidates for such subsets based on their inherent generative power based on the richness of underlying corpuses used in their development, as opposed to employing some algorithmic convergence approach as in [51]. The *SAX LLMSentimentXplain* service (see next section) is deployed for integration with the ML model that is used for SA, complementing it with the elicitation of a sufficient explaining set of terms in a method that is independent of the concrete ML model that is used for SA (see demo explanation in Section 4.2).

3.2.3 Technical Specification

3.2.3.1 Component-level C4 Architecture

In this deliverable, we elaborate further on the details of the core architecture presented in D2.2 (figure 20, FAME SA C4 container diagram). Particularly here, we present the inter-container dependencies and the corresponding inner components corresponding to the SAX techniques container. Overall, the "SAX techniques" at the highest level is instantiated by a subset of containers, mainly splitting between "XAI scoring" and "SAX4xxx" as depicted in D2.2. Here, this design is elaborated (separately) with the details of each. At the context level, SAX techniques enable explainability of the ML predictions or classification, and respectively require interaction with any relevant "ML pipelines" as shown in Figure 4. The data underlying such explainability (and ML) is also accessible to SAX techniques via the relationship with "Data access". The realization of explainability also leverages external services such as LLMs, where such access is depicted via the relationship to "External Services".

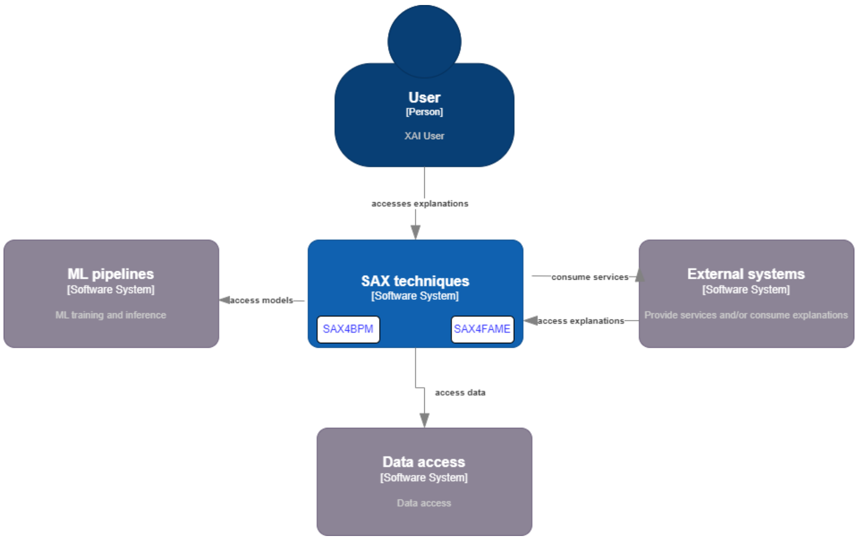


Figure 4 - SAX techniques C4 context diagram.

As presented in Figure 4 and Figure 5, to date SAX techniques are realized in two façade containers: "*SAX4BPM*" which consolidates a set of fundamental functionalities, and "*SAX4FAME*" that extends these core functionalities with project specific services. In both realizations, the functionality is exposed by an API component, which is also wrapped by an UI one to facilitate interaction with the users. Further to this, in the context of the core functionalities, "Situational Context" drives any needed utilization of a CEP engine for context enrichment of the data logs, accessed via the "Data access" container, leverages existing XAI libraries using the "XAI" container which also has access to machine learning models via the "ML access" container. For the additional functionality in "*SAX4FAME*", we also included the "Large Language Models" container to enable access to LLM models (see Figure 6).

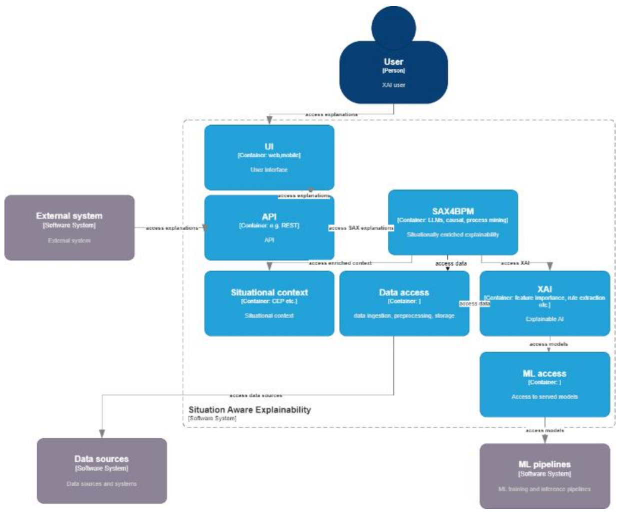


Figure 5 - SAX4BPM C4 container diagram.

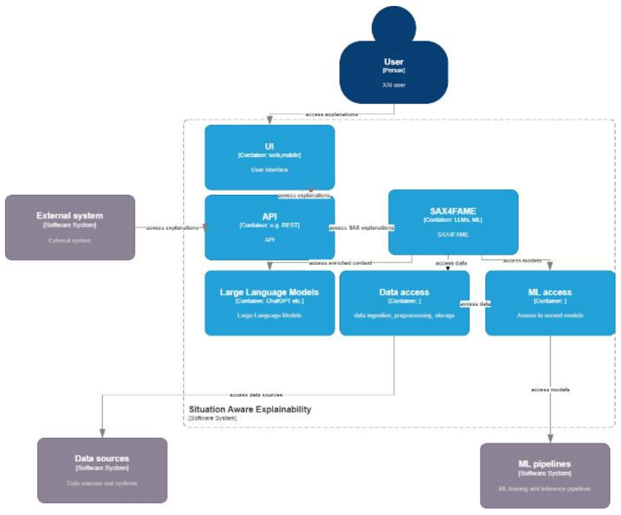


Figure 6 - SAX4FAME C4 container diagram.

With respect to the main two containers of *SAX4BPM* and *SAX4FAME*, we further detail here their corresponding partitioning into underlying components, each related to a more concrete service type. The *SAX4BPM* library currently supports the following services (Figure 7):

- ***Mining4Process*** – This service is aimed at generating the knowledge component of the process model. It takes a given event log as input and uses an external process discovery algorithm (like heuristic miner). The resulting process model is then exported in JavaScript Object Notation (JSON) format and used as input for the LLM.
- ***Causal4Process*** – This service aims to provide the knowledge component for the causal process model. A novel algorithmic approach [53, 54] is used to extract the causal process model from the event log and process model. This algorithm builds on an existing causal discovery algorithm (LiNGAM) and is adapted to account for process execution times. The resulting graph model is provided in JSON format as input to the LLM.
- ***ContextEnrichment*** – The aim of this service is to enhance the event log by adding contextual information related to the process and its environment during execution. For instance, our experiments with complex event processing [55] have demonstrated that incorporating temporal contextual information can enhance the accuracy of explanations provided for process execution instances. The enrichment has a dual effect: it adjusts the importance of features that correspond to the outcome and promotes the accuracy of the surrogate ML model. This service applies situational rules to enrich the initial event log, resulting in an enriched event log that serves as new input.
- ***X4Process*** – The aim of this service is two-fold. Firstly, it generates and determines the importance ranking of the features used in predicting the condition of interest while adhering to the process model's constraints. Secondly, it utilizes the ContextEnrichment service to supplement the event log with broader, contextually relevant attributes related to the same condition. This service calls the ContextEnrichment service from the library. It then retrieves the enriched event log and process model, which include constraints and flow control logic. These are used to limit the search space of an invoked XAI service (such as SHAP or LIME) to ensure compliance with the process [56]. The output is a feature importance vector segmented according to the activities in the process, which is exported in JSON format as input to the LLM.
- ***NLP4X*** – The aim of this service is to integrate different pieces of knowledge and serve as a front-end for user interaction, leading to an LLM. It takes the query (the user's search condition) and the selection of knowledge ingredients as input. The service's main function is

to synthesize input ingredients by invoking corresponding SAX4BPM services and streamlining them via the LLM prompt to elicit an explanation narrative. Although the current implementation uses predetermined leading phrases to facilitate interaction, future work aims to automate this step using dynamic templating. The library currently uses ChatGPT3 from OpenAI as the LLM.

- ***Time4process*** – This service aims at converting time series data into an event table form that can substitute an event log input. This service was not part of the original architecture. However, as the data received from the use cases does not necessarily come from processes, we started to look into ways to still exploit this data. The transformation from time series into an event log relies most fundamentally on the ability to unite the raw timestamped data into segments, each of which reflects some higher-level phase in the system's process. Such segmentation may be accomplished in various ways, and could sometimes be automated, emerging from the density and distribution properties of the raw measurements themselves. The Time4process service attempts to employ a few semi-automated and fully automated approaches to address the pragmatic need to translate raw time series data into a form of a higher-level phase-transition graph as a basis for translating raw time series data into corresponding process event logs. From an explainability perspective, this translation lets us continue focusing on SAX, where explanations are about the need to populate clarifications about concrete conditions that occur during or after process executions. For example, in the context of the ESG (environment, social, and governance) pilot, we can explore explanations about predicted ESG values in relation to input sequences (i.e., processes) of sentiments.

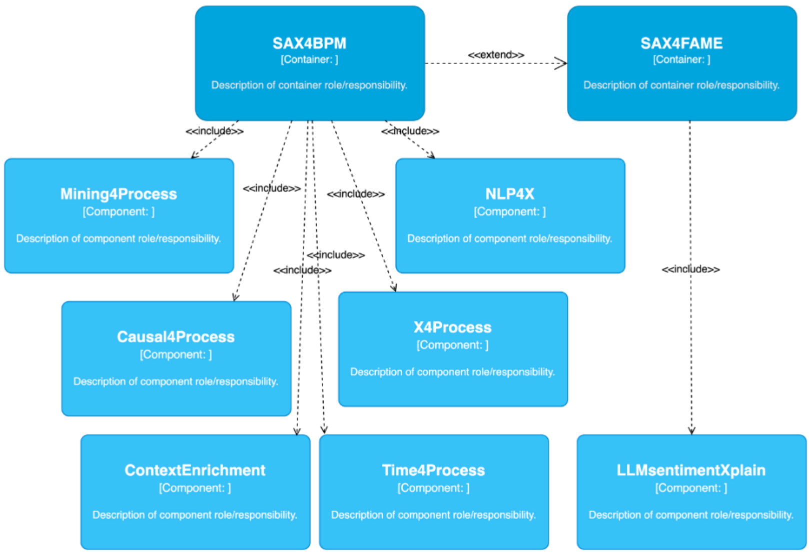

SAX4BPM
[Container:]
Description of container role/responsibility.
<<extend>>
SAX4FAME
[Container:]
Description of container role/responsibility.
<<include>>
Mining4Process
[Component:]
Description of component role/responsibility.
NLP4X
[Component:]
Description of component role/responsibility.
Causal4Process
[Component:]
Description of component role/responsibility.
X4Process
[Component:]
Description of component role/responsibility.
ContextEnrichment
[Component:]
Description of component role/responsibility.
Time4Process
[Component:]
Description of component role/responsibility.
LLMsentimentXplain
[Component:]
Description of component role/responsibility.

Figure 7 - SAX4BPM and SAX4FAME components.

The SAX4FAME library currently supports the following service:

- ***LLMSentimentXplain*** – This service has been added to the *SAX4BPM* library to derive explanations for sentiment analysis. The goal of the service is to identify a subset of words within a given textual narrative that best associate with the sentiment classification of the narrative as concluded by any machine learning model. The purpose is to determine the terms that most affected the prediction made by the machine learning model, while remaining agnostic to both its internal structure and the data that was used for its training (i.e., post-hoc). As such, the implementation of the service is also coupled with some ongoing interaction with the ML model, initiating the overall process of interaction with the LLM without any prior knowledge (i.e., zero-shot).

The *SAX4BPM* library along with the *SAX4FAME* extension will be released as open source at the end of the project.

3.2.3.2 Baseline Technologies and Tools

The realization of the SAX Techniques is implemented as Python libraries and relies on the following baseline technologies (either by providing wrappers for open-source libraries or by invoking external services):

Table 3 – Baseline Technologies and Tools

Baseline Technology	**Description**	**Added value to FAME**
Process Mining - pm4py 2.7.9.4	Core algorithms for discovery of process models over input event traces, utilizing Heuristic and Alpha miner. https://pypi.org/project/pm4py/	The process model is part of the input for SAX4BPM to determine temporal relationships among process attributes.
Complex Event Processing – IBM PROactive Technology ONline (PROTON)	An open-source (Apache 2.0) complex event processing engine developed at IBM Research Israel. https://github.com/ishkin/Proton	The CEP engine is employed in SAX4BPM for process context enrichment. The event log is streamlined via the engine to enrich it with content derived events that are then used as an additional input for better process explainability.
Causal Discovery - LiNGAM	An open-source (MIT licence) library for the discovery of non-gaussian linear causal models. https://github.com/cdt15/lingam	Causal discovery is employed in SAX4BPM to determine causal execution dependencies among process activities as an input for explanation fidelity and feature stratification.
Large Language Models – e.g., GPT 4.0	A set of generative AI models by openAI capable of performing various natural language processing tasks. https://openai.com/gpt-4	LLMs are employed in SAX4BPM for the synthesis of multiple knowledge inputs, generation of interpretable explanations, and post-hoc explainability of ML results.
AIX360 (e.g., SHAP, LIME, and TSlime)	An open-source (Apache 2.0) XAI library that supports interpretability and explainability of datasets and machine learning models and of time series data. https://github.com/Trusted-AI/AIX360	AIX360 is employed in SAX4BPM for the derivation of feature importance in relation to particular process conditions for which an explanation is generated and corresponding visualizations (see example, Figure 8 and Figure 9 below)
Lime-4-Time	Application of the LIME algorithm for time series classification. https://github.com/emanuel-metzenthin/Lime-For-Time	Another alternative to using AIX360 for explainability of time series data.
StreamStory	Analysis tool for multivariate continuously time-varying data streams.	Segmentation of time series data to transform raw data into higher level process data.

	https://github.com/E3-JSI/StreamStory2

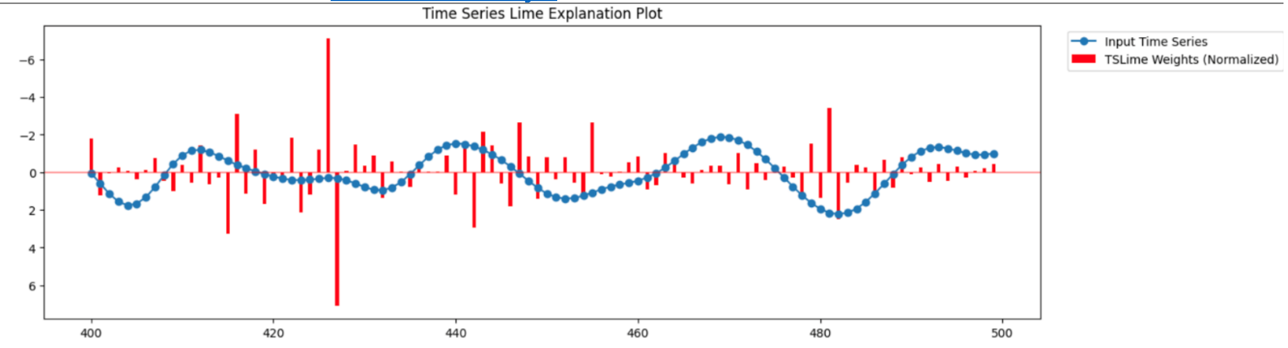


Figure 8 - TSlime baseline example for visualization of time series feature importance.

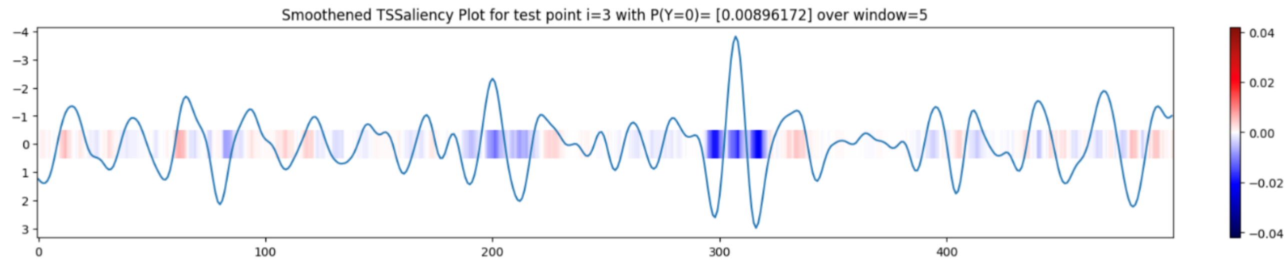


Figure 9 - TSSaliency baseline example.

By exploiting all these technologies, the developed component will offer FAME capabilities of situation awareness, causal process discovery, and process mining, over given event logs. SAX code will be developed as a set of components and services released as the SAX library to the open-source community by the end of the project.

3.2.4 Interfaces

Currently, the SAX sentiment analysis component provides UI integration as the interface for component invocation (See details in Section 4.2).

Code 1 - API for SreamStory.

```
// buildModel Function
{
  "dataSource": {
    "type": "file | internal",
    "format": "json | csv",
    "fileName": "string (conditionally required if type == 'file')",
    "data": "string | array (conditionally required if type == 'internal')"
  },
  "config": {
    "numInitialStates": "int",
    "numHistogramBuckets": "int (default: 10)",
    "attributes": [
      {
        "name": "string",
        "source": "input | synthetic (default: 'input')",
        "sourceName": "string (optional)",
        "label": "string (optional)",
        "distWeight": "float (optional)",
        "type": "time | numeric | categorical | text",
        "subType": "string | float | int (default depends on type)",
        "timeType": "time | float | int (optional, only if type is 'time')"
      }
    ],
    "ops": [
      {
        "op": "timeShift | timeDelta | linTrend",
        "inAttr": "string",
        "outAttr": "string",
        "windowUnit": "samples | numeric | sec | min | hour | day",
        "windowSize": "float | int",
        "timeAttr": "string (conditional)"
      }
    ],
    "decTree_maxDepth": "int (default: 3)",
    "decTree_minEntropyToSplit": "float (default value based on numInitialStates)",
    "decTree_minNormInfGainToSplit": "float (default: 0)",
    "ignoreConversionErrors": "boolean (default: true)",
    "distWeightOutliers": "float (default: 0.05)",
    "includeHistograms": "boolean (default: true)",
    "includeDecisionTrees": "boolean (default: true)",
    "includeStateHistory": "boolean (default: true)"
  }
}
// Output JSON Structure
{
    "status": "ok | error",
    "errors": ["string"],
    "model": {
      "scales": [
        {
          "nStates": "int",
          "areTheseInitialStates": "boolean",
          "states": [
            {
              "stateNo": "int",
              "initialStates": ["int"],
              "childStates": ["int"],
              "parentState": "int (optional)",
              "sameAsParent": "boolean",
              "centroid": [
                {
                  "attrName": "string",
                  "value": "mixed"
                }
              ],
              "stationaryProbability": "float",
              "nMembers": "int",
              "nextStateProbDistr": ["float"],
              "histograms": [
                {
                  "attrName": "string",
                  "freqs": ["int"],
                  "freqSum": "int",
                  "bounds": ["float"],
                  "keys": ["mixed"],
                  "dayOfWeekFreqs": ["int"],
                  "monthFreqs": ["int"],
                  "hourFreqs": ["int"]
                }
              ],
              "xCenter": "float",
              "yCenter": "float",
              "radius": "float",
              "suggestedLabel": {
                "label": "string",
                "nCoveredInState": "int",
                "nNotCoveredInState": "int",
                "nCoveredOutsideState": "int",
                "nNotCoveredOutsideState": "int",
                "logOddsRatio": "float"
              },
              "decisionTree": {
                "nPos": "int",
                "nNeg": "int",
                "entropyBeforeSplit": "float",
                "splitCost": "float",
                "entropyAfterSplit": "float",
                "infGain": "float",
                "normInfGain": "float",
                "splitAttr": "string",
                "splitLabel": "string",
                "children": ["recursive definition of this object"]
              }
            }
          ]
        }
      ],
      "totalHistograms": [
        {
          "attrName": "string",
          "freqs": ["int"],
          "freqSum": "int",
          "bounds": ["float"],
          "keys": ["mixed"],
          "dayOfWeekFreqs": ["int"],
          "monthFreqs": ["int"],
          "hourFreqs": ["int"]
        }
      ],
      "stateHistoryTimes": ["float"],
      "stateHistoryInitialStates": ["int"]
    }
  }
// classifySamples Function
{
    "dataSource": {
      "type": "file | internal",
      "format": "json | csv",
      "fileName": "string (conditionally required if type == 'file')",
      "data": "string | array (conditionally required if type == 'internal')"
    },
    "model": "The entire model structure as returned by the buildModel function"
  }
// Output JSON Structure
{
    "status": "ok | error",
    "classifications": ["int"],
    "errors": ["string"]
  }
```

3.3 XAI scoring framework.

3.3.1 Description

The XAI Scoring Framework stands out as a tool for assessing and quantifying the explainability of various XAI models, AI systems, and data assets. It is designed to bridge the gap between complex AI technologies and end-user comprehension, allowing for a systematic evaluation of how transparent and understandable these technologies are. This framework enables users to select different AI approaches based on their explainability, fostering trust and ensuring that AI solutions are not only technically robust but also user-friendly and accessible. By focusing on user experience and providing a trust gauge through understandable AI decision-making processes, the framework aims to play a crucial role in benchmarking AI technologies, allowing users to make informed decisions about which models align best with their needs. Additionally, the framework assists in accurately pricing AI data assets on platforms like FAME, ensuring that valuations reflect the true utility and explainability of these assets, thus enhancing their market relevance and user trust.

3.3.2 Related Work

Previous research on evaluating XAI methods has focused on various properties, such as fidelity, consistency, stability, comprehensibility, and certainty [27] [28]. However, most existing evaluation frameworks are tailored to specific explanation methods or rely on subjective human judgments, which can vary based on factors like domain expertise and cognitive biases [57] [58]. Consequently, there is a pressing need for a unified multidimensional explainability metric that objectively quantifies the trustworthiness of different XAI models, facilitating their comparison and selection in real-world applications [59] [60].

Recent developments have introduced various frameworks and tools designed to enhance the explainability of AI models and explainable AI (XAI) techniques, including AI Explainability 360, Alibi, and InterpretML. These resources offer guidelines and implementations for a range of explainability methods. Despite these advances, there remains an unmet need for an explainability metric or score that effectively evaluates the trustworthiness and effectiveness of XAI techniques across different scenarios. Our proposed approach seeks to fill this gap by combining quantitative, human-centric evaluation metrics with an understanding of the contextual nuances of XAI algorithms and the explanations they generate, aiming to establish a comprehensive explainability score.

Regarding the quantification and assessment of explainability, several research efforts have adopted varied methodologies. Reference [59] introduced a quantitative framework for interpretability, based on functional decomposition, facilitating the comparison of interpretability scores across models and enhancing our quantitative understanding of interpretability. Contrastingly,a study [57] investigated the human comprehension of machine-generated explanations, yielding insights into the interpretability of various methods from a human-centric viewpoint, thereby underscoring the importance of incorporating human cognition and perception in explainability evaluations. Reference [27] examined explainability in the context of diabetic retinopathy diagnosis through deep learning, using a user study with medical professionals to evaluate the impact of explanations on user performance and system trust, highlighting the practical value of explainability in critical decision-making settings. Moreover, research [61] underscored the significance of not only generating but also effectively communicating explanations, suggesting the application of Information Theory to measure the information conveyed, thus emphasizing communication's role in explainability. Recent initiatives, as mentioned in [62], propose metrics focusing on user-centred aspects such as satisfaction, mental models, curiosity, trust, and the efficacy of human-AI collaboration, offering a broader perspective on explainability that incorporates technical, communicative, and user-focused elements. The XAI Scoring Framework aims to benchmark XAI techniques against a comprehensive set of criteria, integrating these novel perspectives into our ongoing research efforts.

Another critical dimension of explainability evaluation involves comparing various XAI methods. Study [63] explored how different types of explanations affect users' decision-making, comparing model-generated explanations, human-generated explanations, and contrasting cases. Their findings shed light on the complex relationship between explanation types and user behaviour, calling for additional research in this domain. These studies collectively enhance our understanding of explanation methods' strengths and weaknesses, contributing to the development of more effective and comprehensive explainability metrics for XAI models.

3.3.2.1 Challenges in Explainability Quantification

Quantifying explainability poses significant challenges due to its inherently subjective nature and the wide array of stakeholders involved in the evaluation process. Individual differences in preferences and understandings of explanations complicate the creation of a universally applicable metric. Incorporating user feedback into the evaluation, potentially through active learning as suggested in [64], may offer a way to tailor the metric to individual needs and preferences. The relevance of various explainability metrics can vary depending on the context and objectives of the XAI technique. Factors such as the fidelity and stability of explanations may change based on the dataset and model, while the understandability of an explanation could be influenced by the user's domain expertise. These contextual dependencies challenge the development of a unified, multidimensional explainability metric applicable across various tasks, domains, and stakeholder groups [60]. Trade-offs between simplicity, fidelity, and coverage must be navigated; for example, highly detailed explanations may be more complex and less accessible to users, whereas simpler explanations may compromise on detail for greater comprehensibility. Finding the balance among these factors needs consideration of the specific aims and limitations of the XAI approach and the intended audience [65].

3.3.2.2 Advancements beyond the Related Work

The presented solution elevates the evaluation of XAI models by integrating user satisfaction alongside quantitative metrics, acknowledging the critical role of human interaction in AI systems. This holistic approach ensures that models are not only technically sound but also resonate with user expectations and experiences, marking a significant advancement over traditional evaluation methods that may overlook the human element.

Furthermore, by being model agnostic, the XAI Scoring framework sets a new benchmark in the field, enabling the assessment of a wide range of AI models without bias towards specific architectures or domains. This flexibility ensures broad applicability and fosters innovation in AI development, pushing the boundaries of what's considered state-of-the-art in XAI evaluation.

3.3.3 Technical Specification

3.3.3.1 Component-level C4 Architecture

XAI Scoring framework delivers an extensive mechanism for evaluating the explainability of various XAI techniques and AI-related data resources, applicable to numerous ML models and XAI strategies, with the overarching aim of establishing a comprehensive, multidimensional explainability score. The proposed methodology will outline a scoring system that accounts for characteristics of XAI algorithms like fidelity, stability, simplicity, and comprehensiveness, while also incorporating elements centred on the user experience, such as satisfaction, trust, and efficiency in task execution. It aims to provide a thorough evaluation by including performance indicators like accuracy, precision, and recall, ensuring that the benchmarks for explainability do not compromise the predictive accuracy of the models. This framework is designed to produce explainability scores that can be compared with assessments made by human experts, offering a detailed and validated measure of explainability. These features are depicted in the third layer of the C4 diagram (specifically, the component diagram) shown in Figure 10.

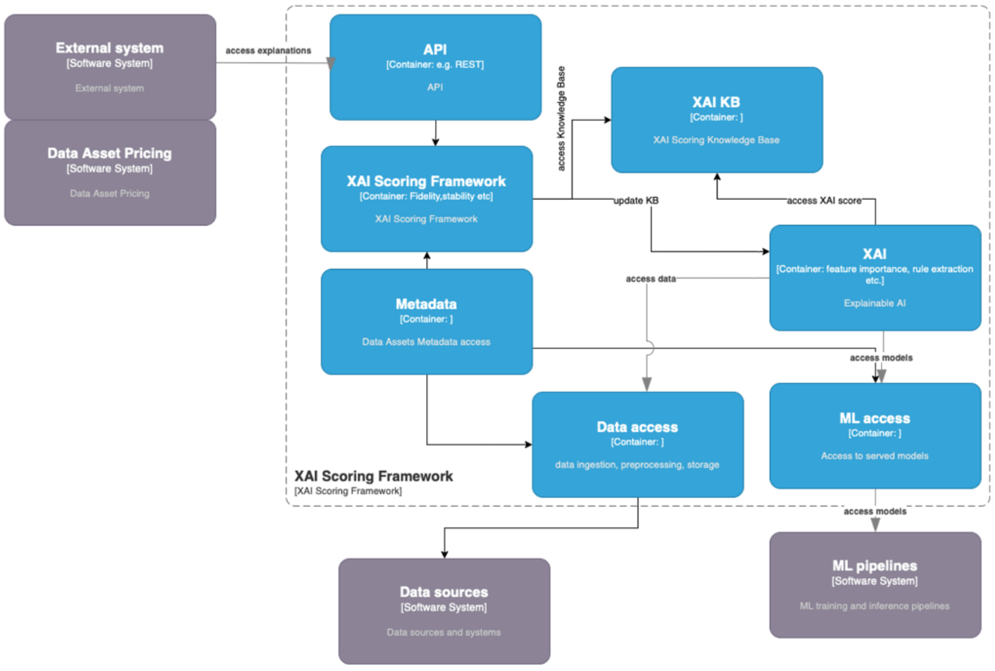


Figure 10 - XAI Scoring Framework component-level architecture.

We create a knowledge base for assessing the explainability score of various types of data assets. The knowledge base is constructed by aggregating the benchmarking results and metadata from AI/ML models, XAI models, and datasets, and then deriving rules, heuristics, or algorithms for estimating the XAI score of a given model (asset) based on its metadata.

In summary, our methodology for benchmarking XAI techniques and creating a knowledge base for assessing the explainability score of each model involves the steps depicted in Figure 11.

Conduct benchmarking experiments on a diverse set of AI/ML models, XAI techniques, and datasets.

- Evaluate the explanations using the XAI quantification scheme and compute the XAI score for each combination of model, dataset, and XAI technique.
- Analyse the benchmarking results to identify trends, patterns, and relationships between the properties of AI/ML models, XAI techniques, and datasets.
- Create a knowledge base for assessing the explainability score, based on the benchmarking results and metadata from AI/ML models, XAI models, and datasets.
- Periodically update the knowledge base with new benchmarking results to ensure its relevance and accuracy.

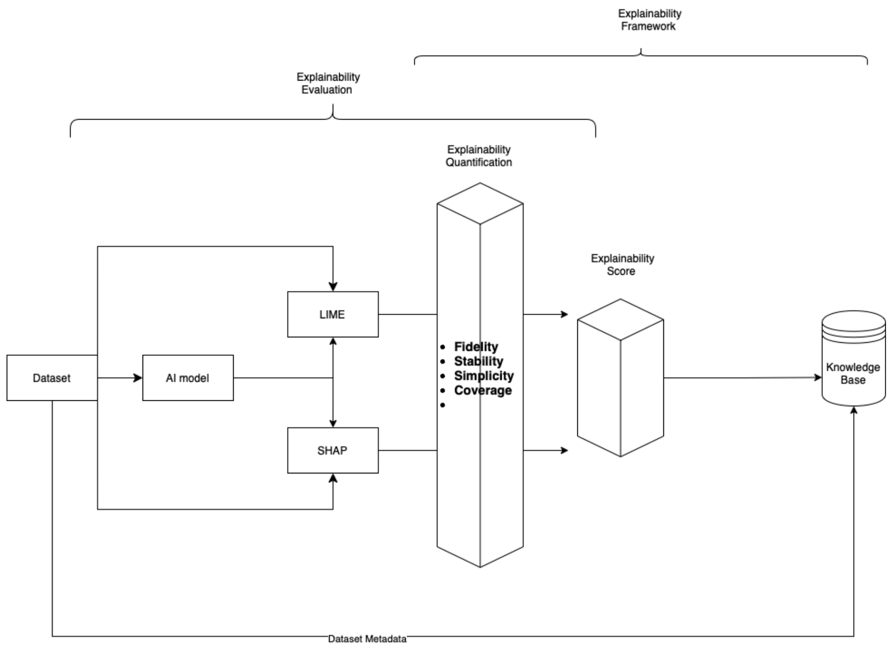


Figure 11 - Benchmarking process.

This section expands upon the component-level architecture outlined in Deliverable D2.1, offering detailed insights into the XAI scoring framework's design and functionality within the FAME project. For complete context and foundational architecture principles, refer to Deliverable D2.1.

Component-Level Architecture of the XAI Scoring Framework

The XAI scoring framework comprises several key components, each contributing to the system's ability to evaluate and score AI model explainability. Below is a detailed breakdown of these components, their functionalities, and how they interact within the framework:

Web Application Interface Component

- **Functionality:** Serves as the primary user interface for inputting AI, XAI, and dataset attributes.
- **Interactions:** Communicates directly with the Estimation Engine to submit user inputs and requests for scoring. Receives results from the Estimation Engine to display XAI scores to the user.

Benchmarking Engine Component

- **Functionality:** Calculates quantitative explainability metrics.
- **Interactions:** Saves results to Knowledge Base.

Estimation Engine Component

- **Functionality:** Acts as the core processing unit, calculating XAI scores based on quantitative and qualitative metrics. It is based on LightGBoost model.
- **Interactions:** Retrieves data from Knowledge Base.

Knowledge Base Component

- **Functionality:** Stores and manages a repository of information related to AI and XAI models, including performance metrics, user satisfaction scores, and historical analysis results.

- **Interactions:** Supplies the Estimation Engine with data necessary for scoring and recommendations. Receives updates from the Estimation Engine based on new analysis results and user feedback.

A draft "view" of the inputs of the knowledge base are presented in the following tables, where in the first 3 of them the features of each data asset are presented while in the fourth table are some indicative results of the matching between these assets.

Table 4 - Used datasets.

dataset_id	dataset_name	size	type	dataset_task	feature_count	domain	benchmark_status
0	IRIS	150	Tabular	Classification	4	Botany	Standard
1	Wine	178	Tabular	Classification	13	Chemistry	Standard
2	Breast Cancer Wisconsin	569	Tabular	Classification	30	Healthcare	Standard
3	UNIPI trajectory data	10000	Spatio-temporal	Regression	N/A	Mobility	No
4	Wafer	1000	Time series	Classification	N/A	Semiconductor	No
5	FordA	1000	Time series	Classification	N/A	Automotive	No
6	MNIST	70000	Image	Classification	784 (28x28)	Handwriting/Digit	Standard
7	CIFAR-10	60000	Image	Classification	3072 (32x32x3)	Object Recognition	Standard
8	Adult Income	48842	Tabular	Classification	14	Socio-Economics	Standard
9	20 Newsgroups	~20000	Text	Classification	N/A	News	Standard
10	Lending Club Loan Data	2260701	Tabular	Classification	145	Finance	No

Table 5 - Used XAI models.

xai_model_id	xai_model	Post-Hoc	Ante-Hoc	Global Explanations	Local Explanations	Attribute based	Example based	Model Agnostic
0	LIME	Yes	No	No	Yes	Yes	No	Yes
1	SHAP	Yes	No	Yes	Yes	Yes	No	Yes
2	PFI	Yes	No	Yes	No	Yes	No	Yes
3	Counterfactuals	Yes	No	No	Yes	No	Yes	Yes
4	Attention	No	Yes	Yes	Yes	Yes	No	No
5	Grad-CAM	Yes	No	No	Yes	Yes	No	No
6	Integrated Gradients	Yes	No	No	Yes	Yes	No	Yes
7	Anchor Explanations	Yes	No	No	Yes	Yes	No	Yes
8	Partial Dependence Plots	Yes	No	Yes	No	Yes	No	Yes
9	LIME-SUP	Yes	No	Yes	Yes	Yes	Yes	Yes

Table 6 - Used AI models.

ai_model_id	ai_model	Linear	Monotone	Interaction	Robustness to Overfitting	interpretability	training_time	ai_task
0	Linear Regression	Yes	Yes	No	Low	High	Fast	Regression
1	Logistic Regression	No	Yes	No	Moderate	High	Fast	Classification

2	Decision Trees	No	Some	Yes	Low	High	Fast	Classification
3	Decision Trees	No	Some	Yes	Low	High	Fast	Regression
4	RuleFit	Yes	No	Yes	Moderate	Moderate	Moderate	Classification
5	RuleFit	Yes	No	Yes	Moderate	Moderate	Moderate	Regression
6	Naïve Bayes	No	Yes	No	Moderate	High	Fast	Classification
7	KNN	No	No	No	Low	Low	Slow	Classification
8	KNN	No	No	No	Low	Low	Slow	Regression
9	Random Forests	No	Varies	Yes	High	Moderate	Moderate	Classification
10	Random Forests	No	Varies	Yes	High	Moderate	Moderate	Regression
11	GBM (Gradient Boosting Machine)	No	Varies	Yes	High	Moderate	Slow	Classification
12	GBM (Gradient Boosting Machine)	No	Varies	Yes	High	Moderate	Slow	Regression
13	AdaBoost	No	Varies	Yes	High	Moderate	Moderate	Classification
14	AdaBoost	No	Varies	Yes	High	Moderate	Moderate	Regression
15	Stacking	Varies	Varies	Yes	High	Low	Slow	Classification
16	Stacking	Varies	Varies	Yes	High	Low	Slow	Regression
17	Bagging	No	Varies	Yes	High	Moderate	Moderate	Classification
18	Bagging	No	Varies	Yes	High	Moderate	Moderate	Regression
19	SVM	No	No	Yes	Moderate	Low	Slow	Classification
20	SVM	No	No	Yes	Moderate	Low	Slow	Regression
21	CNN	No	No	Yes	High	Low	Slow	Image Classification
22	CNN	No	No	Yes	High	Low	Slow	Regression
23	RNN	No	No	Yes	Moderate to High	Low	Slow	Sequence Prediction
24	RNN	No	No	Yes	Moderate to High	Low	Slow	Regression
25	XGBoost	No	Varies	Yes	High	Moderate	Moderate	Classification
26	XGBoost	No	Varies	Yes	High	Moderate	Moderate	Regression
27	LightGBM	No	Varies	Yes	High	Moderate	Fast	Classification
28	LightGBM	No	Varies	Yes	High	Moderate	Fast	Regression
29	CATBoost	No	Varies	Yes	High	Moderate	Moderate	Classification
30	CATBoost	No	Varies	Yes	High	Moderate	Moderate	Regression
31	Elastic Net	Yes	Yes	No	Moderate	High	Fast	Regression
32	LASSO	Yes	Yes	No	Moderate	High	Fast	Regression
33	Ridge Regression	Yes	Yes	No	Moderate	High	Fast	Regression

Table 7 - Obtained results.

dataset_id	ai_model_id	xai_model_id	Fidelity	Coverage	Simplicity	Stability	User Satisfaction
6	12	1	0.9	0.8	0.6	0.7	0.85
7	12	1	0.95	0.85	0.65	0.75	0.9
2	6	0	0.8	0.75	0.7	0.7	0.8
8	11	2	0.7	0.65	0.6	0.6	0.75
9	17	3	0.85	0.7	0.75	0.8	0.8
2	7	4	0.8	0.6	0.65	0.65	0.7
4	5	5	0.7	0.7	0.5	0.6	0.65
3	14	6	0.6	0.8	0.4	0.5	0.6
0	18	7	0.9	0.9	0.9	0.85	0.9
1	17	8	0.85	0.75	0.8	0.8	0.85
4	10	9	0.65	0.7	0.55	0.6	0.7
5	9	0	0.7	0.65	0.6	0.65	0.7

6	13	1	0.6	0.55	0.6	0.6	0.65
7	12	2	0.95	0.9	0.7	0.75	0.9
8	19	3	0.85	0.8	0.85	0.8	0.85
9	16	4	0.65	0.6	0.7	0.65	0.7
0	18	5	0.9	0.85	0.9	0.9	0.9
1	17	6	0.85	0.8	0.85	0.85	0.8
2	6	7	0.75	0.7	0.65	0.7	0.75
3	14	8	0.65	0.6	0.55	0.6	0.6
4	10	9	0.7	0.65	0.6	0.65	0.7

Internal Relations and Data Flow

- **User Interaction:** Users interact with the Web Application Interface to input data or upload files, initiating the scoring process.
- **Data Processing:** The Estimation Engine utilizes Benchmarking Tools and accesses the Knowledge Base to calculate XAI scores, incorporating both quantitative and qualitative evaluations.

3.3.3.2 Baseline Technologies and Tools

Table 8 - Baseline Technologies and Tools

Baseline Technology	Description	Added value to FAME
Python	A programming language that excels in data manipulation, analysis, and machine learning through libraries like NumPy, pandas, and scikit-learn. It's favoured for its ease of use, readability, and extensive support for statistical and machine learning techniques.	Python fast prototyping capabilities allow creation of assets and services
XAI Baseline Models (LIME [1], SHAP[2], Grad-CAM[17])	LIME[1] and SHAP[2] provide model-agnostic interpretability, offering insights into feature importance and decision logic. Grad-CAM[3] offers visual explanations for decisions made by CNNs.	These models, among others, form the baseline model of qualitative and quantitative analysis in FAME, providing diverse perspectives on model behavior. LIME and SHAP aid in understanding feature contributions and decision processes, while Grad-CAM offers intuitive visual explanations for CNN-based models.
LightGBM [14]	A high-performance, gradient boosting framework that provides efficient implementations of decision trees designed for speed and accuracy, particularly on large datasets.	Used to train an AI model that estimates the XAI score based on the features (or metadata) of each data asset. This enables the framework to generate explainability scores that reflect both the qualitative insights from human expert evaluations and the quantitative metrics derived from benchmarking processes. LightGBM,

		without actually run any benchmarks on the data.
Flask	A lightweight WSGI web application framework in Python, enabling the quick and easy setup of web servers with minimal code.	Serves as the backbone for developing RESTful APIs for the FAME framework, allowing seamless interaction with the explainability scoring component and integration with the pricing module.
Docker	A platform for developing, shipping, and running applications in lightweight, portable containers.	Ensures that the FAME microservice is deployable, scalable, and manageable across various environments, facilitating the dissemination and real-world application of the FAME framework.

3.3.4 Interfaces

The API for the XAI Scoring Framework, in its initial implementation, provides two main interfaces that are described in the following tables. Each interface serves distinct purposes within the framework enabling users to interact with the system for estimating XAI scores and retraining the underlying models (as mentioned in the previous subsection), respectively. Below are the descriptions of these interfaces, along with more general interfaces of the API:

Title:	**Root interface**
Endpoint:	{HOST}
HTTP Method:	GET
Description:	This interface returns a list with the API's interfaces that are available to be used by all users. It acts as a roadmap, providing the interfaces along with short information about the functionalities that they trigger. The structure of the information follows a tree approach.
Body Data:	None
Headers:	None
URL Parameters:	None
Query Parameters:	None
Restrictions / Special Features:	None
Successful Response:	API's roadmap in text/plain.
The following is an example of the request in cURL:	
`curl –request GET '{HOST}'`	

Title:	**Estimation interface**
Endpoint:	{HOST}/app/estimation
HTTP Method:	POST
Description:	This interface is designed to receive input data related to AI, XAI, and datasets, and to provide estimations of XAI scores for the provided information / data. This interface facilitates the evaluation of the explainability scoring for the AI model mentioned in the previous subsections, allowing users to gauge the transparency and understandability of these technologies and how reliable they are. This method allows users to submit input data, including attributes of XAI techniques. The API processes this data and returns the estimation of XAI scores, providing insights into the explainability of the given models and techniques.

	This interface requires a POST request, and the body of the request must contain data that will be used for the new estimation, as raw data in JSON format. The schema of the data / content varies and it is flexible.
Body Data:	Raw (JSON) Data (Content-Type: application/json).
Headers:	None
URL Parameters:	None
Query Parameters:	None
Restrictions / Special Features:	None
Successful Response:	JSON Object with the estimation in its content.
The following is an example of the request in cURL:	

```
curl –request POST '{HOST}/app/estimation' \
–header 'Content-Type: application/json' \
--data-raw '{ **valid JSON schema with the data to be used for the estimation** }'
```

Title:	**Retrain interface**	
Endpoint:	{HOST}/app/retrain	
HTTP Method:	POST	
Description:	This interface enables users to initiate the retraining of the underlying model used for estimating XAI scores. By providing mechanisms for model retraining, this interface supports the continuous improvement and adaptation of the XAI scoring framework to evolving data and requirements. Users can utilize this method to trigger the retraining process for the underlying model. The API initiates the retraining based on the provided parameters and data, ensuring that the model remain up-to-date and reflective of the latest trends and patterns in the data. This interface requires a POST request, and the body of the request must contain the data that will be used for model's retraining, as form-data with the key "file".	
Body Data:	Data Type:	Form Data
	Key	Value
	file	Binary data / Path to file
Headers:	None	
URL Parameters:	None	
Query Parameters:	None	
Restrictions / Special Features:	Available only to the administrators of the API.	
Successful Response:	JSON Object with the estimation in its content.	
The following is an example of the request in cURL:		

```
curl –request POST '{HOST}/app/retrain \
--form 'file=@"<full_path_to_asset>"'
```

4 Components Demonstration

4.1 ML/AI Analytics

Among the pilot use cases available within the FAME project, it was decided to use the task of sentiment analysis for financial news to demonstrate the functionalities described for the AI/ML analytics component. For this task, a set of texts is provided containing news fragments related to the company's actions and the financial expert's opinion. All these texts are labelled with the polarity of the fragment. This metric consists of a number between -1 and 1 that indicates how negative or positive the feeling is towards the text.

As a result, two AI-based Natural Language Processing (NLP) models were implemented to address the proposed task. The first is based on the VADER [17] model, while the second is based on the Google NNLM [18] model.

VADER [17] model

This approach exploits the advantages of parsimonious rule-based modelling to construct a computational sentiment analysis engine that ensures that:

1. Works correctly on social media style text, yet easily generalises to multiple domains.
2. Does not require training data, but is constructed from a generalisable, valence-based, human-curated gold standard sentiment lexicon.
3. Is fast enough to be used online with streaming data.
4. It does not suffer from a severe speed/performance trade-off.

The steps taken to set the VADER model in order to estimate the polarity of a sentence are as follows:

1. Examine all lexical features of existing well-established and human-validated sentiment lexicons (e.g. UWC, ANEW, GI)
2. Supplement with additional lexical features commonly used to express sentiment social media text (emoticons, acronyms, slang)
3. Use wisdom-of-the-crowd approach to establish point estimations of sentiment valance for each of 9000+ lexical feature candidates.
4. Kept 7500+ lexical features with mean valence near zero and standard deviation lower than 2.5 as a human-validate gold-standard sentiment lexicon.
5. Use data-driven iterative inductive coding analysis (e.g. Grounded Theory) to identify generalizable heuristics for assessing sentiment in text.

Once the model is fitted, it is ready to infer sentiment polarity from pieces of text.

NNLM-based [18] model

This other approach uses a Neural Network Large Model (NNLM) followed by a classification model, which allows it to learn from data instead of developing a rule-based system. This NNLM model needs to be trained on a significant amount of data to be able to return the significant embeddings of the given texts to be used later as inputs to the classification model.

To avoid training the NNLM model from scratch, a pre-trained version of the NNLM model from Google is used, which returns normalized text embeddings of size 128. This model is later concatenated with a dense classifier network, which is fine-tuned with the financial data provided.

4.1.1 Prerequisites and Installation Environment

Since all AI/ML analytics tools use Docker containerisation technologies to avoid obscure pre-installation requirements, the only tool that needs to be pre-installed on the user's premises is the Docker Engine. Below are links to the installation guides for the three most popular operating systems.

- Windows: https://docs.docker.com/desktop/install/windows-install/
- Linux: https://docs.docker.com/desktop/install/linux-install/
- Mac OS: https://docs.docker.com/desktop/install/mac-install/

4.1.2 Installation Guide

Currently, in order to download the AI assets, users may need to gain access to the provider's Docker repository and obtain the necessary running files (Docker compose files).

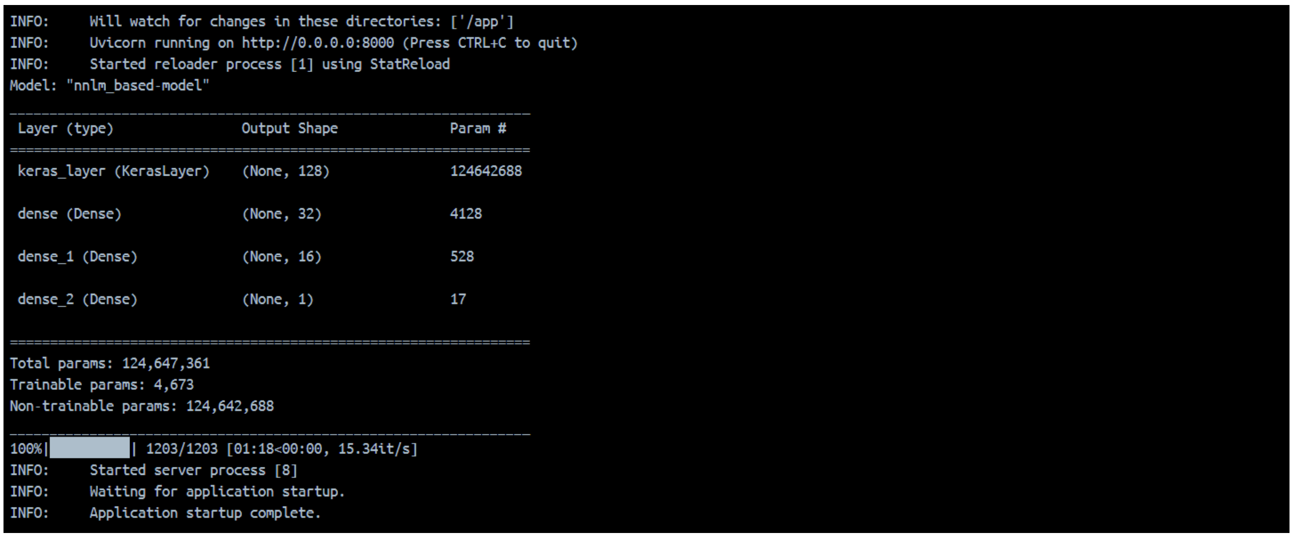
4.1.3 User Guide

Once users have obtained the executable files, all they need to do to start using the AI analytics tools is running them as follows:

```
$ docker compose -env-files /path/to/envs/file -f /path/to/docker/compose/file up
```

After that, the API endpoints, explained on Section 3.1.4, are available for users to interact with the provided tools. It is worth noting that there only are available the "/" and the "/infer" endpoints, since the "/evaluate", "/train" and "/index" endpoints are thought to be developers end points. The functionalities of these last three are already implemented, but they are not accessible through the endpoint.

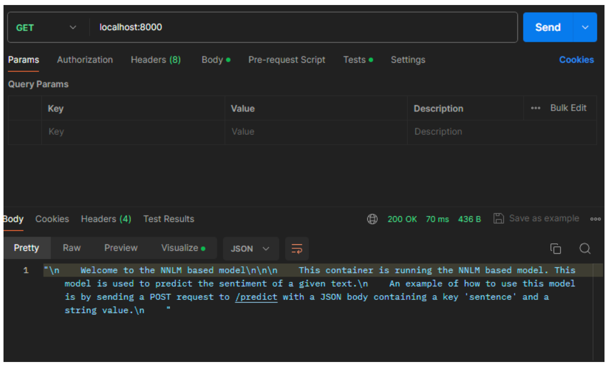
In Figure 12, it can be observed the container logs when the service is up and properly running show that the model (the NNLM-based model in this case) is loaded and tested and the endpoint is started.



```
INFO:     Will watch for changes in these directories: ['/app']
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [1] using StatReload
Model: "nnlm_based-model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 128)               124642688

 dense (Dense)               (None, 32)                4128

 dense_1 (Dense)             (None, 16)                528

 dense_2 (Dense)             (None, 1)                 17

=================================================================
Total params: 124,647,361
Trainable params: 4,673
Non-trainable params: 124,642,688
_________________________________________________________________
100%|██████████| 1203/1203 [01:18<00:00, 15.34it/s]
INFO:     Started server process [8]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Figure 12 - Service start up process.

When accessing the "/" endpoint, a description of the functionalities of the service is returned to the user, as can be seen in Figure 13.a). Moreover, in Figure 13.b) the container logs show the successful completion of the request.



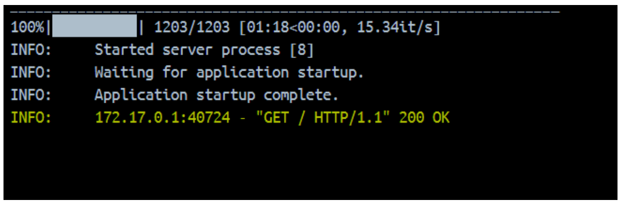
localhost:8000

Query Params

Key	Value	Description
Key	Value	Description

200 OK 70 ms 436 B

```
"\n    Welcome to the NNLM based model\n\n\n    This container is running the NNLM based model. This
   model is used to predict the sentiment of a given text.\n    An example of how to use this model
   is by sending a POST request to /predict with a JSON body containing a key 'sentence' and a
   string value.\n    "
```

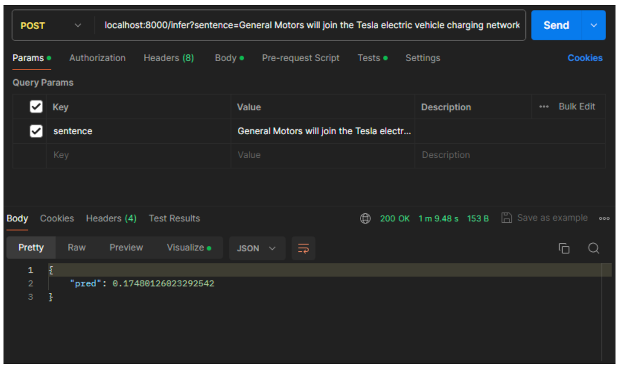


```
100%|██████████| 1203/1203 [01:18<00:00, 15.34it/s]
INFO:     Started server process [8]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     172.17.0.1:40724 - "GET / HTTP/1.1" 200 OK
```

a) b)

Figure 13 – Example of the output returned by "/" endpoint. a) is the JSON response returned by the API and b) is the log prompt within the Docker container.

Finally, when inferring the polarity of a text, this can be performed through the "/infer" endpoint. As is shown in Figure 14.a), it is only needed to pass the text by the key "sentence" and the endpoint returns the polarity of this text. In Figure 14.b) can be seen the container's logs for this process.



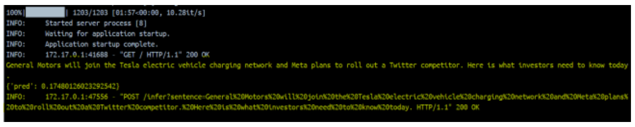
localhost:8000/infer?sentence=General Motors will join the Tesla electric vehicle charging network

Query Params

	Key	Value	Description
✓	sentence	General Motors will join the Tesla electr...	
	Key	Value	Description

200 OK 1 m 9.48 s 153 B

```
{
    "pred": 0.17480126023292542
}
```



```
100%|██████████| 1203/1203 [01:57<00:00, 10.28it/s]
INFO:     Started server process [8]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     172.17.0.1:41688 - "GET / HTTP/1.1" 200 OK
General Motors will join the Tesla electric vehicle charging network and Meta plans to roll out a Twitter competitor. Here is what investors need to know today
.
{'pred': 0.17480126023292542}
INFO:     172.17.0.1:47556 - "POST /infer?sentence=General%20Motors%20will%20join%20the%20Tesla%20electric%20vehicle%20charging%20network%20and%20Meta%20plans%
20to%20roll%20out%20a%20Twitter%20competitor.%20Here%20is%20what%20investors%20need%20to%20know%20today. HTTP/1.1" 200 OK
```

a) b)

Figure 14 – Example of the output returned by "/infer" endpoint. a) is the JSON response returned by the API and b) is the log prompt within the Docker container.

4.2 SAX techniques

As one possible application of SAX, we have developed an application that leverages ML for sentiment analysis on news feeds pertaining to selected companies. For each company, a collection of related news articles is gathered over time. It is generally hypothesized that the series of sentiments extracted from such feeds may correlate with the company's financial performance, as also reflected by financial indicators such as the ESG (Environmental, Social, and Governance) index. Our ultimate goal is to explore the potential to predict and elucidate ESG projections based on the underlying sentiment sequences.

As an initial step here, we demonstrate the use of an LLM to enable ML model-agnostic explainability for a single input narrative and its corresponding sentiment as determined by the ML model. Our goal is to enable the realization of a model-agnostic explainability for sentiment analysis as a SAX service ***LLMsentimentXplain*** as follows. Given some news ad (narrative) about a company and a predicted sentiment by some inference model (e.g., VADER, BERT), explain the sentiment prediction as yielded by the model by highlighting sets of keywords in the news ad that support the sentiment according to the inference model.

Our approach was developed as a post-hoc technique (after model training) to enable the identification of the 'K' most relevant set of terms (i.e., the k-sufficient set) in the input narrative that are sufficient to influence the prediction of the ML model, regardless of which specific ML technique is used for the prediction of the sentiment. Concretely, a k-sufficient set is a subset of words from the original narrative that when provided as an input to the ML model retains the same sentiment output as the sentiment that is generated for the entire narrative. Thus, given some model M and an input narrative text $T = w_1 ... w_n$, we can determine its sentiment as $M(T)$. Respectively, a k-sufficient set in this case is a subset of k words $s_k = w_1 ... w_k \subseteq T$, where $|s_k| = k$, conforming to:

1. $M(s_k) = M(T)$, i.e., the subset of terms retains the same sentiment.
2. There is no subset $s' = w_1 ... w_z \subset s_k$ where $M(s') = M(T)$

We acknowledge that there could be several subsets in T conforming to the above conditions. Our current implementation is agnostic to the differences among such subsets, leaving it to the determination of the LLM. A more advanced form of our approach above could also exploit the computation of weights that signify the relative importance of each word in affecting the result of the ML model. In such a case, our algorithm should be enhanced with a selection of a subset that relies on the value of such weights (e.g., the one with the highest weight average). Toelicit a k-sufficient set of terms from a given narrative, we followed a zero-shot prompting approach leveraging the recent LLM release of GPT-4.0. Our overall iterative interaction scheme is illustrated in Figure 15. We hereby elaborate on its core algorithmic steps:

```
LLMsentimentXplain($M, X, max\_attempts$)

1  $\boldsymbol{result} = $ 'no explanation'
2  $s \leftarrow M(X)$
3  $attempts \leftarrow 0$
4  $\boldsymbol{prompt} \leftarrow$ 'given text:' $+ \boldsymbol{X} +$
   'what are the top k words in it that support its sentiment classification as:' $+ \boldsymbol{s}$
5  do
6      $attempts \leftarrow attempts + 1$
7      $W \leftarrow$ gpt.apply($prompt$)
8      $s' \leftarrow M(W)$
9      if $s \neq s'$ then:
10         $prompt \leftarrow prompt +$ 'the sentiment classification of:' $+ W +$ 'is:' $+ s'$
11 while $\boldsymbol{s \neq s'}$ and $\boldsymbol{attempts < max\_attempts}$
12 if $\boldsymbol{s = s'}$ then: $\boldsymbol{result = W}$
13 return $result$
```

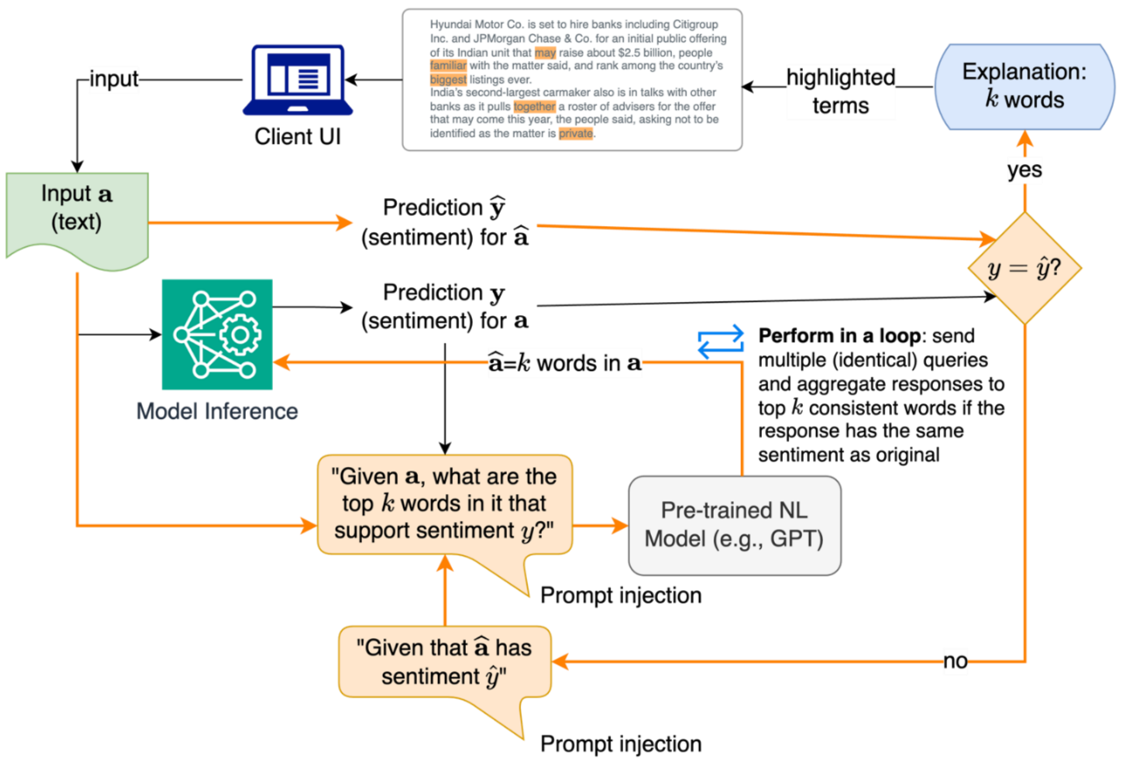

Hyundai Motor Co. is set to hire banks including Citigroup Inc. and JPMorgan Chase & Co. for an initial public offering of its Indian unit that may raise about $2.5 billion, people familiar with the matter said, and rank among the country's biggest listings ever.
India's second-largest carmaker also is in talks with other banks as it pulls together a roster of advisers for the offer that may come this year, the people said, asking not to be identified as the matter is private.
input
Client UI
highlighted terms
Explanation: k words
yes
Input $\mathbf{a}$ (text)
Prediction $\widehat{\mathbf{y}}$ (sentiment) for $\widehat{\mathbf{a}}$
$y = \hat{y}$?
Prediction $\mathbf{y}$ (sentiment) for $\mathbf{a}$
Model Inference
$\widehat{\mathbf{a}}$=k words in $\mathbf{a}$
Perform in a loop: send multiple (identical) queries and aggregate responses to top k consistent words if the response has the same sentiment as original
"Given $\mathbf{a}$, what are the top k words in it that support sentiment y?"
Pre-trained NL Model (e.g., GPT)
Prompt injection
"Given that $\widehat{\mathbf{a}}$ has sentiment $\hat{y}$"
no
Prompt injection

Figure 15 - Sentiment Analysis demo flow.

The core idea of our algorithm is in employing the LLM as a zero-shot XAI model to determine which are the k-sufficient terms corresponding to the sentiment that was determined by the ML model. We instruct the LLM to provide a list of top-K relevant words in the given input narrative that are related to the predicted sentiment. We then present a "masked" narrative with only the top-K terms as input to the ML model and invoke it to get the sentiment that corresponds to these terms. If the newly derived sentiment matches the original predicted sentiment, we return the top-K terms as the explanation and stop the process. Otherwise, and as long as some of threshold attempts have not been exceeded, we extend the prompt to associate the relevant top-K keywords with the new sentiment and repeat the process (see Figure 15).

The above technique has been implemented. A recorded demo illustrating this novel technique can be accessed here.

Subsequent to this step, we foresee as the next steps the possibility of employing SAX over a series of timestamped sentiments that are derived according to the above, using a ML model such as Vader, to derive a sentiment for each individual news ad, and then partitioning such sequences into higher-level process traces with the use of the time4process service. Such traces may then be enriched with additional context related data utilizing ***ContextEnrichment*** and serve as input for the prediction of changes in company's financial indicators such as ESG index. Complementing such sentiment traces also with the k-sufficient terms (along each sentiment label) may also improve the accuracy of such predictions. The time4process service is not only considered relevant to this pilot but seems to also be exploitable in other user cases such as the motor oil (pilot 7), where raw sensor data about machines may be transformed into higher level lifecycle stages. A first exploration of this data already started and is shown below.

In the motor oil pilot, sensors on machines are continuously monitored for the purpose of machine maintenance. Initially, sensor codes are translated into human-readable terms, such as mapping the code "32XI453" to "gearbox axial displacement." The data is then visualized (Figure 16) to provide an overview and pre-processed into a format compatible with the ***StreamStory*** tool, which is used to construct a Markov chain model. The resulting model, as depicted in the Figure 17, identifies recurrent patterns within the data. At the initial scale (scale 0), the model displays two states, each encompassing multiple data points. As the scale increases, additional states become visible on the user interface, with the granularity of states and transitions between them becoming more refined.

In the context of ***StreamStory***, these states could represent different operational conditions or behaviours of the gearbox. The transitions imply a change from one condition to another, and the thickness of the lines might indicate the frequency or probability of transitioning from one state to another. The varying scales suggest that the tool can zoom in on the data for a more detailed analysis or zoom out for a broader perspective. This process facilitates the identification of patterns that could be critical for predictive maintenance, anomaly detection, or understanding the system's dynamics.

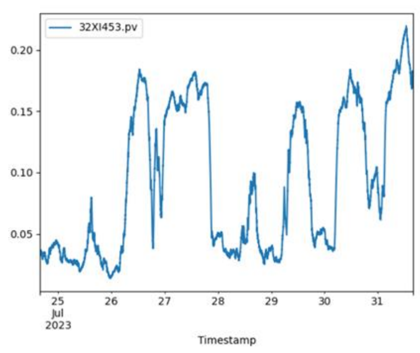


Figure 16 - Unprocessed Signal Data for Gearbox Axial Displacement.

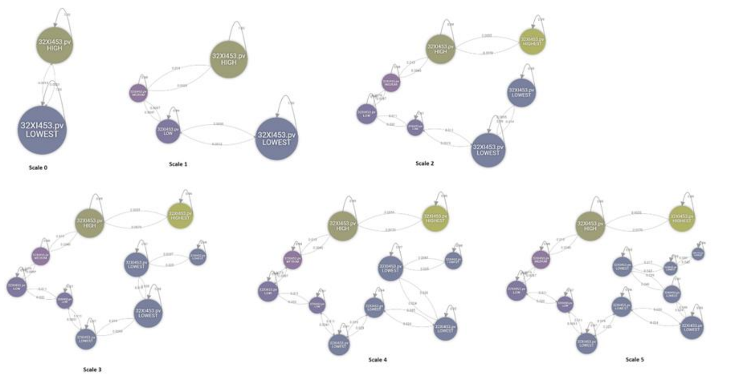


Figure 17 - Markov Chain Model of Gearbox Axial Displacement Signal as Generated by StreamStory.

4.2.1 Prerequisites and Installation Environment

The SAX sentiment analysis module is a python-based docker image. To run it, you should have a docker daemon in your environment, and preferably also docker compose.

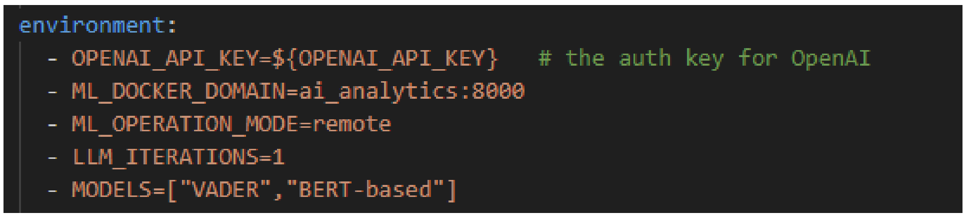
To install docker daemon and/or docker compose follow the instructions here according to your operating system: [Docker Engine installation instructions]

To install docker compose follow the instructions here according to your operating system [Docker Compose installation instructions.]

4.2.2 Installation Guide

To execute the SAX docker component the following steps should be followed:

1. The docker image should be pulled from FAME docker registry (harbor.infinitech-h2020.eu/fame/ai _tools/sax4sentiment).
2. The docker can be run as a standalone, using an internal implementation of VADER model, or as part of the ML pipeline for usage of additional sentiment analysis models (such as BERT)
3. The following parameters for running the docker should be configured:



```
environment:
  - OPENAI_API_KEY=${OPENAI_API_KEY}   # the auth key for OpenAI
  - ML_DOCKER_DOMAIN=ai_analytics:8000
  - ML_OPERATION_MODE=remote
  - LLM_ITERATIONS=1
  - MODELS=["VADER","BERT-based"]
```

a. **OPEN_API_KEY=${OPENAI_API_KEY}**: the OpenAI API access key, since we use the OpenAI GPT model for keywork analysis. The key can be used directly, whether in CLI or docker-compose, or it can be placed in .env file in the same directory for security purposes.
b. **ML_DOCKER_DOMAIN=ai_analytics:8000:** should be configured to the URL or the ai_analytics container for remote ML models (if the ML_OPERATION_MODE = remote)
c. **ML_OPERATION_MODE=remote//local:** can be **remote** or **local.** If remote, will access the ML model inference via the URL provided in **ML_DOCKER_DOMAIN** parameter. If **local** will invoke local VADER model.
d. **LLM_ITERATIONS = N:** N is the number of iterations of GPT for keyword analysis. It can be any value starting from 1. If more than 1, the application will invoke the GPT model N times and return only the keywords which appeared in 70% of invocation results (the "strongest" keywords which cause the sentiment)
e. **MODELS = {"VADER", "BERT-based"}:** names of ML models to invoke. Our application is model independent, additional models can be added to ML pipeline for inference of sentiment analysis, it is enough to add the ML model name as appears in RESTful API of ai_analytics component to the list here so that SAX model can use it.

4. The docker can be run from docker CLI or using ***docker-compose.yml*** file. To run the docker from CLI, use the following command: (assuming you have an .env file with the value of OpenAI key)

```
docker run --env-file .env \
  -e OPEN_API_KEY=${OPENAI_API_KEY} \
  -e ML_DOCKER_DOMAIN=ai_analytics:8000 \
  -e ML_OPERATION_MODE=remote \
  -e LLM_ITERATIONS=N \
  -e MODELS='{"VADER","BERT-based"}' \
 harbor.infinitech-h2020.eu/fame/ai_tools/sax4sentiment
```

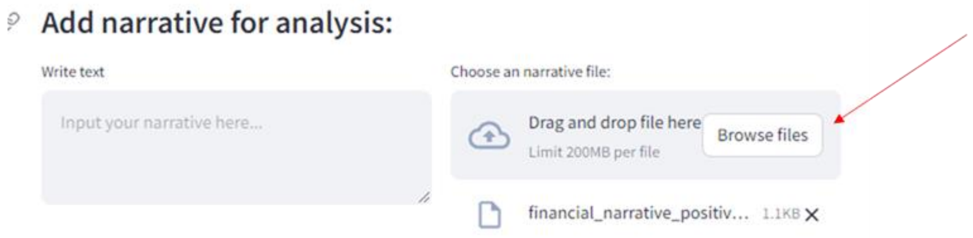
5. Alternatively, if using docker-compose, here is the proposed content of the docker-compose.yml file: (this is the full file including the ML pipeline. To run SAX standalone, remove the ai_analytics container definition and the **depends_on: ai_analytics** statement from the explainer-container definition).

```
version: '3'
services:
  explainer-container:
    build:
      context: .
    image: harbor.infinitech-h2020.eu/fame/ai_tools/sax4sentiment
    depends_on:
      - ai_analytics
    environment:
      - OPENAI_API_KEY=${OPENAI_API_KEY}   # the auth key for OpenAI
      - ML_DOCKER_DOMAIN=ai_analytics:8000
      - ML_OPERATION_MODE=remote
      - LLM_ITERATIONS=1
      - MODELS=["VADER","BERT-based"]
      - PYTHONUNBUFFERED=1
      - PYTHONIOENCODING=UTF-8
    ports:
      - "8501:8501"
    networks:
      - frontend

  ai_analytics:
    image: harbor.infinitech-h2020.eu/fame/ai_tools/ai_analytics
    container_name: ai_analytics
    ports:
        - "8000:8000"
    environment:
      - PYTHONUNBUFFERED=1
    networks:
      - frontend


networks:
  frontend:
    driver: bridge
```

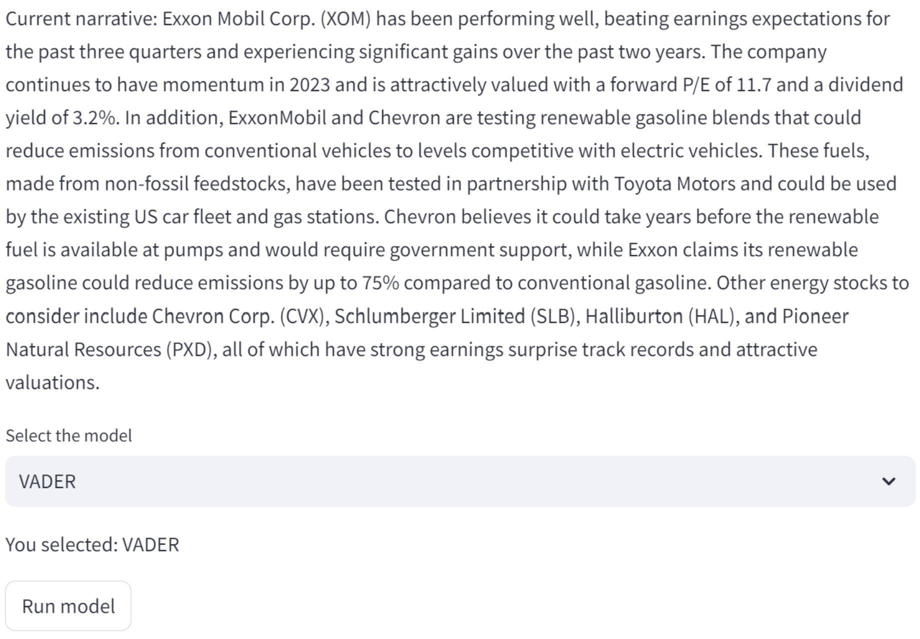
To run the containers, use "***docker compose run –d"*** command. It will pull the required images from the Harbor registry and instantiate the containers

4.2.3 User Guide

1. Once the docker containers are up and running you can access the sentiment analysis UI from your browser on localhost:8501 (8501 is the default port, unless you change the port mapping in docker run command or in docker-compose)
2. In the UI, choose the narrative to analyse: you can upload it as a text file or as text in a text box:


Add narrative for analysis:
Write text
Input your narrative here...
Choose an narrative file:
Drag and drop file here
Limit 200MB per file
Browse files
financial_narrative_positiv... 1.1KB

3. After uploading the narrative, choose the sentiment analysis model and click on "**Run Model**" button.

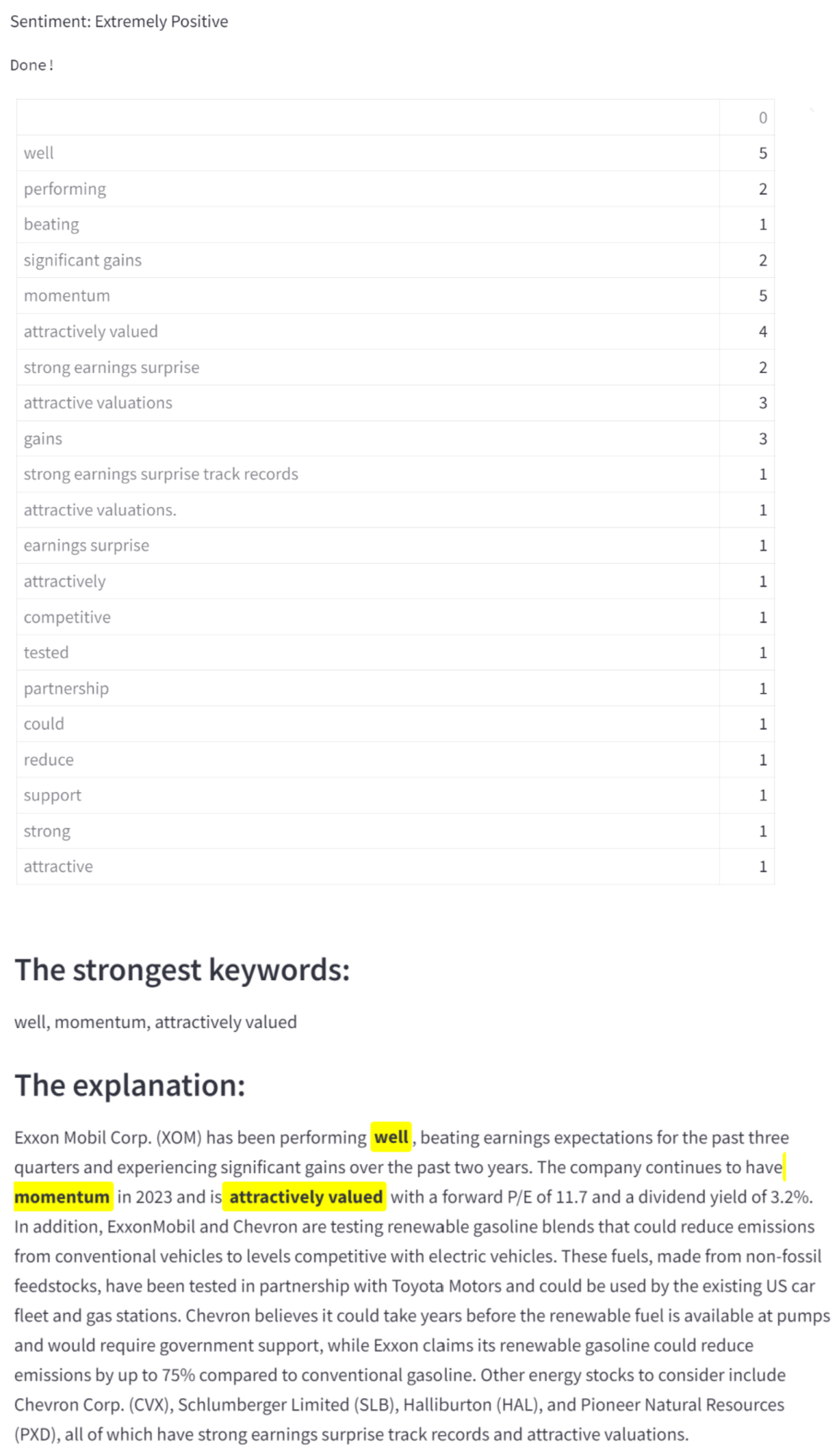

Current narrative: Exxon Mobil Corp. (XOM) has been performing well, beating earnings expectations for the past three quarters and experiencing significant gains over the past two years. The company continues to have momentum in 2023 and is attractively valued with a forward P/E of 11.7 and a dividend yield of 3.2%. In addition, ExxonMobil and Chevron are testing renewable gasoline blends that could reduce emissions from conventional vehicles to levels competitive with electric vehicles. These fuels, made from non-fossil feedstocks, have been tested in partnership with Toyota Motors and could be used by the existing US car fleet and gas stations. Chevron believes it could take years before the renewable fuel is available at pumps and would require government support, while Exxon claims its renewable gasoline could reduce emissions by up to 75% compared to conventional gasoline. Other energy stocks to consider include Chevron Corp. (CVX), Schlumberger Limited (SLB), Halliburton (HAL), and Pioneer Natural Resources (PXD), all of which have strong earnings surprise track records and attractive valuations.

Select the model
VADER

You selected: VADER

4. The application will display the sentiment analysis result for the given narrative, and after a short-while, it will display the keyword analysis: the ChatGPT prompted keywords in the

narrative producing such sentiment, both in the table where we can see the “strength” of the keyword. It will also show the strongest keywords highlighted in text.



Sentiment: Extremely Positive

Done !

	0
well	5
performing	2
beating	1
significant gains	2
momentum	5
attractively valued	4
strong earnings surprise	2
attractive valuations	3
gains	3
strong earnings surprise track records	1
attractive valuations.	1
earnings surprise	1
attractively	1
competitive	1
tested	1
partnership	1
could	1
reduce	1
support	1
strong	1
attractive	1

The strongest keywords:

well, momentum, attractively valued

The explanation:

Exxon Mobil Corp. (XOM) has been performing **well**, beating earnings expectations for the past three quarters and experiencing significant gains over the past two years. The company continues to have **momentum** in 2023 and is **attractively valued** with a forward P/E of 11.7 and a dividend yield of 3.2%. In addition, ExxonMobil and Chevron are testing renewable gasoline blends that could reduce emissions from conventional vehicles to levels competitive with electric vehicles. These fuels, made from non-fossil feedstocks, have been tested in partnership with Toyota Motors and could be used by the existing US car fleet and gas stations. Chevron believes it could take years before the renewable fuel is available at pumps and would require government support, while Exxon claims its renewable gasoline could reduce emissions by up to 75% compared to conventional gasoline. Other energy stocks to consider include Chevron Corp. (CVX), Schlumberger Limited (SLB), Halliburton (HAL), and Pioneer Natural Resources (PXD), all of which have strong earnings surprise track records and attractive valuations.

4.3 XAI Scoring Framework

The FAME project's XAI Scoring Framework is currently in the development stage, providing an initial presentation of how the system will function upon completion. The primary goal of this draft is to illustrate the framework's core functionality, specifically its ability to estimate explainability scores for AI models based on selected attributes. This early version runs locally as a web application, enabling users to interact directly with the system to understand its potential.

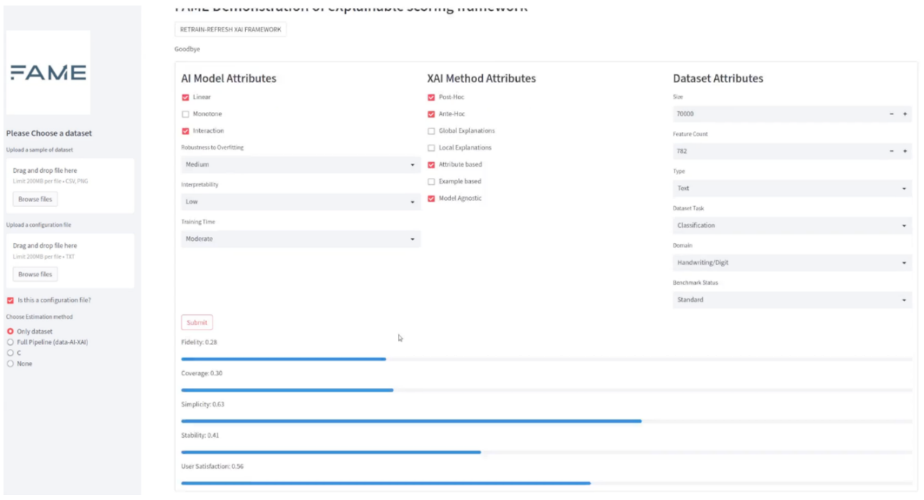
Web Application Interface: The interface allows users to input specific attributes related to their AI model, the XAI techniques of their interest and dataset characteristics. This user-friendly design ensures that even at this early stage, the focus is on ease of use and intuitive interaction.

The current UI (Figure 18) that has been developed for demonstration purposes is divided into three main sections:

- AI Model Attributes: This section has checkboxes for selecting attributes of an AI model such as "Linear," "Monotone," "Interaction," and "Resistant to Overfitting," with a dropdown for "Training Time" that has options such as Low, Medium, Moderate, and so forth.
- XAI Method Attributes: On the top right, there are checkboxes for different explainable artificial intelligence (XAI) method attributes like "Anti-Hoc," "Global Explanations," "Local Explanations," "Attribute based," "Example based," and "Model Agnostic."
- Dataset Attributes: The final section on the right lists attributes of the dataset, including "Size" (70,000), "Feature Count" (784), "Type" (with "Test" selected), "Dataset Task" (with "Classification" selected), and "Domain" (with "Handwriting/Digit" selected), and "Benchmark Status" (with "Standard" selected).

Below these sections, there are sliders for representing the results of XAI scoring framework including "Fidelity", "Coverage", "Simplicity", "Stability", and "User Satisfaction".

On the left-hand side, there's a panel for dataset upload with options to drag and drop files or browse files, and a section for uploading a configuration file. These functionalities will be not yet implemented.



Goodbye

Please Choose a dataset

Upload a sample of dataset
Drag and drop file here
Limit 200MB per file • CSV, PNG

Upload a configuration file
Drag and drop file here
Limit 200MB per file • TXT

☑ Is this a configuration file?

Choose Estimation method
● Only dataset
○ Full Pipeline (data-AI-XAI)
○ C
○ None

AI Model Attributes
☑ Linear
☐ Monotone
☑ Interaction
Robustness to Overfitting: Medium
Interpretability: Low
Training Time: Moderate

XAI Method Attributes
☑ Post-Hoc
☑ Ante-Hoc
☐ Global Explanations
☐ Local Explanations
☑ Attribute based
☐ Example based
☑ Model Agnostic

Dataset Attributes
Size: 70000
Feature Count: 782
Type: Text
Dataset Task: Classification
Domain: Handwriting/Digit
Benchmark Status: Standard

Fidelity: 0.28
Coverage: 0.30
Simplicity: 0.63
Stability: 0.41
User Satisfaction: 0.56

Figure 18 - XAI scoring framework demonstration UI.

Estimation of XAI Scores: Upon submitting their selections and inputs, users receive an estimation of the XAI score. This score encompasses both quantitative metrics such as fidelity, stability, simplicity, and coverage, as well as qualitative assessments, potentially derived from user satisfaction surveys or expert evaluations. This multidimensional score aims to provide a comprehensive overview of the explainability of the selected AI model.

4.3.1 Prerequisites and Installation Environment

- Operating System: Linux is preferred for Docker-based deployments.
- Python Version: Python 3.8 or newer. Ensure Python and pip are installed.
- Docker: The latest version of Docker for container management. Ensure Docker is installed and running on your system.
- Required Python Libraries: Libraries such as LightGBM, Flask, NumPy, pandas, scikit-learn, and specific libraries for XAI models like LIME and SHAP. These can be installed via pip.

- Hardware Requirements: Depending on the size of the datasets and complexity of the AI models, the hardware requirements can vary. A minimum of 4GB RAM and a multi-core processor are recommended.

4.3.2 Installation Guide

Clone the Repository: Clone the FAME framework repository from its Git source to your local machine.

```
git clone <repository-url>
```

Set Up a Python Virtual Environment (optional but recommended):

```
python3 -m venv fame_env source fame_env/bin/activate  # On Windows use `fame_env\Scripts\activate`
```

Install Required Python Libraries: Navigate to the cloned repository’s directory and install the required libraries using pip.

```
pip install -r requirements.txt
```

Docker Containers: If the component is designed to run inside a Docker container, build the Docker image and run it as specified in the project's Dockerfile or docker-compose.yml.

```
docker build -t fame-framework . docker run -p 5000:5000 fame-framework
```

Check that the Docker image has been created:

```
docker images
```

To start a container on a local computer:

```
docker run -d -p <host_port>:<container_port> <image_name>
```

This command starts a Docker container in detached mode, mapping a port of the container to a port on the host machine.

To pull an image from DockerHub, you would use the following command:

```
docker pull <image_name>
```

For deploying a container, especially if referring to an <name> container as an example, the command would be:

```
docker run -d -p <host_port>:<container_port> <name>
```

4.3.3 User Guide

Not yet Available.

5 Conclusions

This deliverable has presented the work done in the context of Task 5.1 and Task 5.2 of FAME, which are devoted to providing AI/ML analytics to FAME users by indexing them in FAME's Federated Data Asset Catalogue (FDAC). Specifically, it documents the developments and advancements of the AI/ML Analytics component, as well as the SAX and XAI ones. These components will be further detailed and completed in the last version of this deliverable, D5.3, which is expected to be submitted in M33. As a summary, Table 9 gathers the KPIs established in the context of the project for these specific components. As can be seen, at M9 all of them have been fulfilled, which proves that we are on the right track to achieve the KPIs defined for the end of the project.

Table 9 – D5.1 Related KPIs

Obj. ID	KPI ID	Measured KPI	Expected Value [*M9*]	Achieved Value [*M9*]	Expected Value [*M18*]	Achieved Value [*M18*]
O5	**5.1**	Number of XAI techniques/models	2	2	4	5[5]
	5.3	Number of AI/ML models	1	1	4	4[5]

The following subsections gather the specific conclusions that can be gathered for each of the components separately.

5.1 AI/ML Analytics

During this period, the main efforts in the AI/ML Analytics component have been focused on:

1) Analysing the business requirements coming from FAME pilots in order to translate them into technical requirements that can be then used to identify potential models that can be of use to solve these business needs.
2) Gathering, studying, and preparing the data coming from the pilots to use them to train the related AI models.
3) Training models, both with real data coming from pilots and with public datasets.
4) Defining the interfaces of the component, to ensure seamless integration with SAX and XAI components, as well as with the rest of the components of FAME architecture.
5) Creating a demo with the real data coming from one of the pilots, pilot 5, to showcase the offerings of this component to potential FAME users (shown in Section 4.1.3).

5.2 SAX techniques

The main effort of the XAI and SAX techniques revolved around two parallel threads: the development of a scoring framework for assessing and quantifying the explainability of various XAI models, AI systems, and data assets; and the development of the SAX4BPM library, that is, a set of services to support novel explanations given to decisions or outcomes resulting from business process executions.

In the SAX4BPM library the following capabilities have been developed: Causal discovery for BP explanation and improvement; LLM for sufficient sentiment explanation; XAI for business process explanations; and time series segmentation for explanation. To cope with the project specific requirements of the use cases, we have extended the library with an additional novel service to

[5] These values are those reached at M15, when this deliverable was submitted

generate (sufficient) explanations for sentiment analysis. A recorded demo that demonstrates this functionality can be found here.

The developed method is post-hoc ML model training and is independent of the concrete ML model type that is used for sentiment classification. As such, our developed explainability service could be independently instantiated and deployed to supplement the functionality of any ML service that a client may be interested in provisioning via the overall FAME platform. It is also foreseen as a valuable input not only for explanation of the sentiment predictions, but also as an input for the forecasting of the ESG index and possibly other financial indicators leveraging news ads as an input. In parallel to this effort, we are also looking to employ time series segmentation and causal process discovery in the context of the other pilots. While our exploration is still in progress, such capabilities may present their value for provisioning as a tradeable service.
The library will be released as open source at the end of the project.

As next possible steps to the work presented here, we may further integrate the explainability of sentiments with the scoring XAI framework and examine possible applications with necessary adaptations to the other use cases (e.g., motor oil). With regards to the use of the StreamStory tool for time series input data, we aim to augment StreamStory with large language model technology by introducing an interactive chatbot. This enhancement will simplify the exploration and interpretation of complex visualizations and results for users. The chatbot will offer real-time assistance, making the tool's advanced analysis of multivariate time series data more accessible and understandable, even for those with minimal analytical expertise. Users will be able to engage in conversations with the chatbot to ask questions, seek explanations, and gain insights into the data and its visual representations. This update is designed to improve user experience by making StreamStory's sophisticated data analysis capabilities more intuitive and engaging through conversational interaction.

References

1. FAME, “D2.2 - Technical Specifications and Platform Architecture I,” 2023.
2. FAME, “D5.2 – Energy Efficient Analytics Toolbox I,” 2024.
3. A. Mavrogiorgou, et al., "FAME: Federated Decentralized Trusted Data Marketplace for Embedded Finance", [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10215814.
4. FAME, "D2.1 - Requirements Analysis, Specifications and Co-Creation I," 2023.
5. FAME, "D6.1 - Use Cases Specification and Pilot Sites Preparation I," 2023.
6. Goodell, J. W., Kumar, S., Lim, W. M., & Pattnaik, D. (2021). *Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis*. Journal of Behavioral and Experimental Finance, 32, 100577.
7. Das, S. R. (2014). *Text and context: Language analytics in finance*. Foundations and Trends® in Finance, 8(3), 145-261.
8. do Prado, J. W., de Castro Alcântara, V., de Melo Carvalho, F., Vieira, K. C., Machado, L. K. C., & Tonelli, D. F. (2016). *Multivariate analysis of credit risk and bankruptcy research data: a bibliometric study involving different knowledge fields* (1968–2014). Scientometrics, 106, 1007-1029.
9. West, J., & Bhattacharya, M. (2016). *Intelligent financial fraud detection: a comprehensive review*. Computers & security, 57, 47-66.
10. Al-Gasaymeh, A. S., Almahadin, H. A., Shehadeh, M., Migdady, H., & Atta, A. A. B. (2023, March). *Investigating the Role of Artificial Intelligence in Embedded Finance on Improving a Nonfinancial Customer Experience*. In 2023 International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-6). IEEE.
11. El Khatib, M., Al Mulla, A., & Al Ketbi, W. (2022). *The Role of Blockchain in E-Governance and Decision-Making in Project and Program Management*. Advances in Internet of Things, 12(3), 88-109.
12. Mărgărit, M. I. (2021). *Considerations on financial technology and the impact on financial system*. In Organizations and Performance in a Complex World: 26th International Economic Conference of Sibiu (IECS) 26 (pp. 133-143). Springer International Publishing.
13. Man, X., Luo, T., & Lin, J. (2019, May). *Financial sentiment analysis (fsa): A survey*. In 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS) (pp. 617-622). IEEE.
14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is all you need. Advances in neural information processing systems*, 30.
15. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *Bert: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805.
16. Fischer, T., & Krauss, C. (2018). *Deep learning with long short-term memory networks for financial market predictions*. European journal of operational research, 270(2), 654-669.
17. Hutto, C., & Gilbert, E. (2014, May). *Vader: A parsimonious rule-based model for sentiment analysis of social media text.* In Proceedings of the international AAAI conference on web and social media (Vol. 8, No. 1, pp. 216-225).
18. Bengio, Y., Ducharme, R., & Vincent, P. (2000). *A neural probabilistic language model.* Advances in neural information processing systems, 13.
19. C. Meske, E. Bunde, J. Schneider, M. Gersch, *Explainable artificial intelligence: objectives, stakeholders, and future research opportunities*, Information Systems Management 39 (2022) 53–63.
20. A. Adadi, M. Berrada, *Peeking inside the black-box: a survey on explainable artificial intelligence (xai)*, IEEE access 6 (2018) 52138–52160.

21. S. Verma, A. Lahiri, J. P. Dickerson, S.-I. Lee, *Pitfalls of explainable ml: An industry perspective* (2021). arXiv:2106.07758.
22. J. D. Lee, K. A. See, *Trust in automation: Designing for appropriate reliance* (2004). doi:10.1518/hfes.46.1.50_30392.
23. M. Weske, Business *Process Management Architectures, in: Business Process Management*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2019, pp. 351–384. doi:10.1007/978-3-662-59432-2_8. URL http://link.springer.com/10.1007/978-3-662-59432-2_8
24. W. van der Aalst, *Process Mining*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. doi:10.1007/978-3-662-49851-4. URL http://link.springer.com/10.1007/978-3-662-49851-4.
25. J. R. Rehse, N. Mehdiyev, P. Fettke, *Towards Explainable Process Predictions for Industry 4.0 in the DFKI-Smart-Lego-Factory*, KI - Kunstliche Intelligenz 33 (2) (2019). doi:10.1007/s13218-019-00586-1.
26. R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, *A survey of methods for explaining black box models*, ACM Computing Surveys 51 (5) (2018). doi:10.1145/3236009.
27. M. T. Ribeiro, S. Singh, C. Guestrin, *"Why should i trust you?" Explaining the predictions of any classifier*, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Vol. 13-17-August-2016, 2016, pp. 1135–1144. doi:10.1145/2939672.2939778.
28. S. M. Lundberg, S. I. Lee, *A unified approach to interpreting model predictions*, in: Advances in Neural Information Processing Systems, Vol. 2017-December, 2017, pp. 4768–4777.
29. A. Chandrasekaran, L. Ramos, Hype *Cycle for Generative AI*, Tech. rep., Gartner (9 2023).
30. P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, G. Neubig, *Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing*, ACM Computing Surveys 55 (9) (2023). doi:10.1145/3560815.
31. B. Meskó, *Prompt Engineering as an Important Emerging Skill for Medical Professionals: Tutorial*, Journal of Medical Internet Research 25 (2023) e50638. doi:10.2196/50638.
32. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving Language Understanding by Generative Pre-Training*. OpenAI.Com.
33. Mutinda, J., Mwangi, W., & Okeyo, G. (2023). *Sentiment Analysis of Text Reviews Using Lexicon-Enhanced Bert Embedding (LeBERT) Model with Convolutional Neural Network*. Applied Sciences (Switzerland), 13(3). https://doi.org/10.3390/app13031445
34. Kheiri, K., & Karimi, H. (2023). *SentimentGPT: Exploiting GPT for Advanced Sentiment Analysis and its Departure from Current Machine Learning*. ArXiv. /abs/2307.10234
35. Fatouros, G., Soldatos, J., Kouroumali, K., Makridis, G., & Kyriazis, D. (2023). *Transforming Sentiment Analysis in the Financial Domain with ChatGPT*. ArXiv. https://doi.org/10.1016/j.mlwa.2023.100508.
36. Fahland, D., Fournier, F., Limonad, L., Skarbovsky, I., & Swevels, A. J. (2024). *How well can large language models explain business processes?* ArXiv. /abs/2401.12846
37. Mavrepis, P., Makridis, G., Fatouros, G., Koukos, V., Separdani, M. M., & Kyriazis, D. (2024). *XAI for All: Can Large Language Models Simplify Explainable AI?* ArXiv. /abs/2401.13110
38. Ding, Q., Ding, D., Wang, Y., Guan, C., & Ding, B. (2023). *Unraveling the landscape of large language models: a systematic review and future perspectives.* Journal of Electronic Business & Digital Economics. https://doi.org/10.1108/JEBDE-08-2023-0015
39. Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. *Explainability for Large Language Models: A Survey*. ACM Trans. Intell. Syst. Technol. Just Accepted (January 2024). https://doi.org/10.1145/3639372
40. Bhattacharjee, A., Moraffah, R., Garland, J., & Liu, H. (2023). *Towards LLM-guided Causal Explainability for Black-box Text Classifiers*. ArXiv. /abs/2309.13340

41. B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale, *A review of temporal data visualizations based on space-time cube operations*, in Proc. Eurographics Conf. Vis., 2014, pp. 23–41.
42. W. Aigner, S. Miksch, H. Schumann, and C. Tominski, *Visualization of Time-Oriented Data*, 1st ed., Berlin, Germany: Springer, 2011.
43. H. Hochheiser and B. Shneiderman, *Dynamic query tools for time series data sets: Timebox widgets for interactive exploration*, Inf. Vis., vol. 3, no. 1, pp. 1–18, Mar. 2004
44. S. Havre, B. Hetzler, and L. Nowell, *Themeriver: Visualizing theme changes over time*, in Proc. IEEE Symp Inf. Vis., 2000, pp. 115–123.
45. R. Peng, *A method for visualizing multivariate time series data*, J. Statistical Softw. Code Snippets, vol. 25, no. 1, pp. 1–17, Feb. 2008.
46. T. D. Wang, C. Plaisant, B. Shneiderman, N. Spring, D. Roseman, G. Marchand, V. Mukherjee, and M. Smith, *Temporal summaries: Supporting temporal categorical searching, aggregation and comparison*, IEEE Trans. Vis. Comput. Graph., vol. 15, no. 6, pp. 1049–1056, Nov. 2009.
47. S. Haroz, R. Kosara, and S. L. Franconeri, *The connected scatterplot for presenting paired time series*, IEEE Trans. Vis. Comput. Graph., vol. 22, no. 9, pp. 2174–2186, Sep. 2016.
48. B. Bach, C. Shi, N. Heulot, T. Madhyastha, T. Grabowski, and P. Dragicevic, *Time curves: Folding time to visualize patterns of temporal evolution in data*, IEEE Trans. Vis. Comput. Graph., vol. 22, no. 1, pp. 559–568, Jan. 2016.
49. H. J€anicke and G. Scheuermann, *Steady visualization of the dynamics in fluids using epsilon-machines*, Comput. Graph., vol. 33, no. 5, pp. 597–606, 2009.
50. Y. Gu and C. Wang, *Transgraph: Hierarchical exploration of transition relationships in time-varying volumetric data*, IEEE Trans. Vis. Comput. Graph., vol. 17, no. 12, pp. 2015–2024, Dec. 2011.
51. Carter, B., Mueller, J., Jain, S., & Gifford, D. (2020). *What made you do this? Understanding black-box decisions with sufficient input subsets*. AISTATS 2019 - 22nd International Conference on Artificial Intelligence and Statistics.
52. F. Mannhardt, M. De Leoni, H. A. Reijers, *Heuristic mining revamped: An interactive, data-Aware, and conformance-Aware miner*, in: CEUR Workshop Proceedings, Vol. 1920, 2017, pp. 1–5.
53. F. Fournier, L. Limonad, I. Skarbovsky, Y. David, *The WHY in Business Processes: Discovery of Causal Execution Dependencies* (10 2023). doi:10.48550/arXiv.2310.14975. URL https://arxiv.org/abs/2310.14975v1.
54. S. Shimizu, Statistical Causal Discovery: *LiNGAM Approach, SpringerBriefs in Statistics*, Springer Japan, Tokyo, 2022. doi:10.1007/978-4-431-55784-5. URL https://link.springer.com/10.1007/978-4-431-55784-5.
55. G. Amit, F. Fournier, L. Limonad, I. Skarbovsky, *Situation-Aware eXplainability for Business Processes Enabled by Complex Events*, in: Business Process Management Workshops, Vol. 460, Springer, Cham, 2023, pp. 45–57. doi:10.1007/978-3-031-25383- 6_5.
56. G. Amit, F. Fournier, S. Gur, L. Limonad, *Model-informed LIME Extension for Business Process Explainability*, in: CEUR Workshop Proceedings, Vol. 3310, 2022, pp. 1–12.
57. I. Lage, E. Chen, J. He, M. Narayanan, B. Kim, S. Gershman, and F. Doshi-Velez (2019), *An evaluation of the human-interpretability of explanation*, arXiv preprint arXiv:1902.00006.
58. M. Narayanan, E. Chen, J. He, B. Kim, S. Gershman, and F. Doshi-Velez (2018), *How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation*, arXiv preprint arXiv:1802.00682
59. C. Molnar, G. Casalicchio, and B. Bischl (2019), *Quantifying interpretability of arbitrary machine learning models through functional decomposition*

60. A. B. Arrieta, N. D´ıaz-Rodr´ıguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garc´ıa, S. Gil-L´opez, D. Molina, R. Benjamins et al. (2020), *Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai*, Information fusion, vol. 58, pp. 82–115
61. D. Pruthi, R. Bansal, B. Dhingra, L. B. Soares, M. Collins, Z. C. Lipton, G. Neubig, and W. W. Cohen (2022), *Evaluating explanations: How much do explanations from the teacher aid students?*, Transactions of the Association for Computational Linguistics, vol. 10, pp. 359–375
62. J. Litman (2023), *Measures for explainable ai: Explanation goodness, user satisfaction, mental models, curiosity, trust, and human-ai performance.*
63. F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. W. Wortman Vaughan, and H. Wallach (2021), *Manipulating and measuring model interpretability*, in Proceedings of the 2021 CHI conference on human factors in computing systems, pp. 1–52.
64. A. Holzinger, M. Plass, M. Kickmeier-Rust, K. Holzinger, G. C. Cris¸an, C.-M. Pintea, and V. Palade (2019), *Interactive machine learning: experimental evidence for the human in the algorithmic loop: A case study on ant colony optimization*, Applied Intelligence, vol. 49, pp. 2401–2414.
65. L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal (2018), *Explaining explanations: An overview of interpretability of machine learning*, in 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA). IEEE, pp. 80–89.