ASSESSMENT OF LEARNING TECHNOLOGIES AND FRAMEWORKS FOR INTELLIGENT AND ETHICAL AI

D3.3 Initial AI tools for multilingual interaction and automatic generation of AI models

Deliverable No:	D3.3
Deliverable Name:	Initial AI tools for multilingual interaction and automatic generation of AI models
Work Package:	WP3
Task:	T3.2, T3.4
Dissemination Level:	PU
Deliverable Type:	R
Lead Organization:	TU/e
Submission month:	M18
Date:	31 March 2026

Funded by the European Union

Funded by the European Union under Grant Agreement 101177912. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the granting authority can be held responsible for them.

Project description

Acronym	**ALFIE**
Title	**Assessment of Learning technologies and Frameworks for Intelligent and Ethical AI**
Coordinator	CATALINK
Reference	101177912
Type	Research and Innovation Actions (RIA)
Programme	**Horizon Europe (HORIZON)**
Topic	HORIZON-CL2-2024-TRANSFORMATIONS-01-06 Beyond the horizon: A human-friendly deployment of artificial intelligence and related technologies
Start	01.10.2024
Duration	36 months
Website	alfie-project.eu
Consortium	**Catalink Limited** (CTL)**,** Cyprus, Coordinator **University of Brighton** (UoB), United Kingdom **Centre for Research & Technology** (CERTH)**,** Greece **Distributed Analytics Solutions LTD** (DAS), United Kingdom **Edge Hill University** (EHU), United Kingdom **Kempelen Institute of Intelligent Technologies** (KINIT), Slovakia **Universitat Autònoma de Barcelona** (UAB), Spain **Robert Bosch Espana SLU** (Bosch), Spain **Eindhoven University of Technology (**TU/e), Netherlands **Diadikasia Business Consulting** (DBC), Greece

Document history

Version	Date	Beneficiary	Description
0.1	02.02.2026	TU/e	Creating initial ToC
0.2	25.02.2026	UoB	Added content and Review
0.3	05.03.2026	EHU	Add content
0.4	11.03.2026	Bosch	Reviewed and proposed amendments
0.5	13.03.2026	EHU	Reviewed and proposed amendments
0.6	19.03.2026	CTL	Reviewed and proposed amendments
1.0	20.03.2026	TU/e	Final version

Authors

Partner	Name(s)
TU/e	Subhaditya Mukherjee
EHU	Chukwudi Uwasomba, Nonso Nnamoko, Yannis Korkontzelos
UoB	Ignacio Cabrera Martin
KInIT	Viktor Kostan
CERTH	Sotiris Diplaris, Alexandros Vassiliades, Stamatis Samaras, Spyridon Symeonidis

Contributors

Partner	Contribution type	Name
Bosch	Review	Juan Antonio Recio Garcia
EHU	Review	Yannis Korkontzelos

Glossary

Acronym	Definition
AI	Artificial Intelligence
ARIA	Accessible Rich Internet Applications
AutoDW	Automated Data Wrangling
AutoML	Automated Machine Learning
AutoML+	Extended Automated Machine Learning
BPS	Business Partner Screening
CD	Concept Drift
DOM	Document Object Model
EDA	Exploratory Data Analysis
ETD-Hub	EthiTech Dialogue Hub
GDPR	General Data Protection Regulation
JSON	JavaScript Object Notation
LLM	Large Language Model
ML	Machine learning
SemKG	Semantic Knowledge Graph
SHAP	SHapley Additive exPlanations
SMOTE	Synthetic Minority Over-sampling Technique
SOTA	State of the Art
STT	Speech-to-Text
TTS	Text-to-Speech
WCAG	Web Content Accessibility Guidelines
XAI	Explainable Artificial Intelligence

Executive Summary

This deliverable presents the initial AI tools co-designed and developed within the ALFIE project, with a primary focus on the AutoML and AutoML+ engines and their role in generating unbiased, ethically sound, and EU-compliant AI models. It outlines the technical foundations, operational workflows, and integration mechanisms that enable robust, transparent, and multilingual AI-driven solutions across ALFIE's trial scenarios.

At its core, the document details the end-to-end model generation process implemented by the AutoML+ engine, explaining how user requirements, expressed in any supported language, are automatically understood, processed, and translated into optimized AI pipelines. This multilingual capability allows users to interact with the system in their preferred language, after which the AutoML engine interprets and acts on the inputs, and subsequently translates the outputs back for the user. The document further describes the distinction between AutoML and AutoML+ and elaborates on the state-of-the-art techniques embedded in the platform, including transfer learning, hyperparameter optimization, meta-learning, and multi-objective optimization. These techniques enable efficient model selection, improved performance, and adaptability across diverse use cases, while balancing accuracy, fairness, interpretability, and computational efficiency.

A complementary contribution of this deliverable is the presentation of the multilingual interaction framework. The architecture supporting both user interface multilinguality and multilingual natural language interaction is described in detail, alongside the supported languages for both the chatbot and the Agentic Core. The conversational interface is designed to simplify complex AI processes, enabling users to interact with the system intuitively and transparently, regardless of linguistic background.

The document further introduces ALFIE's ethical model generation framework. It explains how the AutoML engine incorporates drift and bias detection mechanisms prior to model training, ensuring that datasets are assessed for fairness and representativeness before pipeline construction. In addition, concept drift detection and adaptive model updating mechanisms are applied post-deployment to maintain performance, compliance, and trustworthiness in dynamic environments. Transparency and explainability outputs, as well as comprehensive traceability and logging mechanisms, are embedded throughout the workflow to support accountability and alignment with EU regulatory requirements.

The application of AutoML within the ALFIE trial contexts, website accessibility, autonomous vehicle bias detection, and compliance screening, is also discussed, demonstrating how the platform's capabilities are operationalized in real-world scenarios. These examples illustrate how automated, ethically guided model generation supports diverse domains with varying technical and regulatory constraints.

Finally, the deliverable describes how the AutoML engine works with other ALFIE components, including the AutoDW, the Agentic Core, ETD-Hub, XAI and semantic knowledge graph. These components provide contextual, semantic, and meta-level information that informs and enhances model generation, ensuring coherence across the platform. Through this architecture, ALFIE enables users to create robust, transparent, and unbiased AI models in a structured, efficient, and ethically grounded manner.

Contents

Figures

Tables

1 Introduction

1.1 Overview

The scope of this deliverable is to present the initial AI tools co-designed by the ALFIE partners. In particular, it describes the development of AI models that are unbiased, ethically sound, and fully compliant with EU regulations, as well as the tools that enable multilingual interaction within ALFIE.

This deliverable explores the concepts of AutoML, transfer learning, meta-learning, and other state-of-the-art techniques that support the efficient selection of AI models and pipelines. It explains how these approaches contribute to progress across the three use cases and support upcoming tasks. In addition, it provides a detailed description of the operational workflow of the AutoML and AutoML+ engines, outlining the end-to-end model generation process and the advanced techniques applied throughout.

The document also presents the multilingual interaction framework, including its architecture and supported languages, and describes how the chatbot is designed to simplify and enhance the user experience.

Furthermore, the deliverable details the ethical model generation process. It explains how the AutoML engine performs drift and bias detection prior to model training, as well as concept drift detection and model adaptation after deployment, ensuring continuous compliance and reliability. Given ALFIE's strong emphasis on transparency and logging, these aspects are also thoroughly discussed.

Finally, the deliverable outlines how other ALFIE components, such as AutoDW, the Agentic Core, and ETD-Hub, interact with the AutoML engine to support users in creating new, robust, and unbiased models.

1.2 Relation to other tasks and deliverables

This deliverable is related to the following other ALFIE tasks and deliverables:

Receives inputs from:

Table 1. D3.3 Input from other tasks and deliverables

Deliverable	Due Date	Input for D3.3
D1.4	31 Mar 2025	Data and privacy management plan
D2.1	30 Sep 2025	Ethical gaps and perspective on how to serve AI models
D2.5	31 Jan 2026	EU policy recommendations and guide for designing an ethical AI framework
D3.1	31 Mar 2026	Data Warehouse to retrieve/store models and datasets. Semantic knowledge graph to display useful ethical recommendations to the user based on the task. Intent recognition to identify what task the user wants to perform.

D3.5	31 Mar 2026	Information about XAI and evolving AI models
D4.1	30 Sep 2025	System architecture for the AI tools
D4.5	31 Jul 2025	Use cases to test on
D5.1	31 Jan 2025	User feedback
D5.2	31 Mar 2026	User feedback
D6.1	31 Oct 2024	Ethical information for the AI
D6.2	31 Mar 2025	Ethical information for the AI
D6.3	31 Dec 2025	Ethical information for the AI
D6.5	31 Mar 2026	Ethical information for the AI

Provides outputs to:

Table 2. D3.3 Output for other tasks and deliverables

Deliverable	Due Date	Output from D3.3
D2.2	30 Nov 2026	Ethical models
D2.6	31 May 2027	Integration with Ethical AI and AutoML
D3.2	30 Jun 2027	Outputs saved to AutoDW
D3.6	30 Jun 2027	Models and metadata to XAI and evolving AI
D4.2	31 Jan 2027	AutoML engine
D4.3	30 Apr 2026	Prototype of AutoML engine
D4.4	31 Jul 2027	Prototype of AutoML engine
D4.6	31 May 2026	AutoML models
D4.7	30 Sep 2027	AutoML models

1.3 Structure of the deliverable

The structure of the deliverable is as follows.

Section 1 introduces the objectives and scope of the document, outlines its relation to other ALFIE tasks and deliverables, and presents its overall structure.

Section 2 describes the key terms and concepts that are discussed in this deliverable. It also describes the operational workflow of the AutoML+ engine, detailing the end-to-end model generation process and clarifying the differences between AutoML and AutoML+.

Section 3 outlines the SOTA techniques used in automatically generating unbiased AI models.

Section 4 focuses on multilingual interaction with the end user. It defines the scope of multilingual support at both user interface and natural language interaction levels, presents the system architecture, and lists the supported languages for the web interface, chatbot, and Agentic Core.

Section 5 outlines the ethical model generation framework. It explains the mechanisms for drift and bias detection prior to training, concept drift detection and model adaptation after deployment, and describes the transparency, explainability, traceability, and logging mechanisms integrated into the AutoML workflow.

Section 6 describes how other ALFIE components inform and support model generation. It explains the interaction with AutoDW for meta-information, the coordination with the Agentic Core, the contribution of ETD-Hub, and the role of semantic knowledge graphs in enhancing AutoML model generation.

Section 7 presents the application of AutoML within the ALFIE trial contexts, detailing its use in the website accessibility scenario, the drowsiness detection for autonomous vehicle scenario, and the compliance screening scenario.

2 Key Terms and concepts

2.1 Concepts

AutoML: Automated Machine Learning (AutoML) automates the process of model selection, feature engineering, and training to accelerate AI development. It reduces the need for manual intervention by systematically evaluating multiple algorithms and optimization strategies for the best-performing model.

AutoDW: Auto Data Warehouse (AutoDW) is the centralized system where AI models, outputs, datasets and other artefacts are stored and version-controlled. It ensures consistent data access for training, evaluation, and deployment, integrating metadata and lineage tracking for transparency across ALFIE.

Training AI models: Training AI models involves feeding data into an algorithm that adjusts internal parameters to learn patterns and make predictions. It requires iterative optimization using training datasets, validation checks, and fine-tuning to achieve reliable generalization. This is usually a very costly process.

Hyperparameter optimization: Hyperparameter optimization is the process of automatically tuning configuration parameters that control how a model learns. It uses methods like grid search, random search, or Bayesian optimization to find parameter sets that maximize accuracy or minimize loss.

Compute cost: Compute cost refers to the total amount of computational resources CPU, GPU, energy required to train, test, or serve a model. It significantly impacts scalability and efficiency, influencing decisions about model complexity, training frequency, and deployment strategy.

Green AI: The sustained attempt on choosing models and paradigms that use the least possible resources on a given task. This is a major focus for the AutoML engine, which uses a variety of techniques to ensure the least possible energy footprint by using smaller models, transfer learning etc.

Agentic Core: The Agentic Core is an intelligent controller that determines which AI or software service to invoke based on user intent and context. It coordinates data flow, tool selection, and response synthesis, acting as the central decision-making component in ALFIE.

Bias in AI models: Bias in AI models occurs when algorithms produce systematically skewed outcomes due to imbalanced data or flawed design. It can reinforce existing prejudices or inaccuracies, making fairness auditing and bias mitigation essential in ethical AI workflows.

XAI: Explainable Artificial Intelligence (XAI) encompasses methods that make AI decisions interpretable and understandable by humans. It provides insights into model reasoning through visualizations, feature impact scores, or example-based explanations.

Concept Drift: Concept Drift refers to the phenomenon where the statistical properties of input data change over time, degrading model performance. Detecting and adapting to drift ensures AI systems remain accurate and relevant in dynamic, real-world environments.

2.2 AutoML vs AutoML +

AutoML is used to automate the process of selecting, training, and hyperparameter-tuning models to yield the best single model for a given dataset within a defined compute budget (this budget is either supplied by the user or automatically chosen based on the task). But this paradigm does not fully cover ALFIE's use cases, thus we have AutoML+ that combines AutoML ideas with LLMs and auxiliary tools to tackle end-to-end tasks that require reasoning, data transformation, or multi-step workflows. It aims to orchestrate multiple components (models, prompts, tools) to achieve higher-level objectives than a single model could alone.

AutoML for example would consider model families such as GBMs [14], XGBoost [15], ensemble models, neural networks etc and perform automated preprocessing, feature engineering, and have a hyperparameter optimization loop. There is a focus on maximizing predictive performance under constraints like compute time, memory, and data availability. This is the choice made when the user wants a robust, well-performing model with limited machine learning (ML) expertise and a dataset available.

AutoML+ on the other hand would use LLMs for natural language reasoning, code generation, or guidance. There, it is possible to have integration with external tools or APIs (e.g., evaluation pipelines, data labeling, retrieval, or task-specific modules) and orchestrated decision logic to select which model or tool to apply at each step. Here, the system considers performance with higher-level task success metrics and often leverages human-in-the-loop or domain-specific heuristics. The system chooses this option when domain tasks require planning, multi-step reasoning, or integration with external data sources and tools; helpful when annotated data is scarce but you can leverage external knowledge or automation around tasks.

2.3 End-to-End model generation process

Consider the use case of producing an AutoML model given a tabular dataset. Once the designer user has created a new account on ALFIE using the frontend, they are allowed to directly prompt the Agentic Core to create a new model. The agentic core chats with the user, and once it has verified that there are appropriate training datasets and queries to run or has informed the user sufficiently, the AutoML pipeline begins.

The data is stored in the AutoDW, where the relevant train and testing splits are performed. The bias detector runs on this modified data set and shows it to the user. It attempts to automatically fix minor issues in the data set but does not make any major changes without consent. Once the user confirms the changes, the AutoML engine is called.

The AutoML engine takes into account data from the ETD-Hub about ethical cases and use case-specific recommendations provided by the community at large. Additional information is obtained from the Semantic Knowledge Graph component, which might provide extra information about the specific use case if it exists. Once all of this information has been processed and synthesized, the AutoML identifies the type of data and attempts to generate the best model for the given task. If it is a tabular data set like the current example, it uses AutoGluon[10], a library known for tabular AutoML; if it is a vision task, then a custom PyTorch trainer is used. Once the model has been generated, it, along with its predictions and other meta-information, are stored in the AutoDW.

The XAI tools are then called on this data set and model to generate an explainability report based on the trained model. The user is also provided with information on how to download

and use the given model, along with an approximate estimate of the resource consumption. Note that ALFIE does not deploy the models provided to the user. Since all of these outputs are directly available to the Agentic Core, it is possible to some extent for the user to ask questions based on these outputs and receive responses in real time. There is also a concept drift detector that runs to determine if the data has drifted over time and if the model needs to be retrained. Once all of these steps have been performed, the user can download the model, read the instructions, examine the ethical information provided, and then get up and running with the model. More detailed information about each of these steps and how they relate to automatically generating AI models are present in the sections below.

3 SOTA techniques for automatic generation of AI models

The combination of transfer learning, hyperparameter optimization, knowledge transfer, and multi-objective optimization forms the methodological backbone of the AutoML engine. Transfer learning provides powerful starting points by leveraging pre-trained backbones, hyperparameter optimization systematically tailors these backbones to new tasks, meta-learning accelerates this process by bringing in prior experience, and multi-objective optimization ensures that the final solutions meet real operational constraints. In an applied setting such as ALFIE, this combination allows non-expert users to access robust, efficient models that are adapted to their data and requirements, even when labeled data is limited and computational resources are constrained.

Another advantage of using these methods is the energy footprint that they consume. In order to be more green, the AutoML engine uses the smallest possible model and techniques such as transfer learning that still provide a good balance of accuracy.

This section provides more details on each of these paradigms.

3.1 Transfer Learning

Transfer learning has become a central technique in modern machine learning [11] because it makes it possible to reuse knowledge from large, pre-trained models and adapt it to new tasks with relatively little data. Instead of training a model from scratch, which is computationally expensive and data-hungry, a pre-trained backbone is selected that has already learned rich, generic representations from a broad source dataset. This backbone is then fine-tuned or used as a fixed feature extractor on a smaller, task-specific dataset. In many cases, this approach achieves competitive or superior performance compared with models trained from scratch, while significantly reducing training time, compute cost, and the amount of labeled data required. For systems like ALFIE, where end users may not always possess large, carefully annotated datasets, transfer learning is particularly attractive: it allows them to leverage standardized, well-tested backbone models as a starting point and build reliable solutions from limited supervision.

Within this paradigm, the teacher–student analogy is a useful way to understand how transfer learning can be applied in practice. A large, powerful pre-trained model can be viewed as the “teacher” that has acquired broad knowledge from extensive training, while a smaller or task-specific model acts as the “student” that learns from this prior knowledge. Techniques such as fine-tuning, feature reuse, and knowledge distillation allow the student model to benefit from the teacher’s representations or outputs without needing to see the full volume of data that the teacher was originally trained on. This not only reduces the overall amount of data and compute required for the student but also opens up the possibility of deploying smaller, more efficient models that nevertheless retain much of the teacher’s performance. For ALFIE, this pattern is valuable because it supports training effective models even when users have only modest datasets, while still maintaining feasible inference costs and deployment constraints.

3.2 Hyperparameter optimization

Hyperparameter optimization plays a complementary role to transfer learning by systematically searching for good configurations of training and model parameters under practical constraints. A model's performance often depends heavily on choices such as learning rate, regularization strength, network depth, number of trees, and various preprocessing or augmentation settings. Manually tuning these hyperparameters is time-consuming and error-prone, especially for non-experts. Automated hyperparameter optimization frameworks instead treat this as a structured search problem over a defined space. Strategies can range from simple grid and random search to more sophisticated Bayesian optimization, evolutionary algorithms, and multi-fidelity methods that allocate more resources to promising configurations while early-stopping poor ones. By constraining this process with realistic limits on runtime or compute, the system can discover well-performing configurations without exhaustive experimentation.

In the context of ALFIE, hyperparameter optimization is used to make the most of standardized backbone models. Rather than exploring arbitrary architectures from scratch, the system can fix a small number of strong candidate backbones and focus on adjusting their hyperparameters in a "smart" way. This allows the solution to adapt to the user's data and task while keeping computation manageable. Users benefit by receiving models that are fine-tuned for their specific use case, with minimal manual intervention. Importantly, the optimization process can be guided by metrics that reflect real-world objectives, such as accuracy, F1 score, or other domain-specific measures. By automating this search, the system lowers the barrier to deploying effective machine learning models and reduces the need for deep expertise in model tuning.

Computer vision tasks illustrate especially well how transfer learning and hyperparameter optimization can be combined. Modern vision models, such as convolutional neural networks and vision transformers, are typically pre-trained on large-scale image datasets. These pre-trained models provide generic visual features that transfer effectively to a wide range of downstream tasks, such as classification, detection, or segmentation. For a system like ALFIE, a pre-trained vision backbone can be attached to a lightweight task-specific head and then fine-tuned on the user's images. Hyperparameter optimization can then be used to select learning rates, data augmentation strategies, number of trainable layers, and training duration, subject to constraints on available compute time. In this way, vision models can be adapted efficiently, even when only a relatively small number of labeled examples are available.

Tabular data, which is common in business and scientific applications, presents a different set of modeling challenges but benefits from automation in a similar way. State-of-the-art tabular AutoML frameworks typically rely on strong base models such as gradient-boosted decision trees, random forests, and neural networks, sometimes augmented by automated feature engineering. These frameworks often use ensembles, combining multiple models to improve robustness and accuracy. Hyperparameter optimization is critical here as well: it governs both the configuration of individual models and the structure of the ensemble. By automating these choices, such systems can produce high-quality models without requiring users to design complex pipelines by hand. In ALFIE's case, adopting a mature tabular AutoML backend, together with automated hyperparameter tuning and ensembling, provides a practical and powerful solution for users working primarily with structured data.

3.3 Knowledge transfer

Rather than treating each new dataset as completely isolated, knowledge-transfer methods leverage prior experience to guide model selection [12], hyperparameter choices, and training strategies. Over time, the system can infer which model families and parameter settings tend to work well for certain types of data or problem structures. For an AutoML system integrated into a product like ALFIE, this means that each new user or dataset benefits from knowledge accumulated from previous deployments. As a result, the system can converge more quickly on effective solutions and may require fewer iterations of trial-and-error, thereby improving both performance and user experience.

3.4 Multi-objective optimization in practice

Multi-objective optimization reflects the fact that real-world machine learning systems rarely optimize a single metric [13]. In practice, there are trade-offs among accuracy, latency, memory usage, robustness, interpretability, and even energy consumption. A purely accuracy-driven solution might be unsuitable if it is too slow or too costly to run at scale. Multi-objective optimization explicitly acknowledges these trade-offs by searching for configurations that balance several criteria simultaneously. The result is often a set of Pareto-optimal solutions: configurations where improving one objective would necessarily worsen another. Decision-makers can then choose the most appropriate solution given their priorities, such as preferring slightly lower accuracy in exchange for significantly faster inference.

For ALFIE, multi-objective optimization can be applied to both training and deployment. During training, the system can constrain total training time, memory footprint, and hardware utilization while aiming to maximize predictive quality. During deployment, it can consider constraints like inference latency, throughput requirements, and cost per prediction. Hyperparameters and model choices are then evaluated not only on how well they fit the data, but also on how well they satisfy these operational constraints. This leads to more realistic model selection and ensures that the chosen models are viable in the environments where users plan to run them. By incorporating multi-objective optimization, the system becomes more aligned with practical requirements, rather than focusing narrowly on benchmark-style performance metrics.

4 Multilingual interaction with the end user

This section describes the multilingual natural language interaction that is a part of ALFIE's chat interface. It explains how the user can chat or talk in their own language and get results back in it as well. The engine uses English as an intermediate language and converts to and fro from the user's chosen language and English. This ensures that the user can just talk to the engine in the language of their choosing.

4.1 Scope

The scope of multilingual interaction within the ALFIE platform encompasses the ability of the system to support end users in multiple natural languages, both at the level of the **user web interface (UI)** and at the level of the **intelligent chatbot built upon the agentic core**.

Multilinguality in ALFIE is therefore addressed from two complementary viewpoints: user interface multilinguality and multilingual natural language interaction.

4.1.1 User web interface multilinguality

The ALFIE web interface is localized into selected European languages to ensure accessibility and inclusiveness. This layer ensures that users see the UI in their own language and can interact with the AutoML platform in their preferred language without requiring knowledge of English.

The localization of the user web interface, as well as the integration of speech-to-text (STT) and text-to-speech (TTS) components, will be described in detail in deliverable D4.3 – Initial prototype of AutoML platform and frontend development (M19).

4.1.2 Multilingual natural language interaction

Beyond static interface localization, ALFIE enables multilingual natural language interaction through its interaction layer and agentic core. The chatbot is capable of:

- Processing multilingual user input (free-text queries, task descriptions, requirements, etc.)
- Extracting structured information from multilingual instructions
- Translating user requirements into formal configurations for the AutoML engine
- Generating multilingual responses, explanations, etc.

The multilingual interaction layer contributes directly to the democratization of AI by lowering language barriers and enabling non-technical and non-English-speaking users to create and evaluate AI models through natural language descriptions.

4.2 Multilinguality as part of the system architecture

From the chatbot viewpoint, multilinguality is inherently supported by design, as it relies on Large Language Models (LLMs) forming the core of the ALFIE agentic architecture.

The main AI agent:

- Infers the language of the conversation automatically from user prompts.

- Responds in the language used by the user.

The interaction workflow is as follows:

- The user submits a query in natural language (e.g., describing a machine learning task).
- The interaction layer (LLM-based agent) interprets the request.
- The agent extracts structured intent and configuration parameters independent of the input language and then converts it to English, which is used by the AutoML engine. The agent may also use appropriate tools and request additional information.
- The structured configuration is passed to:
    - The AutoML engine
    - The AutoDW and other components
- Results are returned to the interaction layer.

The LLMs used as the core of the agents are multilingual foundation models trained on a large corpora of multilingual data. They can understand and generate text in a wide range of languages, including the languages of all consortium partners and, more broadly, most European languages. Currently ALFIE is using GPT-4o mini (an LLM by OpenAI).

The multilingual capability is therefore not implemented as a separate translation-only component but is integrated directly into the reasoning and orchestration logic of the agentic core. Where necessary, explicit translation steps may be applied within the conversation context in the future, for instance, when translating dataset metadata or external resources.

4.3 Supported languages

Multilingual support in ALFIE is divided into:

- (A) User web interface languages
- (B) Chatbot and agentic core languages

4.3.1 User web interface languages

The current web interface supports the following languages:

- English
- Greek
- Spanish
- Catalan
- Slovak
- Dutch

These languages correspond to consortium representation and initial deployment priorities. The UI localization framework is designed to be extensible, allowing additional European languages to be integrated in future iterations.

4.3.2 Chatbot and Agentic Core languages

The multilingual capabilities of the chatbot are primarily determined by the capabilities of the underlying foundation LLM.

The currently deployed model, GPT-4o mini, supports understanding and generation in a wide range of languages, including (but not limited to):

- Major European languages (e.g., English, Spanish, French, German, Italian, Dutch, Greek, Portuguese, Polish, Slovak, etc.).
- Selected minor and regional European languages.
- Additional globally used languages.

It should be noted that while the foundation model supports a broad range of languages, performance quality (e.g., fluency, technical terminology accuracy) may vary depending on the language and domain specificity. Continuous evaluation and fine-tuning activities within ALFIE aim to improve robustness across selected European languages relevant to the trial use cases.

The chatbot is able to communicate in multiple languages and switch a language on request. Looking at the Figure 1 below, the model fluently switched from English to Slovak while preserving its ability to understand user queries and respond accordingly.

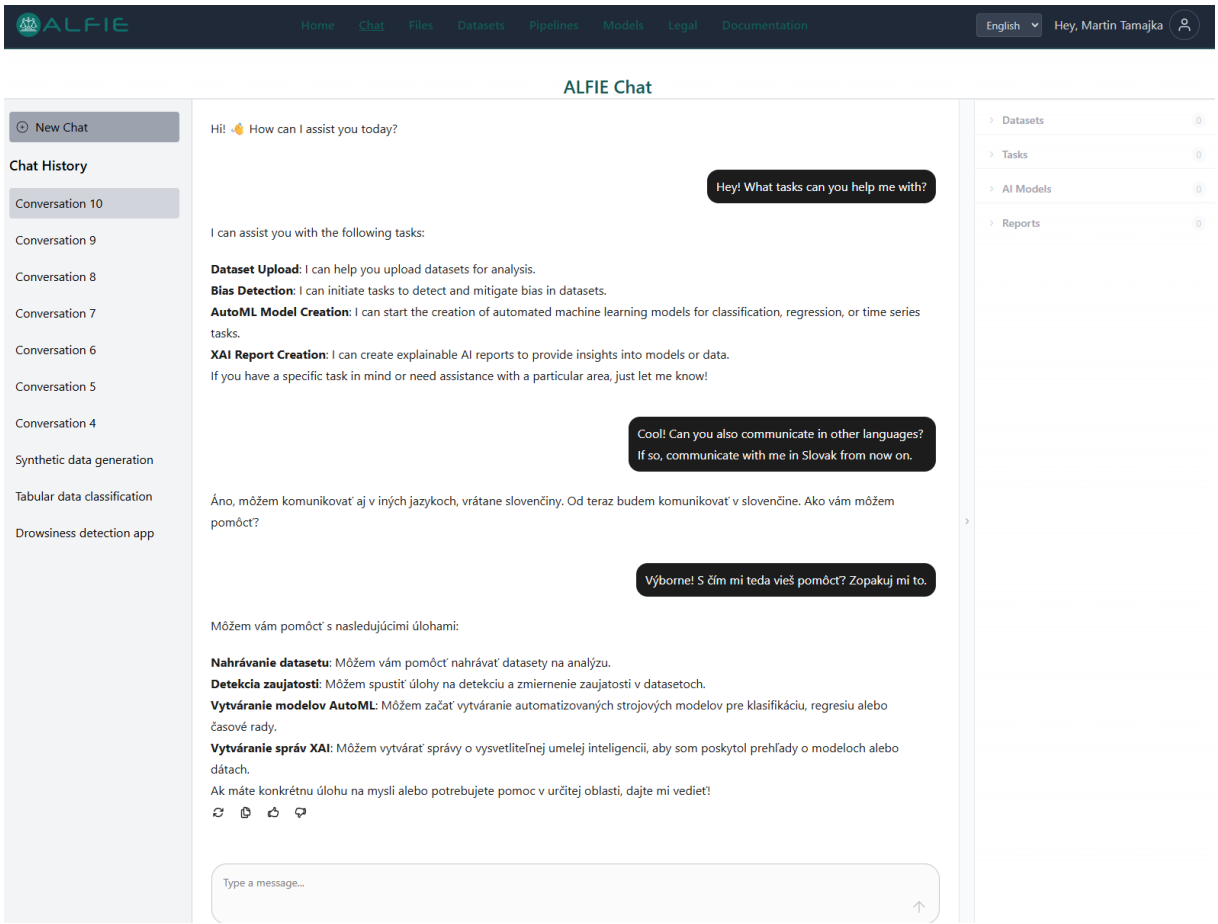

Figure 1. Chatbot switching languages on demand

5 Ethical model generation

Bias is quite a large problem in machine learning. This section describes how the AutoML engine provides the user information about how the model might perform based on the given data, how performance drifts over time and attempts to transparently describe how the model makes its decisions. In the ALFIE system architecture, there are multiple components that ensure ethical models are generated. These are further described in this section.

5.1 Drift and bias detection before training

In the broader shift toward fully automated machine learning, data processing pipelines increasingly operate as “black boxes,” often obscuring underlying biases, structural imbalances, and quality deficiencies embedded within datasets. The Detector/Mitigator component addresses this challenge by serving as an ethical anchor within the human-in-the-loop AutoML paradigm. Rather than treating data preparation as a purely mechanical preprocessing step, it reframes it as a transparent, inspectable, and accountable process that actively supports responsible AI development.

By embedding structured checkpoints and enabling meaningful human oversight, the component operationalizes three core ethical pillars: transparency, fairness, and accountability.

Transparency is established through comprehensive descriptive clarity. Ethical AI begins with a deep understanding of the data that fuels model development. The extensive Exploratory Data Analysis (EDA) conducted by the Detector module reduces opacity by systematically profiling dataset characteristics, including feature names, types, distributions, correlations, and anomalies. This level of profiling enables experts to identify ethically sensitive attributes, such as potential proxies for protected characteristics, before they influence model training. Furthermore, the generation of a detailed transformation log ensures full traceability. Every modification applied to the dataset, from imputation strategies to encoding choices and normalization techniques, is documented. This audit trail not only supports reproducibility and regulatory compliance but also safeguards against inadvertent or unjustified data manipulation.

Fairness is addressed through proactive bias detection and mitigation mechanisms embedded in the Mitigator module. Automated systems, if left unchecked, tend to replicate and amplify historical inequities present in training data. To counteract this, the component introduces targeted interventions. Techniques such as SMOTE (Synthetic Minority Over-sampling Technique) are used to address class imbalances that frequently disadvantage minority groups in predictive performance. Additionally, systematic handling of skewed distributions and outliers through normalization and scaling prevents models from overemphasizing extreme or non-representative data points. These measures contribute to more equitable model behavior across different segments of the population.

Accountability is reinforced through a co-evolutionary interaction between automation and human judgment. While the system performs large-scale statistical analysis and transformation efficiently, human experts retain a strategic oversight role. The EDA reports function as decision-support artifacts that enable informed validation before modeling proceeds.

In summary, the Detector/Mitigator component ensures that AutoML prioritizes not only efficiency and performance, but also statistical rigor grounded in domain awareness and

ethical reflection. By providing mechanisms to inspect, validate, and override automated decisions, it supports the development of AI systems that are not merely optimized, but responsibly aligned with human values and societal expectations.

5.2 Concept drift detection and model adaptation for streams of data

The lifecycle of automated machine learning, generating an initial model is only the beginning. Real-world data environments are dynamic: distributions shift, user behaviors evolve, and external conditions change. The Concept Drift (CD) Detector component represents a critical advancement within the human-in-the-loop AutoML framework by ensuring that automatically generated models remain reliable, adaptive, and aligned with evolving contexts. Rather than treating deployment as a static endpoint, this component reframes model maintenance as a continuous, transparent, and accountable process.

A central contribution of the CD Detector is robustness through systematic stress-testing. Model quality cannot be assessed solely under stable, historical conditions; it must also be evaluated under realistic and evolving data scenarios. To achieve this, the framework reserves a portion of the dataset to simulate a live streaming environment. Specifically, 15% of the available data is used to emulate real-time input, monitored using the ADWIN (Adaptive Windowing) algorithm. This enables continuous tracking of model performance and statistical properties in a setting that approximates operational deployment.

To avoid overestimating robustness in artificially stable conditions, the system also introduces controlled artificial drift at regular intervals, injecting distributional changes at every 10% segment of the data stream. This structured stress-testing allows developers to observe not only how performance degrades under drift conditions, but also how effectively the system recovers following adaptation. In doing so, the component transforms robustness evaluation into a measurable and reproducible process rather than an assumption.

While the CD Detector is capable of triggering automated retraining when drift is statistically detected, it deliberately avoids reverting to a black-box retraining cycle. Instead, adaptation is governed by interpretable statistical signals. Retraining is triggered based on defined parameters, such as a delta threshold and a specified window size, ensuring that updates occur only when meaningful distributional changes are detected rather than on arbitrary or fixed schedules. This reduces unnecessary computational costs while preserving responsiveness.

Crucially, automated detection is complemented by human oversight. Drift reports provide structured, interpretable summaries that allow experts to assess whether detected shifts reflect genuine data evolution, temporary anomalies, or broader domain-specific changes requiring policy-level intervention. This human review layer ensures that adaptation is not purely statistical but contextually informed. Experts retain the authority to validate, refine, or override automated retraining decisions when necessary.

The component further strengthens governance through the generation of auditable artifacts. Each adaptation cycle produces an updated and retrained model reflecting current data conditions, a detailed drift report documenting all detected shifts, and a comprehensive metrics report evaluating performance throughout the streaming simulation. Together, these artifacts create a transparent record of the model's evolution, a traceable "biography" of how data and model behavior have changed over time.

Ultimately, the Concept Drift Detector transforms AutoML from a one-time optimization event into a continuous, responsibly governed cycle. By combining automated large-scale monitoring with human interpretative oversight, the framework supports a co-evolutionary approach in which machines manage the scale and speed of detection while humans provide strategic judgment and contextual reasoning. This ensures that AI systems remain not only technically accurate, but also context-aware, accountable, and aligned with long-term operational and ethical objectives.

5.3 Transparency and explainability (XAI) outputs

Transparency and explainability are integrated as intrinsic capabilities of the ALFIE AutoML and AutoML+ engines. Every model produced through the AutoML pipeline is accompanied by structured metadata, preprocessing documentation, and explainability artifacts that together ensure reproducibility and interpretability (for more information about XAI, please refer to deliverable D3.5).

The ALFIE XAI framework is structured around three core design principles - Model-agnostic explainability, ensuring interpretability services remain stable regardless of the estimator family selected during optimization. Joint data-level and model-level analysis, coupling feature attributions with dataset diagnostics such as missingness, proxy correlations, subgroup imbalance, and distribution shift to reduce the risk of misinterpretation.

Transparency is established at the data preparation stage, where the Detector/Mitigator component profiles dataset structure and records all preprocessing operations including imputation strategies, encoding methods, scaling transformations, and imbalance mitigation in a transformation log. This ensures the training dataset can be reconstructed deterministically and the relationship between raw input data and the final model-ready representation remains fully traceable. At the model generation stage, the AutoML engine persists the selected model family, hyperparameter configurations, validation strategy, performance metrics, and computational footprint, enabling reproducibility of experiments and comparison across iterations. In ensemble-based pipelines, the composition of base learners and stacking logic is also persisted. In AutoML+ scenarios, the structured configuration extracted from the user's natural language prompt is stored alongside model artifacts, creating a traceable link between user intent and model configuration. Operational auditability is ensured through the Task Manager's asynchronous execution architecture, with execution states, timestamps, and output references logged via Kafka events and persisted in the system database.

Following training, the XAI module produces both global and local explainability outputs. Global explanations characterize dominant predictive features and error distributions across the dataset, typically derived using model-agnostic attribution methods such as SHAP, complemented by correlation-oriented analyses and residual diagnostics, providing a stable overview of which variables contribute most significantly to predictions and whether error distributions exhibit structural bias or instability. Local explanations operate at the instance level, providing additive contribution decompositions that indicate how individual feature values influenced the prediction relative to a baseline.

Transparency is not limited to initial model deployment. The Concept Drift Detector continuously monitors performance on streaming data, and when retraining is triggered, the newly generated model is versioned and updated explainability reports are produced. By linking drift detection logs with model versions and explanation artifacts, the system enables

comparison of feature influence patterns before and after adaptation, supporting a coherent narrative of how model behavior evolves over time. Conversational access to all transparency artifacts is further provided through the Agentic Core, enabling users to query preprocessing steps, dominant predictive features, and instance-level decisions without direct interaction with raw technical outputs, lowering the barrier to understanding automated pipelines and strengthening user trust in generated models. Overall, the framework ensures that performance optimization does not come at the cost of interpretability, producing an AutoML environment capable of generating high-quality models while preserving clarity regarding how data was processed, how models were configured, and how predictions are formed.

6 Using ALFIE components to inform model generation

This section describes how each of the components in ALFIE integrate with each other to produce the best model for the job.

6.1 Interaction with AutoDW for meta information

The integration between the AutoML and AutoDW is central to the platform's ability to ensure transparency, reproducibility, and version control across the machine learning lifecycle. This interaction is governed by an asynchronous, event-driven handshake mediated by Kafka and supported by RESTful API transactions for large artifact transfers.

A summary of integration between the AutoML and AutoDW and all the data exchanges that take place between their communication is shown on Table 3.

Table 3. Summary of integration between the AutoML and AutoDW and their data exchanges.

Data Type	Direction	Storage Layer	Purpose
Dataset Pointer	AutoDW → AutoML	Kafka (automl-trigger)	Initiates training with specific data version.
Model Artifacts	AutoML → AutoDW	MinIO (Object Storage)	Stores physical files (PKL, ONNX, etc.).
Training Metrics	AutoML → AutoDW	MongoDB (Metadata)	Records accuracy, F1-scores, and framework info.
Lineage Info	AutoML → AutoDW	MongoDB (Metadata)	Links model vX to dataset vY for reproducibility.

Metadata acquisition and triggering: The AutoML process begins by consuming a trigger event from the `automl-trigger-events` Kafka topic. This message contains critical "meta-pointers," including the `user_id`, `dataset_id`, and the specific `version` (typically v2, containing the pre-processed splits) to be used for training. Before training commences, the AutoML service queries the AutoDW API to retrieve full dataset metadata, ensuring that features such as the `target_column_name` and `task_type` (e.g., classification or regression) are correctly aligned with the user's original intent. An example of a typical `automl-trigger-events` message is shown below:

```
{
  "task_id": "automl_task_xyz789",
  "event_type": "automl-trigger",
  "timestamp": "2025-02-10T14:35:00.000000Z",
  "input": {
          "user_id": "user123",
          "dataset_id": "drowsiness_dataset",
          "dataset_version": "v2",
          "task_category": "tabular",
          "task_type": "classification",
          "target_column_name": "target",
          "time_budget": "10"
  },
  "failure": null
}
```


Automated storage of model artifacts: Upon completion of the training phase, the AutoML service does not merely store a flat model file; it registers a comprehensive "Model Package" back to the AutoDW. This involves:

- **Artifact upload**: Real model files (e.g., `.pkl`, `.onnx`, or PyTorch `.pth` files) and auxiliary files (like label encoders or scaling parameters) are uploaded to the `/ai-models/upload/folder/{user_id}` endpoint.
- **Version tracking**: The AutoDW automatically assigns a version (e.g., v1) to the new model, linking it permanently to the specific dataset version used during training to maintain a strict "Data-to-Model" lineage.

An example of how the saved trained model files look on AutoDW's MinIO UI is depicted on Figure 2.

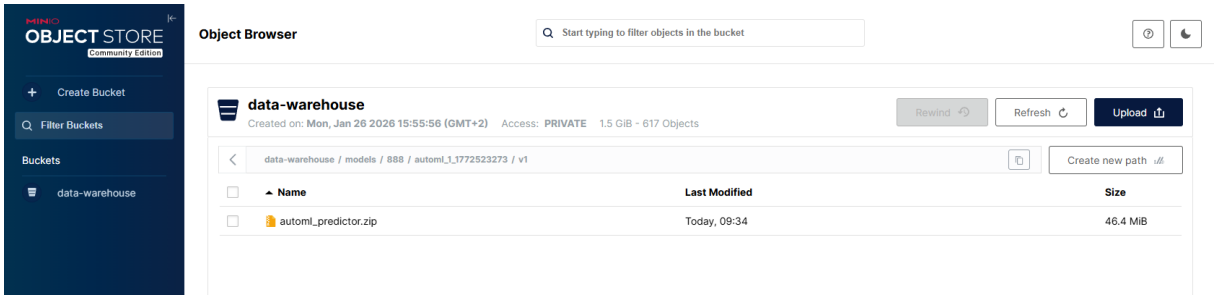

Figure 2. Trained model files example saved on AutoDW.

Persistence of performance meta-information: Beyond the physical model files, the AutoML service pushes detailed performance and configuration metadata to the AutoDW's MongoDB backend. This metadata includes:

- **Training metrics**: Accuracy, F1-score, precision, recall, and any custom optimization metrics used during the search.
- **Framework context**: Information regarding the library used (e.g., AutoGluon, Scikit-learn), the Python version, and dependency requirements (e.g., requirements.txt).
- **Structural metadata**: For folder-based uploads, the AutoDW stores a manifest of all files within the model directory, marking the "primary" model file for downstream consumption by the XAI and Predictor services.

An example of the JSON file stored in AutoDW's MongoDB with all the model meta-information is shown below:


```
{
  "model_id": "automl_1_1772523273",
  "user_id": "888",
  "name": "AutoML Model - automl_1_1772523273",
  "description": "AutoML trained model for tabular data",
  "version": "v1",
  "framework": "sklearn",
  "model_type": "classification",
  "algorithm": "WeightedEnsemble_L2",
  "files": [
    {
      "filename": "automl_predictor.zip",
      "file_path": "models/888/automl_1_1772523273/v1/automl_predictor.zip",
      "file_size": 48694161,
      "file_type": "zip",
      "file_hash": "1470c5cbb777c9e8db1dbe0395954cb6",
      "content_type": "application/octet-stream",
      "is_primary": true,
      "description": "AutoGluon predictor bundle (model + preprocessing + artifacts)"
    }
  ],
  "primary_file_path": "models/888/automl_1_1772523273/v1/automl_predictor.zip",
  "input_shape": [1, 42],
  "output_shape": [1, 1],
  "num_parameters": 0,
  "model_size_mb": 46.43837070465088,
  "training_dataset": "1",
  "training_accuracy": 0.94,
  "validation_accuracy": 0.91,
  "test_accuracy": 0.90,
  "training_loss": 0.23,
  "python_version": "3.11",
  "dependencies": [
    "autogluon.tabular",
    "pandas",
    "numpy",
    "scikit-learn",
    "lightgbm",
    "xgboost",
    "catboost"
  ],
  "hardware_requirements": "CPU (4 cores recommended); 8GB RAM recommended",
  "tags": ["automl", "tabular", "classification", "autogluon"],
  "custom_metadata": {
    "time_budget_seconds": 600,
    "metric": "accuracy",
    "leaderboard_preview": [
```

```
    {"model": "LightGBMXT", "score_test": 0.89},
    {"model": "CatBoost", "score_test": 0.90},
    {"model": "WeightedEnsemble_L2", "score_test": 0.90}
   ],
   "notes": "Example values for deliverable; actual metrics populated after integrating AutoML service outputs."
  },
  "created_at": "2026-03-03T07:34:39.994000",
  "updated_at": "2026-03-03T07:34:40.080000",
  "is_active": true,
  "is_production_ready": false
}
```


Downstream notification: The interaction concludes when the AutoDW emits an `automl-complete-events` message. This event serves as a signal to the Agentic Core that the model and its associated meta-information are now "live" in the repository, enabling the next stages of the pipeline, such as Concept Drift analysis or Explainable AI (XAI) report generation.

6.2 Coordination with Agentic Core

The Agentic Core initiates the AutoML process upon explicit user request. This process is executed asynchronously as a background task. Background task execution and monitoring are handled by the Task Manager service.

The Task Manager is a lightweight orchestration service responsible for initiating background tasks and collecting their outcomes via Apache Kafka topics. It persists task-related metadata, including task identifiers, input parameters, output locations, and timeout specifications, in a PostgreSQL database. This design ensures traceability, fault tolerance, and reproducibility of task execution.

AutoML operates as a worker service within this architecture. It is triggered upon receipt of a designated “trigger” message from a Kafka topic. Once activated, the AutoML component performs model training in accordance with the provided configuration. The resulting model artefacts are stored in AutoDW. Upon completion, the worker publishes a “complete” message to Kafka, containing the metadata required to retrieve the generated outputs from AutoDW.

In the event of an execution failure, the worker likewise emits a “complete” message, supplemented with error-related information to enable diagnostic analysis.

During agent execution, the Agentic Core periodically verifies the status of background tasks within the scope of the active conversation. For successfully completed tasks, it retrieves the corresponding outputs from AutoDW. This mechanism ensures that the AI agent remains informed about task progress and the models produced by the AutoML component.

The sequence diagram (Figure 3) illustrates the execution flow of an AutoML task initiated by the Agentic Core agent. Initially, the agent submits a request to the Task Manager REST API, specifying the task parameters, including the task type and input data. Subsequently, the Task Manager service publishes a message containing the task inputs to the “`automl-trigger-events`” Kafka topic and records the task in the system as scheduled. The AutoML worker consumes this message, performs the required computation, and upon completion publishes a message with the corresponding output metadata to the “`automl-complete-events`” Kafka topic. A background process within the Task Manager, which subscribes to this topic, retrieves the completion message, persists the output metadata, and updates the task state

accordingly. Consequently, when the agent subsequently retrieves the task via the Task Manager REST API, the response includes the results generated by the AutoML process.

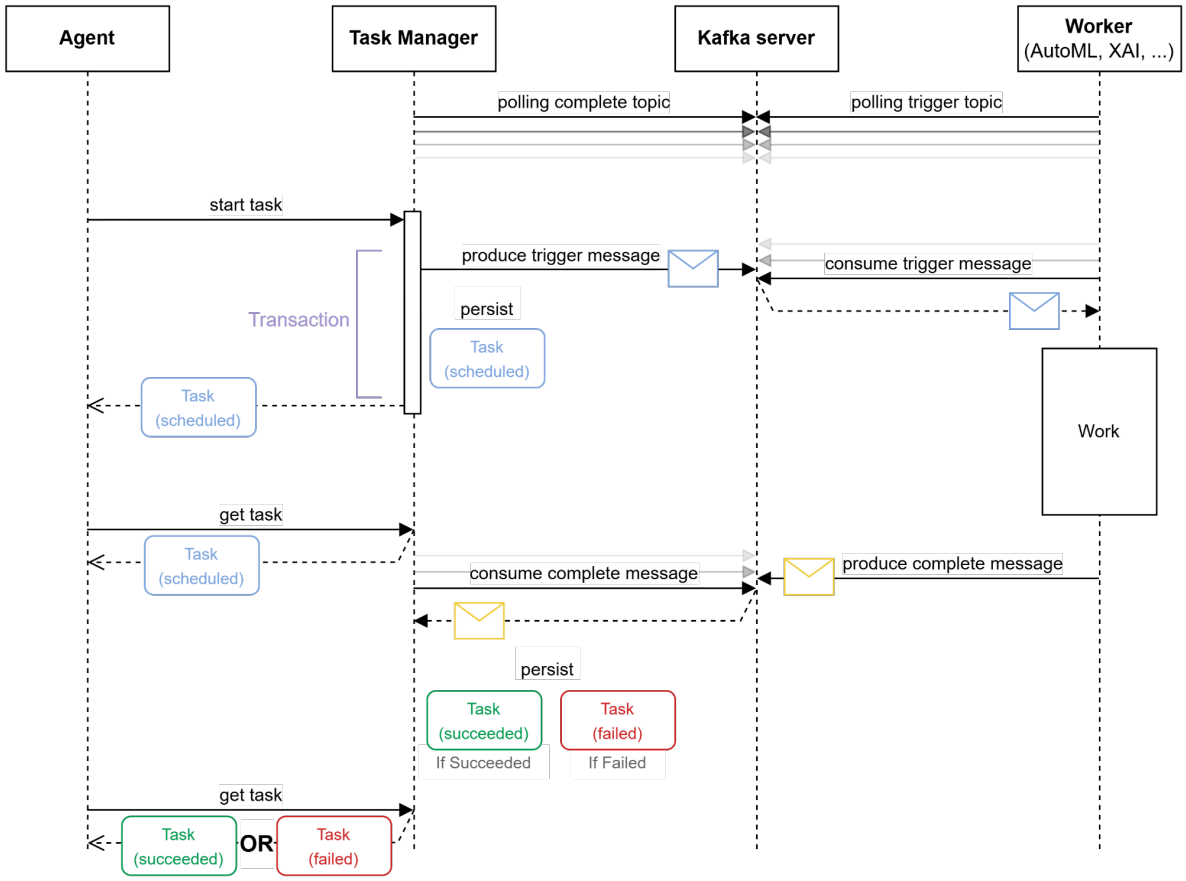

Figure 3. Execution flow of an AutoML task

6.3 Information from ETD-Hub for AutoML models

The EthiTech Dialogue Hub (ETD-Hub)[1] is the ALFIE project's online, multidisciplinary discussion forum designed to bring citizens, policymakers, AI practitioners, and legal experts together to debate the legal, ethical, and societal implications of AI and AI policy. It enables transparent, inclusive consultation across these core stakeholder groups in order to collect ongoing feedback that can be processed and used by SemKG and AutoML to inform model development. The forum operates in a Question and Answer format, addressing specific AI-related questions in open threads, nested under general themes. Themes describe general domain area or related case study in order to aid data organisation within SemKG.

ETD-Hub will serve two main 'types' of data relevant to AutoML and SemKG - Post Metadata and Post Data. These two 'types' of ETD-Hub data are collected for themes (table 4), questions (table 5), answers (table 6), and votes (table 6). Several other datasets are collected, though these are not mentioned here due to limited relevance.

[1] https://etdhub.eu/

Post Data constitutes the actual user-created text content of themes, questions, and answers on ETD-Hub (see tables below). This captures the raw perception of users on debates in AI, allowing direct, granular querying of stakeholder opinion. It is expected that the AutoML system may utilise Post Data to obtain suggestions for approaches to responsible AI development and solve specific development challenges in a feedback loop by requesting new question posts on ETD-Hub.

Post Metadata summarises the non-text-content information for every theme, question, and answer. This includes date/time, theme, and upvote/downvotes (see tables below). Most important for AutoML is the upvote/downvote count for every answer, as this serves as an indicator for overall platform sentiment, and the magnitude of this sentiment, related to a given idea. It is expected that the AutoML system may utilise Post Metadata to help quantify public and expert opinion and support selection of approaches outlined in Post Data for responsible AI development.

Table 4. Summary of theme data served by ETD-Hub to AutoDW.

Data Field	Type	Description	Example
theme_id	Post Metadata	Numeric code unique to theme	1
name	Post Data	Theme name	IRIS Case Study
description	Post Data	Detailed description of theme	The IRIS Case Study concerns…
views	Post Metadata	Unique view count for theme	168
expert_id	Post Metadata	Numeric code unique to expert who created the theme	7
problem_category	Post Metadata	Problem type tag	AI Bias, Datasets, AI Models etc.
model_category	Post Metadata	Model type tag	ML, LLM, MV
domain_category	Post Metadata	Domain type tag	Justice, Social, Education, Cybersecurity
created_at	Post Metadata	Date and time theme was created	2026-03-03T13:12:11
question_count	Post Metadata	Count of questions nested under given theme	54

Table 5. Summary of question data served by ETD-Hub to AutoDW.

Data Field	Type	Description	Example
question_id	Post Metadata	Numeric code unique to question	137

title	Post Data	Question title	Solutions to bias problems in sensitive datasets
body	Post Data	Question body text describing issue	I have a sensitive dataset where I can't ID individuals but need to ensure parity…
created_at	Post Metadata	Date and time question was created	2026-03-03T13:12:11
views	Post Metadata	Unique view count for question	4467
expert_id	Post Metadata	Numeric code unique to expert who created the question	23
theme_id	Post Metadata	Numeric code unique to theme under which question is nested	1
answer_count	Post Metadata	Count of answers nested under given question	98

Table 6. Summary of answer data served by ETD-Hub to AutoDW, including voting information.

Data Field	Type	Description	Example
answer_id	Post Metadata	Numeric code unique to question	45
description	Post Data	Answer body text	Have you tried doing some blind…
created_at	Post Metadata	Date and time answer was created	2026-03-03T13:12:11
question_id	Post Metadata	Numeric code unique to question under which answer is nested	47
expert_id	Post Metadata	Numeric code unique to expert who created the answer	20
parent_id	Post Metadata	Numeric code unique to theme under which question is nested, under which answer is nested	1
vote_count	Post Metadata	Total vote count (upvote + downvote) for the answer	54
upvotes	Post Metadata	Total upvote (positive response) count for the answer	4
downvotes	Post Metadata	Total downvote (negative response) count for the answer	50

6.4 Semantic Knowledge Graphs to inform AutoML model generation

6.4.1 What is Semantic Knowledge Graph

In ALFIE, the Semantic Knowledge Graph (SemKG) is a core component that ingests content from the data fields associated with themes, questions, answers, votes, and experts, and transforms this information into a machine-interpretable knowledge graph. In this graph, entities, relations, and metadata are represented using explicit semantics grounded in ontologies and standardised vocabularies. SemKG encodes knowledge using RDF (Resource Description Framework), typed classes, formally defined properties, and logical constraints, enabling reasoning, validation, and interoperability across systems [2] [3] [4] (for more information about SemKG, please refer to deliverable D3.1).

All of this can be used by the AutoML as additional context when deciding which models to build.

ALFIE SemKG advances the state-of-the-art by shifting the focus of knowledge modelling from static, encyclopaedic entity representation toward structured modelling of deliberative processes and governance dynamics. While established semantic knowledge graphs such as DBpedia, YAGO, and Wikidata primarily capture factual assertions about entities and their relationships [5] [6] [7], the ALFIE SemKG formalises discourse itself as a first-class semantic construct. This represents a conceptual and architectural departure from entity-centric graph paradigms.

Central to this advancement is the TQAVE schema (Theme, Question, Answer, Vote, Expert), which models deliberative structures as semantically typed components within an OWL-compliant RDF framework. Rather than treating discussions, expert inputs, or voting artefacts as peripheral metadata, the SemKG explicitly encodes them as governed classes and relations. Actor participation, evidential grounding, extraction confidence, and expert voting are attached to reified relation nodes, enabling contextualised, explainable assertions. This discourse-native modelling approach operationalises governance interactions as structured semantic knowledge, rather than as loosely attached annotations.

6.4.2 Semantic Knowledge extraction pipeline

SemKG pipeline integrates automated semantic extraction with ontology-driven typing, deterministic IRI generation, SHACL-based constraint validation, and reified n-ary modelling. Existing knowledge graph construction pipelines often emphasise entity linking and triple extraction [8] but do not uniformly enforce structural constraints at deployment time or model contextual provenance as a core design principle. By embedding SHACL validation [10] into the pipeline and requiring each relation to be associated with evidence and provenance metadata, the ALFIE approach strengthens data integrity and traceability beyond many extraction-driven systems [9].

Furthermore, the SemKG is designed for explainable AI integration. Each assertion is represented as a semantically typed relation node connected to evidence, confidence, discourse identifiers, and expert votes. This design enables AI components to retrieve justification chains directly from the graph, supporting transparent reasoning and auditable decision support. In contrast to traditional knowledge graphs that prioritise large-scale data aggregation, the ALFIE SemKG prioritises governance, explainability, and deterministic reproducibility as primary design objectives.

6.4.3 Infrastructure and deployment architecture

The SemKG is serialised into standard RDF formats:

- Turtle (`.ttl`)
- RDF/XML (`.rdf`)
- JSON-LD (`.jsonld`)
- OWL (`.owl`)

These formats allow deployment into any standards-compliant triple store or graph database.

SemKG deploys to a GraphDB instance hosted within AutoDW via a configurable client module. Upload to the triple store is controlled by a configuration flag (`ENABLE_GRAPHDB_UPLOAD`).

This enables:

- Local validation without persistence
- Controlled production deployment
- Batch-based upload strategies

Before upload, the graph undergoes SHACL validation to ensure structural compliance and datatype correctness. Only conformant graphs are eligible for deployment.

6.4.4 Storage layer (Triple Store / Graph Database)

Once deployed into a triple store (GraphDB) in the AutoDW, the graph is accessible via a SPARQL endpoint.

This enables:

- Structured semantic queries
- Pattern-based graph traversal
- Aggregation and analytical queries
- Federated querying with external RDF sources

The graph design supports SPARQL-friendly querying because:

- All relations are reified as etd:Relation nodes
- Subject, predicate, and object are explicitly modelled
- Evidence nodes are attached via etd:hasEvidence
- Datatypes are explicitly declared (xsd:string, xsd:integer, xsd:float)

6.4.5 Reified relation query pattern

Because relations are represented as first-class etd:Relation nodes, SPARQL queries follow a consistent four-step pattern:

1. Match the relation node
2. Retrieve etd:subject
3. Retrieve etd:predicate
4. Retrieve etd:object

Optional joins may include:

- etd:confidence for threshold filtering
- etd:hasEvidence for retrieving supporting text
- Inference metadata (e.g., inference_type) for distinguishing extracted vs inferred assertions

This design ensures uniform query structure across extracted and inferred knowledge.

7 Application of AutoML in ALFIE trial contexts

This section describes how the AutoML engine is used to generate results for each of the three trial use cases defined by ALFIE partners.

7.1 AutoML in website accessibility scenario (UAB use case)

In the website accessibility use case, the AutoML engine employs a pipeline that begins with parsing the input, either raw HTML source code or a live URL into discrete chunks using Python-based DOM traversal libraries. Each block undergoes parallel evaluation by GPT- 4o mini, deployed on EU-compliant servers on Azure to ensure data sovereignty and low-latency inference. This LLM not only assigns a granular accessibility score out of 10, calibrated against WCAG criteria like perceivable, operable, understandable, and robust principles, but also generates context-specific diagnostics, such as flagging insufficient focus indicators for keyboard navigation or non-descriptive button labels that hinder screen reader usability.

Complementing textual analysis, the system integrates static readability metrics computed via the textstat library, quantifying aspects like Flesch Reading Ease (factoring syllable density and sentence complexity), difficult word frequency, lexical diversity, and mean sentence length to gauge cognitive load across user demographics. These metrics feed into a weighted aggregation algorithm within the AutoML engine, producing a holistic site-wide accessibility index that prioritizes blocks with cascading impacts, such as navigation menus affecting global usability.

For visual elements, a dedicated multimodal pathway invokes another GPT-4o mini instance to semantically parse image content, detecting objects, scenes, and textual overlays before cross-referencing against provided alt attributes, flagging mismatches like images lacking alt text to the best of the LLM's capability.

The pipeline extends to formatting and contrast validation using rule-based checkers that scan CSS properties for WCAG compliancy, language declaration consistency via HTML lang attributes, and structural integrity like proper ARIA landmarks. The output is a report with per-block issues, prioritized fix suggestions e.g., "*Replace inline styles with semantic classes for better maintainability*".

7.2 AutoML in autonomous vehicle bias detection scenario (CTL use case)

Within ALFIE's autonomous vehicle scenario, the AutoML engine executes a pipeline tailored for driver drowsiness detection and stress recognition, ingesting diverse data like facial video frames and physiological time-series from in-vehicle sensors. For the vision task, it automates a comprehensive search over state-of-the-art architectures, ranging from EfficientNet and ResNet variants to Vision Transformers, employing PyTorch for transfer learning from pre-trained weights. The system performs hyperparameter optimization via Bayesian methods and tunes learning rates, batch sizes, and applies various data augmentation strategies (e.g., random brightness for lighting invariance). A choice was made to not use neural architecture search even though it would explore depth and attention mechanisms to balance accuracy with edge deployment constraints due to it's extremely high compute and budget requirements.

The pipeline takes into account vision and some tabular features, then applies AutoML-driven model selection across models to find the best one for the job. To enforce fairness, the AutoML engine implements stratified k-fold validation that partitions data to make a more robust model.

7.3 AutoML in compliance screening scenario (Bosch use case)

ALFIE's compliance screening leverages the AutoML engine to process high-volume Business Partner Screening (BPS) alerts, fusing tabular metadata (e.g., risk scores, transaction volumes, entity types) with voluminous unstructured text from violation narratives and audit trails. Initial preprocessing employs domain-adapted tokenizers for compliance lexicon, normalizing terms like "sanctioned entity" or "anti-bribery clause", followed by embedding generation. These embeddings concatenate with numeric features into a unified input space, where AutoGluon orchestrates bagging and stacking ensembles from base learners including LightGBM, TabNet, and RBF neural networks, automating stacker selection via greedy layer-wise construction to find the best model.

8 Conclusions

In conclusion, this deliverable presented the initial design and implementation of the AI model generation capabilities developed within the ALFIE project, with a particular focus on the AutoML and AutoML+ engines. The document described how these components enable the automated creation of machine learning models while ensuring alignment with ethical principles, transparency requirements, and European regulatory frameworks.

The deliverable detailed the operational workflow of the AutoML+ engine, explaining how user requirements are translated into optimized AI pipelines through an end-to-end model generation process. It outlined the distinction between the standard AutoML engine and the extended AutoML+ functionality, highlighting the advanced techniques integrated into the platform. These include transfer learning, hyperparameter optimization, meta-learning, and multi-objective optimization, which collectively support efficient model discovery while balancing performance, fairness, interpretability, and computational efficiency.

A key aspect addressed in the document is the multilingual interaction framework that allows users to interact with the ALFIE platform in multiple languages. Both the multilingual user web interface and the conversational natural language interaction with the chatbot and the Agentic Core were presented, demonstrating how complex AI processes can be made more accessible and transparent for users with diverse linguistic backgrounds and varying levels of technical expertise.

The deliverable also introduced the ethical model generation mechanisms embedded within the AutoML engine. These mechanisms include drift and bias detection prior to model training, ensuring that datasets are assessed for fairness and representativeness before model construction begins. Additionally, post-deployment concept drift detection and adaptive model updating processes were discussed, enabling the system to maintain performance, reliability, and regulatory compliance in evolving data environments. Transparency, explainability outputs, and comprehensive traceability mechanisms further support the trustworthiness and accountability of the generated models.

The application of the AutoML framework across the ALFIE trial scenarios, website accessibility, autonomous vehicle bias detection, and compliance screening, illustrated how the platform's capabilities can be applied to real-world contexts with diverse technical and ethical challenges. These scenarios demonstrate the flexibility of the AutoML engine in addressing domain-specific requirements while maintaining consistent ethical safeguards.

Finally, the deliverable described how the AutoML system interacts with other ALFIE components, including AutoDW, the Agentic Core, the Detector/Mitigator, ETD-Hub, the XAI module, and the semantic knowledge graph. These integrations provide contextual, semantic, and governance-related information that enhances model generation and ensures coherence across the platform's architecture.

Overall, this deliverable establishes the foundations for ALFIE's automated and ethically guided AI model generation framework. As the project progresses, further refinements and extensions of these components will be explored in subsequent deliverables, supporting the continued development of transparent, trustworthy, and user-centric AI systems.

References

[1] W3C. (2014). RDF 1.1 Concepts and Abstract Syntax.

[2] W3C. (2012). OWL 2 Web Ontology Language Document Overview.

[3] Uwasomba, C.F., Lee, Y., Yusoff, Z. and Chin, T.M., 2022. Ontology-based methodology for knowledge acquisition from groupware. *Applied Sciences*, 12(3), p.1448.

[4] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). DBpedia: A nucleus for a web of open data. ISWC 2007.

[5] Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). YAGO: A core of semantic knowledge. WWW 2007.

[6] Vrandečić, D., and Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. Communications of the ACM, 57(10).

[7] Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web Journal.

[8] Knublauch, H., and Kontokostas, D. (2017). Shapes Constraint Language (SHACL). W3C Recommendation.

[9] W3C. (2017). SHACL – Shapes Constraint Language.

[10] Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2020). Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*.

[11] Iman, M., Arabnia, H. R., & Rasheed, K. (2023). A review of deep transfer learning and recent advancements. *Technologies*, *11*(2), 40.

[12] Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *International journal of computer vision*, *129*(6), 1789-1819.

[13] Gunantara, N. (2018). A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, *5*(1), 1502242.

[14] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

[15] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).