# AI Product Card Specification

AI Products represent a software solution that, starting from available data, uses advanced techniques based on artificial intelligence to provide the expected functionality. Considering the entities defined above, the AI Product prepares one or more models or services based on them to carry out typical AI functions: classification, inference, prediction, recognition, etc. These functions can be used in decision support processes, to carry out simulations, participate in advanced interaction chains, etc.

The conceptual model of how an AI Product is organized is represented in the following figure.

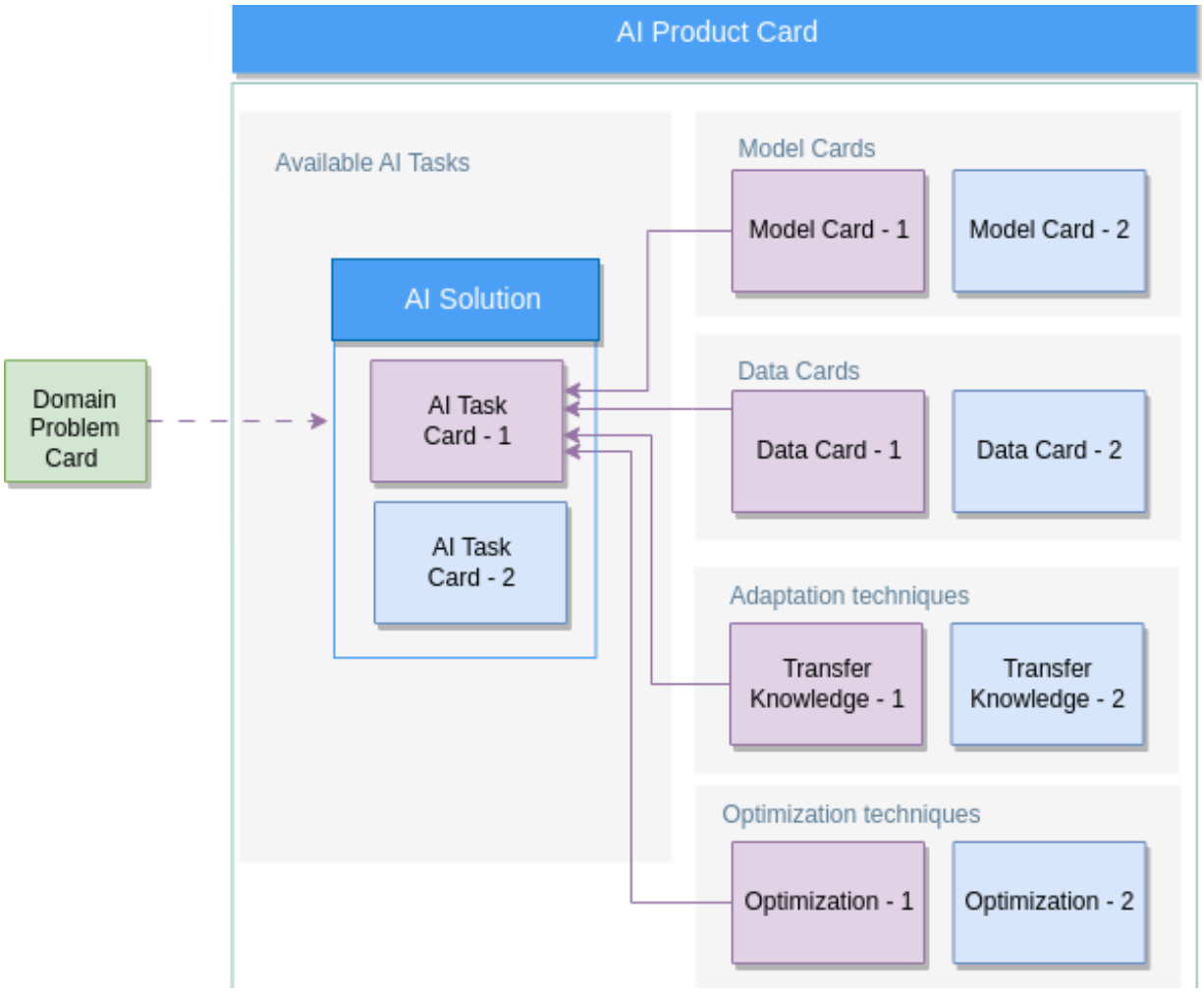


*Figure 1: AI Product model.*

AI Product involves one or more elements, AI Task, each of which is aimed at solving a specific AI-related and AI Model-based task. Each AI Product aims at implementing a problem defined

by stakeholders or policy makers, Domain Problem. AI Tasks that make up the solution are described by a series of information, such as the characteristics of the AI model, code necessary to produce it, aspects related to optimization and adaptation, information regarding the data used and necessary, compatibility and validation requirements.

In this model, Domain Problems are used to explicate problems that emerge in the domain and characterize a specific need that emerges in decision chains. An example of a domain problem could be the classification of official documents (resolutions, ordinances) with respect to the legal reference taxonomy, for example EuroVoc; public transport recognition service in a traffic light video stream; solution to predict traffic intensity in a point of the city based on data from the last 30 minutes. It is important that each Domain Problem is characterized by well-defined inputs and outputs (with the corresponding data models) and optionally by a series of quantifiable requirements that allow validating a possible solution to the problem (for example, a benchmarking dataset with associated labeling, data distribution metrics). Furthermore, it is possible to associate the risk profile, characterization of ethical requirements, etc. to a Domain Problem.

The AI Product definition document, AI Product Card, is specified by those who build the solution and intend to document all the entities involved and their characteristics: models, services, data. AI Product refers to a Domain Problem it implements; it also contains various AI Tasks, their relationships and dependencies. For example, in the context of dialogue tools, an AI Product can respond to the problem of responding in natural text to a user request using a supporting database (for example, answering what movie you can watch in the city evening, using the cultural events database). This AI Product is composed of an AI Task that interprets a user request and transforms it into a query to the DB, and another AI Task that takes the list of films and generates the response text.

The specification document for AI Products is used to increase transparency by communicating key information about the solution in order to facilitate the reproducibility, reusability and fairness of AI Products. The idea is that the solutions released are accompanied by

- documentation in order to (1) clarify the intended use cases of machine learning models and (2) minimize their use in contexts for which they are not suitable.
- Structured, machine-readable metadata to facilitate solution integration and deployment using AI platform components and extensions.

The definition of a single AI Task is represented with the following elements:

- **ML model** used to solve the problem and corresponding implementation details: code to extract features and train the model, model artifacts of different versions, model services.
- **Data model** used to build the ML model: schema, properties, and quality requirements for the data. This subsequently includes both data quality policies and the metrics/tests needed to evaluate the compliance of the data with the task (e.g. to see that the model can be used as is, can be adapted or should be completely retrained ). This model can also optionally be accompanied with the data itself, both as a complete dataset and as a benchmarking dataset necessary for profiling and validation.

- **Adaptation specification**: information on how the model (if available) can be adapted to the context. That is, what techniques, data and code are needed for adaptation and its validation.
- **Optimization specification**: information on how the model can be optimized (code, execution parameters, etc.).

# 1. Domain Problem Specification

This portion of the AI Product Card serves to outline the domain problem that the Card aims to address. It provides a comprehensive explanation of the issue from the perspective of the policy maker who will utilize the solution offered by the Card. The AI Product Card approach offers the advantage of examining the AI system from various viewpoints, including those of policy makers, AI practitioners, and domain experts with specialized knowledge in AI.

The catalogue containing the list of the Domain Problems that have been formulated and used by diverse public institutions may be considered as a potential contribution to the implementation of the FAIR principles within the public sector. Providing open access to commonly used governmental use cases, the catalog with the Domain Problems aim to facilitate their reuse, thereby reducing the time and effort required for their adoption and realization of benefits. While existing efforts to provide open access to datasets and AI models[1] are important, further work is needed to contextualise these models and data to the public administration domain in order to transform them into real AI products.

This section describes the problem within the context of public administration and aims to make it easily discoverable for future reuse by other public authorities. The catalog of Domain Problems contained within our proposed framework will enable policy makers to quickly identify and access previously solved problems within their own domain. Additionally, the framework offers indicators of the applicability level of the AI Product that addresses the Domain Problem, based on mechanisms for detecting, measuring, and reporting deviations in target domain data distribution. Some of the attributes of the formal definition of this specification may include the unique name of the problem, its description, available input data specifications, output data specifications, regulatory requirements needed.

This section of the AI Product Card may contain the following attributes:

- **id** - fully qualified name, unique for the task. Necessary for the references.
- **name** - name of the task
- **description** - detailed description of the task (markdown)
- **version** - version of the document
- **use_case** - describe the single problem we are trying to solve within the given domain problem (markdown)
- **inputs** - define input data specifications that the task should expect to work with. Formally as an array of data specs + constraints specs.
    - **name**
    - **specification** - optional reference to the data spec document.

[1] https://huggingface.co/models

- **outputs** - optionally define the output data specification produced as a result of the task. Formally as a set of dataset specs + constraints:
    - **name**
    - **specification** - reference to the data spec document
- **service** - optionally define the API of the task (as an OpenAPI specification or reference to an existing one)
- **validation** - optional specification/reference of the validation requirements to use as a certification for the solution.

## 2. AI Solution Specification

An effective AI documentation system must establish a structure that allows for a balance between considering the big picture and focusing on the details, aligning general objectives with specific design choices.
Therefore, the AI solution serves as a comprehensive wrapper for a series of cooperative AI tasks that work together to tackle a complex issue outlined in the Domain Problem Specification. This section of the Card can be likened to the Domain Problem, as it has the ability to address its requirements and solve the specified challenge. On the other hand, this section of the Card achieves a balance between the broader perspective and the finer details of implementing a final AI product that effectively addresses the problem at hand.
This section of the AI Product Card may contain the following attributes:

- **id** - fully qualified name, unique for the solution. Necessary for the references.
- **name** - name of the solution
- **description** - detailed description of the solution (markdown)
- **version** - version of the solution
- **source_code** - reference to the source code repository for the solution. To be compatible with the platform, it is expected that the repository contains the definition of the composing elements documented in a standard manner (project, data items, code references, services, functions, pipelines, ecc.).
- **domain_problem** - domain problem the solution implements (required). It is expected that the solution contains the implementation of the services the task expects and is capable of managing the inputs/outputs of the tasks. Specified as fully-qualified id of the task.
- **tags** - label the solution so that to be easily filtered and accessible
- **implementation** - definition of the entry point for the solution as the implementation of the domain problem functionality. Should be formally represented with
    - **kind** - the type of the implementation understandable by the platform. For example, function, pipeline, service, etc.
    - **reference** - reference to a specific entity within the solution (e.g., name of the function or service within the metadata, reference to the source code, etc).

- **inputs** - define input data specifications that the solution should expect to work with. Formally as an array of data specs + constraints specs. Should match those of the corresponding domain problem.
    - **name**
    - **specification** - optionally reference to the data spec document. May be omitted if the domain problem already specifies that.
- **outputs** - optionally define the output data specification produced as a result of the task. Should match those of the domain problem. Formally as a set of dataset specs + constraints:
    - **name**
    - **specification** - optionally reference to the data spec document. May be omitted if the domain problem already specifies that.

Semantics:

- If the input / output specifications of the domain problem declare the corresponding schemas, the task implementation should accept / produce data in that schema.
- If the domain problem declares the validation, the constraints should be satisfied by the validation.

# 3. AI Task Specification

AI task specification defines a single component of a (possibly composed) AI solution based on ML/AI model. The AI task specifications encompass the formal documentation of the various components that have been proposed in the literature such as data, model, validation, optimization and adaptation techniques. An AI task constitute the discrete elements of an AI solution. It serves as the primary building block for a complex system that can be integrated into different AI solutions to address specific Domain Problems.

The goal of the AI Product Card framework is to facilitate a wide range of AI tasks, with a focus on ease of reuse and adaptability. This section contains the metadata that matches the descriptive documentation parts of the task with the corresponding executable scripts or commands. In other words, it provides different scripts that implement the logic behind running inference on a particular model or executing a training phase.

This section pertains to the practical details and resources required for the training and inference phases of the AI task. Additionally, it provides an opportunity to assess the quality and compliance of the available data, ensuring that it aligns with the characteristics and distribution of the data used to train the model. This is an essential step in ensuring trustworthy AI development, as it helps to prevent the incorrect use of the task on data that exhibit different characteristics or distributions. Furthermore, the AI task should reference the adaptation techniques that may be employed to specialize the model for its specific use case.

This section of the AI Product Card may contain the following attributes:

- **id** - Identifier of the task
- **name** - name of the task
- **description** - description of the task (markdown)

- **tags** - label the task so that to be easily filtered and accessible
- **version** - version of the document
- **task** - AI domain task category (e.g., image classification, object tracking, ecc). A value from a predefined (yet extendable) domain.
- **inputs** - define input data specifications that the task should expect to work with. Formally as an array of data specs + constraints specs.
    - **name**
    - **specification** - reference to the data spec document
- **outputs** - optionally define the output data specification produced as a result of the task. Formally as a set of dataset specs + constraints:
    - **name**
    - **specification** - reference to the data spec document
- **training** - optional reference to the function/pipeline that implements the training procedure.
- **service** - optional reference to the service specification (implementation, deployment), which is needed for the deployment of the inference service.
- **model** - reference to the ML Model specification
- **data_compliance** - validation specification to be applied in order to check whether the task may be engaged in the context. This refers to the data checks to be evaluated on the input data (e.g., train data) to see whether the model applies.
- **model_compliance** - validation specification to be applied in order to check whether the resulting model and its execution fit the expected requirements.
- **adaptation** - adaptation specification that defines when the task may be adapted to the given context and how.
- **optimization** - optimization specification that defines how the solution may be optimized to a context.
- **resources** - specification of the resources required / recommended for the task. Defined with
    - **training** - resources needed/used for training. Defined as resource profile and metrics
        - **profile** - definition of resources
            - **architecture** - (proc spec)
            - **ram** - in GB
            - **cores** - number
            - **disk** - in GB
        - **metrics** - performance metrics and values
            - **type** - metric fully-qualified name
            - **value**
    - **serving** - resources needed for the service running. Defined as resource profile and metrics.

Semantics:

- If the task declares data_compliance, the constraints should be satisfied for the input data passed to training or service. Otherwise the context is incorrect.

- If the task declares model_compliance, the constraints should be satisfied for any output data produced by the service. Otherwise the context is incorrect.
- If the adaptation specification is defined, then
    - The compliance should be satisfied;
    - The result of adaptation produces a new model version in the context;
    - The model should pass the model_compliance spec.
- If the optimization specification is defined, then
    - The result of the optimization produces a new model;
    - The model should pass the model_compliance spec.

# 4. Model Specification

The appropriate specifications inside this section include the formalised version of the proposed Model Card. Some related work has already been done in order to provide a formalised and machine-readable format for the representation of the Model Card[2]. Additional studies aim to build the ontology corresponding to the details within the sections of the Model Card. Considering that Model Card discloses information about different aspects of model development and evaluation, we propose the following attributes as part of the formal specifications for replicating it in an automated manner. This representation of the model facilitates seamless integration into the AI task definition:

- **id** - Identifier of the model
- **name** - name of the model
- **description** - description of the model (markdown)
- **tags** - label the model so that to be easily filtered and accessible
- **version** - version of the document
- **license** - specification of the license (https://hf.co/docs/hub/repositories-licenses)
- **license_link** - reference to license in case of custom (other)
- **library_name** - name of the library underlying the model, such as keras, spacy, etc (es., https://github.com/huggingface/hub-docs/blob/main/js/src/lib/interfaces/Libraries.ts)
- **metrics** - metrics of the model specific to the type of the library
    - **type** - fully qualified metric ID
    - **value** - evaluated metric value
    - **name** - human-readable metric name
    - **args** - dictionary of parameters for the metric to be applied

Notes:

- *artifact* section should provide the parameters that produced the particular model artifact
    - Is it correct to have in the yml file the

Framework
Param
Metric

[2] https://github.com/huggingface/hub-docs/blob/main/modelcard.md?plain=1

data

# 5. Validation Specification

The validation specification defines the constraints that should be respected for the given dataset. It can be used to verify the compliance of the data, the applicability of the solution task, etc. The validation specification enables the provider to develop a certifiable AI Product that encompasses all crucial aspects for assessing the suitability of existing solutions from the catalog. It is essential to guarantee that the solution can be effortlessly adapted to a new context. The validation outcome provides a percentage of confidence for the repurposing of the solution, enabling policymakers to evaluate its applicability and efficacy. Data validation is often considered an iterative process that addresses the quality and the distribution of real world-data that are continuously evolving, leading to a phenomenon known as domain shift.
Research on how to deal with domain shift has been extensively conducted in the literature. However, systems incorporating AI Products in the public sector nowadays require a mechanism to identify, measure, and notify deviations in terms of changes in data distribution and other profiling features, including the quantity of data accessible. In our work, the framework will provide to the AI practitioners the ability to design AI Product Cards with a domain shift mindset, aimed at breaking the assumption that source and target data are independent and identically distributed (i.i.d).
To this end, the framework will integrate and establish mechanisms to identify the level of data drifting and offer essential assistance to alleviate its impact for the purpose of developing models that are robust to domain change or domain shift. As a potential open source library for detecting data drifting in specific categories of data types, evidently[3] may be well-suited for conducting test suites on relevant data sets. This section of the AI Product Card may contain the following attributes

Attributes:

- **description** - description of the validation (markdown)
- **runtime** - notation / library used to perform the validation (e.g., evidently, frictionless, …)
- **specification** - language-specific definition of the validation configuration with its own structure definition
    - **implementation** - optionally, definition of the implementing function for the validation ready for execution.
- **input_data** - list of dataset names used to consider for validation
- **reference_data** - list of reference datasets (test data, benchmarks)
    - **name**
    - **reference**

Semantics:

[3] https://docs.evidentlyai.com

- Input data should refer to the data items in the current context, to the outputs of the function (e.g., compliance, adaptation, etc), or to the elements from reference_data list (by name).
- If input data refers to one of the output names, the output should be generated first with the current implementation of the task, having inputs populated from reference data or input data.

# 6. Adaptation Specification

The adaptation specification defines when and how the specified task implementation may be adapted in the current context. This section of the AI Product Card aims to address the issues related to the efficient adaptation of the AI Product into different context. When referring to context, we are describing different target domain data characteristics. Taking into account the intended reuse of the same AI system, this section will outline the necessary steps and specifications to the reference implementations of the specialized solution. Considering the use case of object classification and detection from traffic cameras in the city of Tallinn, it is highly beneficial to reproduce the same AI Product in a different context, whether it is another urban area or even another city. On the other hand, when addressing Natural Language Processing (NLP) tasks, domain adaptation techniques are commonly used to minimize the disparity between different domains. For example, the task of classifying legal documents with multilingual Eurovoc thesaurus descriptors is commonly employed by various EU institutions, each of which employs distinct concepts that are categorized into distinct areas or subareas. Although the same document classification task may be applicable in diverse domains, it may result in incorrect classification due to domain shift caused by disparate distribution of target labels or classes. It would be highly advantageous in terms of time-saving and optimizing resource usage to adapt or fine tune an existing AI Product for document classification to a different government body, rather than creating a new system from scratch that would perform the same task. This section of the AI Product Card may contain the following attributes:

- **description** - description of the validation (markdown)
- **runtime** - notation / library used to perform the adaptation (e.g., fine tuning, transfer learning, etc.)
- **specification** - specific definition of the adaptation configuration with its own structure definition
    - **implementation** - definition of the implementing function for the adaptation ready for execution.
- **compliance** - validation specification to be applied in order to check whether the task may be engaged in the context with this adaptation spec.

Semantics:

- If the implementation is not defined, it should be possible to build executable function out of the specification only;
- The result of the execution produces a new model that replaces the base one;

- If the compliance is declared, then it should be engaged to check whether the adaptation makes sense.

# 7. Optimization Specification

The optimization specification defines how the solution may be configured to reduce the resources at deployment- or run-time given the specified requirements. This section presents the metadata format for the optimization techniques that may be utilized for AI-based solutions. It includes the machine-readable version of the prescriptive approach outlined in the Method Card, which provides AI engineers with the necessary guidance to optimize the system. Smaller public sector institutions might encounter challenges when fine-tuning a pre-trained model since it will maintain all the parameters of the original model resulting in high resource utilisation. Many recent studies have put forth an array of optimization techniques that could potentially influence the efficacy of the AI product in terms of time and resources allotted to it. Parameter-efficient fine tuning and Low-rank adaptation (LoRA) are widely recognized and highly effective techniques for adapting custom LLMs in an efficient manner on downstream tasks. Accordingly, the optimization section will contain the most effective configurations for implementing them in various applications. These configurations may encompass the base model, the batch size, the matrix rank, the alpha parameter, the target dataset, and the adaptation scripts that need to be applied on it.
This section of the AI Product Card may contain the following attributes:

- **description** - description of the optimization technique
- **runtime** - notation / library used to perform the optimization (e.g., early exit, quantification, etc.)
- **specification** - specific definition of the optimization configuration with its own structure definition
    - **implementation** - definition of the implementing function for the optimization ready for execution.
- **variables** - metrics that may vary for the model and may be optimized. Each variable is defined as
    - **metric** - the metric specification
    - **range** - possible value ranges (intervals, lists, etc)
- **scenarios** - list of the optimization scenarios that can be proposed (as examples)
    - **description** - description of the scenario.
    - **objectives** - list of metrics with the corresponding goals
        - **metric** - name of the metric
        - **value** - objective value as range, value, or target.

## Semantics:

- At optimization time the user may define
    - objectives - the goals for the optimization as ranges of values or targets (like ‘min’, ‘max’)

- The algorithm should provide as an outcome a new version of the model where the selected objectives are respected (if possible)

## 8. Data Specification

The Data Specification section contains the set of the parameters that represent the sections of a typical Data Card documentation approach. Data Cards play a crucial role to understand the structure and content of the dataset used for training and validating the model. They are used to assess the discrepancies between the characteristics of the training data and the characteristics of the production data, which may be utilized by public administrations. The specifications for a dataset should not only provide its structural schema, but also include any necessary constraints that the dataset must meet.
This section of the AI Product Card may contain the following attributes:

- **id** - Identifier of the dataset instance
- **name** - name of the dataset
- **description** - description of the dataset (markdown)
- **type** - macro category of data (e.g., tabular, document, audio, video, geo, …)
- **reference_taxonomy** - if any, reference to the domain ontology
- **tags** - label the domain dataset so that to be easily filtered and accessible
- **version** - version of the document
- **example** - reference the dataset implementation example (link, embedded)
- **schema** - reference to the schema of the dataset (depends on the type of the data).
    - **runtime** - specification dialect for the schema
    - **definition** - actual schema in the specified dialect

# Project template

The components of the AI Product Card are represented in three formats:

1. The descriptive or prescriptive format (Model Card, Method Card, Data Card, Adaptation Card, Optimization Card)
2. The machine-readable formal specifications
   The specifications include the set of declarative configurations that orchestrate the execution of the deployment and adaptation of the AI Product.
3. The implementation references which represent the real code that produce the components of the final product

One of the key elements of an AI product is for it to be **reproducible and reusable** by others. Having this in mind when structuring the project will allow others to look at the code, understand it well enough to be able to recreate the same results and also further **adapt** the solution to specific goals.

Keeping this in mind while structuring the project will enable others to review the code, comprehend it sufficiently to replicate the results, and **adapt** the solution to meet specific objectives.The following is a project template with a specified folder structure:

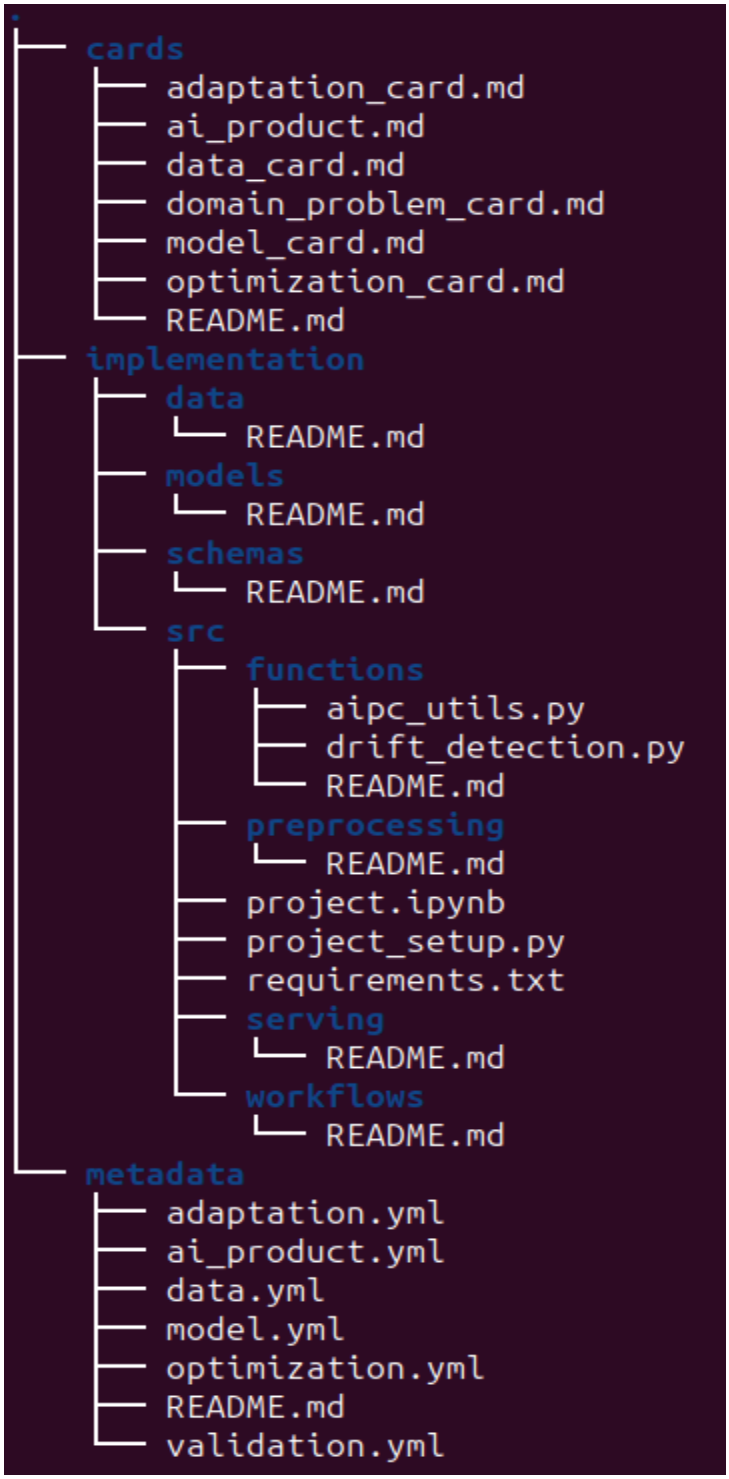

Figure 2: Folder structure of the AIPC project

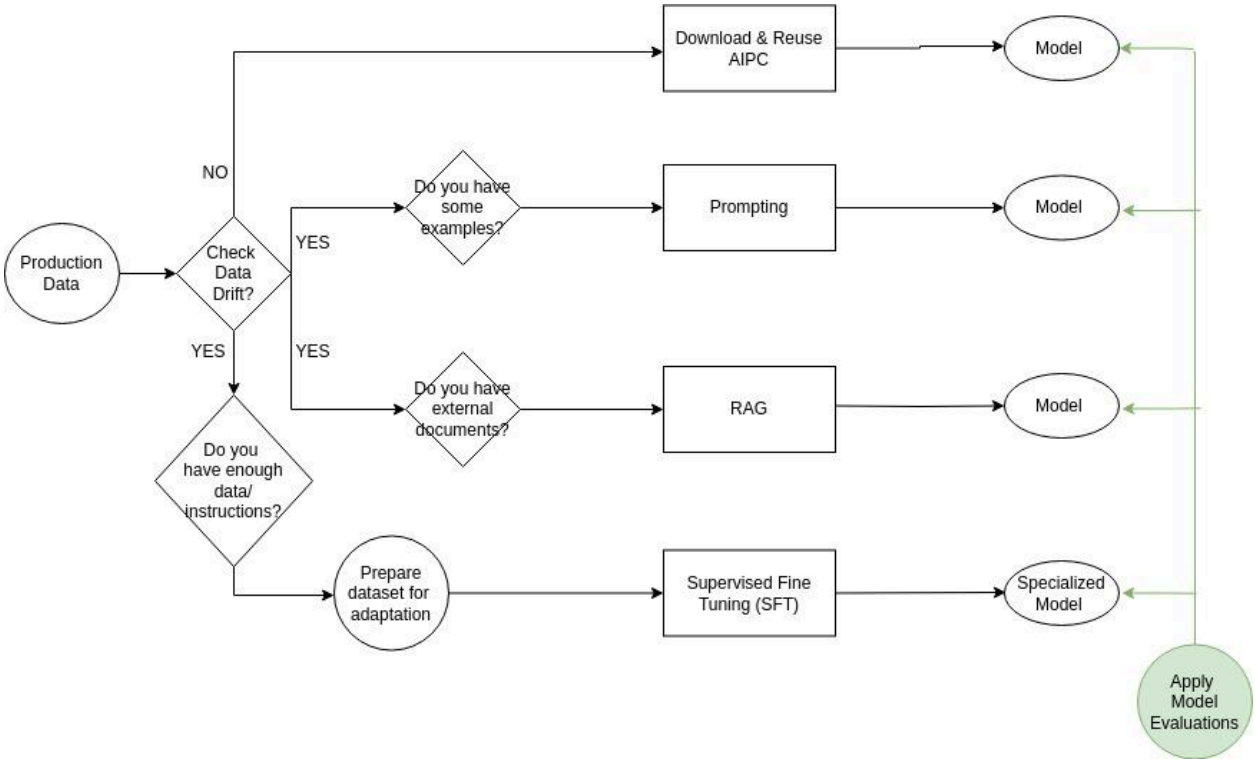


Fig 3. AIPC framework adapted for LLMs