

HumanE AI Net:

The HumanE AI Network

Grant Agreement Number: 952026
Project Acronym: HumanE AI Net

Project Dates: 2020-09-01 to 2024-08-31
Project Duration: 36 months

D8.6 Final Version of the Virtual Laboratory including benchmarking and challenge infrastructure

Author(s): Martin Weiß
Contributing partners: FHG, SU, ORU
Date: 24.09.2023
Approved by: Paul Lukowicz
Type: Report
Status: final
Contact: martin.welss@iais.fraunhofer.de

Dissemination Level

PU	Public
----	--------

X

DISCLAIMER

This document contains material, which is the copyright of *HumanE AI Net* Consortium parties, and no copying or distributing, in any form or by any means, is allowed without the prior written agreement of the owner of the property rights. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the *HumanE AI Net* Consortium as a whole, nor a certain party of the *HumanE AI Net* Consortium warrant that the information contained in this document is suitable for use, nor that the use of the information is free from risk, and accepts no liability for loss or damage suffered by any person using this information.

This document reflects only the authors' view. The European Community is not liable for any use that may be made of the information contained herein.

DOCUMENT INFO

0.1 Authors

Authors	Institution	e-mail
George Kampis (GK)	DFKI	George.Kampis@dfki.de
Martin Weiß	FHG	martin.welss@iais.fraunhofer.de

0.2 Document History

Revision		
Date	Lead Author(s)	Comments
30.09.2020	GK	Empty template
24.09.2023	MW	Review version

TABLE OF CONTENTS

0.1	Authors	2
0.2	Document History	2
Table of Contents		3
Executive Summary		4
1.	Introduction	5
2.	Container Specification	7
3.	Pipeline Examples	8
4.	Use Cases for research and education	11
	The Exploration Use Case	11
	The Training Use Case	11
	Education Use Case 1: Teaching	11
	Education Use Case 2: Student Lab	11
	Certification Use Case	11

EXECUTIVE SUMMARY

The Virtual Lab is part of the European AI on Demand (AIoD) Platform based on the AI4EU Experiments subsystem and provides the following main features to the AI community:

- Catalog of re-usable AI resources with clear license information (open source or proprietary)
- Visual Editor to easily create AI pipelines from the building blocks in the catalog
- Deployment of pipelines and single models to an execution environment
- Rating and commenting of the resources by the users
- Work in mixed teams on models and solutions (also privately)

The foundation for the flexible combination of AI resources are the packaging format and the communication framework, which are both based on proven open source technologies and programming language agnostic. The Virtual Lab (AI4EU Experiments) is developed in an open, distributed development process, where anybody can contribute.

The contributions from HumanE-AI-Net to the Virtual Lab are as follows:

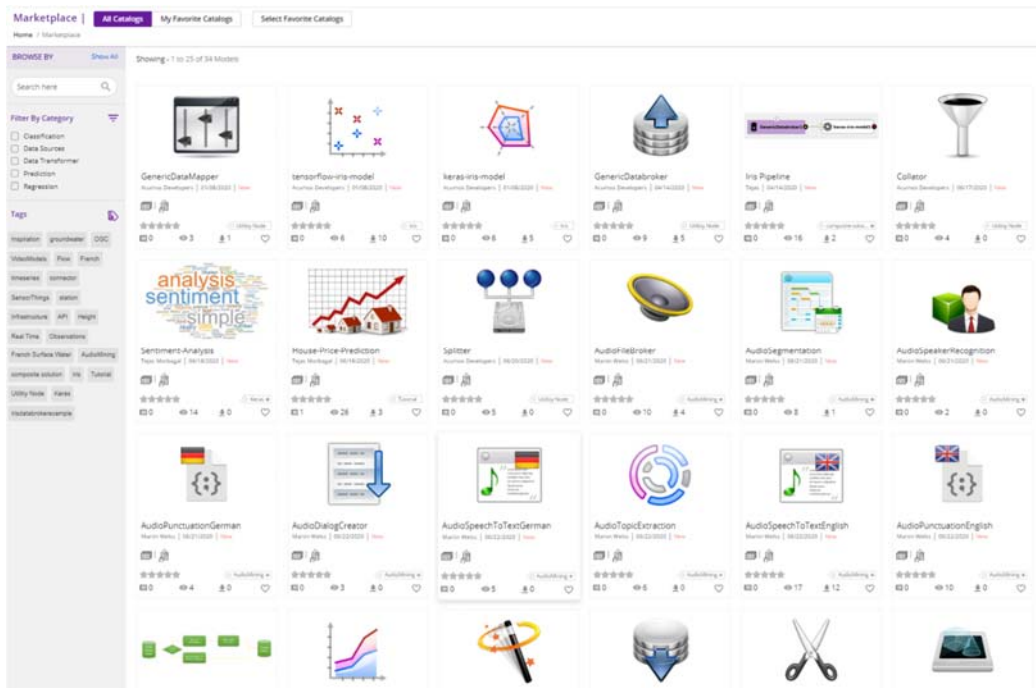
1. The Jupyter-Connect-Feature for easy interactive exploration of AI-Modules in the catalog ("One-click-deploy")
2. Definition of a dataformat to model pipeline execution runs as well as metrics of a pipeline run.
3. Extension of the AI Playground to actually capture those data of an execution run
4. Extension of the Catalog to manage challenges, including a Leaderboard
5. Extension of the Catalog to receive, save and compare execution runs

This approach and datastructure will not only allow to support benchmarking and challenges, but in the future also reproducibility and certification.

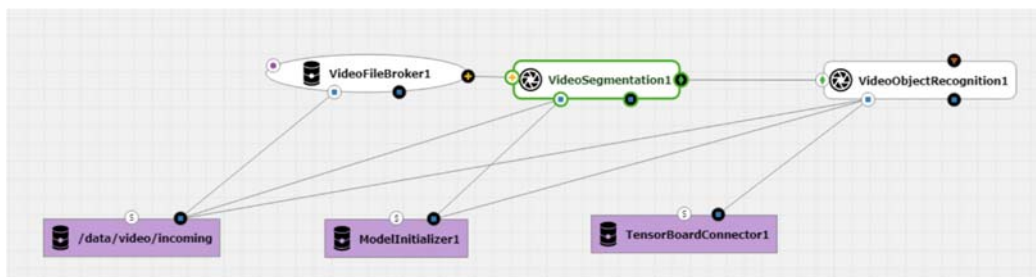
1. INTRODUCTION

AI4EU Experiments is based on the open source Eclipse Graphene project hosted by the Eclipse Foundation.

AI4EU Experiments (and hence the Virtual Lab) is a customized version of Acumos. The main entry point is of course the catalog (or marketplace) of AI building blocks:

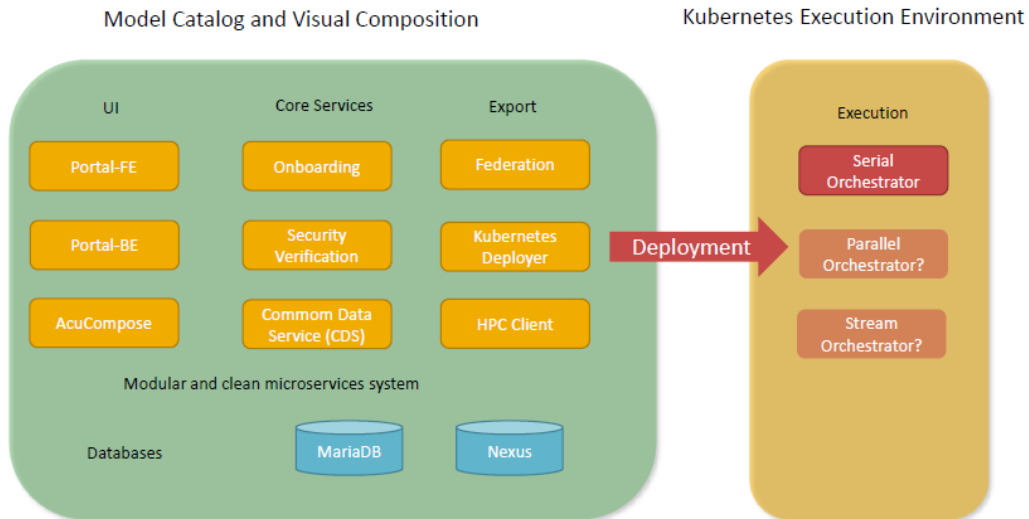


The visual pipeline editor makes it possible to create AI pipelines from the models and datasets (the re-usable building blocks) from the catalog:

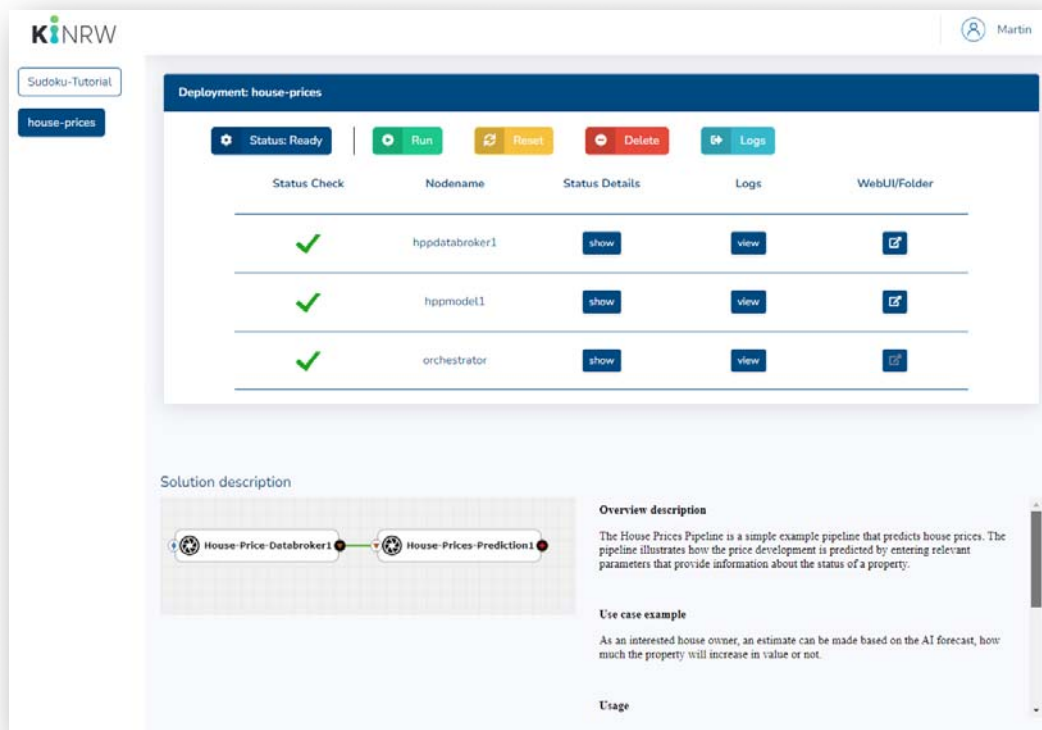


The editor supports the user by checking the connectivity of the models and ensures that only matching services are connected.

Graphene itself is based on a clean and easy to extend microservices architecture, but it is important to understand that Graphene is NOT an execution environment:



The target execution environment is Kubernetes, which can be either a hosted system or a local system operated by the user's organisation. The AloD Platform provides an execution environment for free with limited resources called the AI Playground. In the picture below we see the details of one of the deployed pipelines.



Deployment: house-prices

Status: Ready | Run | Reset | Delete | Logs

Status Check	Nodename	Status Details	Logs	WebUI/Folder
✓	hppdatabroker1	show	view	
✓	hppmodel1	show	view	
✓	orchestrator	show	view	

Solution description

House-Price-Databroker1 → House-Prices-Prediction1

Overview description

The House Prices Pipeline is a simple example pipeline that predicts house prices. The pipeline illustrates how the price development is predicted by entering relevant parameters that provide information about the status of a property.

Use case example

As an interested house owner, an estimate can be made based on the AI forecast, how much the property will increase in value or not.

Usage

2. CONTAINER SPECIFICATION

The container specification is the foundation for the interoperability of the building blocks and has the following properties:

- Based on free and open source technologies
- Docker containers
- programming language and tool agnostic
- public interface as protobuf definition
- simple and clear interface specification
- gRPC communication
- efficient and standardized communication
- code generators available for many programming languages
- makes automatic generation of artefacts possible

https://github.com/ai4eu/tutorials/tree/master/Container_Specification

```
// set used version of protobuf
syntax = "proto3";

// set unique package name
package fraunhofer.demo.cpp.iris.v1

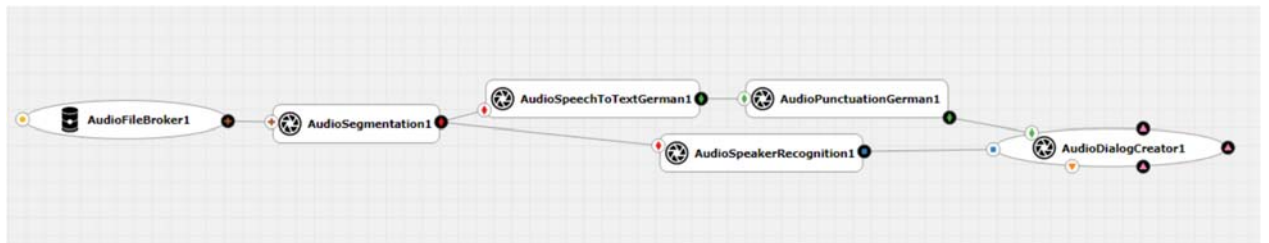
// define input data structure
message IrisDataFrame {
    repeated double sepal_length = 1;
    repeated double sepal_width = 2;
    repeated double petal_length = 3;
    repeated double petal_width = 4;
}

// define output data structure
message ClassifyOut {
    repeated int64 value = 1;
}

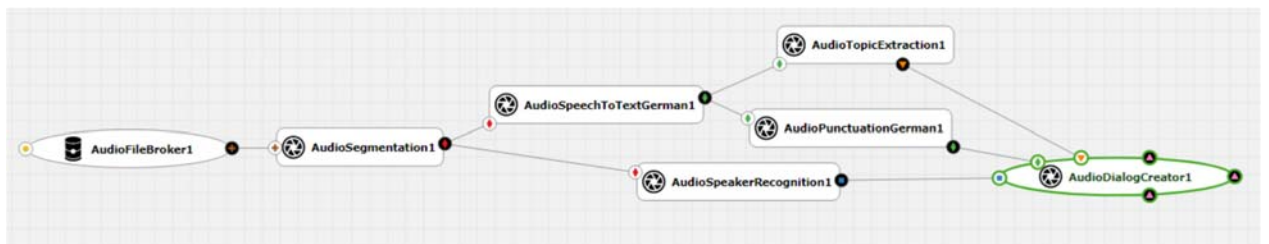
// define exposed service
service Model {
    rpc classify (IrisDataFrame) returns (ClassifyOut);
}
```

3. PIPELINE EXAMPLES

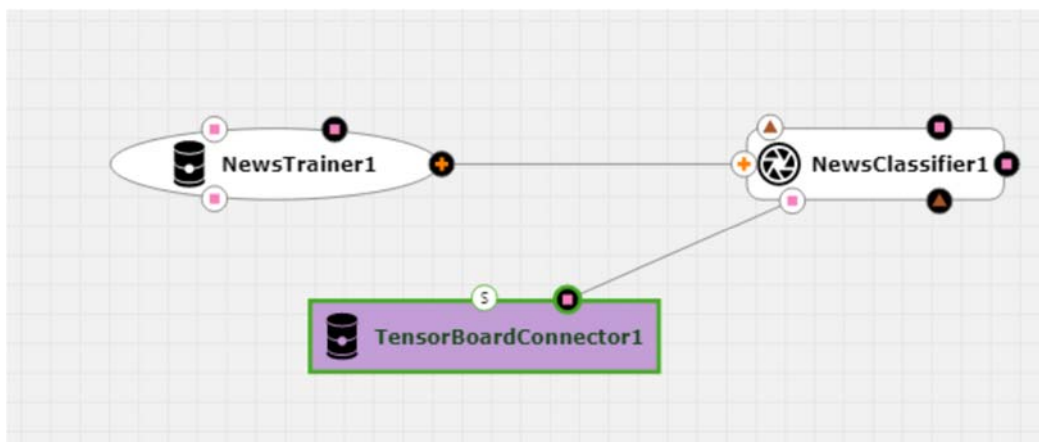
Here are examples of audiopipelines built from matching building blocks which demonstrates the level of re-usability:



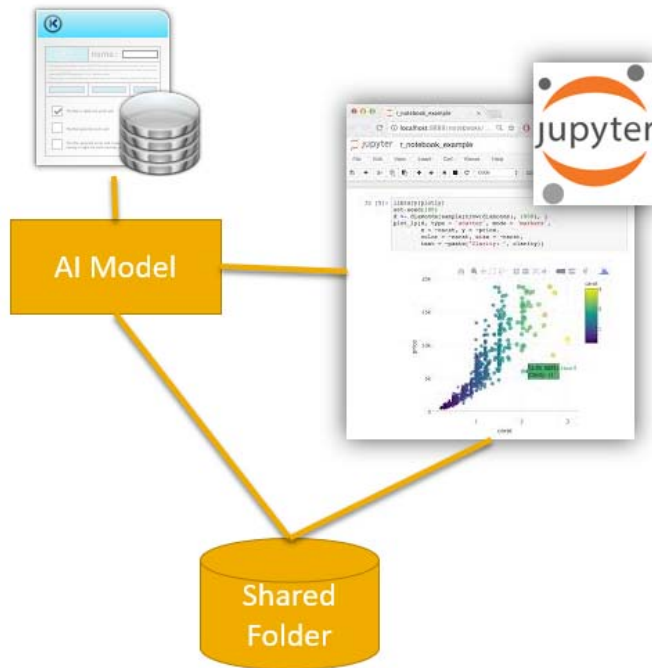
and here with TopicExtraction added:



For a training pipeline, specific nodes are needed to manage the training and the model of course needs to provide the corresponding training interface. Optionally, diagnostic nodes like a Tensorboard can be connected.



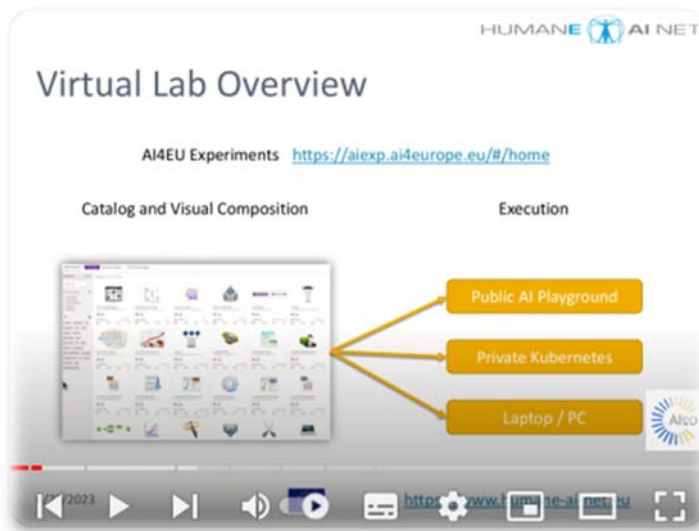
If the user wants to work interactively with a given model, there is an automated "Jupyter Connect" deployment, that will implicitly create a pipeline like this:



Here is the workflow to use this Feature:

- Choose single model from Catalog
- Click on Deploy for execution:
 - Playground or generic Kubernetes
- The Virtual Lab creates all necessary plumbing:
 - Jupyter Container
 - Shared Folder
 - Communication stubs based on gRPC code generation
 - ready to use notebook

There is a video tutorial available that demonstrates the Jupyter-Connect use case:



[Jupyter Connect Video Tutorial](#)

4. USE CASES FOR RESEARCH AND EDUCATION

There have been several areas of use cases identified where the Jupyter-Connect-Feature can help to support the goals of HumanE-AI-Net.

THE EXPLORATION USE CASE

- to quickly checkout a model
- choose model and deploy with one click as jupyter connect
- check web-ui (if available)
- explore model using the jupyter notebook via the generated grpc stubs
- download notebook for later use

THE TRAINING USE CASE

- The model needs to expose a training method in protobuf
- Load training data into the shared folder using the notebook
- start training via grpc stubs from notebook
- watch, explore and analyze the results
- download trained model for further use

EDUCATION USE CASE 1: TEACHING

- The Lecturer chooses AI models from the catalog
- Deploys them for execution as jupyter connect
- Live demo and interaction during the lecture
- Save notebooks for later distribution among students

EDUCATION USE CASE 2: STUDENT LAB

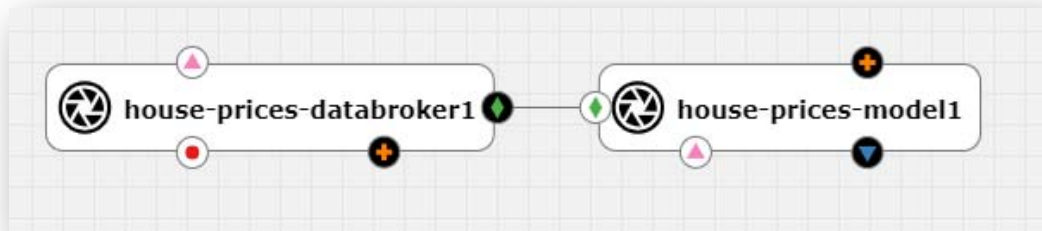
- The instructor chooses AI tools from the catalog
- Each student deploys the model as jupyter connect
- Each student has now a personal environment
- The tasks are prepared in the jupyter notebooks
- Each student can work on the tasks and submit the notebook to the instructor

CERTIFICATION USE CASE

- Deploy model to certify
- Do certification steps in Jupyter Notebook
- Save and digitally sign the notebook
- Result is a reproducible protocol of the certification

5. THE EXECUTION-RUN DATA STRUCTURE

The execution-run.json is an extension of the blueprint.json which represents the pipeline topology and is the basis for deployments. It contains the nodes of a pipeline, the corresponding references to the docker images and the edges, which represent service calls between nodes. Let's have a look at this two-node-pipeline:



It has this corresponding blueprint.json:

```

{
  "probeIndicator": [
    {
      "value": "false"
    }
  ],
  "nodes": [
    {
      "proto_uri": "org/acumos/6a69881f-918f-4f64-a5a3-d0e8c9db9bf5/hppdatabroker/1.0.0/hppdatabroker-1.0.0.proto",
      "image": "cicd.ai4eu-dev.eu:7444/tutorials/house_price_prediction/hpp_databroker:v1",
      "node_type": "MLModel",
      "container_name": "hppdatabroker1",
      "operation_signature_list": [
        {
          "connected_to": [
            {
              "container_name": "hppmodel1",
              "operation_signature": {
                "operation_name":
"predict_sale_price"
              }
            }
          ]
        }
      ],
      "operation_signature": {
        "operation_name": "hppdatabroker",
        "output_message_name": "Features",
        "input_message_name": "Empty",
        "output_message_stream": false,
        "input_message_stream": false
      }
    }
  ]
}
  
```

```

    },
    "checksum": "docker-pullable://cicd.ai4eu-dev.eu:7444/tutorials/house_price_prediction/hpp_databroker@sha256:6d8c5f203a4bcc8a794b54d82a7428b36ffeb9f53b164968f576442e2bdde89f"
  },
  {
    "proto_uri": "org/acumos/81c8b420-98ba-48ed-a110-dd7d386f3d87/hppmodel/1.0.0/hppmodel-1.0.0.proto",
    "image": "cicd.ai4eu-dev.eu:7444/tutorials/house_price_prediction/hpp_predictor:v1",
    "node_type": "MLModel",
    "container_name": "hppmodel1",
    "operation_signature_list": [
      {
        "connected_to": [],
        "operation_signature": {
          "operation_name": "predict_sale_price",
          "output_message_name": "Prediction",
          "input_message_name": "Features",
          "output_message_stream": false,
          "input_message_stream": false
        }
      }
    ]
  },
  "checksum": "docker-pullable://cicd.ai4eu-dev.eu:7444/tutorials/house_price_prediction/hpp_predictor@sha256:c81afacc482a0e92afabeeb4c94e64fda90f269f9fdae6b0ad764edb752daf4"
}
],
"name": "hppsolution",
"version": "1",
"input_ports": [],
}

```

The Graphene Deployment Service creates the necessary Kubernetes config files which can then be applied in the execution environment, notably the AI Playground. After completing the execution of the pipeline, the “Run”, an extended version of the blueprint.json is written by the playground-app. The execution-run.json contains then additional information about the execution like the system-info and different types of metrics. Here is an example of the execution-run.json, the added objects are highlighted in light-blue:

```

{
  "probeIndicator": [
    {
      "value": "false"
    }
  ],
  "nodes": [
    {
      "proto_uri": "org/acumos/6a69881f-918f-4f64-a5a3-d0e8c9db9bf5/hppdatabroker/1.0.0/hppdatabroker-1.0.0.proto",
      "image": "cicd.ai4eu-dev.eu:7444/tutorials/house_price_prediction/hpp_databroker:v1",
      "node_type": "MLModel",
      "container_name": "hppdatabroker1",
      "operation_signature_list": [

```

```
{
  "connected_to": [
    {
      "container_name": "hppmodell1",
      "operation_signature": {
        "operation_name":
"predict_sale_price"
      }
    }
  ],
  "operation_signature": {
    "operation_name": "hppdatabroker",
    "output_message_name": "Features",
    "input_message_name": "Empty",
    "output_message_stream": false,
    "input_message_stream": false
  }
},
{
  "checksum": "docker-pullable://cicd.ai4eu-
dev.eu:7444/tutorials/house_price_prediction/hpp_databroker@sha256:6d
8c5f203a4bcc8a794b54d82a7428b36ffeb9f53b164968f576442e2bdde89f",
  "dataset_features": {
    "type": "aiod-dataset/v1",
    "datasetname": "The Reuters Dataset",
    "description":
"http://kdd.ics.uci.edu/databases/reuters21578/README.txt",
    "size": "4MB",
    "DOI_ID": "Not available"
  }
},
{
  "proto_uri": "org/acumos/81c8b420-98ba-48ed-a110-
dd7d386f3d87/hppmodel/1.0.0/hppmodel-1.0.0.proto",
  "image": "cicd.ai4eu-
dev.eu:7444/tutorials/house_price_prediction/hpp_predictor:v1",
  "node_type": "MLModel",
  "container_name": "hppmodell1",
  "operation_signature_list": [
    {
      "connected_to": [],
      "operation_signature": {
        "operation_name": "predict_sale_price",
        "output_message_name": "Prediction",
        "input_message_name": "Features",
        "output_message_stream": false,
        "input_message_stream": false
      }
    }
  ]
},
{
  "checksum": "docker-pullable://cicd.ai4eu-
dev.eu:7444/tutorials/house_price_prediction/hpp_predictor@sha256:c81
afacc482a0e92afabeefb4c94e64fda90f269f9fdae6b0ad764edb752daf4",
  "metrics": {
    "type": "classification-metrics/v1",
    "date_time": "2023-09-07 07:28:51",
    "accuracy": 0.921697199344635,
    "validation_loss": 0.9244863986968994,
```

```

    "status_text": "success"
  }
},
{
  "name": "hppsolution",
  "version": "1",
  "input_ports": [],
  "running_time": "345s",
  "system_info": {
    "system_name": "mwtest",
    "fqdn": "202.61.249.225",
    "cpu": "10",
    "gpu": "",
    "memory": "65851412Ki"
  }
}

```

The added checksum property will help to ensure reproducibility because it uniquely identifies the docker image that has been used in the run and can easily compared to other similar runs.

Different types of metrics are supported that depend on the type of AI model. Classification models have different metrics than Knowledge-Graphs or OCR.

EXTENSION OF THE CONTAINER SPECIFICATION

In order for the execution metadata capture to work, the execution environment needs to be able to somehow read the AI module specific metadata like dataset-features and metrics. This is done by scanning the logs of the containers for metakeys:

Leveraging Databroker for enhanced metadata with dataset's attributes

The dataset features can be incorporated into the Databroker. The information about the dataset type, dataset name, associated description, size, DOI ID, etc., may be found in the logs generated out of this node.

When the Pipeline is launched, these initial logs are recorded. The implemented method reads and finds the logs with the meta key - **dataset_features**. The extracted logs are then added to the metadata file (execution_run.json) as a Python dictionary.

A sample representation of the logs for the news_training Pipeline is as follows,

```

INFO:root:{'dataset_features': {'type': 'aiod-dataset/v1(TensorFlow
Dataset(tfds)', 'datasetname': 'The Reuters Dataset', 'description':
'http://kdd.ics.uci.edu/databases/reuters21578/README.txt', 'size':
'4MB', 'DOI_ID': 'Not available'}}}

```

Metrics aggregation

We can determine information about the model's performance by using a variety of metrics, including classification, regression, clustering, anomaly detection, ranking, GNN-specific, custom metrics, etc. Obtaining and updating these metrics in the metadata is paramount to understanding the need to fine-tune the model or improve the outcomes. As a result, by including the metrics from the nodes/containers they are contained in, the created metadata file, `execution-run.json`, may be constructed more concisely.

The model provider should ensure the following additions in order to accomplish the acquisition of the metrics,

1. Have message fields for the metrics defined in the protobuf definitions.
2. Adding an initialization flag (`has_metrics = True`) indicating the presence of metrics in the particular node/module and logging a message (`logging.info('MetricsAvailable')`) accordingly. In doing so, we only output a list of nodes containing the metrics when the Pipeline is launched.
3. Have the metrics collected and updated after the training process. For this purpose, a gRPC routine/method, `get_metrics_metadata(self, request, context)`, is subsequently called after the training process concludes.
4. The Python logging should be enabled in order to see all the logs that are obtained.

Please refer to the Additional Information section to further comprehend the topics mentioned above.

Enhancing the playground-app repository with metadata functionality

When the Pipeline is started, the initial logs indicated in 2. are captured, allowing merely the nodes with metrics to be retrieved. When the Pipeline is executed, these nodes are further checked for `metakey - metrics`. Typically, these log entries will be at the end. A Python dictionary containing the retrieved logs is then appended to the metadata file (`execution_run.json`).

A sample representation of the metrics logs for the `news_training` pipeline is as follows,

```
INFO:root: {'metrics': {'date_time': '2023-09-15 08:17:22', 'type': 'classificationMetrics/v1', 'accuracy': 0.9247897267341614, 'validation_loss': 0.9067514538764954, 'status_text': 'success'}}
```

Note : Another field in the metadata is 'type,' which represents the model's metrics type. This field is critical not only for model evaluation but also for the model's performance to the precise goals and requirements of the application. Adding this feature allows one to group the metrics into a primary group or sub-group of metrics indicating the training and testing phases.

A sample further extension of the metadata is as shown below,

```
{
  "metrics": [
    {
      "type": "classification-training-metrics/v1",
      "date_time": "2023-09-07 07:28:51",
      "accuracy": 0.9216,
      "validation_loss": 0.9244,
      "status_text": "success"
    },
    {
      "type": "classification-testing-metrics/v1",
      "date_time": "2023-09-07 07:28:51",
      "F1 Score": 0.85,
      "Specificity": 0.90,
      "ROC-AUC": 0.92,
      "status_text": "success"
    }
  ]
}
```

LEVERAGING THE EXECUTION-RUN METADATA

After the execution of a pipeline is completed, the user can view or download the execution-run metadata for further examination or for reference.

In case the execution generated satisfactory results, the user can decide to send the execution-metadata back to the catalog so that it can be associated with the AI model and can be further used for comparison, benchmarking and ranking in challenges. It will show up in the “Model Artifacts” panel on the model details page.

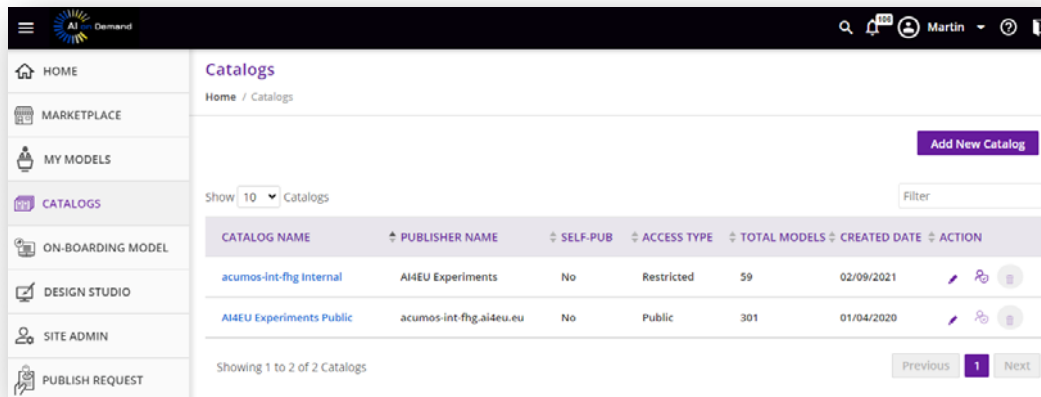
The screenshot displays the 'i-nergy-load-forecasting' model page. The left sidebar contains navigation links: Description, License Profile, Signature, Documents, Model Artifacts (selected), Author/Publisher Details, and Tags. The main content area is titled 'MODEL ARTIFACTS' and contains a table with the following data:

ARTIFACT NAME	VERSION	MODIFIED ON	SIZE	ACTION
license-1.0.0.json	1.0.0	10/05/2021	486 Bytes	Download
docker.io/glampropo/i-nergy-load-forecasting:1.0	1.0.0	10/05/2021	N/A	Download
OnboardingLog-1.0.0.txt	1.0.0	10/05/2021	6.2 KB	Download
i-nergy-load-forecasting-1.0.0.proto	1.0.0	10/05/2021	324 Bytes	Download
execution-run-1.0.0.json				

6. THE CHALLENGES CONCEPT

The remaining time of the project will be used to implement the challenges feature in the catalog, for which we already have a concept that will be described in the following sections.

First of all, the overall “catalog” is actually an aggregate view of several sub-catalogs and each sub-catalog has an access-type and contains a distinct set of models (= AI Modules):



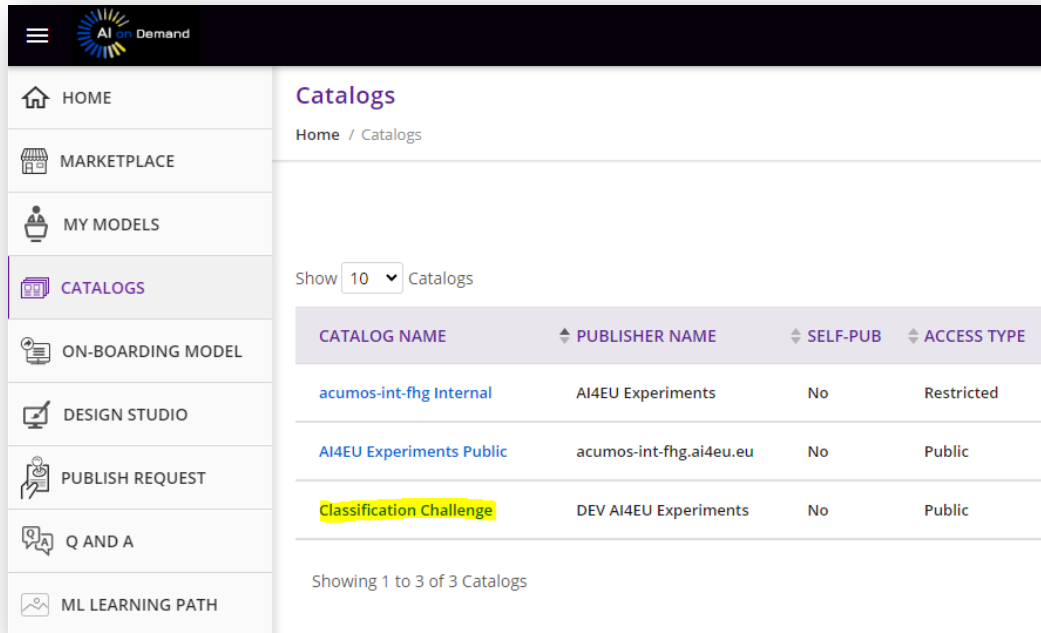
CATALOG NAME	PUBLISHER NAME	SELF-PUB	ACCESS TYPE	TOTAL MODELS	CREATED DATE	ACTION
acumos-int-fhg Internal	AI4EU Experiments	No	Restricted	59	02/09/2021	Edit Share Info
AI4EU Experiments Public	acumos-int-fhg.ai4eu.eu	No	Public	301	01/04/2020	Edit Share Info

Showing 1 to 2 of 2 Catalogs

Previous 1 Next

A challenge is basically a special kind of catalog in the sense that it has an special view called “Leaderboard” where the entries are ordered by the metrics of execution-runs, best models first.

So, to start a new challenge, the challenge owner has to create a new catalog of type “Challenge” for it:



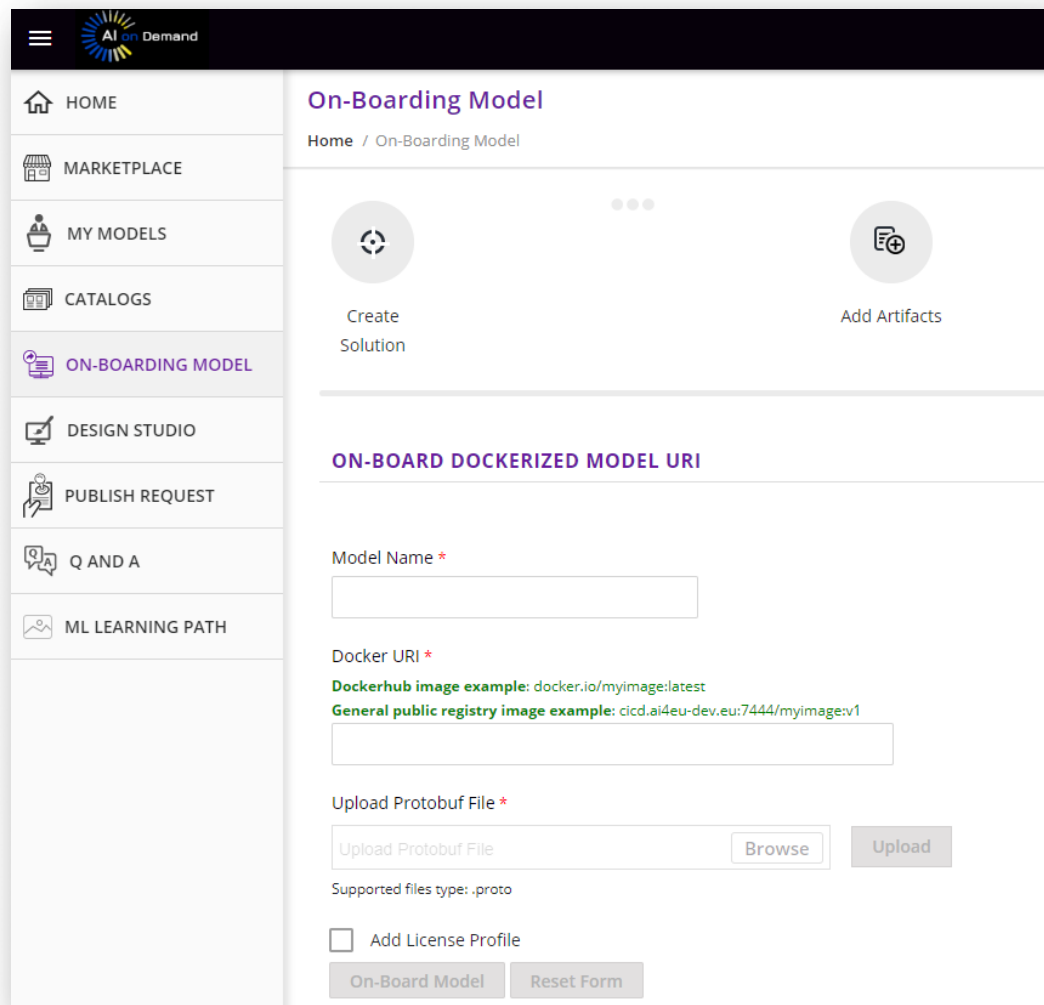
Home / Catalogs

Show 10 Catalogs

CATALOG NAME	PUBLISHER NAME	SELF-PUB	ACCESS TYPE
acumos-int-fhg Internal	AI4EU Experiments	No	Restricted
AI4EU Experiments Public	acumos-int-fhg.ai4eu.eu	No	Public
Classification Challenge	DEV AI4EU Experiments	No	Public

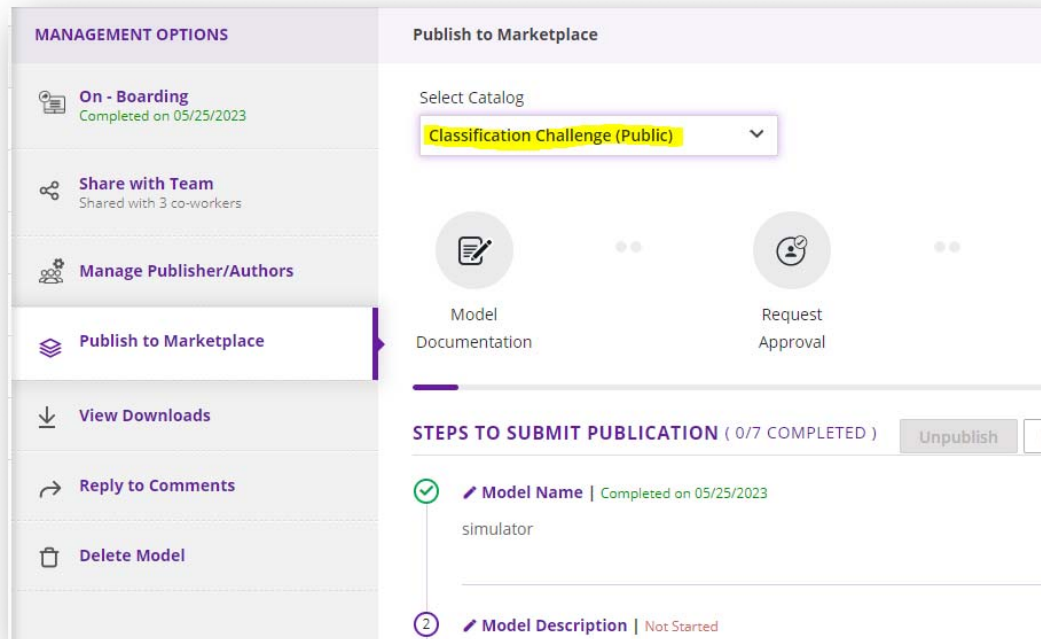
Showing 1 to 3 of 3 Catalogs

The challenge participants can then do the regular Onboarding of their AI-Modules



The screenshot shows the 'On-Boarding Model' page in the HumanE AI Net application. The left sidebar contains a navigation menu with the following items: HOME, MARKETPLACE, MY MODELS, CATALOGS, ON-BOARDING MODEL (highlighted in purple), DESIGN STUDIO, PUBLISH REQUEST, Q AND A, and ML LEARNING PATH. The main content area is titled 'On-Boarding Model' and includes a breadcrumb 'Home / On-Boarding Model'. Below the title, there are two circular buttons: 'Create Solution' and 'Add Artifacts'. A horizontal line separates this section from the 'ON-BOARD DOCKERIZED MODEL URI' section. This section contains three form fields: 'Model Name *' with a text input, 'Docker URI *' with a text input and two example lines ('Dockerhub image example: docker.io/myimage:latest' and 'General public registry image example: cld.ai4eu-dev.eu:7444/myimagev1'), and 'Upload Protobuf File *' with a text input, a 'Browse' button, and an 'Upload' button. Below these fields, there is a checkbox for 'Add License Profile' and two buttons: 'On-Board Model' and 'Reset Form'.

To actually take part in the challenge, the model must be published to the challenge catalog, in this example it would be the Classification Challenge Catalog:



Publication is a two-step process: in the first step, the user submits the module for publication and in the second step, a reviewer must approve the publication.

Finally, the participant must create a pipeline that connects their model with the Challenge input data (a “Databroker”) and maybe add other necessary AI-Modules. A minimal pipeline looks like the one at the beginning of chapter 5 with just two connected nodes.

RANKING, BENCHMARKING AND LEADERBOARD

Now everything is prepared to associate execution-runs with the AI-Module competing in the challenge. For that, the participant deploys the pipeline to an execution environment, for example the AI Playground and start a pipeline run. When a satisfactory result is achieved, the user can send the execution-run metadata back to the catalog. Based on this metadata, comparisons like benchmarking and ranking can be done.

So, based on the collected metrics, the AI-Module is ranked against the other submissions in the Challenge catalog and put in the right place on the Leaderboard:

The screenshot displays the HumanE AI Net Marketplace interface. The top navigation bar includes a search icon, a notification bell, a user profile icon labeled 'Martin', and a help icon. The left sidebar contains a menu with options: HOME, MARKETPLACE, MY MODELS, CATALOGS, ON-BOARDING MODEL, DESIGN STUDIO, PUBLISH REQUEST, Q AND A, and ML LEARNING PATH. The main content area is titled 'Marketplace' and shows a list of models under the 'Classification Challenge' catalog. The models are ranked 1 to 5:

- SmartRiver** by author466 | on 10/13/2022 | New. Rating: 5 stars. 0 reviews, 19 views, 4 downloads.
- SWE predictor** by author1083 | on 10/12/2022 | New. Rating: 5 stars. 0 reviews, 18 views, 1 download.
- lexatexer-ai4hydro-proxy** by author522 | on 10/11/2022 | New. Rating: 5 stars. 1 review, 25 views, 0 downloads.
- audio-file-broker** by author207 | on 05/24/2022 | New. Rating: 5 stars. 0 reviews, 19 views, 2 downloads.
- AI REGIO DSS4TB** by author1198 | on 05/24/2022 | New. Rating: 5 stars. 0 reviews, 19 views, 2 downloads.

The interface also includes a 'Filter By Category' section with checkboxes for Classification, Data Sources, Data Transformer, Prediction, and Regression. A 'Tags' section lists various keywords like semantic web, model baseline, word2vec, artificial intelligence, anonymization, purchasing, ai4agri, GUI, ontology, Dataset assessment, energy, ML, python, AI, gpt, language model, Pose Estimation, conditionmonitoring, classification, Forecasting, Planning under uncertainty, AI REGIO, MultiClass Classification, planning, data quality, and prediction.

7. REFERENCES

1. AI4EU Experiments / Virtual-Lab [Container Specification](#)
2. AI4EU Experiments / Virtual Lab production System: [AI4EU Experiments](#)
3. Docker <https://docs.docker.com/>
4. gRPC <https://grpc.io/docs/>
5. Protocol Buffers <https://protobuf.dev/>