IVVES

Industrial-grade Verification and Validation of Evolving Systems

Labeled in ITEA3, a EUREKA cluster, Call 5

ITEA3 Project Number 18022

D2.4 – Final version of Validation Methods and Techniques for ML

Due date of deliverable: March 31, 2022
Actual date of submission: March 25, 2022

Start date of project: 1 October 2019 **Duration:** 39 months

Organization name of lead contractor for this deliverable: Solita

Author(s): Tijana Nikolic, Almira Pillay, Sogeti; Timo Lehtonen, Solita; Harri Pölönen, Janne Merilinna, Johan Plomp, VTT; Mark Van Heeswijk, Tatu Aalto, F-Secure; Lalli S. Myllyaho, Tuomas P. Halvari, Jukka K. Nurminen, Zafar Hussain, Dennis Muiruri, University of Helsinki; Mahshid Helali Moghadam, Sima Sinaei, RISE; Steven Boylan, Ana Soto, CCTL; Marko Koskinen, Techila Technologies; Eva Garcia Martin, Ekkono Solutions; Mark Pijnenburg, Philips

Status:	Draft
Version number:	1.0
Submission Date:	25-March-2022
Doc reference:	IVVES_Deliverable_D2.4._V1.0.docx
Work Pack./ Task:	WP 2 / T2.2
Description: *(max 5 lines)*	This document describes the final version of validation methods and techniques for ML considering use case requirements.

Nature:	**X** R=Report, ☐ P=Prototype, ☐ D=Demonstrator, ☐ O=Other		
Dissemination Level:	**PU**	Public	**X**
	PP	Restricted to other program participants	
	RE	Restricted to a group specified by the consortium	
	CO	Confidential, only for members of the consortium	

DOCUMENT HISTORY

Release	Date	Reason of change	Status	Distribution
V0.1	27/01/2022	First draft, kick-off version	Draft	All
V0.2	25/03/2022	First version	Concept	Authors
V1.0	25/03/2022	Approved by PMT, to be submitted to ITEA3	Final	Uploaded to ITEA

Table of Contents

1. Glossary

Abbreviation or acronym	**Description**
ADAS	Advanced Driver-Assistance System
AI	Artificial Intelligence
AN	Agglomerative Nesting
ANN	Artificial Neural Networks
BC	Branch Classifiers
BraTS	Brain Tumor Segmentation
BNN	Bayesian Neural Network
CART	Classification and Regression Trees
DL	Deep learning
DevOps	Development and operations
DIP-VAE	Disentangled Inferred Prior Variational Autoencoder
DNN	Deep Neural Network
DOE	Degree of Correctness
DQW	Data Quality Wrapper
DTM	Document Term Matrix
EDA	Explorative Data Analysis
ETL	Extract, Transform, Load
GAN	Generative Adversarial Network
KG	Knowledge Graph-based
KNN	K-Nearest Neighbour
LDA	Linear Discriminant Analysis
LIME	Local Interpretable Model-Agnostic Explanations
ML	Machine Learning
MLOps	Machine Learning Operations
NBDT	Neural-Backed Decision Trees
NLP	Natural language Processing
OS	Operating systems
OT	Operational Technology
PDP	Partial Dependence Plot
PDS	Pedestrian Detection System
PLC	Programmable Logic Controllers
ProtoDash	Prototypes with Importance Weights
QAIF	Quality Artificial Intelligence Framework

RL	Reinforcement Learning
SHAP	SHapley Additive exPlanations
SUT	System Under Test
SVD	Singular Value Decomposition
WEKA	Waikato Environment for Knowledge Analysis
XAI	Explainable Artificial Intelligence

2. Executive Summary

This document (D2.4) describes the final validation methods and techniques for machine learning (ML) in project use cases. During the previous phases of the IVVES project, the state of the art of the validation techniques for ML were described in document D2.1. The work continued in deliverable D2.2 where an initial version of validation methods and techniques for ML was developed. The document acted as a project plan for deliverable D2.3, where the actual tool development was started. Finally, this deliverable (D2.4) presents the final version of methods and techniques for ML in project use cases. The document is based on previous deliverable documents D2.1, D2.2 and D.2.3. Moreover, the latest advances of tool development are included in this deliverable with directions for future work in the last part of the project.

The objectives of this work package are to:

- Assess data collection methods and techniques that produce data with adequate quality to be used by ML algorithms.
- Analyze data quality to support testing of trained ML algorithms.
- Improve training data quality by enriching small or incomplete training data sets.
- Develop methods, techniques and tools that address quality aspects of AI and ML models.
- Devise techniques to assess quality aspects of ML models after the model training phase.

Based on the above objectives, three tasks are derived:

- Task 2.1 – Model Quality.
- Task 2.2 – Incoming Data QA.
- Task 2.3 – Testing techniques for ML.

This document is divided into three parts based on the above tasks. First, a section with title **Incoming Data QA**, contains descriptions of synthetic data cases in the healthcare sector and cyber security. Additionally, a section dedicated to quality data is included.

Second, the **Model Quality** section describes the topic of Explainable AI in financial investments and industrial environments. This section also discusses the MLOps topic, focusing on drift detection when monitoring ML Models in the cyber security field and on inference scalability.

Finally, the section for **Testing techniques for ML** contains information on ML-assisted testing, testing learning algorithms, metamorphic testing, and testing model robustness. The testing techniques can be applied in many sectors, for instance, in finance, robotics, healthcare and industrial automation.

The outputs are novel. The tools were developed to address several topics in using artificial intelligence, especially machine learning, to create value in many sectors of society. Since the previous document deliverable, we have introduced three new tools:

- Knowledge distillation by Ekkono in section 4.3.2.
- Data harvester by Solita in section 3.2.2.
- Adversarial example generator by RISE in section 5.2.

The following tools were discontinued and will not be included in this deliverable:

- Testing learning algorithms by CRIM.
- Neural backed decision trees by Sogeti NL.

This deliverable is a continuation of deliverable 2.2. which was the initial version. The final version of the methods and techniques are explained in their designated chapters. At the end of this document, we describe how the following tools can be used together to operationalize ML tasks and automate the quality assurance checks for model development and deployment which follows the Quality AI Framework (QAIF) best practices described in previous deliverables.

3. Task 2.2 - Incoming Data QA

Task lead: Solita

In this chapter, we introduce the important topic of incoming data quality assurance. The focus is on machine learning [1]. Quality assurance in general is a wide topic in the industry [2]. Incoming data quality and its assurance is a key element in every machine learning solution. If the quality of data is poor, the model built on top of the data will also be poor. In this chapter, we present synthetic data in different domains, for instance healthcare. Moreover, a data quality wrapper approach is presented. Special focus has been put to acoustic analysis. In addition to the topics mentioned, semi-natural language data, training data quality and data quality for text driven ESG investment systems are presented. There are several tools related to this chapter. First, the data quality wrapper (DQW) tool. DQW tool has been applied to several use cases. Moreover, several tools for data collection have been implemented.

3.1. Synthetic Data

3.1.1. Simulated data for healthcare

Solution provider: VTT

For verification of the MRI scanner functionality and software, large clinical data sets of the brain are needed. Data sets are typically created using volunteers during development and verification/validation activities. The availability of volunteers, however, is limited. Moreover, unlike the patients to be scanned at the clinic, the volunteers typically are healthy and do not contain tumors and other anomalies of interest. Using volunteers during the verification execution is also time consuming, resulting in long lead times and strict planning of activities, limiting flexibility of execution. In addition, volunteer scanning cannot be automated, typical sample sizes are small and data privacy rules and regulations prevent the sharing and using of data from hospitals.

Synthetic MRI data created with AI models can help to solve the above issues. There are AI models that can adopt the essential characteristics of a given data set and after intensive training learn how to generate similar - but not exact duplicates - of the given data samples. Most widely used such AI models are generative adversarial networks [3] (GANs). In case enough clinical MRI data samples are available, a GAN model can be trained to produce massive amounts of synthetic MRI data. Synthetic MR data created this way is anonymous and can be used in combination with automated test execution.

Data privacy rules slow down acquisition of adequate clinical MRI datasets significantly and demand major restrictions to share data with research partners. Therefore, a data plan/strategy is required. We inspect also whether it is better to create “raw” (spatial frequency and phase information) or “imaging” synthetic data. It also needs to be investigated how to feed the synthetic data into the test framework and validate the data for its intended use.

As next steps, we aim to create a synthetic data plan defining what data to generate, how the training/generation is performed, how much data is needed, how to validate, data privacy impact analysis, etc. The AI model trained partly with synthetic data will be tested on customer data, when available. We will define product equivalent interface for synthetic data insertion and validate quality of synthetic data with clinical specialists. While all the previous steps are completed successfully, we will extend the AI model to create synthetic data for other anatomies i.e., knee, spine, breast.

The synthetic MRI data can be generated either via physics-based model or via a data driven approach. Physics based model tries to imitate the whole scanning process including human anatomy, scanner technology and physical phenomena such as magnetic field distortions. In data driven approach an algorithm is used to generate samples like given training data.

Here we concentrate on data driven approach using GAN models and public MRI data sources. There have already been a lot of development efforts on GAN models and in recent years also three-dimensional GAN approaches have emerged. As the algorithm development is done widely on open-source philosophy, most of the suitable GAN algorithms are available publicly. We concentrate on applying these algorithms to our case instead of developing a new algorithm from scratch. First, we start with public data sets and later aim to obtain clinical data from our partners.

Data driven solutions depend heavily on the availability and quality of the data. The first application we investigated was low-grade and high-grade gliomas i.e., brain tumors [4, 5, 6]. The public data set we obtained had less than 400 patients with MRI images. We managed to apply the state-of-the-art GAN algorithms to this data, but the quality of the generated synthetic data was far from the original data used in training. Next, we obtained a public data set of MRI images from 1112 healthy young adults as the training data [7]. With this dataset we were able to produce synthetic MRI images with much better quality and anatomical correctness (see Figure 1). A medical expert confirmed that the anatomy in these images looks correct. However, it is worth mentioning that the human brain is a relatively easy target generative AI model for our approach as the anatomical variation is small in that region. Applying the algorithm e.g., to abdominal region data may require much more training data.

We checked that the synthetic MRI data generated from 1112 healthy young adults seem to be unique in the sense that we managed to confirm that the generated volumes are not replicas of any of the training MRI images. However, as AI models are very complex containing millions of parameters, it is very challenging to prove that privacy is 100% preserved. Although privacy issues are not in focus in this challenge, we keep an eye on the progress on this field of science.

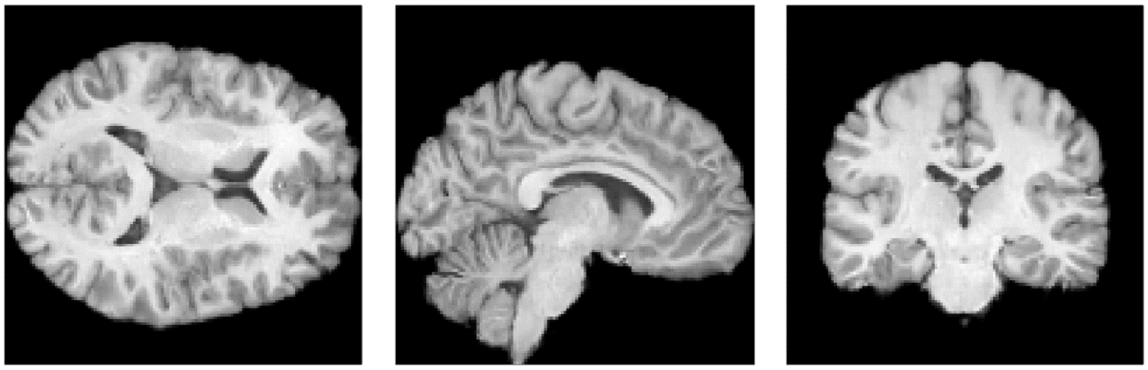


Figure 1. Synthetic 3D MRI data created from a dataset of 1 112 healthy young adults

As the GAN model [8] we used seems to work well with healthy young adults, we continued to investigate new datasets and use cases. As we are working with medical data and are interested in various pathologies, the shortage of data will always be an issue. Thus, we are investigating methods to process the training data in various ways in order to expand the available data for the GAN model. Note that this process is different to commonly implemented data augmentation schemes already included within many AI models as we modify the training data itself. One approach that we have noticed to work to some extent is to perform non-rigid registration between pairs of original training data samples and create a “new” data sample from half-transformed sample. We also investigate pre-processing methods to modify the training data more consistent, for example via co-localization and intensity normalization. The Alzheimer's Disease Neuroimaging Initiative [9] (ADNI3) dataset is used in this study. See Figure 2 for the first results with this limited (366 Alzheimer’s patients) dataset.

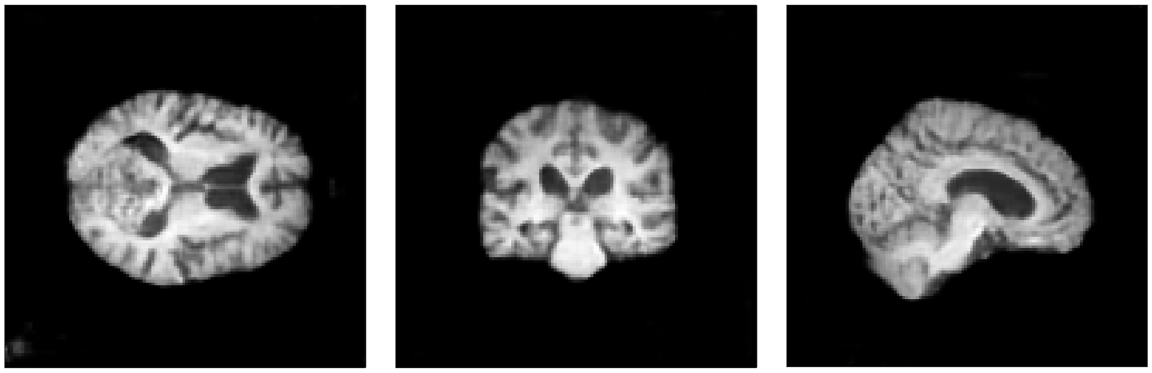


Figure 2. Synthetic 3D MRI data created from a dataset of 366 Alzheimer's patients

We also keep an eye on state-of-the-art on this field and try out any new available GAN algorithms that are published. The generated synthetic datasets will be used in training and testing existing AI algorithms. Final output is planned to consist of an AI algorithm that can reliably generate synthetic three-dimensional medical data like the given training data, understanding of the capabilities and limitations of the proposed AI algorithm and one or more synthetic datasets created with the AI algorithm.

3.1.2. Simulated data for cybersecurity

Solution provider: F-Secure

The main issues with quality training data in the OT domain is the confidentiality of the customer data, extensive use of proprietary undocumented protocols and the challenges in obtaining such data from the highly sensitive control system networks. To battle all the issues, the only way to get good quality data and to get the data for multiple scenarios is to simulate the data in as close to the real setting as possible. In practice, this means using the same or similar OT components typically found in the OT setting. This includes PLCs (Programmable Logic Controllers) and real sensors and actuators.

To enable this F-Secure built a flexible platform that doesn't take up a lot of space but allows the simulation of wide variety of different scenarios where OT can be found, this includes but is not limited to industrial facilities, marine vessels, and trains. The platform makes it possible for a small scale but accurate representation of the scenario in question, this includes correctly simulating the process flow, physical simulation of the process and generating simulated data that is accurate for the scenario.

The final version of the F-Secure simulation platform is built using Siemens S7-1200 PLCs, Siemens HMI displays, 3 simulation PCs and bunch of cyber physical sensors and actuators.

Figure 3. F-Secure simulation platform

The platform is a full-sized facility in a miniature form factor. It accurately follows and simulates all of the levels in the Purdue reference model for OT environments. The simulation platform can be used to simulate a wide range of attack scenarios against OT systems and to collect data for further analysis and AI training.

The platform also enables the testing of AI based threat mitigation methods and their effectiveness in OT environments and whether there are adverse effects on the production systems.

3.2. Incoming data QA methods

This section focuses on ensuring quality in data prior to the model training phase. The final version of the methods focuses on more data structures than in the initial version, with, most notably, audio data being added. Furthermore, VTT's contribution to training data quality focuses on feature importance and engineering, giving an MLOps perspective to handling data.

3.2.1. Data Quality Wrapper

Solution provider: Sogeti NL

The Data Quality Wrapper (DQW) [10] is an automated EDA and data selection tool. This tool has potential to be used in any AI workflow, in any industry in the Data Understanding and Data Preparation phase of the QAIF (See more information in chapter 6).

The methods and approach to the preprocessing are deployed in a Streamlit [11] app. Streamlit is an app framework that ensures effortless and streamlined application development, specifically tailored for machine learning and data science. Streamlit can be used for educational and demonstration purposes, so we found the approach of making these methods available as an application to a broader audience is beneficial for the ITEA IVVES project-related work dissemination.

The packages used in the tool have undergone some changes due to the validation of approaches. The updated package table is below.

(Sub)section	Description	Visualisation	Selection	Package	Reference
Synthetic tabular	✓	✓		Table-evaluator	[12]
Tabular	✓	✓		Sweetviz	[13]
Tabular	✓	✓		Pandas Profiling	[14]
Tabular, text			✓	PyCaret	[15]
Text			✓	NLTK	[16]
Text			✓	SpaCy	[17]
Text	✓		✓	TextBlob	[18]
Text	✓	✓		WordCloud	[19]
Text	✓		✓	TextStat	[20]
Image	✓	✓		basic-image-eda	[21]
Audio	✓	✓		librosa	[22]
Audio	✓	✓		dtw	[23]
Audio	✓	✓		AudioAnalyser	[24]
Audio			✓	audiomentations	[25]
Report generation	✓			Fpdf	[26]
Report generation	✓			pdfkit	[27]

Table 1. The packages used in the DQW.

The final version of the methods used to preprocess and ensure quality for the following data structures:

1. Structured data analysis – improved feature:
   a. Added more elaborate analysis methods.
2. Text data analysis – improved feature:
   a. Improved runtime.
   b. Improved preprocessing.
3. Audio data analysis – new app feature.
4. Image data analysis – new app feature.

Structured data

Since the initial version of the methods included the EDA of one dataset, we have validated this approach and found that having more flexibility in this app section brings a lot of value. Therefore, we added more subsections in this part of the app:

- One file analysis with pandas-profiling, an automated EDA package. This package outputs a report that is used to share EDA findings with team members, ensuring operationalization and transparency of the data understanding process in AI model development.
- One file preprocessing with PyCaret – a very useful package for workflow automation. In the DQW, we rely on a PyCaret function which creates the preprocessing pipeline. Transparency into the pipeline is ensured by displaying a diagram of the preprocessing steps taken. Furthermore, a pipeline pickle is provided so it can be used in model training, especially in case of **imbalanced class mitigation** with SMOTE. The sampling needs to happen within training folds, so you will not be able to see any impact of this method on the datasets themselves, but you will be able to see the difference in model performance when you use the pipeline pickle file.
- Comparison of two files with Sweetviz – another automated EDA library in python that is extremely useful in cases of comparison of two files. This package also outputs a report that can be shared with relevant team members.
- Comparison of original and synthetic data with table-evaluator. This package offers descriptive statistics' comparison of the synthetic and original dataset, together with PCA analysis of both. Additionally, the package can be used to train various ML models (Logistic Regression, Random Forest, Decision Tree Forest, MLP Classifier) and compare the F1 scores obtained from the training results on the synthetic and original dataset.
- Report generation is also added to the final version of this method. Reports are useful for operationalization of a technical process like data preprocessing.

Text data

The text data part was well received in the validation part, but it has been noted that the flexibility in languages needs to be improved as the app only supports English. Due to the time-consuming nature of adding other languages to the app, Sogeti has decided to abandon this improvement and focus on text data preprocessing and runtime improvement.

The preprocessing has been set to the following:

- Tokenizing with NLTK. The process of splitting input text into tokens that can be passed to subsequent steps in the preprocessing pipeline.
    - Normalizing
    - Removing email and URL patterns
    - Removing non-ascii characters [28]
    - Converting to lowercase
    - Removing punctuation
    - Replacing numbers with words
- Removing stop-words with NLTK.
- Lemmatizing with NLTK.

The topic analysis has been improved by adding the LDA *u_mass* coherence score [29] for automatic selection of the topic number based on coherence scores. This is an advanced, automated method of LDA performance analysis which can save time in the text preprocessing pipeline because the developer doesn't need to experiment on their own. This feature was added in collaboration with Helsinki University.

Audio data

A data source that was not recognized in the first version of the methods, audio data is used in audio signal processing algorithms such as music genre recognition and automatic speech recognition. We have realized that this part of the app is very useful as the audio data pipeline is complex and developers would benefit from having a standardized way of audio data analysis and augmentation. Furthermore, this data structure preprocessing and quality methods were validated and added in collaboration with Solita as they are utilizing audio data (see 3.2.2.) for a predictive maintenance use case.

In this section, we have added:

- **One file analysis** where we provide plots that can be saved as potential training data for classification algorithms or simply used to understand the audio data file:
    - Waveform plot [30].
    - Power Spectrum plot [31].
    - Short-time Fourier transform plot [32].
    - Mel Frequency Cepstral coefficient plot [33].
- **1 file augmentation** with audiomentations, useful for increasing training data robustness as audio data is known for being difficult to collect.
- **Comparison of two files** with:
    - Spectrum compare, a method to compare two audio files based on their power spectrums, with applied thresholds.
    - Time Warping (DTW) [34], a method of analyzing the maximum path to similarity of 2 sequences with different lengths. This method is added to the DQW because of its application to speech recognition, where words may mean the same but are pronounced differently.

Images

Image preprocessing (augmentation) was added to the final version of the methods due to the importance of having a versatile image training set for computer vision algorithms. Furthermore, image augmentation can be used as a method of testing data collection (for metamorphic tests).

- **EDA of images** for basic understanding of provided images, including file type, size and name information, image sizes, color histograms and color channels.
- **Augmentation of the images** using Pillow – the app offers several augmentation methods, including image resizing, applying noise, contrast, and brightness adjustment.

Conclusion

To sum up, DQW is a tool for automatic preprocessing of data. Additionally, it offers report generation to operationalize the Data Understanding and Data Preparation phase of the QAIF. Through operationalization and standardization, we aim to offer methods to quantify and record observations made in these two phases. Finally, these methods support the data-centric AI approach, which puts data handling, quality, and validation to the forefront of the AI project.

3.2.2. Audio data QA

Solution provider: Solita

Industrial grade evolving systems (ES) consist of software and hardware. On the Edge, there may even be thousands of devices which evolve all the time. New versions of software are released continuously. Firmware of the device is updated. Hardware has been replaced with a newer one over the years. The devices are often connected to each other by a local network or internet of things where machine to machine (M2M) communication is needed. Cloud computing is often the endpoint which then sends feedback to the devices and vice versa. These kinds of complex evolving systems in the industry provide a very interesting challenge in many ways, especially when AI is involved. Furthermore: how to execute AI on the Edge with limited resources real-time?

How to verify and validate a complex ES? One answer is to develop special methods, techniques, and tools. In this context, in co-operation with IVVES partners, we have developed methods and techniques for validating data-driven ML solutions. In this subsection we focus on incoming data QA, especially audio data which is a very promising data source for many industrial-grade use cases.

Sound is a very promising data source for building industrial-grade machine learning models. The methodology developed in this project context is heavily based on experimentation. To conduct an industrial-grade experiment for a business relevant use case, e.g., predictive maintenance, the preparation must be very careful. How to conduct an experiment? Which phenomenon is validated? It may be assumed that an experiment is part of a proof-of-concept (PoC) implementation. However, what is the concept that is proved?

At this stage, we'll have to define sound data, often called audio data. Audio data is everywhere. Humans communicate with voice by producing nouns and vowels. In addition to this, industrial-grade machines produce sounds. The question is: how to apply traditional speech recognition techniques to audio data of machines?

There are several audio data file format types available. The format types can be divided into three categories. First, uncompressed audio formats (for example WAV, AIFF, AU and raw) save the audio data as is. Second, lossless compression (for example, FLAC, Monkey's audio, WavPack, TTA, ATRAC, ALAC (.m4a), MPEG-4 SLS etc.). Finally, lossy compression (for example, MP3, Vorbis, AAC, Musepack, WMA etc.) provide a way to store audio data in a smaller place [35].

From data farming point of view [36], audio data is a very potential data channel. In data farming, it is very important to get any data. For instance, data farming in audio data can be applied to, e.g., machines in an industrial-grade setting.

Audio data analysis often begins with tagging the data. There are two kinds of tags in this case. First, low level tags can be applied to any data sample. They characterize raw data. Second, logical tags characterize the dataset on a higher abstraction level. For instance, in case of an industrial-grade setting, a machine on a factory floor may produce a waveform of sounds that characterize how the machine works.

When the tags are in place, for instance, deep learning in audio data analysis is a topic that has been widely researched [37]. Deep learning solutions produce analysis that are often hard to interpret. Moreover, principal component analysis (PCA) of audio data is another approach. From explainable AI (XAI) point of view, the results are easier to explain due to the mathematical approach instead of neural networks.

We have conducted several experiments in co-operation with IVVES partner. The first experiment was an A/B test at the office. We tried to tell if the machine is ok (A) or broken (B) based on the sound. The second experiment was at the machine rental shop. A contact microphone was attached to the machine. An air microphone records sound, i.e., vibration through air. No digital metadata tool was available at that time. Therefore, we've implemented a tool called Data Harvester, which is used for collecting data.

In the IVVES project with our partners, we went to a local machine rental shop to conduct an industrial-grade experiment. The setup was set up very carefully. The goal of the industrial-grade experiment in a real industrial environment (a machine rental shop), was to recognize a machine by its audio finger printing. This has been applied to consumer products for music recognition. One of the most famous services is Shazam [38]. It can recognize practically any piece of music ever published. The solution is based on audio fingerprinting or audio hashing [39]. When the sound of an industrial machine is recorded, sound data quality is a key concept. To put it short, sound data must have at least three attributes in place, which are depicted in the following.

First, gain of the recording [40]. This is a very special attribute in the real world. If gain is too low or high, the collected audio data can be corrupted. The gain level must be adjusted according to the environment. If the machine to be recorded is very loud, any peaks in the data can be bad from audio data quality point of view. Naturally, automatic gain control systems exist. Although already in the 1950's, automatic gain control systems were presented [40], there is no solution that would always work out-of-the-box.

Second, background noise filtering while recording audio [41]. This problem has been addressed earlier [41]. The problem of background noise is very relevant in an industrial environment. For instance, the factory floor is a place where there may even be thousands of machines operating at the same time. Moreover, outdoor environments related to one of the partners, namely Bombardier, where the trains operate, background noise is a relevant challenge.

Finally, extra resonance with equipment nearby. In the conducted experiments, nearby magnetic fields, for instance, have been an issue affecting audio data quality. The phenomenon to be recorded may be affected by many potential sources of error.

In this project, we have been piloting three different topics in focus described below.

First, data collection with good enough quality. This topic has been an interesting problem. The quality of the microphone of the recording device affects data quality at scale. With our partners, we have been experimenting with very cheap microphones. This is because a microphone may be damaged in an industrial environment. Many consumer devices, e.g., mobile phones, have microphones by default. We have been trying to use such readily available devices. In the future, an experiment related to the comparison of high-quality microphones with very cheap ones, is apparent. A formally setup experiment would provide information on this important topic.

Second, data analysis with novel methods. In the well-managed IVVES consortium project, the partners have been able to co-operate. Novel mathematical methods, for instance, combination of self-similarity matrices and principal component analysis, have been developed. Moreover, applications of applying machine learning in combination with traditional mathematical methods, have been developed further. The complementary of the developed tools will help to develop a methodology for sound data analysis in the future.

Finally, the data has been analyzed as part of the consultancy business. Sensor software consulting for sensor fusion. In the consultancy business, we have an empirical context with real-world problems to be solved. Initial pilots with customers have been planned.

In the future, we are planning to implement a data collection tool for fault detection by remote observation, namely the Data Harvester. The tool collects any sensor data on the Edge. The incoming data is processed before it is sent to the cloud, in this case, Amazon Web Service and Microsoft Azure. In the future, the multi-cloud approach will most likely contain Google Cloud Platform, too.

Predictive maintenance is an important part of the life cycle cost of any machine in industrial context. IVVES project has involved several research organizations and partners with whom we are able to conduct real-world experiments with real-world business feedback. With this approach we can make an impact that lasts. How to identify the need to replace a consumable part? (e.g., drill bit or grinding wheel)

By collecting voice and radar data in an experiment in cooperation with VTT, where A = machine in good condition and B = need for maintenance. Repeat experiments A and B alternately so that the heating of the machine or material is not affected

Two more pairs of A and B are collected, but these samples have been kept hidden from the beginning, so it has not been possible to learn these from the data in advance.

Data farming [36] is process where data is grown in an experiment. It is a metaphor which means that is possible to generate your own data if no data exists [42]. In some of the use cases, a data farming approach has been used [36]. Furthermore, Edge devices can analyze the data real-time. The latest analysis initially included MFCC-coefficients computed from spectrogram, but it turned out that Mel bucketing [33], i.e., grouping frequencies similarly to how humans hear, may not work since machines use different frequencies than humans.

To record data in an industrial-grade experiment, the setting is very important. In this particular domain where we handle real-world machines, a key point for failure is the experimentation sequence. In the case of audio data, the quality of incoming data is, naturally, often distracted. There are many sources of distraction. For instance, another machine may disrupt the recording. Moreover, human factors may produce problems with sound.

There are many solutions for improving the quality of incoming audio data. The first option is to design the experiment with rigor. The experimentation setup is essential. In the case of, for instance, A/B testing, there are many potential problems related to audio data setup. In one of the experiments in the industrial-grade setting, a very basic mistake took place. The problem was with the heating of the machine which is demonstrated in the following.

Method 1: AAAAA BBBBB

Method 2: A B A B A B A B

In the industrial-grade A/B testing context we had two options where A=machine is ok and B=broken machine (for instance, ventilation of the machine was shut down on purpose). Then, we recorded the sound of the device with method 1 depicted in the sequence above. The results of the experiment were very promising since the model had a very high accuracy when we repeated option A five times and then option B five times in a row. Unfortunately, the model accuracy in case A was due to the rising temperature of the machine. The correct way to design the experiment would have been to have method 2, i.e., ABABABA where the temperature of the machine behaves differently. In the beginning, the machine is cold. In the end, the machine is hot. Due to the approach in method 2, the experiment was based on accurate data collected with rigor.

If high quality data doesn't exist, then it must be collected. It is crucial to design the method of collection (i.e., the experiment setup) from a data-centric point of view while keeping the model use case in mind. For instance, if we were to record the sound of several identical industrial machines in order to build a classifier, it is tempting to capture long audio clips of each machine one after the other. However, for many industrial grade machines their operational sound can take a long time to stabilize (if it happens at all) as the machines warm up, which means that there is considerable drift in the distribution of the data recorded during such a period. Therefore, if we want to be able to classify the machines without having to run them for lengthy periods of time, then a proper experiment design would be to interleave the recordings by capturing shorter clips of each machine sequentially, multiple times, thus ensuring that the state of the machines stays constant.

Once we have designed our data collection procedure to ensure a stable distribution, we have to take steps to mitigate any potential sources of noise that could impact the data quality during recording. Continuing with our prior example, suppose that our machines run at very high cycles causing strong airflow near them. With improper setup, this airflow can introduce excess noise in our recordings which heavily deteriorates the quality of our data. There are, of course, certain standard metrics, such as clipping and dynamic range, we can measure during the recording to mitigate this, but often this is not enough. However, if we recognize the risk beforehand, we can overcome it by equipping our recording devices with proper windshields. Another typical risk in an industrial setting is a noisy environment. For example, factories and warehouses are full of varying levels of unpredictable background noise. One way to alleviate its impact could be to record samples of the background noise and use methods, such as spectral subtraction or denoising variational autoencoders, to try to separate the machine sounds from the noise. We could also try to augment our collected data set by overlaying it with various samples of separately recorded background noise and hoping the model learns to distinguish the machines' sound from the noise. This, however, is costly, since more time must be spent both collecting data for augmentation and for validating that the augmentation and denoising is having the desired effect on our model. On the other hand, we found that a cost-effective pragmatic solution to this problem is to also employ a contact microphone (also known as a piezo microphone) in our data recording by attaching it to the machine. This automatically filters out background noise and requires no further effort from us after the data is collected, thus consistently ensuring high quality of incoming data.

Having obtained our data, we can start building our model. Since the amount of data is limited (since we have to painstakingly collect it on location in real-time), we cannot rely on deep neural networks to do the heavy lifting for us, but we have to put actual thought into the model design and to carefully determine which features we want our model to use. The standard method in literature is to use the mel-frequency cepstral coefficients (MFCCs)—which capture the harmonies and periodicities present in the short-term power spectrum of the input signal—as the main feature. However, they are sensitive to noise and, due to their scaling which mimics the human auditory system, do not respond well to machines creating high-pitch sounds.

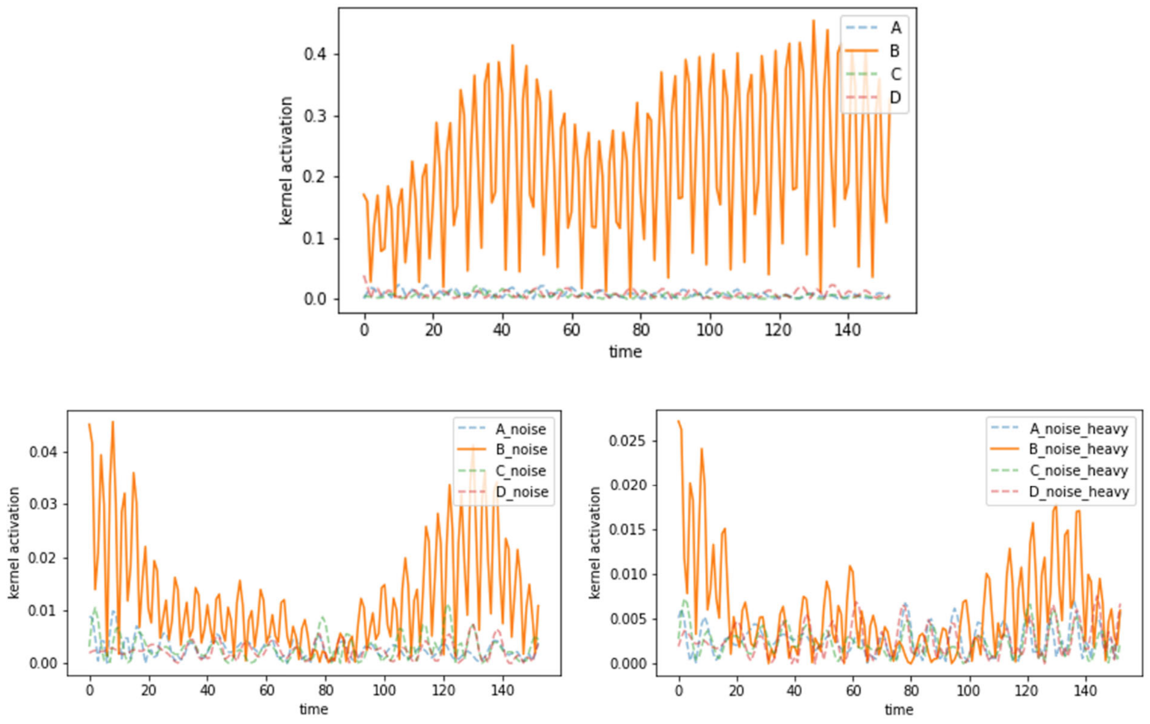


***Figure 4. Top: Activations of machine B's** fingerprint for the sound of four identical machines A, B, C and D. Bottom: activations for the sounds that have been augmented with moderate (resp. heavy) additive random noise on the left (resp. right).*

On the other hand, we found that by introducing a custom scaling tailor-made to the task at hand along with other representative features, such as periodicities of the log-spectral energy, we produce better-performing more robust models. For example, in our hypothetical setting of classifying machines, we built a system to create audio fingerprints for each machine based on representative kernels of the self-similarity matrix of their operational sound. By detecting the patterns unique to each machine, via kernel activations computed with sharpened cosine similarity, we were able to consistently identify the correct machine even when introducing heavy random noise to the input signal. In Figure 4, we show how the machine classification with audio fingerprint performed for four identical machines (A, B, C and D) with fuel combustion engines. At the top we show how the detector built for machine B reacts to the original audio recorded from each of the four machines. In the bottom we have introduced additive Gaussian noise to the input clips. Notice how the activation of the detector for machine B stays relatively small for machines A, C and D even with the added noise.

3.2.2.1. Future work related to audio data with IVVES partners

In IVVES consortium, we have an opportunity to collaborate with partner organizations in the industry. At the time of writing, we are looking for a partner to collaborate on research with Solita's artificial intelligence solutions. An EU-wide project provides a plethora of research organizations, for instance, to collaborate with.

For a future experiment, we need an industrial machine to research. From a business point of view, predictive maintenance is an important part in managing the life cycle cost of industrial equipment. If maintenance is done only, when necessary, cost savings may be available.

How to identify the need for machine maintenance? For example, how could the need to replace a wearing part be detected by sensors? The part to be replaced could be, for example, a drill bit or a grinding wheel.

In the experiment, we'll use microphones and radar as sensors since some of the partners have special expertise in these fields.

The plan is to collect audio and radar data in a test set-up in cooperation with a research organization. In case A of the test set-up, the machine is in perfect condition, but in case B there is some need for maintenance. In a half-day experiment, voice and radar data will be recorded by repeating cases A and B in a sequence (ABABABA). In our previous experiment, it has been shown that rotation is important instead of a serial sequence (AAABBB) so that, for example, the heating of the machine or material does not affect the quality of the data collected. Some of the collected A and B samples are set aside immediately at the end of the experiment so that the models being developed, such as machine learning models learning from data, are not aware of all the results in advance.

In addition to data collection with high quality microphones, we use microphones that already exist in the environment. For instance, one of the partners has medical devices which have a microphone. Moreover, every mobile device has a microphone. What kind of data quality does it produce?

In research, we need real-world problems. We hope to find a target to be researched during the upcoming weeks already. Initial technical solution of the experiment in real-world context real-time is depicted in the following:

First, a machine operates in an industrial context. In this case, we assume that the machine produces products I, L and O. The letters depict the physical form of the objects. Product O is round whereas products I and L are not round. In the real-world setting, there may be a lot of background noise, for instance, other machines or humans communicate with their voices. The hypothesis is that when the machine produces product O, the sound is different than when it produces product L and I.

The sound is captured with four sensors. First, an air microphone records the sound. Second, a contact microphone attached to the machine physically records sound. Third, a mobile device records the sounds. Finally, a radar captures a signal of the machine. With these four datasets we are able to recognize machine operations.

An intelligent Edge solution operating in the factory stores the sensor data. Features may be mined at this stage already. Edge computing is good from, for instance, information security point of view. The Intelligent Edge sends the data to multi-cloud. There is an API that receives the data and stores it to one of the clouds, for instance, AWS, Azure or GCP. The feedback loop back to the factory is, for instance, 6 seconds. When there is a problem recognized, the multi-cloud warns the domain experts working in the factory. With this setting, continuous learning is enabled.

3.2.3. Quality Assurance of Semi-Natural Language Data

Solution provider: Helsinki University

This research direction addresses similarity between commands users enter the terminal windows or which are automatically generated by other applications. The work is driven by the F-Secure needs to detect abnormally behaving users.

We aim to visualize the similarities and differences between two commands. We develop new ways to measure command similarity. For this we use NLP methods and NLP models. In addition to just focusing on the command text we use the text of the official documentations and manuals data to complement it. The work targets Windows, Mac and Linux commands. The commands in these environments have rather different structures and types of documentation reflecting the historical development of these systems. The processing of command lines is a mixture of formal and natural language processing. We expect the work to create results useful for future command line syntaxes and for their documentation. Since various command lines are underlying today's graphical user interfaces, we expect our basic technologies to also have other uses besides F-Secure's immediate needs.

Semi-natural languages, such as markup languages, algorithms, command-line commands, and processes are hard to analyze in the absence of a ground truth and without any syntactic and semantic knowledge.

To add the extra layer of knowledge, expert opinions can be useful in creating a set of rules. These rules act as grammar, add syntactic and semantic meanings, and help in understanding the sentence (commands) structures just like natural languages. In general, all the possible analyses can be performed using a set of rules, but a rule-based system is not efficient when the data size is increasing exponentially and insights from data are needed continuously. Maintaining a rule-based database is also an exhaustive and expensive task. To solve these problems, we studied a hybrid approach of a rule-based system with a machine learning system. This approach is useful in finding similar commands, understanding the hierarchy and dependency of the prevalent flags and parameters, and learning the structure of the commands.

The approach of combining a rule-based system with machine learning is not common because machine learning has the capability to solve the problems related to quantitative data, image data, and natural languages solely. Since our data is semi-natural and unlabeled, applying machine learning exclusively is not a feasible solution. With this hybrid approach, we managed to solve this problem, and this approach can also be useful for other use cases where data is of semi-nature and expert opinions are needed. For example, a lot of research has already been done to detect code clones, but this hybrid approach has not been explored before. To detect code clones or to analyze code structures, chunks of codes can be clustered. Following the proposed approach, experts can help in labeling the pieces of codes as function, declarative statement, conditional statement etc. and then using a carefully created set of rules, codes similarities can be detected. Another possible use for this approach can be in the conversion of algorithms to code, where parts of an algorithms can be labeled, such as instruction, variable, condition, loop etc. Then applying set of rules created by the domain experts it is possible to detect similar algorithms. This can be helpful in evaluating how structurally similar codes can be generated against the similar algorithms of a cluster.

Though this approach is proven beneficial in certain cases as discussed earlier but it needs a lot of human effort in the beginning. The customized labeling of tokens also requires extensive domain knowledge and adequate time. Experts need to be careful while creating a set of rules, as they can keep increasing with the complexity of the data. Besides all these limitations, this hybrid approach is a promising solution to a problem which involves complex and semi-natural data, and where the experts' opinions are required to understand the sensitivity of the domain.

For the future work, we plan to cluster the commands and upon receiving a new command would map it to any existing cluster. This will help us in analyzing the structure of the commands by calculating inter-class and intra-class similarities. If a new command does not map with any of the existing clusters, a new cluster will be started if the command is safe, otherwise the new command will be part of risky cluster of commands.

Finally, a new research direction was found related to a collaboration in the topic of DQW with F-Secure due to the text data support in Sogeti’s tool: ensuring quality of semi-natural language data used in in the semi-natural language analysis model. The collaboration and validation of the DQW text analysis, with a special focus on topic analysis with LDA has been limited to the first part of the pipeline – data scraping of the text of the official documentation.

3.2.4. Data Quality for text driven ESG investment systems

Solution provider: Concatel/Netcheck

To solve the problem of data quality in complex ESG investment systems, SII CONCATEL & NETCHECK have designed tools to evaluate the quality of the data, considering that the data handled will be obtained from different sources [43]. The evaluation of the quality of this data will allow us to determine if this data is suitable to be used in the procedures followed to train the Machine Learning models implemented in the Fintech domain.

For a better understanding and contextualization of this concept, the concept of Fintech will be briefly explained. The term Fintech comes from the English words Finance and Technology. This term refers to all types of companies whose functions include activities involving innovation and new technological developments in the design, supply or provision of financial services or products.

Fintech companies make it possible to offer consumers much more efficient management of their personal finances and a wide range of possibilities for comparing different financial products. They provide information on the status of financial products and services, hence the importance of being able to control and provide the highest quality of data handled in this sector.

Different aspects will be considered to treat this information since data handling is very sensitive and all this procedure must be always in line with the GDPR (General Data Protection Regulation) law, which is a regulation within the EU framework that controls the way in which companies or organizations use personal data [44]. To this end, the tools developed will use techniques to ensure that all ML (Machine Learning) components comply with these directives, certifying that the data is of high quality and that robustness and security can be guaranteed.

One of the added values that Fintech bring to the sector is that they generate added value to the market by creating large volumes of data (Big Data), which allows the use of various artificial intelligence techniques on them [44].

Airflow is one of the tools that will be used to process this large volume of data, as this tool is one of the most powerful in terms of workflow automation and allows processing and managing large workloads dynamically. Airflow is a so-called "workflow manager" type of tool, i.e., it allows us to manage, monitor and plan workflows, being used as a service orchestrator. A service orchestrator acts as a dedicated component for organizing systems and integrating services from different sources.

Other tools such as MLflow and Kafka will be used to maintain the quality of the data so that it can persist over time through the different machine learning (ML) mechanisms. They are briefly described below so that their function or performance can be understood.

MLflow is an open-source platform for managing the entire machine learning lifecycle. MLflow allows for comprehensive logging and tracking of all training execution metrics and artifacts of the model being trained with the data obtained.

Kafka, on the other hand, is a messaging system and a complete real-time data processing and streaming platform. Being able to handle real-time data provides great benefits such as the ability to publish and process data streams in a scalable and fault-tolerant manner. Kafka also allows data to be subscribed to so that it can be obtained immediately from multiple sources and this data can be distributed.

Additionally, another objective of this task is to develop methods, techniques and tools that address the quality issues resulting from deviations and biases in the training data that would eventually result in poor performance of the trained ML algorithm. The methods and techniques devised in this task will be able to uncover hidden patterns and correlations as well as biases in the training data. In addition, coverage criteria and synthesis techniques will be devised from training data.

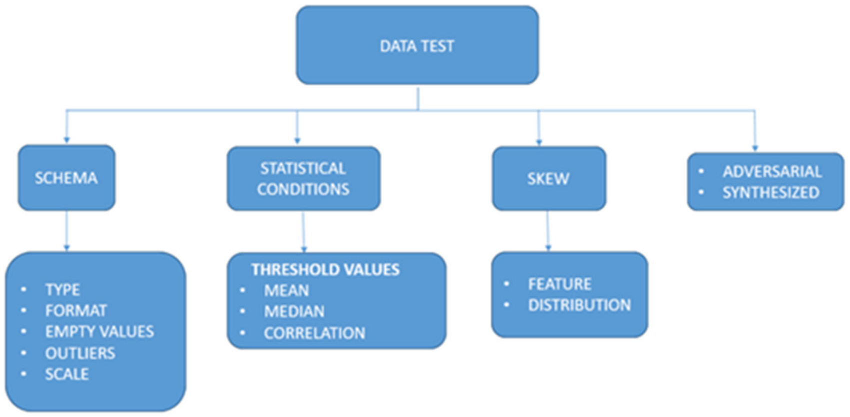


Figure 5. Testing dataset

The Data Validator [43] compares the data properties with a previously generated schema. The schema may have data types, data formats and any other conditions on the characteristics, such as "non-null" or "non-duplicate", etc. This component produces a reference schema, which attempts to capture meaningful properties of the data. With each new data ingest, the Data Validator recommends updates to the initial schema.

This system supports data validation for three different cases. Single batch validation checks for anomalies in a single batch of data. Inter-batch validation evaluates training data against service data and evaluates data from multiple batches. In the third component, the system checks if there are any assumptions in the training code that are not represented in the data. The Data Analyzer calculates a predefined set of statistical properties such as mean, median, etc. The data is validated against these properties based on some threshold values. The Model Unit Tester detects errors in the training code using synthetic data. This synthetic data is generated based on the schema and statistical properties.

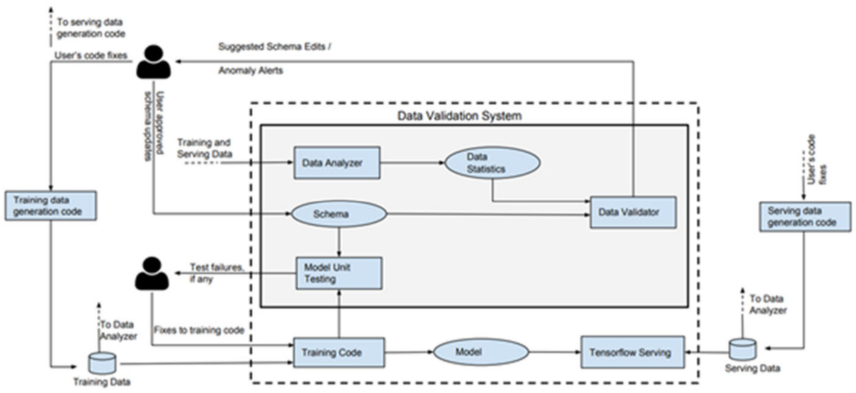
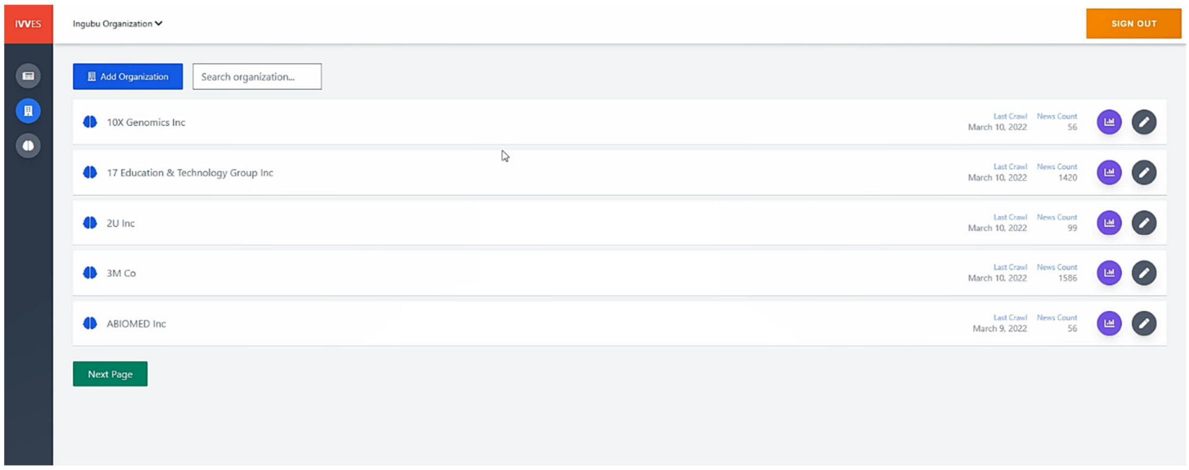
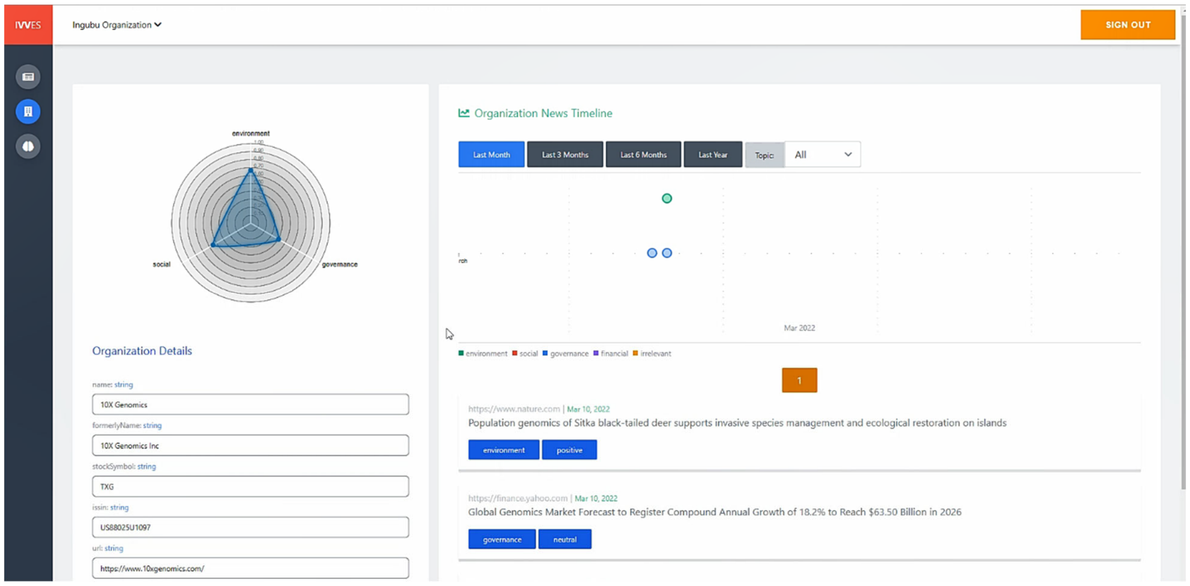


Figure 6. System Analysis of Data Quality.



	Last Crawl	News Count
10X Genomics Inc	March 10, 2022	56
17 Education & Technology Group Inc	March 10, 2022	1420
2U Inc	March 10, 2022	99
3M Co	March 10, 2022	1586
ABIOMED Inc	March 9, 2022	56

Figure 7. Demo of UI CCTL/NTCHK.



Organization Details

name: string
10X Genomics

formerlyName: string
10X Genomics Inc

stockSymbol: string
TXG

issin: string
US88025U1097

url: string
https://www.10xgenomics.com/

https://www.nature.com | Mar 10, 2022
Population genomics of Sitka black-tailed deer supports invasive species management and ecological restoration on islands
environment positive

https://finance.yahoo.com | Mar 10, 2022
Global Genomics Market Forecast to Register Compound Annual Growth of 18.2% to Reach $63.50 Billion in 2026
governance neutral

Figure 8. Demo of UI CCTL/NTCHK.

4. Task 2.1 - Model Quality

Task Lead: Techila Technologies

In the following sections, we discuss the metrics, approaches and methods used to ensure machine learning model quality. Explainable AI approaches used to provide transparency in the ESG field describe novel approaches such as mutation validation for avoiding the pitfalls of typical cross validation and presents a context-aware outlier and novelty detection approach that consider *what* is to be predicted.

Performance of machine learning models is discussed in 5.3 with the focus of comparing the inference scalability with different network protocols and on the benefits of knowledge distillation, which is a novel approach for addressing model scalability while maintaining predictive performance. Drift detection (4.4) expands the MLOps aspects and describes how a monitoring approach was designed, implemented, and deployed to track the performance of a host behavior classification system. This system uses a machine learning classification model to predict as many classes as there are types of host behavior to identify.

4.1. Explainable AI Toolkit

Solution provider: VTT

VTT realizes that developing trustworthy machine learning (ML) models requires tackling a multifaceted complex challenge which remains to be unsolved. We approach the topic by discussing the challenge from three viewpoints and introduce, some, supporting methods accompanied with their implementations. To be noted, some of the background work is still unpublished but the presented work is tempting enough to be discussed here.

1. How to ensure that the utilized data for fitting a model is of high quality given the task to be modelled? That is, the data is not analyzed in isolation but instead analyzed from a viewpoint if it

facilitates predicting what is to be predicted. Detecting data corruption is of particular interest and how to detect it automatically. The methods are based on [45] and [46].

2. Mutation Validation (MV) discussed in yet to be published paper [47] suggests complementing the traditional cross-validation (CV) to tackle some of the caveats of CV. The claim is that MV can better catch when the model is under-fitted and when the models start to be over-fitted. MV is also claimed to able catching when the model seems to be over complex for the task therefore there exists a simpler model with similar performance. Thus, whereas 1) tackles the data quality, 2) strives to enable developing models which are complex enough for the task, but not too complex.

3. In real-world applications, it is not always guaranteed that there would not be, even drastic, covariate shift, or any other types of shifts, in the data. Covariate shift causes performance degradation as the model is forced to extrapolate. When extrapolating, epistemic uncertainty of the model will be elevated but unfortunately, not typically expressed in any way. Here, we present a simple wrapper method to incorporate a proxy metric for epistemic uncertainty for a random forest model, to LightGBM[1] due to performance reasons. The proxy metric can capture when the model is forced to operate in a feature-space which is off the training-data.

Having a quantifiable metric for data quality, particularly a quantifiable metric for each training data sample, facilitates developing more robust models as removing samples of bad quality have a negative impact on the model performance. Additionally, if a sample is of no value, then such a sample can be removed altogether. In addition, detecting which samples are of good or bad quality enables further analysis of why the case is as such.

After filtering samples out which harm the model, MV, aside to CV, enables developing models which are complex enough, but not too complex, for the task at hand. Avoiding developing unnecessarily complex models facilitates having a model which is less brittle, and perhaps less vulnerable for adversarial attacks.

In the end, even though the model would be fitted with high quality data and it would be of just the right complexity, the model must be monitored when deployed. In mission critical applications, it is necessary to monitor if the model is forced to extrapolate. The discussed method does just that, sample by sample, therefore giving a chance for e.g., human operator to overwrite the predictions or to temporarily replace the model with a simpler, less accurate but more robust model.

4.1.1. Training data quality

Training data QA, in this context, refers to assessing the quality of the training data with respect to the task to be modelled. Evaluating the quality of the data in isolation is excluded but the quality of data when utilized for fitting a specific model to predict a specific target is of interest. Thus, the quality is assessed given the dependent variable and the model accompanied with its hyper-parameters.

A model-agnostic approach based on Shapley values is discussed, which claims to be more robust than Leave-One-Out (LOO) in assessing the usefulness of the samples. In LOO, the sample usefulness is assessed by fitting a model with and without a sample of interest and calculating the performance difference between these two models. If the model performs better with the sample, then the sample is deemed to be useful. In case the performance degrades, the sample is considered harmful.

The method discussed in [45] is somewhat like LOO except it draws inspiration from game theory where the goal is to distribute the usefulness of the samples fairly. The difference in essence is that in Data Shapley all permutations of the samples are evaluated when calculating the usefulness. Calculating the usefulness of each sample is a computation heavy procedure. Whereas in LOO, one is required to fit and evaluate the model N + 1 times where N is the number of samples, in Data Shapley the number of times

[1] https://lightgbm.readthedocs.io/en/latest/

the model is to be fitted and evaluated grows exponentially to the number of samples thus in practice rendering the vanilla method intractable even for toy cases. Fortunately, permutation truncation means, and Monte-Carlo sampling of permutations discussed in the paper makes the method tractable for more than toy-cases. The algorithm is also trivially parallelizable and distributable therefore enabling computing slightly more samples in reasonable time depending on the utilized model. Nevertheless, the algorithm can be considered very heavy in all cases.

In [46], a fast K-Nearest Neighbour (KNN) specific version of Data Shapley is discussed which enables computing the exact Data Shapley values of the samples in O(NlogN) thus making is feasible for larger datasets too than the previous version of the algorithm. Additionally, at least Data Shapley for classification is trivial to compute in parallel or taking advantage of distributed computing. For larger dataset, Locality Sensitive Hashing (LSH) method is also discussed which improves performance.

Implementation

In IVVES, implementations for [45] and [46], for both regression and classification, were implemented. For [45], a parallel processing enabled implementation was developed as a single-core implementation was considered prohibitively slow. However, even when utilizing multiple CPU cores, [45] does not scale beyond much above toy-cases. Thus, [46] was implemented where for both classification and regression parallel processing capabilities were implemented as the algorithm is straightforward to make parallel along the test-set. For larger datasets, the current implementation is not feasible therefore a proper distributed computing enabled implementation must be developed. Nevertheless, both implementations based on [46] are multitudes faster than [45] although suffering from inherent restriction of using KNN as an internal model.

Experiments

In this experiment, classification and regression based on [46] are experimented with in detecting if the algorithm can detect samples which have corrupted targets, i.e., the dependent variable. It is experimented, similarly to in [45], how the performance of the model improves when removing the worst samples by Data Shapley value. For binary classification, Wisconsin Breast Cancer[2] dataset consisting of 569 samples with 30 features was utilized whereas for regression, California Housing dataset consisting of 20640 samples with 8 features was used. The train/val/test splits for both datasets are 50/25/25. For classification 25% of the training-set targets are corrupted, i.e., the label is flipped. In the regression case, 25% of the targets are corrupted by zero-mean two training-set target standard deviation Gaussian noise.

As [46] is based on KNN which is not exactly a very powerful model, additional means are taken advantage of before calculating the Data Shapley values. First, LightGBM random forest model is fitted with the training data to get the feature importance values. In this case, we utilize mean absolute Shap values of the features instead of vanilla tree-based feature importance scores. Second, the features are scaled between 0 and 1. Third, the scaled features are multiplied by the Shap value-based feature importance to emphasize those features which on average are relevant for the prediction task. This step sometimes makes sense as KNN is based on simple distance metrics and if the not useful features dominate in scale, KNN model does not necessarily perform very well.

In Figure 9 a histogram of non-corrupted and corrupted samples in Breast Cancer dataset is depicted. X-axis depicts Data Shapley values where positive values stand for samples with positive impact on the model performance and negative the other way around. As depicted, non-corrupted samples tend to have a positive Data Shapley value as expected and the main body of the corrupted samples have negative values. This is what one could expect.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

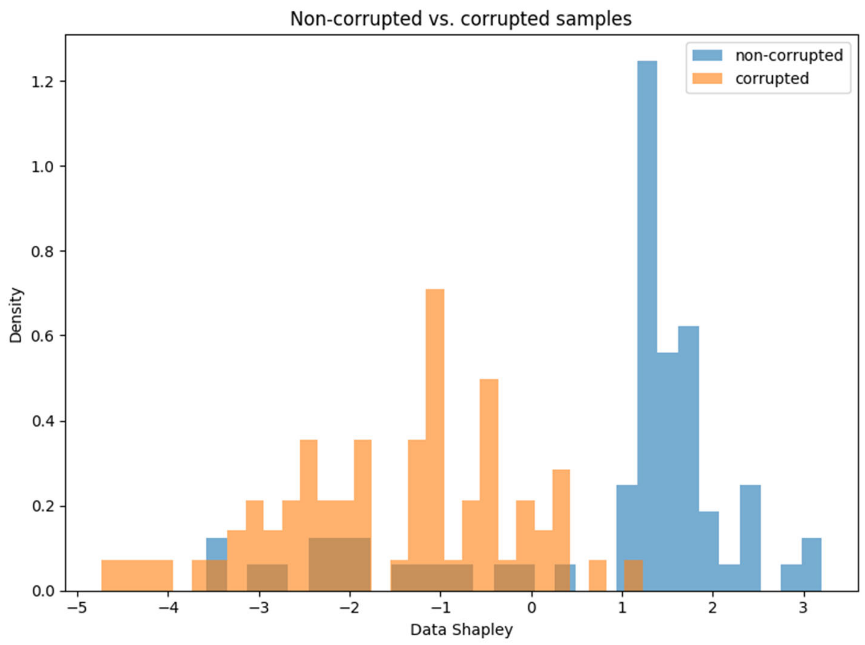


Figure 9. Corrupted and non-corrupted samples in Breast Cancer dataset.

In Figure 10, the impact of removing worse samples by Data Shapley values is depicted. The samples are sorted by Data Shapley value, then the samples are gradually removed from the training-set and model performance is then evaluated again. As depicted, removing circa 30% of the samples results in perfect training-set performance. However, high training set performance is not the goal but high test-set performance instead. Validation-set performance is not directly utilized in the model training phase, but Data Shapley values were computed against it therefore essentially making the validation set a part of the training-set as well. As depicted, test-set performance (ROC AUC) improves till circa 30% of the samples are removed from the training-set. This is again, as expected.

In the Breast Cancer dataset, Data Shapley seems to behave exactly as expected. Removing samples with low or negative Data Shapley values have a positive impact on the model performance in out-of-sample data.

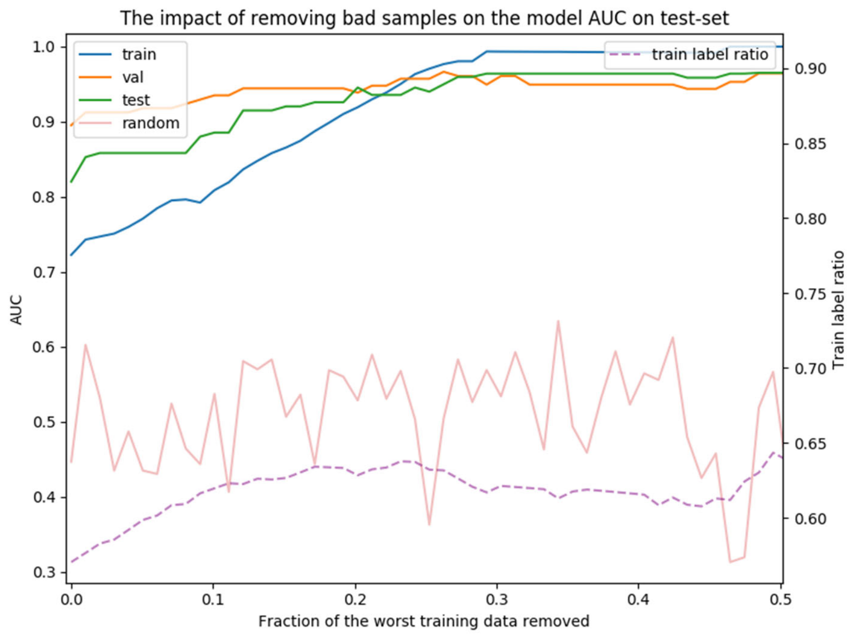


Figure 10. The impact of removing worst samples in Breast Cancer dataset.

In the regression case, similar behavior can be observed as depicted in Figure 11. As the noise is sampled from a Gaussian distribution, plenty of the additional noise has not much impact on the target variable which can also be observed. The non-corrupted and corrupted samples overlap but corrupted samples have significantly fatter left tail indicating that there are plenty of samples which hurt the performance.

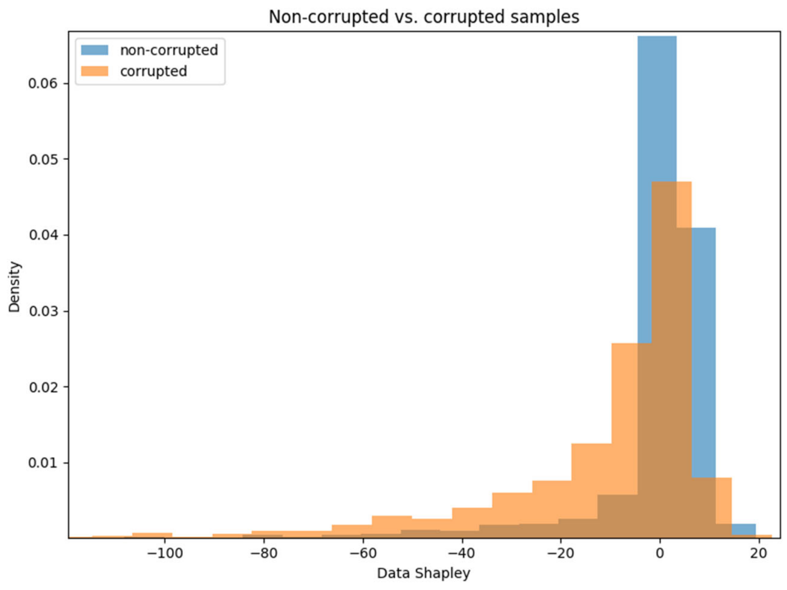


Figure 11. Corrupted and non-corrupted samples in California Housing dataset.

Similarly, as it is in the Breast Cancer experiment, removing worst samples by Data Shapley value from the training-set has a positive impact on the model performance as depicted in Figure 12. As can be observed mean squared error (MSE) of test-set gets lower till circa 10% of the worst samples are removed which also reflects the histogram above. Clearly, removing samples with worst Data Shapley values has a positive impact on the model performance.

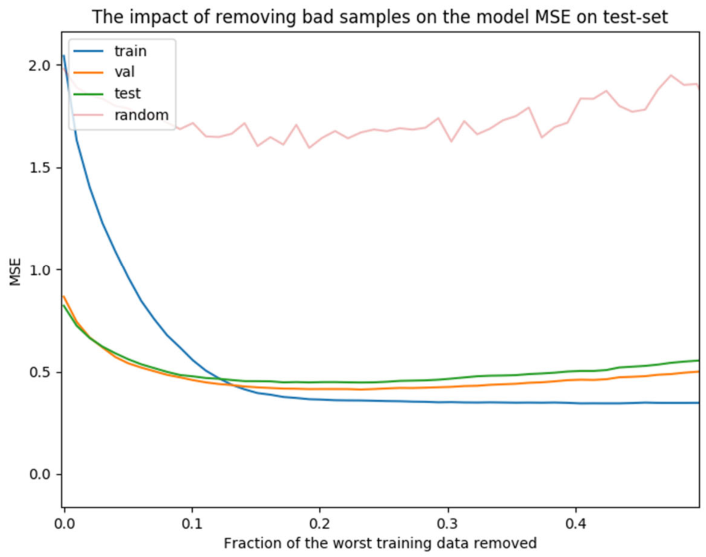


Figure 12. The impact of removing the worst samples in California Housing dataset.

4.1.2. Model Quality Assurance

Industry best practice for model QA is to conduct cross-validation (CV). However, as briefly discussed in [47], out-of-sample validation has the following limitations:

1. Test-set can be too small to represent the real out of sample distributions.
2. The model performance has a large variance across different runs.
3. The test-set has exactly the same distributions as the train-set therefore leading to inflated test-score.
4. Overfitting to the test-set.

Mutation Validation (MV) is a new unpublished method which strives to avoid the pitfalls of typical cross validation by using only the training-set to validate the model. MV is based on an idea of mutation testing and metamorphic testing where both combined is argued to lead to a method which is able to capture when a model is under-fitted and when over-fitted. In essence, MV is argued to indicate when the model is too simple or unnecessarily complex.

Implementation

Implementation for binary classification is as discussed in [47]. In the case of multi-class classification, label shifting is replaced by randomly picking a different label for the mutated samples. Regressions were not discussed in the original paper, but an experimental version of it is implemented but further discussion omitted here.

Experiments

The Wisconsin Breast Cancer dataset is utilized as an experiment. As a model LightGBM was utilized. The goal is to find maximum number of leaves for a random forest.

Figure 13 depicts the result of both 10-fold CV and MV. As depicted, the maximum and stable ROC AUC is reached at 18 *n_leaves* after which the performance does not improve. Considering MV, first, MV starts to climb higher while reaching the peak at circa 10 *n_leaves* followed by declining MV score. This curve is as discussed in the paper. That is, first the model is underfitting, i.e., it is too simple, then it hits the sweet spot at 10 *n_leaves*, and then the model starts to get unnecessarily complex. The model does not necessarily overfit as CV remains stable but the model is unnecessarily complex. That is, adding more leaves does not improve the CV but the model begins to have capability to learn mere noise which is unwanted.

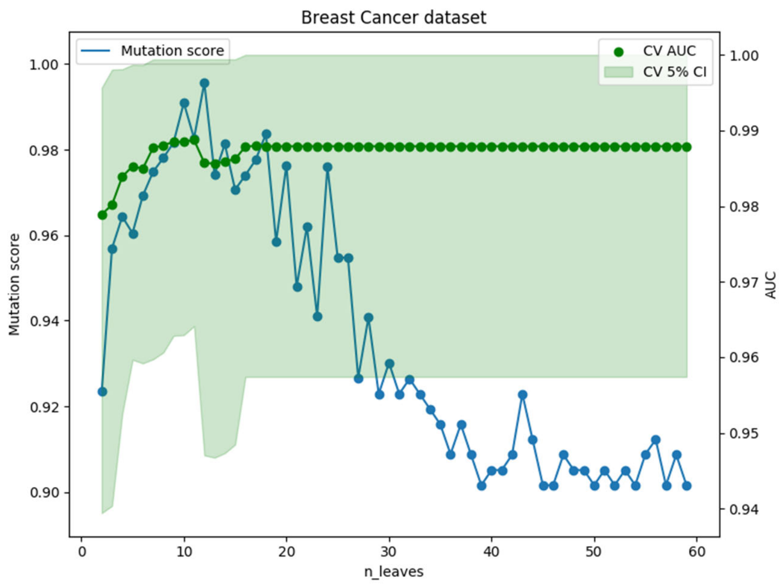


Figure 13. Mutation Validation and 10-fold cross-validation in Breast Cancer dataset.

4.1.3. Prediction Quality Assurance

In classification, in the case of models which are capable of providing prediction probabilities, probabilities are sometimes considered indicating confidence. In essence, probabilities, excluding Bayesian realm, indicate the frequency of similar samples being classified as predicted. Now, this behaves correctly if, and only if, the probabilities are well calibrated, but this fails when the samples fall out of the training-data distributions. That is, the model is forced to extrapolate. In the regression case, confidence intervals are typically utilized to manifest confidence but unfortunately not all machine learning models are capable of providing such.

Outlier and novelty detection are typical means to detect drift in the covariates. However, the vanilla outlier and novelty detection means do not take into account the task which is being modelled. The outliers are evaluated against other training data samples without knowing which features are even meaningful for the prediction task. Thus, the prediction task must also be taken into account to detect when the model is forced to extrapolate or is otherwise forced to operate in non-optimal conditions.

Here, in this section, we briefly discuss one attempt in providing an uncertainty metric for random forest classifier and regression which is capable of providing uncertainty score for each sample. Before moving into the specifics, trivial synthetic data which is easy to visualize and comprehend is introduced for classification and regression.

In Figure 14, four 2D moons are depicted where dots colored in red are of label 1 and the blue ones 0. The dots are utilized as training data in this binary classification experiment. As a model, LightGBM extremely randomized tree model is utilized, whose decision surface is as depicted.

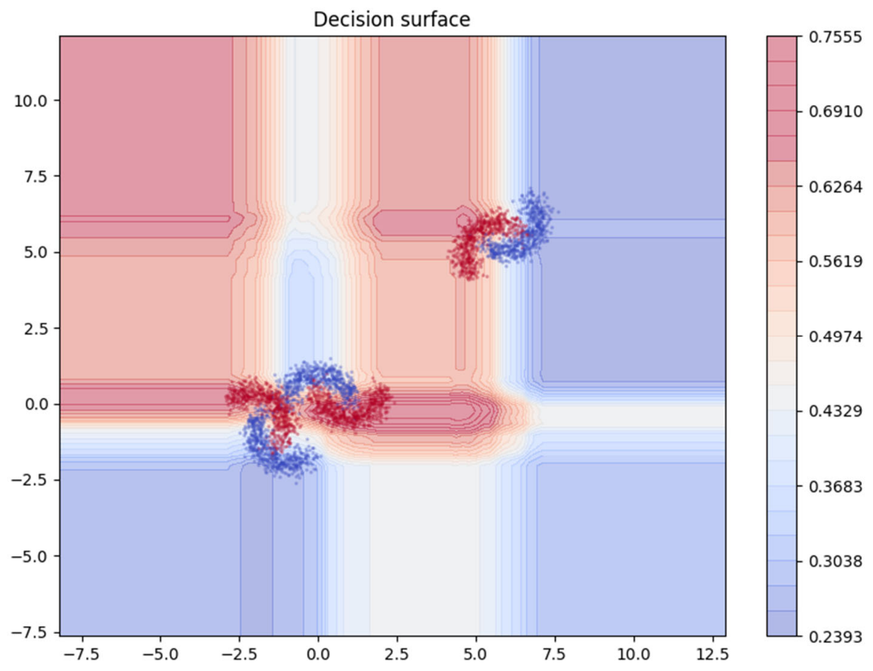


Figure 14. Decision surface of a binary classifier.

As can be observed, the decision surface is as what is typical of tree-based models. The model completely fails in providing any sense of uncertainty when extrapolating as depicted. The model is capable of very accurately predicting the samples but when moving away from the training-data space, the model repeats the predictions. This is typical of tree-based models, and many others (excluded from further discussion) where e.g., advanced piece-wise linear regression is not utilized.

As random forest is an ensemble of many trees, one can also study how the trees independently predict, collate the independent predictions and calculate the spread of the predictions to come up how well the trees agree as depicted in Figure 15. The purpose of such an experiment is to effectively get an impression of the *aleatoric uncertainty*, i.e., knowing when the model knows when it does not know. If the spread is *high*, the trees in the forest disagree therefore there must be something in the data which causes such disagreement.

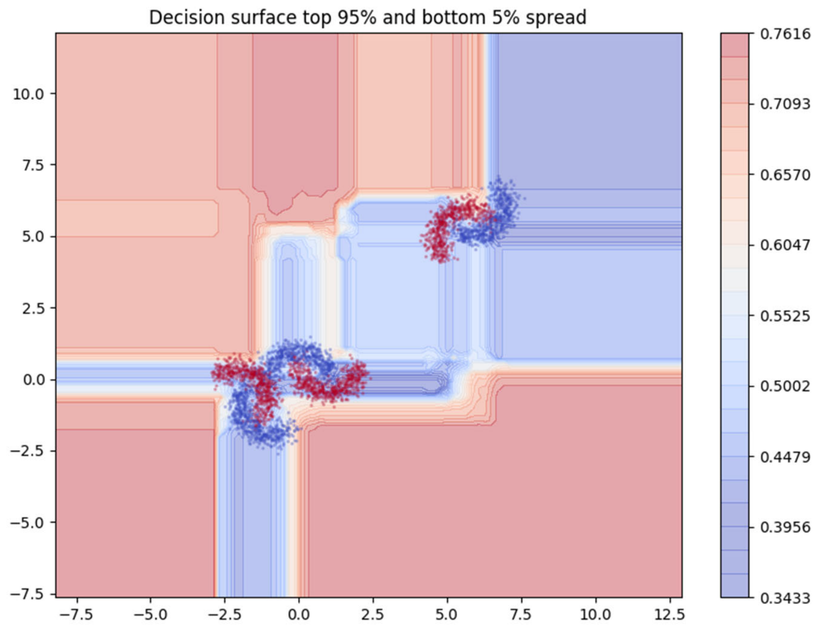


Figure 15. Decision surface spread.

As depicted in Figure 15, the trees in the forest disagree in some areas of the decision surface whereas e.g., between the clusters of the *moons*, the disagreement remains low. The disagreement is high in the class boundaries in the decision surface and then such a disagreement is extrapolated as it is. This can hardly be considered optimal. Observing only the disagreement of the trees cannot be considered as a good metric of uncertainty, especially in the extrapolation case.

In the case of regression, the same is true as illustrated in Figure 16. In Figure 16, a noisy sine wave is to be predicted from X. 20% of the sine wave is removed from the middle and 20% is removed from the end. Now, as depicted in Figure 16 and more explicitly in Figure 17, the prediction spread is high in the mid-section mainly due to how the wave behaves just before and after a part of it is deleted. This is typical of tree-based models and how trees function in general. If the wave had been cut differently, i.e., at the same level, the prediction spread would be significantly smaller. This phenomenon is visible at the end of the wave. The model keeps predicting the last encountered value and also maintains the same spread of the predictions. This clearly is unwanted behavior as one would like to see widening spread in the predictions due to the elevated uncertainty. In this case, the confidence interval is abnormally low at the end which is the opposite of what one would like to have.

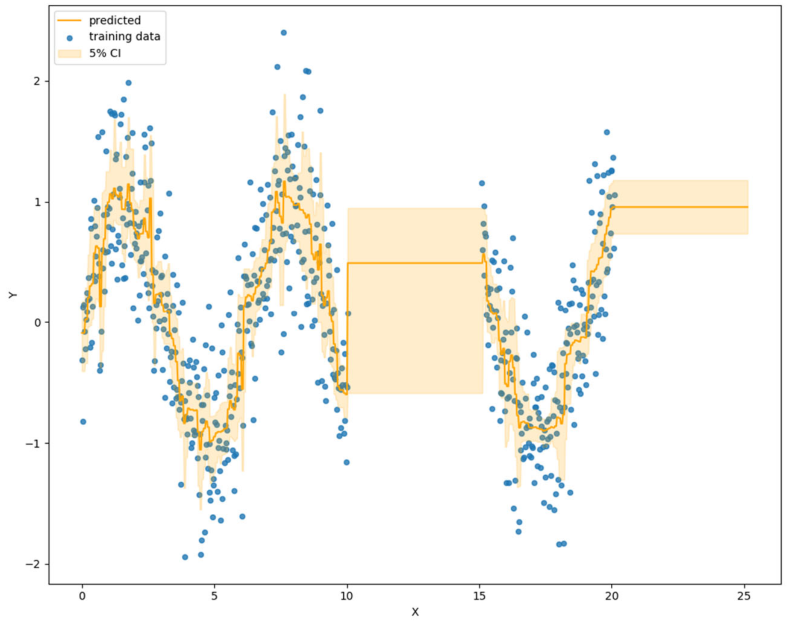


Figure 16. Regression with confidence intervals.

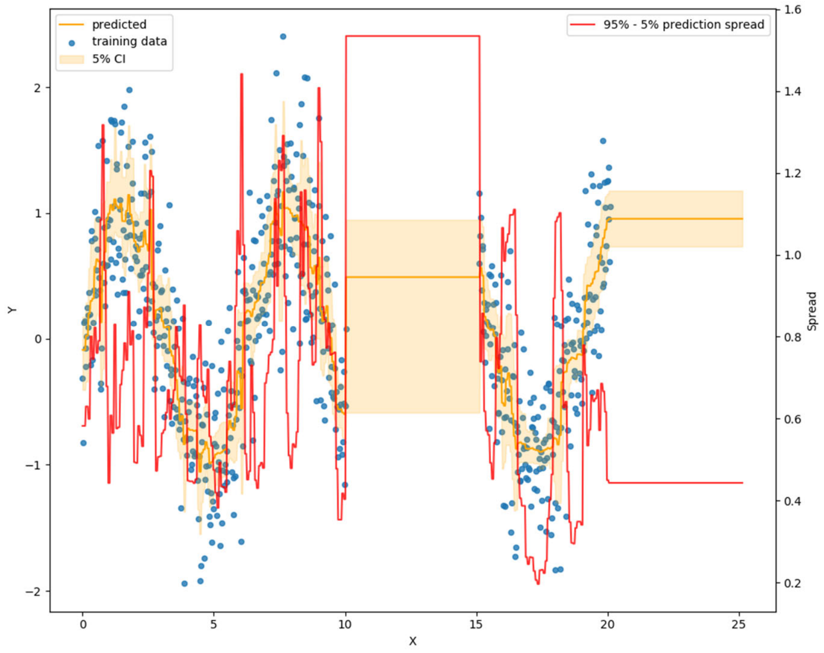


Figure 17. Regression with CI and prediction spread.

Implementation

LightGBM is a state-of-the-art algorithm very similar to its more known counterpart, XGBoost[3], but is significantly faster but tends to have the same performance, or better. LightGBM is utilized as the basis of

[3] https://xgboost.readthedocs.io/en/stable/

the implementation even although any tree-based algorithm is applicable. We start with a pre-fitted model. Thus, we take the model as given.

To model uncertainty given the prediction task, leaves of the trees are of interest. Each sample falls into one leaf in each tree even though a sample would be very far from the training-set distributions. This is as trees are in essence just a set of if-then statements which can be computed no matter how peculiar a sample would look like. Nevertheless, each sample is, in essence, clustered together with their closest peers conditioned by the prediction task. We will take advantage of this.

The idea is to fit a novelty model for each leaf in all trees therefore there will be a significant amount of separate novelty models for one LightGBM model. To be more specific there will be *n_leaves* * *n_trees* amount of novelty models. Each novelty model is fitted with samples which fell into a specific leaf and only the features which were utilized to land on such a leaf will be considered. This is conducted as it is wanted to compute the novelty score by considering the same features the tree itself utilized for *clustering* the samples. As a novelty model One Class SCM is currently utilized with Radial Basis Function kernel.

As fitting many separate novelty models is computation intensive process, the work is computed in parallel to significantly shorten the computation time. The algorithm itself is trivially distributable similarly to it is parallelizable, but now only single CPU solution is currently implemented.

Experiments

Let's use the same data as above but now visualize the mean novelty of the samples. In Figure 18, areas with positive values (red) are considered inliers and negative (blue) are considered outliers. In Figure 19, the same is depicted but as binary colored to facilitate intelligibility. As can be seen, the training samples, i.e., the dots, are considered inliers, and areas of extrapolation as outliers as expected. Now, when predicting, when a sample is considered an outlier, one should seriously consider should one trust the prediction in the first place.

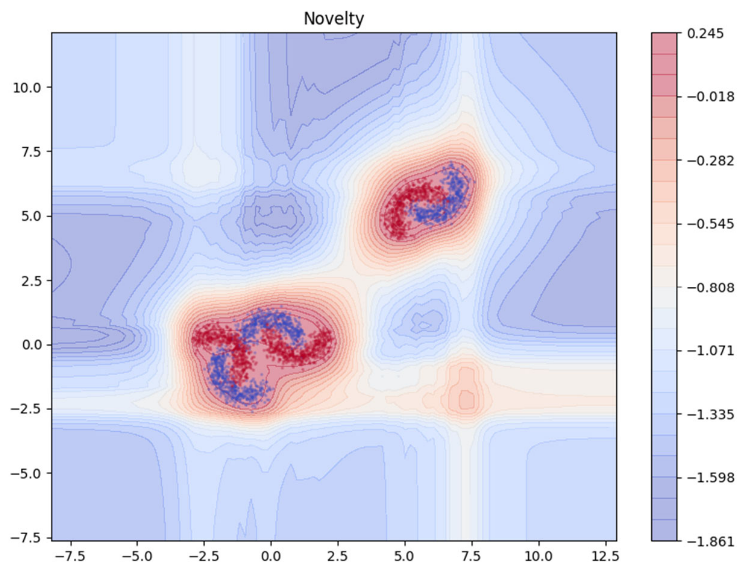


Figure 18. Novelty surface of the classifier.

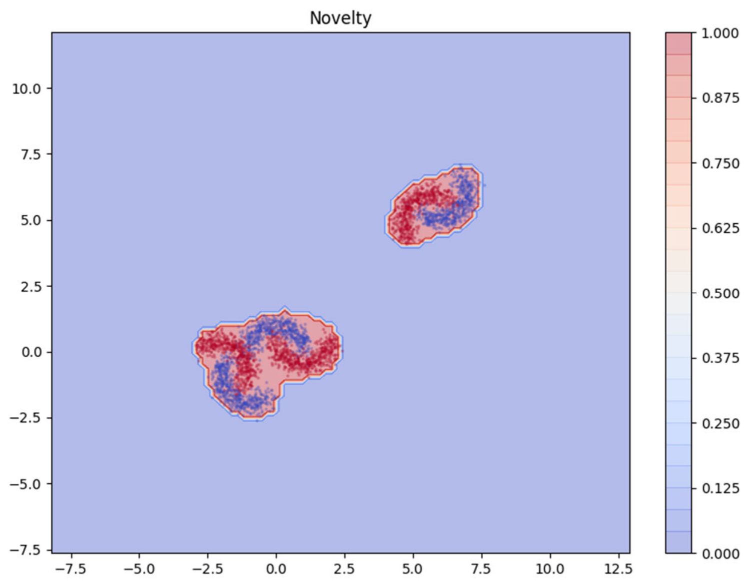


Figure 19. Novelty inliers and outliers of the classifier.

The same behavior is illustrated in Figure 20 for the regression case. As discussed above, the confidence intervals of the predictions are not necessarily to be trusted but the novelty scores might assist.

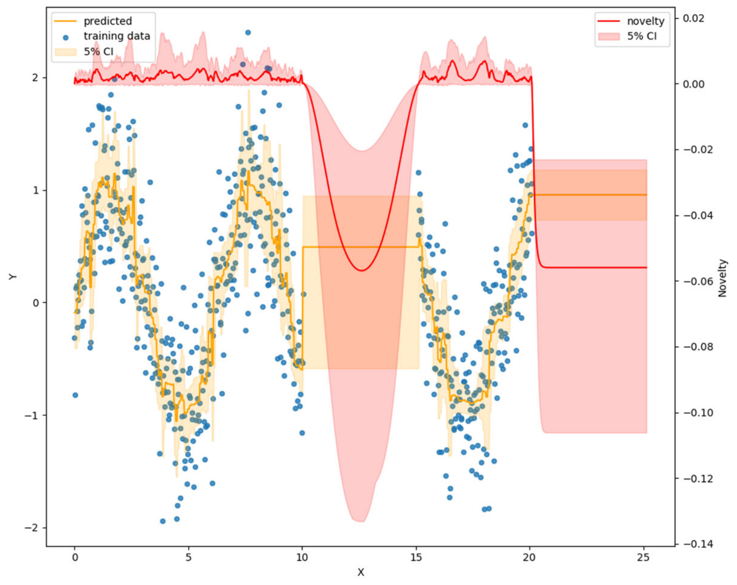


Figure 20. Regression novelty.

As depicted, when residing in the known space, the novelty tends to be positive meaning that the samples are considered inliers. The mid-point area where the sine wave has been cut was detected also by inspecting the prediction spreads but only due to mere luck, in essence. That is, if the wave had been cut in a way where the end and starting point of the cut would be at the same level, the prediction spread would have been smaller. Here, the novelty clearly indicates with negative values that the samples are off the training-set distributions. It is the same with the removed end of the wave. When prediction confidence intervals are not to be trusted, novelty clearly indicates that the model is forced to extrapolate.

Although both presented cases are synthetic and very trivial, and perhaps the extrapolation would have easily been expressed by utilizing any widely available prediction model independent means, that is not the case in non-trivial datasets. The outlier or novelty detection algorithms which do not consider what is to be predicted inspect only the raw data without knowing how the downstream prediction model *sees* the data. This is not the case in the presented algorithm. The presented algorithm *knows* exactly how the model has landed on its predictions therefore the novelty is computed in that context

4.2. Model Quality for text driven industrial environments

Solution provider: Keyland

Artificial intelligence, predictive machine learning in the context of this project, is often presented as an opaque method, which takes data and generates some values. Between these two phases, there is a black box that solves the mapping, the equations. This is partially true, especially when using state-of-the-art nonlinear models [49], which can be notoriously difficult to interpret when compared to more traditional linear solutions.
Research has approached the challenge of explainability from different angles, as each application domain has its own needs and objectives. In addition, different audiences require different types of explanation. For example, some of the standard terms used in the literature to refer to XAI objectives are reliability, fairness, causality, transparency, and justification [50]. All of these indicate the importance of explainability in practice. Developers can use explainability not only to understand the black box model, but also to debug the system by detecting unexpected model behavior or any bias in the training data set or model. Therefore, AI explainability can also facilitate model improvement and the recovery of new hidden features.
In Machine Learning, some models are interpretable models by design and can have different levels of transparency. Such as algorithmic transparency, transparency based on simulation capability and transparency based on decomposition capability. This category includes linear/logistic regression, decision trees, K-nearest neighbors, rule-based learning, general additive models, and Bayesian models [51]. On the other hand, there are algorithms such as Artificial Neural Networks (ANN), which are cumbersome to analyze and interpret as they often include even millions of parameters. Therefore, XAI is a challenge consisting of numerous subfields and challenges, as described in the XAI Taxonomy.

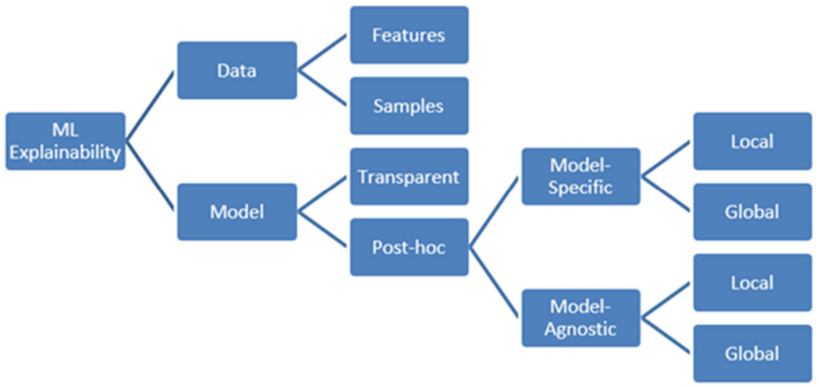


Figure 21. XAI Taxonomy

In general, XAI can be divided into explaining 1) the data itself and 2) the model. Since the model is based on the data, these two sections overlap and intersect. Explaining the data is concerned with exploratory data analysis (EDA). It consists of numerous techniques, including statistical analysis and unsupervised machine learning methods, such as clustering, latent variable analysis, and anomaly detection. In addition to explaining the data in isolation, the data can reveal more when using means of model interpretation.

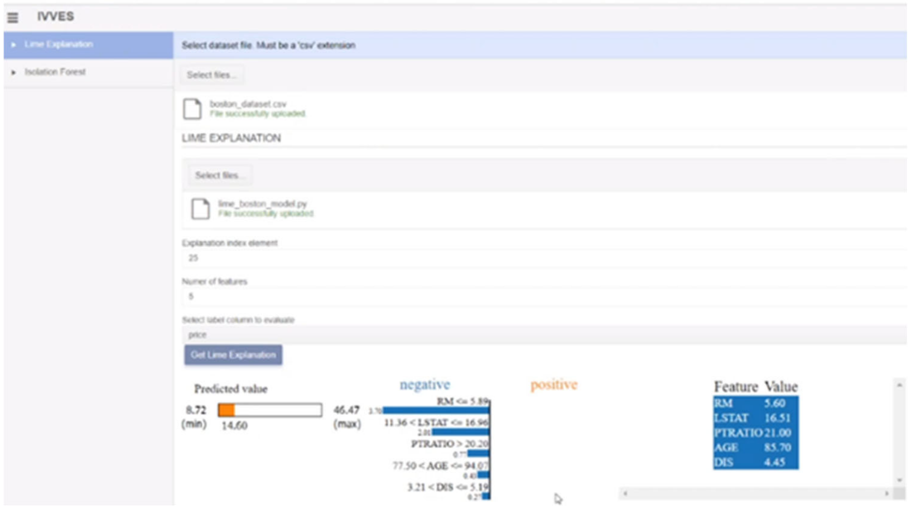
KEYLAND has analyzed benchmark techniques and their benchmark implementations, performing different proofs of concept applicable to the industrial context.

Algorithm	Domain	Scope	Data Type	Implementation	Reference
G-REX	Model-agnostic	Global/Local	Tabular	NA	NA
QII	Model-agnostic	Global/Local	Tabular	QII	NA
ELI5	Model-agnostic	Global/Local	Any	ELI5	[53]
LIME	Model-agnostic	Local	Any	LIME	[54]
Anchors	Model-agnostic	Local	Tabular, Text	Anchor	[55]
Real-Time Image Saliency	Model-agnostic	Local	Image	pytorch-saliency	[56]
Real-Time Image Saliency	Model-agnostic	Local	Image	SaliencyMapper	[57]
ASTRID	Model-agnostic	Global	Any	astrid-r	[58]
KernelSHAP	Model-agnostic	Global/Local	Any	KernelExplainer	[59]
TreeSHAP	Tree ensemble	Global/Local	Tabular	TreeExplainer	[60]
DeepLIFT	Deep learning	Local	Any	DeepLIFT	[61]
DeepSHAP	Deep learning	Global/Local	Any	DeepExplainer	[62]
Grad-CAM	Deep learning	Local	Image	ELI5	[63]
Grad-CAM	Deep learning	Local	Image	Captum	[64]
Deep Visualization	Deep learning	Local	Image	DeepVis	[65]

Table 2. The techniques and implementation.

LIME uses the metaphor of the black box, where the input variables are on one side and the output on the other, with no access to the inside of the box. The L (Local) of LIME focuses on locality and operates on the model test set (test) with both the subset of correctly classified and unclassified instances.

For each instance, close (local) dummy data are created. Distances between the false data and the original data are calculated and predictions are made based on this new data. The minimum number of variables m that generates the maximum probability of correctness is determined and this is considered an explanation of why the instance was classified with the assigned class.



Select dataset file. Must be a 'csv' extension

boston_dataset.csv
File successfully uploaded

lime_boston_model.py
File successfully uploaded

Explanation index element
25

Numer of features
5

Select label column to evaluate
price

Predicted value
8.72 (min) 14.60 46.47 (max)

RM <= 5.89 3.70
11.36 < LSTAT <= 16.96 2.01
PTRATIO > 20.20 0.77
77.50 < AGE <= 94.07 0.40
3.21 < DIS <= 5.19 0.27

Feature	Value
RM	5.60
LSTAT	16.51
PTRATIO	21.00
AGE	85.70
DIS	4.45

Figure 22. Front-end KEYLAND – LIME Evaluation.

Finally, the outline of Keyland's development is summarized in the figure below.

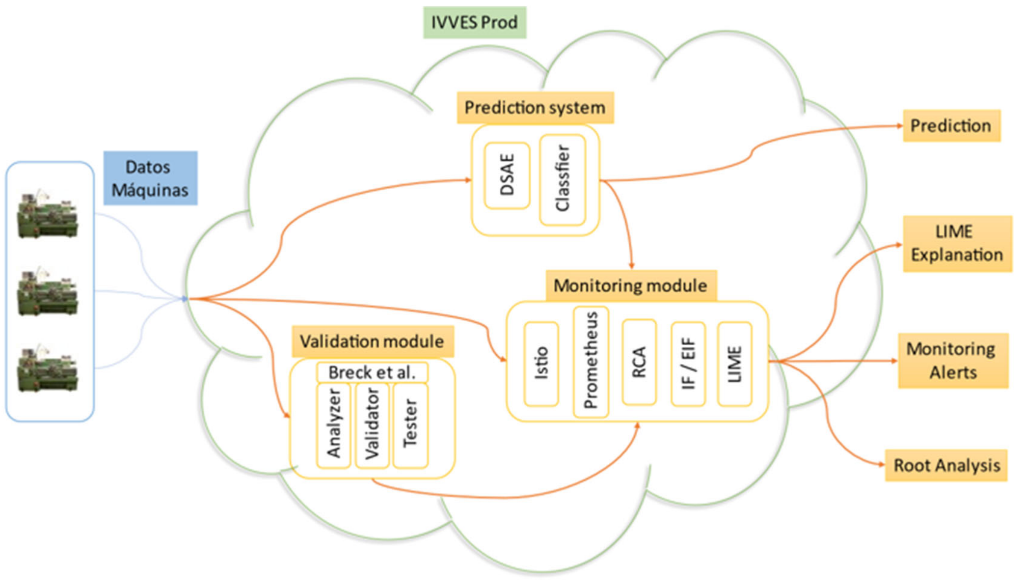


Figure 23. Keyland final scope.

For this purpose, Keyland has focused on the following developments, highlighting the following WP2 tasks:

- ARLean-style generation application [48]. Keyland has investigated GAN for creating synthetic data to train with. This is part of T2.3.
- Validation system based on DATA VALIDATION FOR MACHINE LEARNING [49] which validates that the data corresponds to what is expected, corresponding to T2.2.

4.3. Model Inference Scalability

Performance is yet another perspective related to model quality. Often, model accuracy is traded with model performance where neural networks (NN) with denser hidden layers and using high precision number representation formats such as 64-bit floating point often led to higher level of accuracy. On the other hand, a less dense network and using lower forms of numeric precision for example 32-bit integers are known provide enhanced performance at the expense of higher accuracy. These performance tuning approaches (typically pruning and quantization) [66] are used at the development stage of the ML lifecycle or at compile time. In this study, we consider performance at the deployment and inference stage of the ML life cycle or what may be considered at a model's runtime.

4.3.1. Protocol scalability

Solution provider: Helsinki University

Models are either served using REST or gRPC protocols, the choice between these two protocols is largely influenced by the system or business requirements. REST is more dominant due to its popularity in classic web development and it's utility in public API design. GRPC on the other hand has gained traction as a more performant alternative to rest more so in low-latency environments due to its support of streaming among other technical features.

In this study, we establish the empirical performance characteristics between REST and gRPC in order to characterize the latency associated with the two architectures. The test setup involves a client and a server each running on their own virtual machine having three virtual cores and 4GB of memory. The client generates requests (REST/gRPC) to the server currently a tensorflow serving server. We use the Locust performance testing framework to generate a target test load.

Figure 24 presents data obtained from latency experiments on single requests (non-batch). The setup simulates 500 users spawned 10 users/sec and the experiments are run for 5 minutes each. The first subplot shows that for the same number of users and HW resources, a gRPC architecture would result in a significantly higher number of requests compared to a rest architecture, at maximum 1392 requests/sec, 707 requests/sec respectively. This is attributed to gRPC's multiplexing feature. The second subplot shows the average response time across the two architectures, gRPC has a significantly lower latency compared to REST. The shape of the latency graphs shows gRPC to have an almost linear shape compared to REST suggesting that gRPC would consume resources in an linear manner. This has a bearing on the rate of resource utilization, gRPC would require fast provisioning of resources if the traffic increases rapidly. Such a scenario is proven in the third subplot which shows that gRPC begins to experience failures at the peak load (no other resource were available to support the traffic).

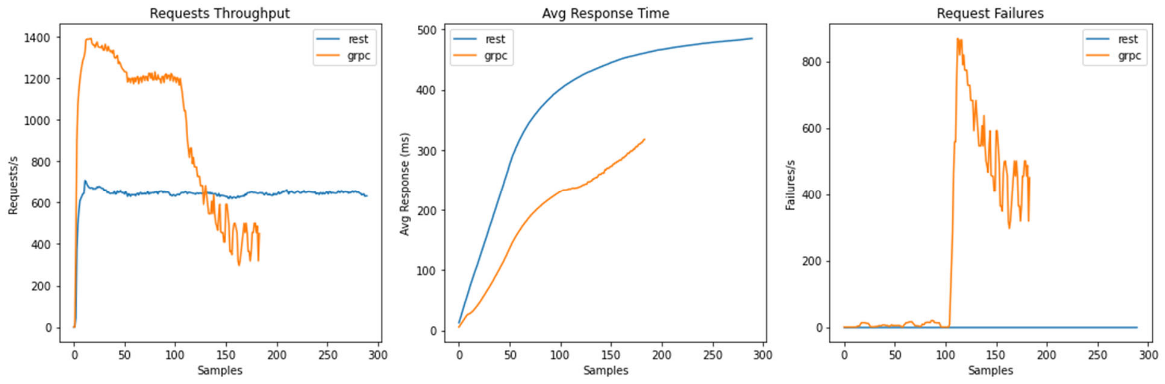


Figure 24. Comparison of gRPC and REST at a peak load of 500 users

An alternative view is the generation of users and latency development as shown in Figure 25 below. In both cases the system simulates 500 users, the latency increases much faster in gRPC as users are added to the system compared to REST where the latency increases but in a non-linear manner.

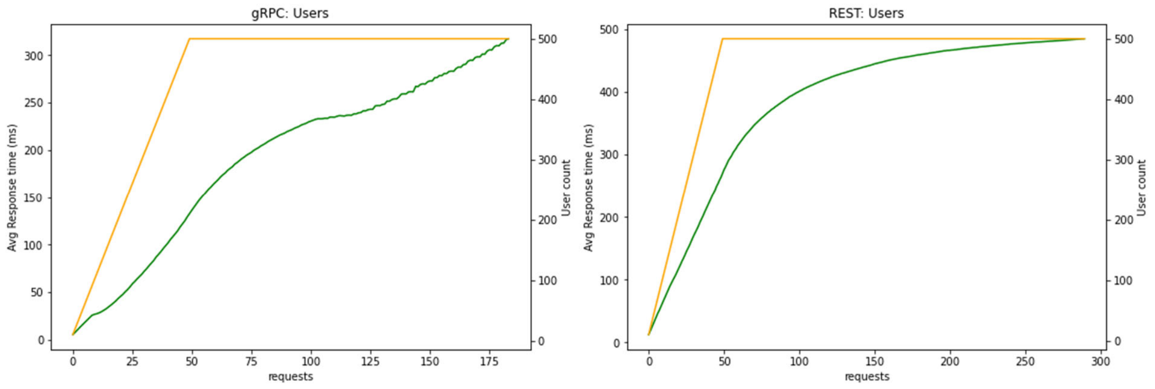


Figure 25. User addition and latency

Figure 26 shows experiments conducted in a batch handling setting. We compare batch performance across gRPC and REST by increasing the batch size from 4, 8 and 16. As the batch size increases the number of requests handled by REST is reduced by a factor of 4 while gRPC request handling reduces by a factor of 1.7. This outcome has implications on the scaling designs required across each architecture.

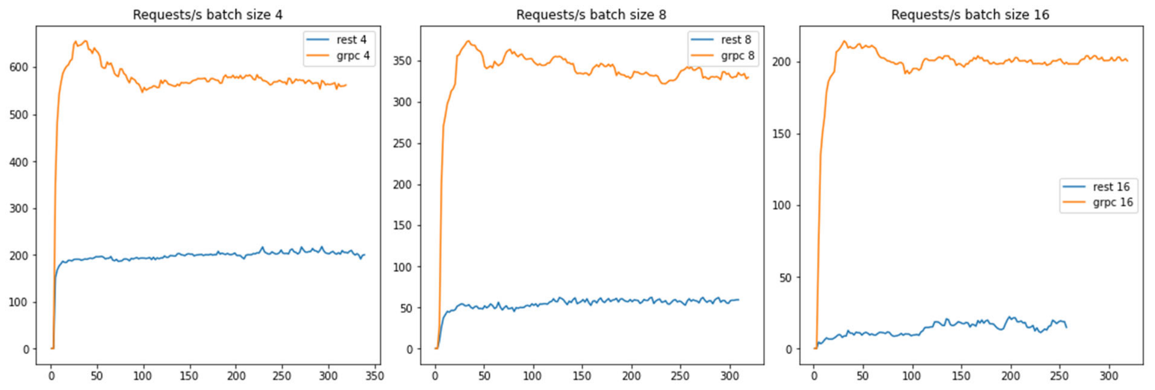


Figure 26. Latency and batch sizes in gRPC and REST

At higher batch sizes, the behavior across both protocols is shown in Figure 27 below. In this setting the simulation involved 100 users and the prediction was carried out using a batch size of 32. The Request throughput is shown in the first subplot and indicates a significant disparity in REST and gRPC request throughput. The large payload results in a very low throughput when using REST compared to gRPC due to the single request-response communication channel used in REST. The second subplot shows how users are generated across the systems and in this view, we note that the first few REST-based requests block the simulation to the extent that the load generation framework does not record the early stages of spawning users. Finally, the third subplot shows a significant difference in latency across the two protocols. With large batch sizes, REST based requests would need to be asynchronous to avoid the observed blocking behavior. GRPC on the other hand has built-in asynchronous communication by default but can be configured to operate in an asynchronous way to match a RESTful pattern.

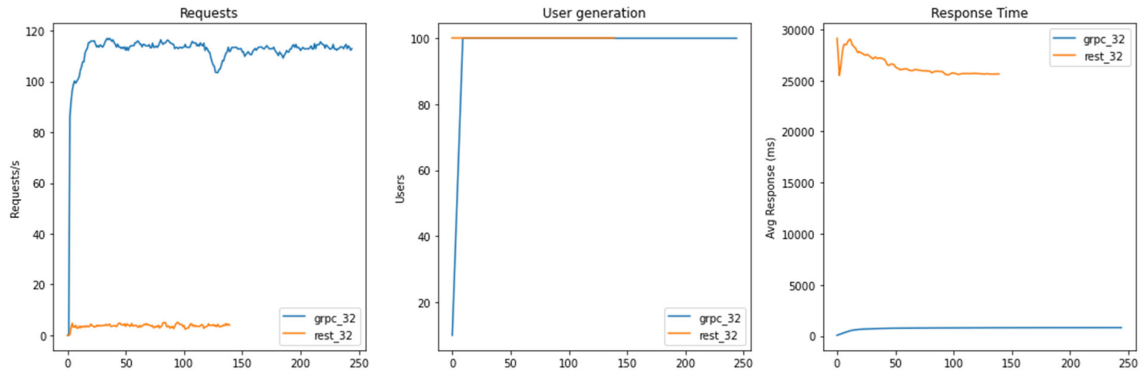


Figure 27. Higher batch sizes

Overall, gRPC benefits from the streaming capabilities provided by HTTP2 which allows multiplexing and asychronous requests by design compared to REST which uses HTTP1.1 and therefore sends one request/response for every TCP connection. REST remains highly popular due to its ease of use among the developer community and is still widely adopted in non-performance critical environments. These experiments motivate more research questions on how the protocols differ, for example in the serialization stage of data transfer and their impact on low-level HW resource utilization, for example memory, network bandwidth and CPU clocks.

4.3.2. Knowledge Distillation

Solution provider: Ekkono

Knowledge distillation is a novel approach to address model scalability while maintaining predictive performance (accuracy). Model compression [86] was introduced in 2006 as an attempt to compress larger, complex models (e.g., ensembles) into smaller, more scalable models, without significant loss in predictive performance. This approach was later extended into knowledge distillation [87] and the student teacher framework. The key idea is to build a student model (small version, scalable) that is trained on the soft labels of the teacher model (bigger, too complex) rather than on the original dataset that the teacher was trained on. The theory suggests that these soft labels are easier to be correctly approximated by a smaller model that would, on the other hand, not be able to correctly learn from the complex non-linearities of the original dataset. The process is usually the following. First, a teacher model is trained on the original dataset; the teacher then labels that dataset; and the student is trained on the labels (called soft labels) from the teacher model. The authors [87] introduced the distillation loss, a loss function that takes into account how good the student is at learning the labels of the teacher (soft labels) and also the hard labels from predicting on the original dataset (ground truth).

At Ekkono, we investigate knowledge distillation for model compression in the context of federated learning. The setup for federated learning consists of a central server that is orchestrating the decentralized learning of individual ML models across different devices (e.g., IoT devices). These devices can't operate with big

ML models due to computational and memory constraint, thus having smaller distilled models that can keep up with the required predictive performance is key.

In this context, we have set up a framework that can orchestrate this sharing of knowledge by using knowledge distillation, focusing on two aspects.

Data compression

First, we designed a solution that allows devices to share the knowledge of their model with the central server by only saving small proportions of the data. With that summary of the data, plus the model, we can generate new data (thus helping privacy-aware scenarios), that is used to update the general model at the server. That model then labels the data, which are later used to train the student model. That small student model can then be sent back to the device, contributing with new more general knowledge.

While most of the knowledge distillation and investigation papers assume that the data to train the student model is easily available, in the context of federated learning that is not feasible. The reason is that it is not possible to save large amounts of data at the edge. We have designed a new approach, called the Online Data Compressor, that can create a very accurate summary of the observed data with only a few instances. Part of the investigation in this project has been to evaluate this approach and compare it with other data summarization approaches, the results showing that the online data compressor can create very good representations of the data with only a few data points (50 in many cases). We are continuing to benchmark this solution and the plan for the upcoming deliverables is to disseminate this in the form of a paper and empirical experiments.

Distillation approaches

There are several ways to create the teacher model that resides in the server. That is, how the generated dataset is going to be labeled, which will then be used to train the student model. This teacher model should be general enough to work in different scenarios, and accurate enough to output good results on those scenarios. We have been investigating several approaches to do the training for the teacher under different datasets from different devices. Such as a weighted ensemble of all the models sent to the cloud, each model's weight being proportional to *how good it approximates the generated data*; a standard ensemble; and a combination of the models where each model labels their generated dataset. The combination of the models works best in those scenarios where there is a clear separation of the domain between the devices, i.e., each device has learned from a different domain so a general model can't generalize well between devices.

Some of the use cases where we have been investigating knowledge distillation for federated learning are the following:

- Model warm-up: in case we are going to deploy a new model on a completely new device to conduct incremental training on it, it's a good idea to start the training from a pre-trained model that has already gathered knowledge from similar devices in similar or different environments
- Improved personalized learning: personalized learning at the edge can be improved by increasing the diversity of the data that a specific device has observed. That way, that specific device will have observed data not only from its particular region, but also from many other ones. This makes the model more generalizable and robust.

4.4. Drift Detection and Response

Solution provider: F-Secure

Machine learning models are trained at a particular time and with data. Under the assumption that the environment (and thus the distribution of the data) does not change, the model will likely keep performing well. However, in practice, the environment in which a model operates (and thus the distribution of the data) changes and, as a result, the performance of this model may deteriorate. Therefore, it is important to monitor machine learning models and their data once deployed in order to make sure that the models keep performing as expected. Once it is detected, through monitoring, that the performance of a model decreased, we can respond and mitigate the issue with the help of information gathered as part of the monitoring.

At F-Secure, as part of this project, such a monitoring approach was designed, implemented, and deployed to track the performance of a host behavior classification system. This system uses a machine learning classification model to predict as many classes as there are types of host behavior to identify (say class A, B and C). The model uses information about the host activity as input, e.g., processes launched, domains accessed, and files opened by a given unknown host. It is trained in a supervised manner using this same activity information from hosts from which the behavior types are known. The output of this machine learning model is then used in downstream tasks that help protect these hosts.

At any point after deployment, observability is desired into whether the model still performs as expected or whether the model deteriorated due to e.g., covariate shift or concept drift [67]. Furthermore, since the data is temporal and the environment may change in some seasonal (or otherwise regular) way, it can be useful to monitor how performance metrics vary with time, and whether the latest model still performs better than previous models.

Typical metrics for performance monitoring (e.g., recall or precision) require availability of ground truth labels. However, labeled data is not always available or labels can be incorrect due to data poisoning or mislabeled samples. Therefore, it is useful to have performance monitoring that does not rely on labels or ground truth being available.

An alternative approach to monitoring model metrics is to directly measure differences in data distribution between training data and inference data, and to quantify data drift that way. The advantage of this approach is that it can be applied regardless of availability of labels, and it can highlight changes in the data before they have a negative impact. However, data drift can be harmless in practice, and it was observed to not always translate into worse performance. Data drift does not always correlate well with actual model performance. Therefore, we take the approach of detecting potential issues through observed performance metrics, and then respond by investigating those samples that likely caused the deterioration of metrics.

To obtain observability in current and historical model performance, starting from the day a model is trained, daily predictions are gathered on a collection of benchmark data sets, and metrics of interest are computed. These metrics are then made available through a dashboard, where the end-user can select what time, models, and data to visualize as depicted in Figure 28.

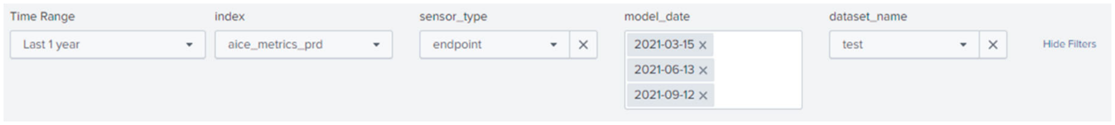

Time Range: Last 1 year
index: aice_metrics_prd
sensor_type: endpoint
model_date: 2021-03-15, 2021-06-13, 2021-09-12
dataset_name: test

Figure 28. Filtering section of the dashboard.

Since the approach needs to work for data for which ground truth labels are not (yet) available, our main metric of interest is **average model uncertainty** (entropy of predicted class probabilities), aggregated over all samples per predicted class.

First, we compute the per-sample model uncertainty (entropy) for each sample i , given the predicted class probabilities $yprob_{ij}$

$$entropy_i = \Sigma_j \, yprob_{ij} \log(yprob_{ij})$$

Then, we aggregate these into per-class $modeluncertaintyavg_c$ metrics, for each of the classes C :

$$modeluncertaintyavg_c = avg(entropy_i, predicted\ class\ for\ sample\ i\ is\ C)$$

The intuition behind this metric is that once samples move closer to decision boundaries (and the model becomes more uncertain of its predictions), they are more likely to be misclassified. Therefore, model uncertainty (entropy of predicted class probabilities) can be a proxy for actual model performance: if model uncertainty goes up, the model performance goes down, and vice versa.

To validate the approach, it is evaluated on data for which we have ground truth labels available. First, we look at actual model performance, computed using ground-truth labels, per-class recall and precision:

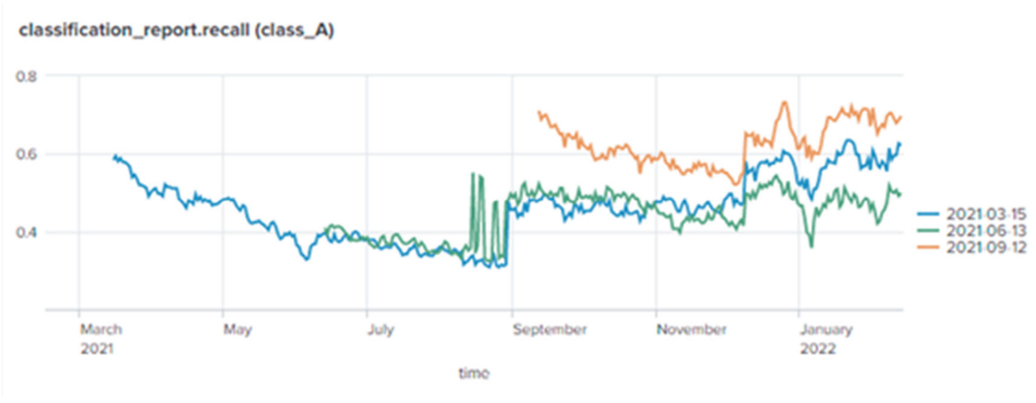


Figure 29. Evolution of performance metric (recall for class A) for 3 different models (2021-03-15, 2021-06-13, 2021-09-12), since the day they were trained.

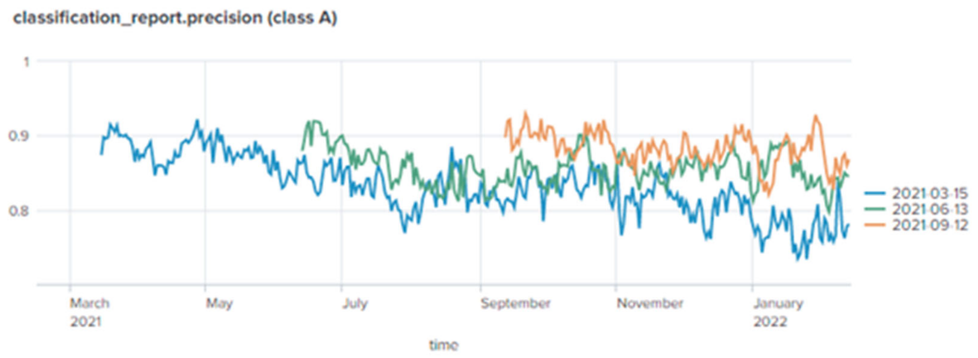


Figure 30. Evolution of performance metric (precision for class A) for 3 different models (2021-03-15, 2021-06-13, 2021-09-12), since the day they were trained.

From Figures 29 and 30, we can see that:

- retraining a model generally improves its performance.
- model performance varies over time, and after an initial drop, it increases back (which may point at the existence of seasonal trends, although more data is needed to say for sure).
- around the turn of the year, all models suddenly show a drop in performance.

Next, we look at whether this is reflected in the average model uncertainty metric, which can be computed without having ground truth labels available:

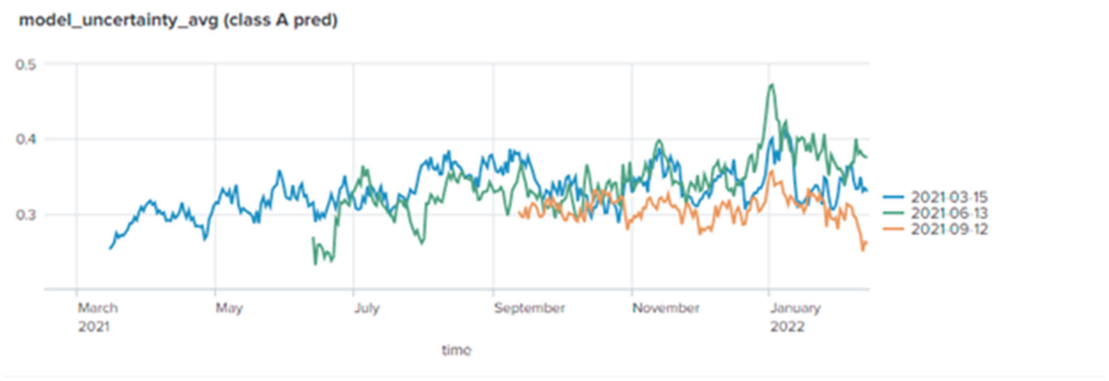


Figure 31. Evolution of performance metric (model_uncertainty_avg for class A) for 3 different models (2021-03-15, 2021-06-13, 2021-09-12), since the day they were trained.

From Figure 31, we can see that:

- As model performance deteriorates, average model uncertainty increases.
- Average model uncertainty shows similar trends over time, albeit not in such an apparent way.
- Around the turn of the year, average model uncertainty for all models suddenly peaks, which is well correlated with a decrease in precision and recall.

Based on these observations, the **average model uncertainty provides a good indication of actual model performance**, and it can be used as an indicator of deteriorating model performance. An analyst works with these metrics through the top part of the dashboard, as depicted in Figure 32.

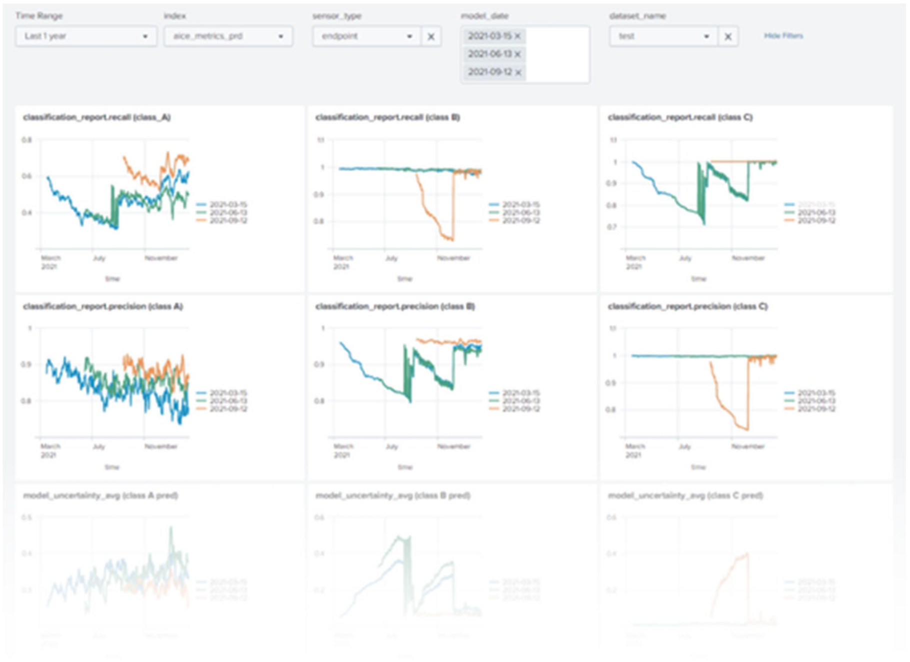
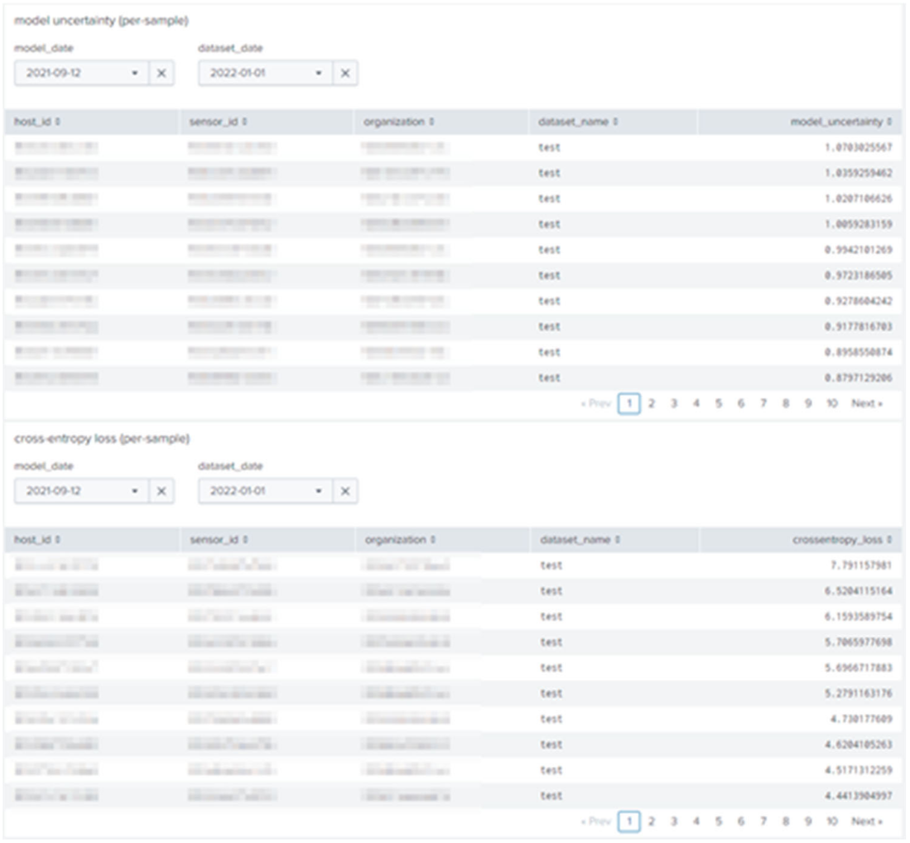

Time Range: Last 1 year
index: aice_metrics_prd
sensor_type: endpoint
model_date: 2021-03-15, 2021-06-13, 2021-09-12
dataset_name: test
Hide Filters
classification_report.recall (class_A)
classification_report.recall (class B)
classification_report.recall (class C)
classification_report.precision (class A)
classification_report.precision (class B)
classification_report.precision (class C)
model_uncertainty_avg (class A pred)
model_uncertainty_avg (class B pred)
model_uncertainty_avg (class C pred)
2021-03-15
2021-06-13
2021-09-12

Figure 32. Main part of the dashboard used by the analyst to select what data and models to inspect and check for potential issues

Once likely deterioration in model performance is detected, several responses for mitigation are available:

- **Model retraining**: the model can be retrained on the latest data (which improves the performance metrics, as our results indicated):
- **Refining data**: the samples of interest can be investigated and correctly labeled, i.e.:
  - o **Active learning**: provide labels for those unlabeled samples that the model is most uncertain on (have highest $entropy_i$, as defined earlier), since these samples will likely have the biggest impact on the model, if labeled correctly as part of an active learning loop.
  - o **Label correction**: investigate and potentially relabel samples that the model gets most wrong (have the highest loss $crossentropy_i = -\Sigma_j ytrue_{ij} \cdot \log(yprob_{ij})$), since the samples with highest loss are potentially mislabeled.

These per-sample statistics are provided in two tables (sorted by metric value) at the bottom of the dashboard, as shown in Figure 33, where the analyst can inspect them, and then decide whether to take further action of labeling / relabeling certain samples.


model uncertainty (per-sample)

model_date: 2021-09-12
dataset_date: 2022-01-01

host_id	sensor_id	organization	dataset_name	model_uncertainty
[redacted]	[redacted]	[redacted]	test	1.0703025567
[redacted]	[redacted]	[redacted]	test	1.0359259462
[redacted]	[redacted]	[redacted]	test	1.0207106626
[redacted]	[redacted]	[redacted]	test	1.0059283159
[redacted]	[redacted]	[redacted]	test	0.9942101269
[redacted]	[redacted]	[redacted]	test	0.9723186505
[redacted]	[redacted]	[redacted]	test	0.9278604242
[redacted]	[redacted]	[redacted]	test	0.9177816703
[redacted]	[redacted]	[redacted]	test	0.8958550874
[redacted]	[redacted]	[redacted]	test	0.8797129206

« Prev 1 2 3 4 5 6 7 8 9 10 Next »

cross-entropy loss (per-sample)

model_date: 2021-09-12
dataset_date: 2022-01-01

host_id	sensor_id	organization	dataset_name	crossentropy_loss
[redacted]	[redacted]	[redacted]	test	7.791157981
[redacted]	[redacted]	[redacted]	test	6.5204115164
[redacted]	[redacted]	[redacted]	test	6.1593589754
[redacted]	[redacted]	[redacted]	test	5.7065977698
[redacted]	[redacted]	[redacted]	test	5.6966717883
[redacted]	[redacted]	[redacted]	test	5.2791163176
[redacted]	[redacted]	[redacted]	test	4.730177609
[redacted]	[redacted]	[redacted]	test	4.6204105263
[redacted]	[redacted]	[redacted]	test	4.5171312259
[redacted]	[redacted]	[redacted]	test	4.4413904997

« Prev 1 2 3 4 5 6 7 8 9 10 Next »

Figure 33. Per-sample model uncertainty and cross-entropy statistics.

In summary:

- The average model uncertainty metric provides an effective way to monitor the effects of harmful data drift, because it is a proxy for the actual model performance and can be computed without having ground truth labels available

- Therefore, it can be used to monitor models in production, where only unlabeled data is available, and can be used to detect deteriorating model performance.
- In response to detected model deterioration, the model can be retrained when needed and/or specific samples can be investigated based on either model uncertainty (for unlabeled data), or cross-entropy loss (for labeled data), and correctly labeled.

4.5. Model Transparency with Neural Backed Decision Trees

Solution provider: Sogeti NL

In D2.3 Sogeti demonstrated the use of Neural-Backed Decision Trees (NBDTs) as a solution to interpreting intermediate model decisions. The process is to convert a neural network into a decision tree and then create hierarchical explanations to explain the sequential decisions. Sogeti implemented this solution with open-source image data and built a Streamlit front-end to visualize the model decisions. Because the use case that was planned to be used to validate this approach was delayed from the beginning of the IVVEs project, this solution was a challenge because it has remained general and trained on open-source data.

Sogeti, together with the contributors and partners of WP2, have collectively decided to terminate the development of this solution due to the challenges and to a shift in effort of Sogeti (Sogeti has assumed the role of WP2 lead). However, Sogeti will still investigate and implement XAI methods for their contribution in WP3 T3.1 and T3.2. This includes developing a code quality solution that can explain model predictions. The current implementation of XAI for this WP3 solution is SHAP.

5. Task 2.3 - Testing techniques for ML

Task Lead: Sogeti NL

In the following sections we describe the final methods that the IVVES consortium is developing to test machine learning models. These smart testing methods apply to the 'Model Evaluation' and 'Model Deployment' phases of the QAIF mentioned in the SoTA Deliverable 2.1 [18]. These are the last phases in which we determine quality before the model is deployed to production. We describe innovative methods such as bio-inspired search-based optimization techniques to generate failure-revealing test scenarios effectively and efficiently for the ML-driven system under test.

The oracle-centered approach to evaluate learning algorithms by CRIM has been put on hold and will therefore not be included in this deliverable. A new method of adversarial testing of convolutional neural networks is introduced by RISE.

Additionally, RISE presents the final version of their evolutionary test generators for testing a DNN-based lane-keeping system, a prevailing advanced driver assistance system (ADAS) in the automotive industry.

Furthermore, SII Concatel presents the final version of metamorphic testing techniques to automate black box testing by generating label preserving perturbations to inputs to scale test creation.

Finally, the AutoML methods to generate ML models and compare the performance of these autonomously generated models to manually created ML models are presented. This is to test and evaluate model robustness to optimize hypermeter tuning and model configuration for utmost performance and quality assurance. The partners in collaboration of the AI-driven testing initiatives include RISE, CONCATEL & NetCheck, University of Helsinki and Techila Technologies.

5.1. Bio-Inspired Search-Based Testing of ML-Driven Systems

Solution provider: RISE

Nowadays, with the growing use of machine learning components in many software-intensive systems, there is a big demand for sophisticated and though pragmatic approaches for testing ML-driven systems. Many of these ML-driven systems in the areas such as automotive, manufacturing, and health care are subject to strict safety requirements. There are new technical guidelines and safety standards being developed to support quality assurance of ML systems. Self-driving cars are prevailing examples of safety-critical AI systems and in this regard, in the automotive domain, ISO/PAS 21448 Safety of the Intended Function (SOTIF) standard is a new under development standard focusing on safety assurance of the systems relying on ML [66]. Meanwhile, AMLAS (Assurance of Machine Learning in Autonomous Systems) is another framework, developed by the University of York, supporting the development and quality assurance of safety-critical ML systems [67].

AMLAS pints out different required activities for the assurance of ML systems. Those activities include assurance of ML safety requirements, data management, model learning, model verification, and model deployment. AMLAS introduces testing and verification procedures at different stages, e.g., model verification that can be regarded as unit testing for ML components and system-level (integration) verification that is carried out after integration and deployment of the model in the system. Besides the model verification, there is a strong need for system-level testing of ML components, particularly in safety-critical systems. For example, in autonomous vehicles, as a prevailing use case of ML-enabled systems, ML components are also connected to other components and the actual functionality of the systems is realized through the integration of ML-based components and some other advanced electronics such as cameras, sensors, and LiDAR technologies. More interestingly, for instance in parallel with the rapid growth of the application of these systems in the automotive domain, there is also an increase in the number of car malfunctions, accidents and crashes that involve the autonomous cars. Therefore, there is an essential need to verify and test at the "*system level*" to ensure the intended correct functionality of the system, particularly in safety-critical domains.

In practice, generating effective test inputs which could lead to malfunctions or improper functionality of the ML model or ML system is often a challenging task. Test input data that can reveal failures, depending on the test level, could be for instance images, as used in DeepTest [68] and DeepXplore [69] or test scenario configurations as used in GA-driven ScenarioGenerator in [70] and Deeper [71].

Generating failure-revealing test input data could be done through input data mutation and test scenario manipulation. Mutating input data involves generating new input through applying various transformation techniques such as using metamorphic testing techniques [68, 69] and evolutionary search-based algorithms [72, 73] to the existing data.

Test scenario manipulation involves going through the search space of the possible test scenarios to find the failure-revealing scenarios. Many of the works in this category use search-based techniques to discover the critical failure-revealing scenarios [74, 75, 76]. In this regard, simulators as a form of digital twins have been considered an effective complementary solution to field testing that can strongly help with capturing the critical scenarios. Nowadays, several high-fidelity simulators have productively contributed to this area. CARLA, BeamNG.tech, SVL simulator, and Pro-SiVIC are among the ones that have been widely used for testing automotive ML-driven systems [77].

The current test subject (ML-driven system under test) in our research is a smart driving agent---a BeamNG-integrated autonomous driving agent, which utilizes optimization techniques to plan the driving trajectory w.r.t speed limit constraints while keeping the car inside the road lane. It has been equipped with a deep neural network-based lane-keeping system, which involves onboard cameras and a deep learning steering angle prediction module. The functionality that is intended to be tested is the capability of keeping the car inside the lane (Lane-keeping).

In our work, we propose and develop a bio-inspired computation-driven test generator, called Deeper, which can generate effectively and efficiently failure-revealing test scenarios. In this work, failure is defined in terms of episodes in which the car drives partially outside the lane, w.r.t a tolerance failure threshold. This threshold specifies the percentage of the car's bounding box required to be outside the lane to regard that driving episode as a failure.

Deeper in its current version utilizes a set of bio-inspired search-based techniques, genetic algorithm, $(\mu + \lambda)$ and (μ, λ) evolution strategies, and particle swarm optimization to generate failure-revealing test scenarios. The problem is basically considered an optimization problem. In this regard, we use a heuristic-based objective function to guide the search process and leverage a quality population seed to boost the search, meanwhile, we develop domain-specific evolutionary operations (crossover and mutation) for the presentation model used for modelling the test scenarios in the tool. Our empirical evaluation of Deeper shows that the test generators in Deeper can perform effectively and efficiently to provoke a considerable number of failure-revealing test scenarios with respect to different target failure severity (e.g., high tolerance failure threshold), driving constraints (e.g., speed limits), and the available test budget (I.e., the amount of time allocated for generating test scenarios).

They prove to be able to trigger several diverse failures even under limited test budget, speed limit constraints and high failure tolerance threshold. These developed test generators show significant effectiveness in terms of the ratio of detected failures to the total valid test scenarios generated and prove to be reliable test generators in reaching the test objective, i.e, triggering diverse failures in the ML system under test.

5.2. Generating Adversarial Examples for increasing ML Robustness

Solution provider: RISE

An adversarial example is a slightly perturbed image, still easily recognizable by human observers, generated by an adversary with the goal of producing a wrong output from the correct target class. An adversarial example $\acute{x}$ is generated as $\acute{x} = x + \epsilon$ where x is the original sample and ϵ is the added perturbation. The aim of an adversary is to find $\acute{x}$ to deceive network f such that:

$$f(x) \neq f(\acute{x}) \land \nabla(\acute{x}, x) \leq \epsilon$$

Where $\nabla(\acute{x}, x)$ indicates the difference between the original data and the adversarial example. Generally, the strength of the adversaries is limited by ϵ which is the amount of change they are allowed to apply to the original sample. Adversarial examples may be targeted in which their goal is for adversarial example $\acute{x}$ is to be classified as a specific class t such that:

$$f(\acute{x}) = t \land f(x) \neq f(\acute{x})$$

∇ is commonly defined by a L_p_norm distance metric. Three of the most popular choices for L_p are:

- L_0: For image samples, this metric limits the number of pixels that the adversary is allowed to perturb to generate an adversarial example. The amount of perturbation on each pixel is unlimited.
- L_2: This metric is the Euclidean distance between the initial sample x and the adversarial example $\acute{x}$. In this method the adversary is allowed to tweak any pixel of the image if the L_2 distance is smaller than a certain value.

- L_∞: This metric limits the amount of perturbation that an adversary is allowed to apply to each pixel. The adversary may alter any number of pixels if the amount of perturbation for each pixel is smaller than a predefined value.

We have used several different attack algorithms to evaluate the models in our experiments. These attacks generate their examples based on different metrics, one pixel attack [78] uses L_0 norm, DeepFool [79] uses L_2 norm and L_∞ norm attacks such as FGSM [80], BIM, and PGD [82]. Carlini and Wagner attack (C&W) [72] supports all of the aforementioned metrics. The following is a detailed description of how these attacks generate their adversarial examples. Here, we refer to the difference between the original and the perturbed image as $\eta = x - \acute{x}$.

Pixel Attack: This adversarial attack generates its examples based on differential evolution (DE) method where the attacker is only allowed to perturb one pixel from the original image. This process is formulated into an optimization problem defined as:

$$\min P_{adv}\big(x + e(x)\big) \;\; subject\ to\!: \; \| \, e(x) \, \|_0 \; \leq 1$$

Where P_{adv} shows the probability of x being misclassified as the wrong class adv, and $e(x)$ is an additive vector with values of $e_1, \dots, e_n$ $w$here n is the number of pixels. According to the optimization problem, only one of the elements of $e(x)$ is allowed to have a non-zero value. Ultimately, solving the optimization problem from the pixel attack determines the amount of perturbation and the pixel it should be applied to.

Fast Gradient Sign Method (FGSM): FGSM is an efficient method that benefits from the liner nature of neural networks to generate adversarial examples. Cost of the attack is controlled by measuring the η using the L_∞ metric. FGSM assumes that the amount of perturbation is similar in every dimension which is limited by the parameter ϵ. The following formula describes how FGSM calculates perturbation using gradient vector of the loss function:

$$\eta = \; \epsilon.sign(\nabla_x J(f(x), y))$$

where $J()$ is the loss function that is used in the training phase of DNNs and y is the correct classification label of the input x. The goal of the FGSM is to perturb the input x by maximizing the loss $J()$ based on a transformed gradient.

Basic Iterative Method (BIM): This method is the iterative version of the FGSM attack. Here, the perturbation ϵ is added at small steps instead of applying it at once. The adversarial example at the m_{th}iteration is defined as:

$$\acute{x}_{m+1} = \acute{x}_m + Clip_{x,\epsilon}(\alpha.sign(\nabla J(f(\acute{x}_m)), y)))$$

The clipping function, $Clip_{x,\epsilon}$, performs clipping on each pixel of η to ensure that the $\acute{x}$ remains in the ϵ -neighborhood of the original sample x. This approach gives the adversary more control over generating the adversarial example and has proven to be more effective than FGSM.

Projected Gradient Descent (PGD): This method uses L_2 and L_∞ and the concepts of back-propagation training to find the smallest perturbation possible to maximize the loss function. Basically, PGD is an advanced version of FGSM and BIM where they have adopted the momentum method to escape from local maxima while finding an adversarial example. Momentum method accumulates a velocity vector with the gradients at each iteration to accelerate gradient descent algorithms. Velocity vectors are calculated as:

$$g_{i+1} = \mu . g_t + \frac{\nabla_x J(\hat{x}_i, y)}{\| \nabla_x J(\hat{x}_i, y) \|_1}$$

After the calculation of the gradients at the i_{th} step, adversarial examples are generated as:

$$\hat{x}_{i+1} = \hat{x}_i + \alpha . sign(g_{i+1})$$

Adversarial examples generated by PGD are more effective than FGSM and BIM attacks in fooling DNNs.

DeepFool: This method finds adversarial examples with minimal required perturbation. The perturbation is calculated by finding the distance from the original image to the closest decision boundary of the classifier. In case of linear classifiers, where decision boundaries are made of linear planes, output classes are represented as polyhedrons (the planes of which are the classifier defined boundaries). To generalize the non-linear classifiers, DeepFool adopts an iterative approach to estimate the linear polyhedron of classifiers. The perturbation is computed as:

$$\arg min \| \eta_i \|_2 \;\; subject\ to\colon f(x_i) + \nabla f(x_i)^T \eta_i = 0$$

Where f is the approximate linearized version of the classifier at each iteration.

Carlini & Wagner (C\&W): This targeted attack is stronger than other known targeted attacks while requiring small amount of perturbation. C\&W attack has three different versions that generate their examples based on L_0, L_2 and L_∞ norms. An objective function g is defined in C\&W such that:

$$min \| \eta \|_p + c . g(x + \eta) \; subject\ to\colon \; x + \eta \in [0, 1]^n$$

Where p indicates the used norm and $g(\acute{x}) \geq 0$ if and only if $f(\acute{x}) = \acute{l},$ in which $\acute{l}$ is the intended target of the attack.

In our research we proposed using attacks that generate their examples to evaluate the robustness of neural networks. We select adversarial examples that are generated using these metrics $L_0\ and\ L_\infty$, as opposed to any other combination of metrics, can enhance the robustness evaluation in neural networks. We have used this method to create a set of pre-generated adversarial examples using different threshold values for $L_0\ and\ L_\infty$, metrics, 1, 3, 5 and 10 respectively. To be inclusive among various neural network architectures, our adversarial examples set is targeted at four well-known CNNs including DenseNet, LeNet, Resnet, and VGG-16. During our search process, we evaluate the robustness of our models as their accuracy against our generated adversarial set. Our approach also assures our robustness evaluation is not biased towards certain adversarial attacks.

5.3. Metamorphic Testing for text driven ESG Investment systems

Solution provider: Concatel/Netcheck

The tools described in 4.2.5 and 5.1. have been used to generate a component that validates and verifies AI models developed for news tagging, within the FinTech sector. Most notably, MLflow records the training metrics as well as trains, saves, verifies, and validates the AI models that are used in the platform.

The following components are highlighted in this validator:

- **Background_Worker**: Based on C#, this component asks every N minutes if there is a new model registered - it has to fulfill some conditions, test tags coming from Mlflow and when it is verified that there is a model with those tags a message is sent to Kafka that this model is available and that it solves a task and passes it the most influential tags.
- **Evaluator**: this component based on Python, collects those messages with the selected model to the task and evaluates that model against a dataset not seen in training. With that evaluation the training metrics are available. Model-drifting can be captured (the model is trained with a starting dataset but with the passage of time it can make wrong predictions, it indicates that it must be re-trained with other datasets) once re-trained, the model is re-validated, and it is checked if it is useful for the trained dataset.

In parallel, a tool has been developed to validate, verify, and correctly classify ESG news for responsible investment and detection of erroneous behavior. The necessary models have been generated to classify the news according to its ESG topic and its sentiment. In addition to all the necessary components to carry out this task automatically on all the news that is published on the internet in real time. This tool is composed of:

- **News Scrapper**: component that collects news from various sources and applies an AI model of classification for its division into E(environmental), S(social), G(governmental).
- **Sentiment classification transformers**: this component applies a sentiment analysis AI algorithm which indicates whether the news, understanding its origin (E, S or G), is positive or negative within that group.
- **Topic classification transformers**: this component extracts the most relevant keywords from the article by applying an XAI algorithm.
- **Graph database for persistence**: this component aims to obtain a better visualization of the news in real time.
- **Docker** for service orchestration.
- **Web UI**, graphical interface based on TypeScript.
- **Rest API** based on TypeScript through which all components communicate.
- **Amazon Cognito** for user and permissions management.

To conclude, with the type of technology available on the market today, ensuring that software is specifically tailored to your audience can make all the difference against competitors. That's why domain knowledge is such a valuable and coveted asset. Within the FIntech use case, CONCATEL and NETCHECK want to give extra value within domain knowledge as it collects data validation from experts that translate into improved algorithm training.

5.4. Robust AutoML

Solution provider: Helsinki University

AutoML is a promising direction to further automate the creation of ML models and choose the optimal models and parameters matching the needs. A potential risk for a highly optimized model is that it becomes very sensitive to its input data. While the training data should contain a wide set of examples it may not fully represent all possible cases where anomalous input is used for inference. In order to ensure that the system is as robust as possible to unexpected inputs, we investigate how robust the models generated by different AutoML approaches are. We use and extend a previously implemented dpEmu data fault generator [83] for this task.

The key observation is that the different AutoML systems work in very different ways. As we report in detail in [84] the resulting models differ in type, size, and performance. Also, there are major differences in the computational resources needed. The same applies to robustness. The difficulty is that predicting in advance how robust the models will be is not easy.

In particular, we have investigated AutoML for time series data coming from different kinds of sensors. Sensor data collection in IoT networks is sensitive to the malfunction of sensors and communications. Hence, it is important that models using the data work in a reasonable way even when there are some, potentially temporary, problems. We have used temperature and taxi data as examples and analyzed how the models generated with different AutoML systems tolerate faults in their input data. We have emulated faults arising from common problems in wireless sensor networks such as missing readings, anomalous values, and readings stuck to a constant value. The analyzed AutoML systems are Microsoft's Azure AutoML, Intel's Analytics Zoo AutoML, and Facebook's Prophet. As a result, we rank AutoML systems based on their performance with respect to data faults and their severity. In addition, we show how the AutoML generated models differ given the data fault type. As an observation, it was interesting to see that some systems performed well with certain data fault types and level combinations but did not do so well with another setup. Also, the generated models were quite different given the type of fault in the training data. Therefore, for a certain problem, one of the systems might be much more effective than others, but in general, this can be found out by running suitable tests. For cases where training must be fast, Prophet is clearly the best, with its forecasting performance comparable with other tested systems.

While all compared systems provide similar features, they all are clearly different, making it difficult to compare them technically in terms of other than input, output, and performance. With LSTM models, deeper layers are used to capture more high-level or abstract features from the data, which in the case of time series would correspond to longer-term trends. Analytics Zoo seems to exclusively generate two-layer LSTM models. So, it can be said that the second layer is used to capture more general trends in the training data, while the first layer is used to describe the more immediate changes. On the other hand, the width of each LSTM layer, namely the number of units on a layer, determines how strongly the relationships between the inputs to a specific layer are taken into account by the model. So, too few units on a layer can lead to underfitting, while too many units can lead to overfitting. In addition, both LSTM layers have an associated Dropout layer in our generated models. The Dropout layer helps to reduce overfitting by excluding some units from updates. More detailed results are available in [85].

Our currently ongoing work is trying to reuse the models developed during the AutoML search. Normally, the non-optimal models are discarded. We, instead, store a sample of the explored models and in case of concept drift, try to find an existing model matching the changed data. In this way we hope to make reacting to concept drift more agile.

6. Operationalizing tasks - From incoming data to a validated model

During the work on this work package, we have noticed three recurring themes:

- Proper data handling and quality assurance.
- Model quality in training and development phase.
- Testing of models after the development phase.

These three points are mapped to the tasks in the work package. For the final version of the methods, we conclude that these three points should not be treated separately in an ML development lifecycle. They need to be brought together so ML algorithms can be tested and developed as traditional software. This ensures a valid basis for a fundamental shift in software and brings about Software 2.0 [83]. In Software 1.0, software is explicitly programmed and usually consists of one team working on the development together. In Software 2.0, we have a lot of moving parts which are mapped to the tasks – training data preparation, architecture, modelling, and behavior of the model after training. These moving parts are handled by different teams. To make them communicate appropriately, we present the Quality AI Framework (QAIF) that can be used to operationalize ML model development.

Inspired by the CRISP-DM Framework, QAIF is a cohesive, generic framework that can be tailored to any AI solution. It is designed to help product managers and business owners identify and mitigate potential risks at each stage of the AI lifecycle. The QAIF can be used to govern the AI development cycle. We add a gate to each phase – just like in traditional software testing – to ensure certain quality control checks are completed. The following paragraphs describe the quality control tests that should be conducted at each phase of the AI lifecycle to ensure quality and ethical adherence. The QAIF has 5 phases – Business Understanding, Data Understanding and Preparation, Model Development and Evaluation, and Deployment.

The process of developing and implementing an AI solution always starts with the business case. A business problem is defined and scoped and then turned into design requirements for data scientists – this is called the Business Understanding phase. Next, training and test data are collected in the Data Understanding phase. In the Data Preparation phase, the data is analyzed, sampled, and pre-processed for the chosen AI model. Then the Model Development phase can begin. In this phase, the AI model is trained, tuned, and evaluated. Once the Model Evaluation is optimized and meets the business requirements, the model can be tested and Deployed. This is when the AI system goes into production and performance is monitored. These phases are iterative and illustrated in Figure 34.

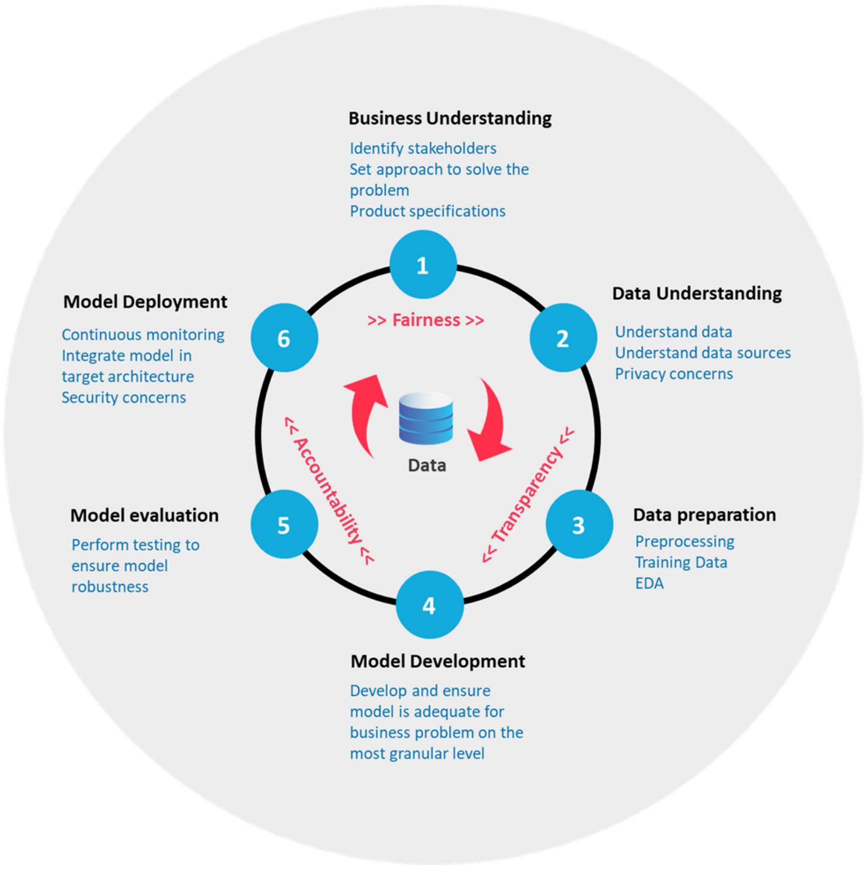


Figure 34. The QAIF gates and main activities.

The methods introduced during the work in this work package correspond to the gates of the QAIF. We will highlight the end-to-end AI product lifecycle by highlighting those methods through the phases.

Business Understanding

In this phase, the tasks of identifying stakeholders, product requirement specifications, technical design specifications, performance metrics and ethical/legal compliance will be completed and understood for the development process to be initiated. No specific tool has been developed to accommodate for this phase as it is a theoretical one, but we have used the principles tied to it to set up experiments in the ITEA IVVES project.

Data Understanding and Preparation

The data phases are the most important in the AI project lifecycle. Data is the fuel for AI, so treating data as a first-class citizen is of outmost importance in AI project management and development. The DQW is here to serve as an automated way of approaching data quality assessment. Synthetic data is used to accelerate this phase and mitigate privacy risks.

As part of this work package, we can provide an example of how this phase looks for one of the use cases around audio data by Solita. The data handling in the use case has many moving parts, two of which are **sensor fusion** and **data farming**. It is important to understand these methods because they impact the following phases of model development. The mapping of the methods happens in the initial phase of the use case, with all relevant parties (stakeholders, developers, product owners, etc.). The output is a document containing specifications of all moving parts of the use case. Two examples are below.

Sensor fusion [84] is a process where several sensor data sources are combined into one. Sensor data quality plays a very important role in many contexts. For instance, Internet of Things (IoT) is an emerging technology where sensor fusion can be applied. Machine learning has many applications in sensor fusion. In the use cases of this project, we have found many ways to apply ML to sensor fusion which are described in the following. In some of the use cases, a data farming [36] approach has been used.

If we understand the nuances of data handling in a use case, we can successfully mitigate risks caused down the line.

Model Development

The AI Model Development phase starts with high quality training data. Model developers have the main responsibility in this phase - ensuring that the AI model they are developing is suited for the application and works with the data prepared in previous phases. To be sure of this, performance metrics are drawn from the model and presented to the stakeholders. Furthermore, we test the model performance and functionality on the most granular level. The testing is focused on features, where we can use XAI to assess feature importance. Furthermore, AutoML can be used to infer optimal hyperparameters for a given model architecture.

Model Evaluation

The model evaluation phase is mapped to the final task of this work package, meaning that all tools explained in its section contribute to this phase of QAIF. To reiterate, you can execute metamorphic and adversarial tests to ensure the model is robust enough to be deployed. Metamorphic tests aim to assess model impact by transforming the inputs of the model and then testing the model with the augmented inputs. Adversarial testing aims to generate adversarial attacks to stress test the model. Furthermore, you can execute user acceptance tests using the XAI outputs that were implemented in the previous phase.

Model Deployment

Once we have a transparent and understandable model, we can enter the final phase with a focus on monitoring, real-world model performance and maintenance.

To ensure robustness, fairness and transparency, a monitoring dashboard should be set up to track model performance in production. The production performance metrics include:

- Model performance metrics from Model Development.
- Bias metrics from Model Evaluation.
- Drift detection metrics like concept drift and data drift detection

In this final phase, we can confidently check all the quality control boxes and pass through all the gates of development, but that doesn't necessarily mean we are done with the AI lifecycle. When the model needs to be retrained or adjusted, we can always turn back and revisit any of the phases, as the AI project life cycle is an iterative process.

Finally, we have a wholesome approach that is in line with the framework and encompasses all three tasks in this work package. From CCTL and NTCHK, an ESG solution that encompasses the application of artificial intelligence for news classification within the Fintech ecosystem is presented. Thanks to this new and innovative development, the verification and validation of artificial intelligence models involved in the classification of news is of great value and complies with the Human In The Loop (HITL) concept.

NETCHECK has contributed its development by designing tools to assess the quality of data from different sources for the training phase of ML algorithms in the FinTech domain, considering GDPR and regulatory constraints. As a result, techniques for erroneous or bad performance of ML algorithms have been provided.

Moreover, CONCATEL has focused on assessing the reliability of datasets from multi-domain sources, including open data initiatives considering specific constraints (GDPR) for Fintech and RegTech (regulatory technology). In addition, techniques to enable KRR from Deep Learning models have been analyzed and components for transparency of ML-based Fintech solutions will be implemented.

Finally, CONCATEL and NETCHECK have generated a tool with great added value within the FinTech community, which goes hand in hand with experts to preserve domain knowledge.

This wholesome approach to the ESG use case and ML adoption in the FinTech industry is impressive and shows that a framework with guidelines on ML model development is of utmost importance when building these solutions.

With these state-of-the-art tools, we can automate the quality control checks at every phase of the QAIF and AI development cycle. The tools enable efficient testing and operationalizing of ML. Many of these tools can be deployed as micro-services to automate the development and testing of AI. This is in line with DevOps & the shift left movement and should be a standard when developing AI products and services.

In the next deliverable, 2.5, we will deliver the final version of the tools along with validation results. We will also report on the dissemination and commercial activities of the tools in the market.

7. References

[1] Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." Science 349.6245 (2015): 255-260.

[2] Fox, Michael J. Quality assurance management. Springer, 2013.

[3] Goodfellow, I. et al., “Generative adversarial networks,” Commun. ACM, vol. 63, no. 11, pp. 139–144, Jun. 2020.

[4] Menze, B.H., A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)", IEEE Transactions on Medical Imaging 34(10), 1993-2024 (2015) DOI: 10.1109/TMI.2014.2377694

[5] Bakas, S., H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J.S. Kirby, et al., "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features", Nature Scientific Data, 4:170117 (2017) DOI: 10.1038/sdata.2017.117

[6] Bakas, S., M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, et al., "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge", arXiv preprint arXiv:1811.02629 (2018)

[7] Van Essen, D.C., S. M. Smith, D. M. Barch, T. E. J. Behrens, E. Yacoub, and K. Ugurbil, “The WU-Minn Human Connectome Project: An overview,” Neuroimage, vol. 80, pp. 62–79, Oct. 2013.

[8] https://github.com/neuronets/nobrainer

[9] Mueller S. G. et al., “The Alzheimer’s disease neuroimaging initiative,” Neuroimaging Clinics of North America, vol. 15, no. 4. NIH Public Access, pp. 869–877, Nov-2005.

[10] Data Quality Wrapper: Clean, describe, visualise and select data for AI models. Available at: https://share.streamlit.io/soft-nougat/dqw-ivves/app.py

[11] Streamlit to the rescue! https://medium.com/sogetiblogsnl/streamlit-to-the-rescue-7d5f2f663465

[12] https://github.com/Baukebrenninkmeijer/table-evaluator

[13] https://github.com/fbdesignpro/sweetviz

[14] https://github.com/pandas-profiling/pandas-profiling

[15] https://github.com/pycaret/pycaret

[16] https://github.com/nltk/nltk

[17] https://github.com/explosion/spaCy

[18] https://github.com/sloria/TextBlob

[19] https://github.com/amueller/word_cloud

[20] https://github.com/shivam5992/textstat

[21] https://github.com/python-pillow/Pillow

[22] https://github.com/librosa/librosa

[23] https://github.com/pierre-rouanet/dtw

[24] https://github.com/iver56/audiomentations

[25] https://github.com/QED0711/audio_analyzer

[26] https://github.com/Setasign/FPDF

[27] https://github.com/JazzCore/python-pdfkit

[28] Dynadot. What is ASCII and what are ASCII vs. Non-ASCII domains? Available at https://www.dynadot.com/community/help/question/what-is-ascii

[29] Zvornicanin, E. When Coherence Score is Good or Bad in Topic Modeling? Available at https://www.baeldung.com/cs/topic-modeling-coherence-score

[30] Comeau, J. Let's learn about waveforms, available at: https://pudding.cool/2018/02/waveforms/

[31] Spectral Density. Available at https://en.wikipedia.org/wiki/Spectral_density

[32] Short-time Fourier transform, available at https://en.wikipedia.org/wiki/Short-time_Fourier_transform

[33] Nair, P. The dummy’s guide to MFCC. Available at: https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd

[34] Zhang, J. Dynamic Time Warping. Available at https://towardsdatascience.com/dynamic-time-warping-3933f25fcdd

[35] Wikipedia: Audio file format. Available at https://en.wikipedia.org/wiki/Audio_file_format

[36] Wikipedia: https://en.wikipedia.org/wiki/Data_farming

[37] Chamberlain, Daniel, et al. "Application of semi-supervised deep learning to lung sound analysis." 2016 38th annual international conference of the IEEE engineering in medicine and biology society (EMBC). IEEE, 2016.

[38] Wang, Avery. "The Shazam music recognition service." Communications of the ACM 49.8 (2006): 44-48.

[39] Haitsma, Jaap, Ton Kalker, and Job Oostveen. "Robust audio hashing for content identification." International Workshop on Content-Based Multimedia Indexing. Vol. 4. 2001.

[40] Hathaway, J. L. "Automatic Audio Gain Controls, Part 1." Journal of the Audio Engineering Society 1 (1950): 16-18.

[41] Malik, H. "Acoustic environment identification and its applications to audio forensics." IEEE Transactions on Information Forensics and Security 8.11 (2013): 1827-1837.

[42] Sanchez, Susan M. “Data Farming: Methods for the Present, Opportunities for the Future.” ACM transactions on modeling and computer simulation 30.4 (2020): 1–30. Web.

[43] Liu, Y., Yang, T., You, Z., Fan, W., & Yu, P. S. (2020). Commonsense Evidence Generation and Injection in Reading Comprehension. arXiv preprint arXiv:2005.05240.

[44] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

[45] Ghorbani, A., Zou, J., Data Shapley: Equitable Valuation of Data for Machine Learning, pre-print: arXiv:1904.02868

[46] Ruoxi, J., et al., Efficient Task-specific Data Valuation for Nearest Neighbor Algorithms, Proceedings of the VLDB Endowment, Volume 12, Issue 11, July 2019, pp 1610–1623, https://doi.org/10.14778/3342263.3342637

[47] Zhang, J. M., et al., Model Validation Using Mutated Training Labels: An Exploratory Study, pre-print: https://arxiv.org/pdf/1905.10201.pdf

[48] Christopoulos, Athanasios, et al. "ARLEAN: An Augmented Reality Learning Analytics Ethical Framework." Computers 10.8 (2021): 92.

[49] Breck, E., N. Polyzotis, S. Roy, S. E, Whang, , and M. Zinkevich, “Data validation for machine learning,” IEEE Transactions on Software Engineering., In SysML, 2019.

[50] Carletti, Mattia, et al. Explainable machine learning in industry 4.0: Evaluating feature importance in anomaly detection to enable root cause analysis. En 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). IEEE, 2019. p. 21-26.

[51] Anastasi, S., Madonna, M., & Monica, L. (2021). Implications of embedded artificial intelligence-machine learning on safety of machinery. Procedia Computer Science, 180, 338-343.

[52] https://github.com/hovinh/QII

[53] https://github.com/TeamHG-Memex/eli5

[54] https://github.com/marcotcr/lime

[55] https://github.com/marcotcr/anchor

[56] https://github.com/plasmashen/pytorch-saliency

[57] https://github.com/karanchahal/SaliencyMapper

[58] https://github.com/bwrc/astrid-r

[59] https://github.com/slundberg/shap/tree/master/notebooks/kernel_explainer

[60] https://github.com/slundberg/shap/tree/master/notebooks/tree_explainer

[61] https://github.com/kundajelab/deeplift

[62] https://github.com/slundberg/shap/tree/master/notebooks/deep_explainer

[63] https://github.com/TeamHG-Memex/eli5/blob/master/notebooks/keras-image-classifiers.ipynb

[64] https://github.com/pytorch/captum

[65] https://github.com/yosinski/deep-visualization-toolbox

[66] Liang, T., et al. "Pruning and quantization for deep neural network acceleration: A survey." Neurocomputing 461 (2021): 370-403.

[66] Road Vehicles – Safety of the intended Functionality. International Organization for Standardization, Tech. Rep. ISO/PAS21448:2019, 2019.

[67] Hawkins, R., Paterson, C., Picardi, C., Jia, Y., Calinescu, R., & Habli, I. (2021). Guidance on the assurance of machine learning in autonomous systems (AMLAS). arXiv preprint arXiv:2102.01564.

[68] Tian, Y., Pei, K., Jana, S., & Ray, B. (2018, May). Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In Proceedings of the 40th international conference on software engineering (pp. 303-314).

[69] Pei, K., Cao, Y., Yang, J., & Jana, S. (2017, October). Deepxplore: Automated whitebox testing of deep learning systems. In proceedings of the 26th Symposium on Operating Systems Principles (pp. 1-18).

[70] Ebadi, H., Moghadam, M. H., Borg, M., Gay, G., Fontes, A., & Socha, K. (2021, August). Efficient and Effective Generation of Test Cases for Pedestrian Detection-Search-based Software Testing of Baidu Apollo in SVL. In 2021 IEEE International Conference on Artificial Intelligence Testing (AITest) (pp. 103-110). IEEE.

[71] Moghadam, M. H., Borg, M., & Mousavirad, S. J. (2021, May). Deeper at the sbst 2021 tool competition: ADAS testing using multi-objective search. In 2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST) (pp. 40-41). IEEE.

[72] Carlini, N., & Wagner, D. (2017, May). Towards evaluating the robustness of neural networks. In 2017 ieee symposium on security and privacy (sp) (pp. 39-57). IEEE.

[73] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016, March). The limitations of deep learning in adversarial settings. In 2016 IEEE European symposium on security and privacy (EuroS&P) (pp. 372-387). IEEE

[74] Abdessalem, B., R., Nejati, S., Briand, L. C., & Stifter, T. (2016, August). Testing advanced driver assistance systems using multi-objective search and neural networks. In Proceedings of the 31st IEEE/ACM international conference on automated software engineering (pp. 63-74).

[75] Abdessalem, R. B., Nejati, S., Briand, L. C., & Stifter, T. (2018, May). Testing vision-based control systems using learnable evolutionary algorithms. In 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE) (pp. 1016-1026). IEEE.

[76] Haq, F. U., Shin, D., Briand, L. C., Stifter, T., & Wang, J. (2021, July). Automatic test suite generation for key-points detection DNNs using many-objective search (experience paper). In Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 91-102).

[77] Borg, M., Abdessalem, R. B., Nejati, S., Jegeden, F. X., & Shin, D. (2021, April). Digital twins are not monozygotic–cross-replicating adas testing in two industry-grade automotive simulators. In 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST) (pp. 383-393). IEEE.

[78] Su, J. , D. V. Vargas, and K. Sakurai, “One Pixel Attack for Fooling Deep Neural Networks,” IEEE Transactions on Evolutionary Computation, 2019.

[79] Moosavi-Dezfooli, S.M., A. Fawzi, and P. Frossard, “DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks,” in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016.

[80] Goodfellow, I.J., J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in 3rd International Conference on Learning Representations (ICLR), 2015.

[83] Karpathy, A. Software 2.0. https://karpathy.medium.com/software-2-0-a64152b37c35

[84] Teh, Hui Yie, et al. "Sensor data quality: a systematic review." Journal of Big Data 7.1 (2020): 1-49.

[82] Madry, A., A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in 6th International Conference on Learning Representations (ICLR), 2018.

[83] Nurminen, J.K., Halvari, T., Harviainen, J., Mylläri, J., Röyskö, A., Silvennoinen, J., and Mikkonen, T., "Software Framework for Data Fault Injection to Test Machine Learning Systems". 4th IEEE International Workshop on Reliability and Security Data Analysis (RSDA 2019) at 30th Annual IEEE International Symposium on Software Reliability Engineering (ISSRE 2019), Berlin, Germany, October 2019

[84] Halvari, T., Nurminen, J.K., and Mikkonen, T., “Testing the Robustness of AutoML Systems," 1st Workshop on Agents and Robots for reliable Engineered Autonomy (AREA) at 24th European Conference on Artificial Intelligence (ECAI 2020), September 2020

[85] Halvari, T., Nurminen, J.K., Mikkonen, T., “Robustness of AutoML for Time Series Forecasting in Sensor Networks,” IFIP Networking Poster Session, virtual event, June, 2021

[86] Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression." Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. 2006.

[87] MLA, Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." arXiv preprint arXiv:1503.02531 2.7 (2015).