IVVES

Industrial-grade Verification and Validation of Evolving Systems

Labeled in ITEA3, a EUREKA cluster, Call 5

ITEA3 Project Number 18022

D2.2 – Training data quality

Due date of deliverable: December 31, 2020
Actual date of submission: December 18, 2020

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

Organisation name of lead contractor for this deliverable: Solita Oy

Author(s): Harri Pölönen, Janne Merilinna, Heba Sourkatti, VTT; Tia Nikolic, Almira Pillay, Sogeti; Marko Koskinen, Techila Technologies; Tommi Mikkonen, Zafar Hussain, Sasu Mäkinen, University of Helsinki; Sanna Öster, Solita; Matvey Pashkovskiy, Janne Taponen, Perttu Ranta-aho, Marcin Kowiel, F-Secure; Kirsi Nurmilaukas, Philips Finland; Mahshid Helali Moghadam, RISE; David Caubilla Quevedo, Keyland; Omer Nguena Timo, CRIM

Status: Approved by PMT, to be submitted to ITEA3

Version number: 1.0.

Submission Date: 31-December-2020

Doc reference: IVVES_Deliverable_D2.2._V1.0.docx

Work Pack./ Task: WP 2 / T2.2

Description: ***(max 5 lines)*** This document describes initial version of validation methods and techniques for ML in project use cases.

Nature:	✓ R=Report, □ P=Prototype, □ D=Demonstrator, □ O=Other		
Dissemination Level:	**PU**	Public	**X**
	PP	Restricted to other programme 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	7/10/2020	First draft, kick-off version	Draft	All
V0.2	6/11/2020	First version	Initial	Authors
V0.3	4/12/2020	Version for the last review	Reviewed	Authors
V0.4	11/12/2020	Version to submit	To submit	PMT
V1.0	21/12/2020	Approved by PMT, to be submitted to ITEA3	Final	Uploaded to ITEA

Table of Contents

1. Glossary

Abbreviation / 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 (2.2.) describes initial validation methods and techniques for ML in project use cases. The state of the art of the validation techniques for ML are described in the document D2.1., which is used as a basis for this document. The work continues in deliverable 2.3. where the actual tools are developed. Therefore, this document can be considered as a project plan for the deliverable 2.3.

The objectives of this task are:

- Assess the quality of data to be used for the ML algorithms;
- Analyze data to support testing of trained ML algorithms;
- Improve training data by boosting 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 training phase.

The document is divided into three parts based on the tasks of the WP2.

First, the Data Quality section contains descriptions of synthetic data cases in the healthcare sector and cyber security sector. The other sub-section under the Data Quality section is incoming data quality assessment. In this section, we introduce a data quality wrapper (DQW) and use cases around the DQW such as automatic training example selection and quality assurance of semi-natural language, like command line parameters.

Second, the Model Quality section contains Explainable AI in Financial investments and the Industrial environment. This section also addresses an MLOps pipeline, model reproducibility and a use case regarding continuous monitoring of ML Models in the cyber security field.

Finally, Testing techniques for ML contains ML-assisted testing, testing learning algorithms, metamorphic testing, and testing model robustness. These testing techniques can be applied in finance, robotics and healthcare.

The novelty of the outputs is either entirely new tools or new approaches and methods to utilize existing tools.

The proposed new tools are:

- Techniques for credibility assessment based on knowledge graphs;
- Knowledge Graph-based XAI;
- Oracle-centred approach to evaluate learning algorithms for decision trees;
- Unit and performance test case generator;
- Model-agnostic structured XAI.

The proposed new methods and approaches, using existing tools in an innovative way are:

- Creating synthetic data;
- Data Quality Wrapper with automated data selection;
- XAI with Decision backed Neural Networks (DBNNs).

3. Introduction: Summary of the Deliverable 2.1. State of the Art

This deliverable is a continuation of the deliverable 2.1. (State of the art of validation methods and techniques for ML). The topics in the deliverable 2.1. that are continuing in the D2.2. are Explainable Artificial Intelligence, Synthetic Data, Metamorphic Testing and MLOps.

Explainable Artificial Intelligence refers to techniques that interpret the predictions of machine learning systems in understandable terms to humans. Various exchangeable terms are utilized to refer to XAI such as *Interpretable ML*, *Explainable AI*, and *Transparent AI.* The common goal is to improve the trustworthiness of AI-systems and provide clear explanations of AI decisions to a non-technical audience. In the deliverable 2.1. we addressed the fact that XAI is an evolving field with diverse methods and that it is a challenge to be classified into exclusive categories. Nevertheless, a brief reference list for the available implemented and frequently utilized XAI techniques and toolkits were introduced. Also, five model-agnostic algorithms were briefly introduced to provide an overall comprehension of how some of the advanced methods function. Those were:

- **Disentangled Inferred Prior Variational Autoencoder (DIP-VAE)** which is an unsupervised representation learning algorithm that will take the given features and find a new representation that is disentangled.
- **ProtoDash** which is an example-based data explanation method that understands the data through extracting prototypes.
- **Data Shapley** which is a data metric that exceptionally fulfils several properties of equitable data valuation where high Shapley values will identify the type of required samples to improve model performance.
- **Local Interpretable Model-agnostic Explanation (LIME)** which objective is to set a minimization function that fits a linear model locally to the original model that needs to be explained.
- **SHapley Additive exPlanations (SHAP)** which integrates LIME with Game Theory capabilities to provide a unique solution with fast computation.

Synthetic data has attracted lots of scientific interest in recent years. For example, in many medical applications like MRI, it is not practically feasible to solve the need for additional data by simply acquiring more images. The deliverable 2.1. concentrates mainly on improving a deep learning model's accuracy through data augmentation. Generative adversarial networks (GANs) are behind many examples that have received lots of public attention and various methods have already been developed for data augmentation purposes with medical data.

Metamorphic testing is a useful technique for ML-based systems because testing ML systems is more complex than other, more traditional systems. This technique is the creation of follow-up cases based on the already created or tested cases. It uses the available test inputs to create additional test cases and predict the output of these new cases. For example, if a function f has the property that f(x+2) = f(x) then metamorphic testing verifies that f(x+2) = y if f(x)=y. Any other output instead of y indicates the existence of errors in the system.

When testing this on a Ranking system, it was argued that instead of original values of attributes, their relative values determine the model. It follows that adding any constant value to every feature or multiplying each feature by a constant, should not affect the model performance.

Two more properties of metamorphic testing, inclusive and exclusive, can also be tested based on the domain knowledge and model's behavior. By adding a new element in the input, if an output can be predicted, the inclusive property is satisfied. Similarly, if excluding an element, the model's outcome can be predicted, the exclusive property is verified.

MLOps is an engineering practice that is mainly intended to apply DevOps principles to the development of machine learning systems and unify the ML development and ML system operation. It involves automation of all steps of ML development including integration, testing, deployment, and infrastructure resource management. Building an ML system is not a challenge itself, the main challenge is building an integrated ML system and operating it continuously in production. The use of MLOps tools presented in deliverable 2.1 include; AWS SageMaker, Azure, Databricks MLflow and Kubeflow.

4. Initial Version of Validation Methods and Techniques for ML in Project Use Cases

4.1 Data Quality

Data quality is an essential subject considering trustworthiness of ML systems. Data quality issues can be caused by many reasons, like missing data/values, outliers in the data or imbalance of the data. To respond to these data quality issues, we introduce methods for generating simulated data and validating incoming data in this chapter.

The main drivers for the simulated data are compliance and small training dataset issues. Both challenges are especially related to highly regulated industries that have sensitive data applications or if the data is limited, such as MRI data from specialised diseases. We introduce how to generate synthetic MRI image data for brain cancers. Another use case for the simulated data is the operational technology domain, where the data is highly confidential.

As a solution for validating incoming data, we introduce a tool by Sogeti, in collaboration with the University of Helsinki: Data Quality Wrapper (DQW). The phases of the DQW are Data Preparation, Data Description, Data Visualisation and Data Selection. The designated use cases for DQW are outlier detection by Philips Finland and quality assessment of semi-natural language by F-Secure. The third use case for incoming data QA is an ESG investment application, where we develop and implement tools to detect biased content and sentiment analysis, especially when introducing new sources and new content. The main goal here is to ensure the credibility of the sentiment analysis, so that we can also guarantee a decent explanation of the results.

4.1.1 Simulated Data for Healthcare

4.1.1.1 Ethics and compliance as a driver for simulated data

Acquiring Data for AI training is often slow and costly. Privacy requirements like GDPR restrict for example medical image collection and the data that is collected according to GDPR, cannot be used for other purposes than what was originally defined. This creates the need to investigate if simulated data could be created and used instead of real patient data in the training and validation of AI, and in research related to AI. This way, the amount of training data can be dramatically increased and better AI products with a less costly data acquisition phase can be realized. Also, collaboration with partners would be easier if there were no privacy concerns with the data. The training dataset may also have a limited representation of certain type of data cases, and with simulation tools, more such cases could be available (for example, simulating tumours in healthy volunteers).

With Simulated data, a basic problem is yet unresolved: when can the simulated data be different enough from the original data that was used as input in the simulation process so that GDPR no longer applies to the simulated data?

4.1.1.2 Results and experiences on generating synthetic brain MRI data

Synthetically generated MRI data would be very useful in e.g. AI model development, validation and quality check. In previous Deliverable 2.1 [18] we went through the state-of-the-art of generating synthetic brain MRI images with deep learning methods and back then we were not able to find any out-of-the-box solutions for the problem. Therefore, we started to browse and test publicly available methods to see which of them are the most promising and the best candidates for further development. Here we present some experiences and results from the approaches we took into closer inspection.

Data

We tested the methods using MICCAI conference Brain Tumour Segmentation (BraTS) Challenge 2020 data that is publicly available at the challenge website [1]. This dataset and its previous instalments are widely used in the literature and thus provide a good place to compare the methods, at least visually. The training dataset contains four different MRI sequence volumes (T1, T1ce, T2, Flair) from 369 patients with either high or low grade glioma (brain tumour). The dataset is described more in detail in [2] - [6]. See an example of an MRI T1 scan in **Figure** 2**1**.

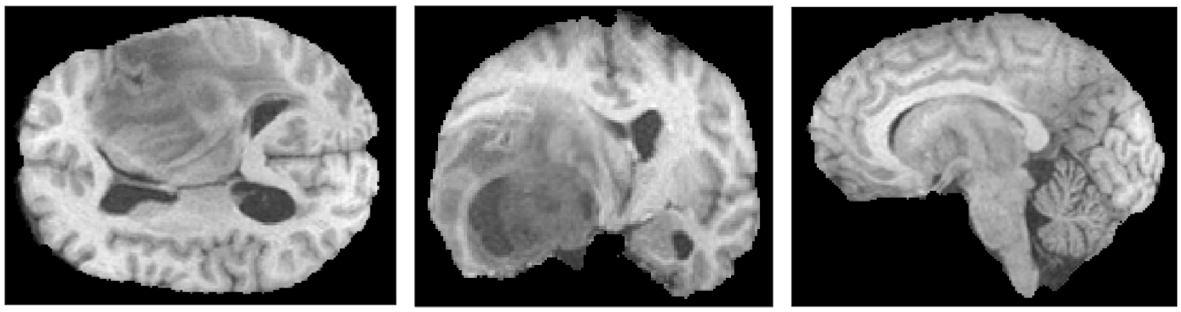

Figure 1. Example of a T1 MRI scan in the BraTS dataset from three orthogonal orientations

Translation from segmentation mask to MRI image

Although direct creation of random synthetic 3D MRI data is still under investigation, translation from one 3D volume to another using AI has given good results in many publications. For example, Yu et al [7] used edge-aware GANs to translate MRI T1 volume into MRI T2 and into MRI Flair volumes with good quality. There are also approaches to translate between different imaging modalities, for example Pan et al [8] successfully estimated positron emission tomography (PET) data from input MRI T1 sequence.

One reason why translation from one volume to another seems to work well is the local similarity of different volumes. MRI and CT contain the same anatomical structures and differ only on the textures and voxel intensity levels. Thus, this problem can be solved locally, i.e. the data can be processed in small patches instead of trying to fit the whole volumes into the physical memory of a GPU (or CPU).

The good results in volume translation motivated us to try if realistic MRI data could be generated by giving a segmentation mask as an input to a deep learning network. If this succeeds, it would be possible to then manipulate the segmentation masks according to our needs. For example, if small tumours on the left side of brain are under-represented in the training dataset, we could

easily shrink and move the tumour manually or automatically in the segmentation mask and generate corresponding unseen MRI data.

BraTS dataset contains simple manual labels for three different tumour phases (necrotic and non-enhancing tumour core, peritumoral edema and GD-enhancing tumour), shown in **Figure 2**. We added a whole brain mask from MRI T1 sequence as a fourth label to help the AI algorithm learn the overall brain shape.

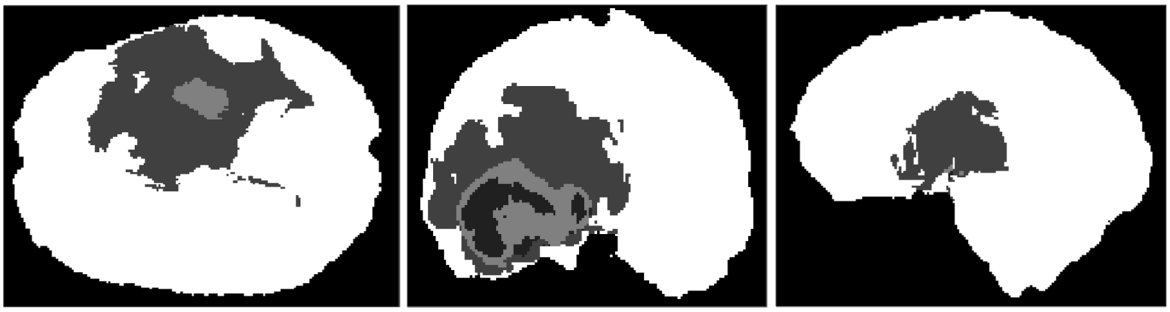

Figure 2. Simple mask for the brain and the tumour shown in three different orthogonal orientations. Grey areas show different tumour tissue types.

We set-up a customized three-dimensional UNET network to map segmentation labels to MRI T1 data. However, despite of tuning hyper-parameters and training the network with different inputs, we were not able to achieve realistic results. Although the output of the network shows general structure of a human brain (see **Figure 3**), it lacks all the necessary details such as sulci and gyri of the brain cortex. One reason for this is probably the lack of details in the input mask volume. We think this approach is worth investigating further though, by using more detailed input mask and a GAN-style network as a translation network.

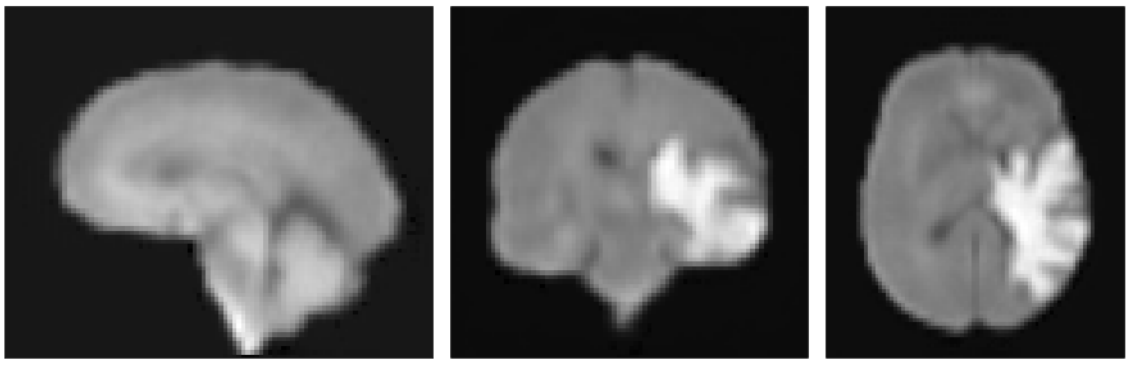

Figure 3. Results from trying to translate segmentation masks to MRI data

Style transfer from one or few patients

In this approach, we try to teach the AI network the specific texture and “style” of the brains of a single patient or of few similar patients. This is basically over-training the network to match the input data very closely and the idea is to transfer the specific style of one patient to the anatomical structure of another patient’s MRI data, and thus create new unseen data. The approach is motivated by [9] where only few patients were used in segmentation algorithm.

We constructed a 3D network with U-NET style architecture and trained it with a randomly chosen patient using segmentation mask volume as an input and the MRI volume of the patient as the

output. The network's architecture contained a narrow bottleneck layer in the middle so that network should not be able to memorize the input data completely. However, it was quite difficult to set the architecture and network training hyper-parameters so that the results would be good. Either the network learned the details of the training case(s) too well and the output is basically a copy of the training data, or the anatomical details were mostly missing from the output (see **Figure 4**).

Although this approach is intuitively feasible in some sense, in practice it is hard to set the parameters of the network robustly. Moreover, even if we managed to make this approach work, we would face privacy issues because it is possible that the model records the training data too accurately. Therefore, this approach will not be our primary direction to continue in the future.

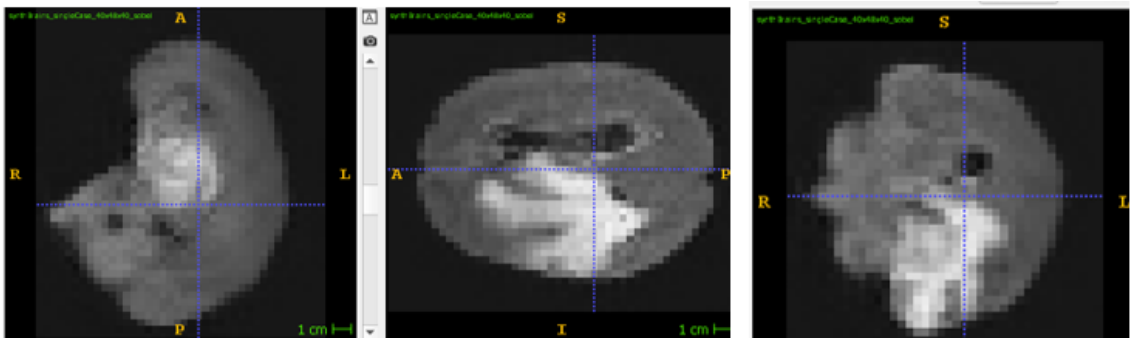

Figure 4. Output of a style transfer from one patient to another

Vox2vox

Pix2pix [10] has been used e.g. to create very realistic facial 2D images that are practically impossible to detect from real photos by eye. The 3D extension of this method is named as Vox2vox [11] and there are some implementations of it available publicly. We tried two public implementations available at github.com and also created our own implementation. However, none of these implementations provided good results (see **Figure 5** for a single example). Probably the amount of cases in the training dataset is simple too small for this kind of algorithm. For example, with facial 2D images it is possible to use millions of images in the training set but with our brain tumour dataset we have 369 patients. So, we conclude that this algorithm is not worth investigating any further with this dataset.

Figure 5. Example of an output of Vox2vox network

Generating only tumours

As one way to ease out the problem of generating realistic human brains with all the details is to use GAN networks to generate only the tumour. There are multiple aspects why this should be easier to solve, e.g. the needed number of voxels is considerably lower within a tumour than within the whole brain, there are no fixed anatomical structures within the tumour that need to be learned and the tumour can be in any orientation enabling better data augmentation in AI network training phase.

We extracted only the tumour data from the BraTS dataset and built a simple 3D GAN network. The data resolution was set to 64×80×64 voxels for faster processing and training data was augmented with random rotations around all three axes. An example of the results can be seen in **Figure 6**. At least to an untrained eye the generated tumours look quite realistic, although a bit too smooth. At least the generated tumours seem to contain more details than the GAN model output of the whole brain.

However, the problem is that even if we replace the real tumour with the synthetic one in MRI data, the rest of the MRI volume still remains the same. The data around the tumour contains all the information from which the patient can be identified. Thus, this approach will not help much from the privacy aspect of the data. The method itself seems to work ok, so we will keep on investigating further possibilities with this.

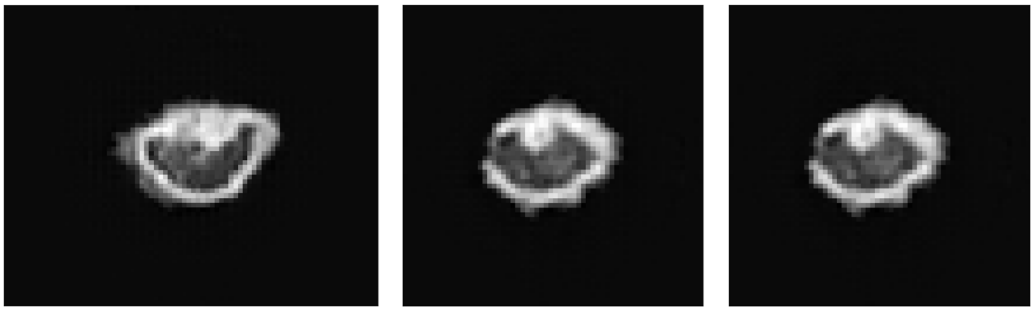

Figure 6. Example of a synthetic tumour generated with a GAN network from orthogonal dimensions.

Assisted AI

It is a common practice with GAN methods that the synthetic data is generated from a random vector. Thus, it is assumed that the AI can learn everything it needs to know in order to generate realistic output from the training data. However, the overall shape of the (healthy) human brain is very well-known a priori i.e. every human brain has two hemispheres, cerebellum, two hippocampi and so on. It is kind of a waste of resources (AI network power) not to give this prior knowledge as an input. Thus, we have started experimenting with a network that takes anatomical atlas of a brain as one of the inputs as an additional channel.

The anatomical atlas we use is the MNI Colin atlas [12] shown in **Figure 7**. The atlas is based on multiple (27) scans of a single subject and is used in numerous previous studies. We registered the atlas to the patient brain in BraTS dataset using classical machine learning techniques i.e. non-rigid registration with mutual information metric.

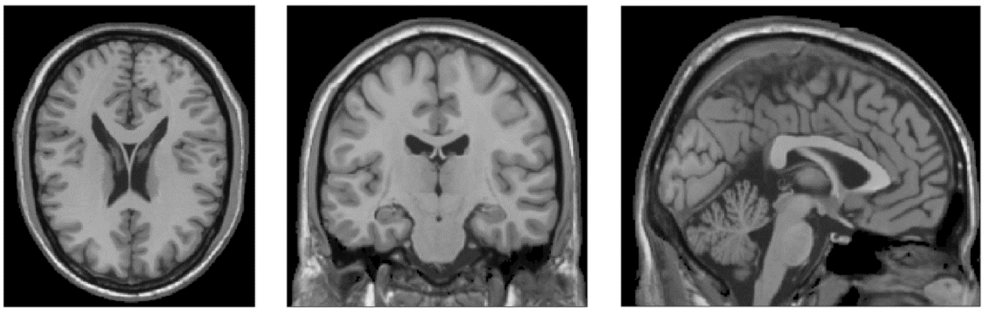

Figure 7. MNI Colin atlas from three orthogonal dimensions

According to our initial tests with 3D convolution networks, using this atlas as one of the input channels improves the amount of output details a bit and may be worth further investigation. It is again tricky to set the network architecture and hyper-parameters right, so that the output is not just an elastic deformation of the atlas but something more genuinely like new data. Including the atlas may not be enough to solve the synthetic data problem by itself, but we will keep this trick in mind for the work in future.

State-of-the-art update

Since our previous state-of-the-art review, new papers have emerged constantly, and synthetic 3D medical data seems to a relatively hot topic. Most of the publications dealing "*MRI synthesis*" deal about translation from one imaging sequence or modality to another, but also papers about generating truly unique synthetic data without input volume seems to be arising.

First, Eklund et al [13] presented a method to generate realistic looking MRI data with size 64×64×64 using a 3D progressive GAN. These volumes were generated from a random vector and thus may not be that useful e.g. in improving segmentation algorithms because the generated volumes lack corresponding "ground truth" segmentations. Then Eklund continued his work together with Foroozandeh [14] by combining multiple steps using BraTS brain tumour dataset. Because BraTS datasets contains only tumour segmentations, FSL software ([15]) is used to add more detailed anatomical segmentations to the dataset. Then progressive GAN [16] network is used to generate new synthetic segmentations according to the BraTS segmentations. In the final stage, they used semantic image synthesis method, SPADE [17], to convert generated 3D segmentations into actual synthetic MRI data. The results (see **Figure 8**) are impressive compared to previously published efforts. However, this work is done on 2D slices instead of 3D volumes.

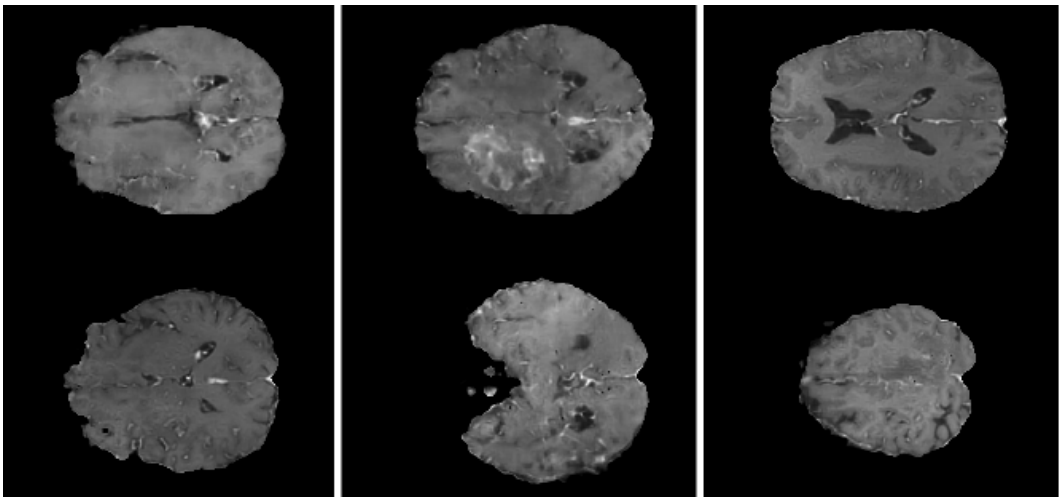

Figure 8. Example synthetic brain slices taken from [7].

Future plans

As Eklund's three step method seems to provide the best synthetic MRI data so far in the literature, albeit in 2D only, we will start following his footsteps. First, we are aiming to expand this approach to 3D. It can be expected that the hardware requirements increase and therefore we have already started collaboration with parallel computing experts at Techila Technologies. We are already running experiments on VTT's computation servers but for large scale tests more computer power and distributed approaches are needed. We are also investigating possibilities to harness the massive computational capacity of CSC – IT Center for Science, Finland, into this problem.

As the synthetic data seems to be a hot topic currently, we will continue monitoring the state-of-the-art constantly. We also had a (virtual) meeting with Philips Netherlands to compare our approaches and share our thoughts about the case. Although our approach is completely data-driven and theirs is based on very detailed simulation of the MR device and the full imaging process, there may be common topics as well in the future. We will also still investigate the option to include anatomical atlas, such as Colin27, into the model to help the network find the details even better.

When we are able to generate synthetic data, we can also start testing the applicability of synthetic data in improving and validating AI algorithms. We will pick a publicly available brain tumour segmentation method designed for BraTS data, train the algorithm further using synthetic data and see if the results are improved.

4.1.2 Simulated Data for Cybersecurity

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 of the aforementioned 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.

The plan is to build 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 should make 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.

4.1.3 Incoming Data QA

4.1.3.1 Data Quality Wrapper

The basis of a transparent and reliable ML project is making sure the quality of the data that goes into a given ML model is suitable and representative. This spans over all industries that apply such models, including healthcare, cybersecurity and finance. Exploratory Data Analysis (EDA) is a group of tools and techniques used for data validation. For more information on this topic, please review ITEA IVVES D2.1 (chapter 3.2.1) [18].

Sogeti's proposed innovation will be integrating SOTA tools and methods, together with data preparation and data selection techniques, in collaboration with Helsinki University, into a **Data Quality Wrapper** (DQW*)* to be used in the initial, Data Understanding and Data Preparation phase of The Quality AI Framework [19] for an efficient and transparent AI model development cycle. The DQW can be applied to different data sources.

The phases of the DQW are (**Figure 9**):

- Data Preparation;
- Data Description;
- Data Visualisation - Understanding relationships and new insights through plots,
- Data Selection based on step 2 and 3:
    - Handling missing data;
    - Handling outliers;
    - Data sampling and handling imbalanced datasets.
    - Automatic selection of data samples, collaboration with Helsinki University.

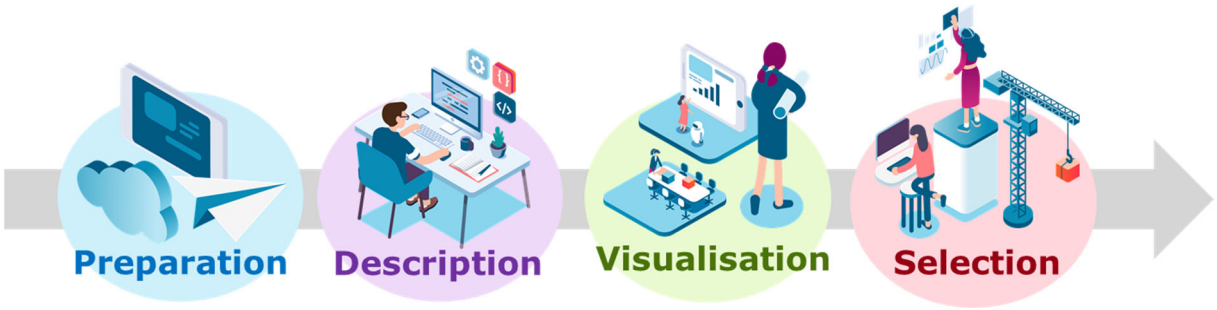

Figure 9: An illustration of phases in the Data Quality Wrapper.

During the ITEA IVVES project, we will be reviewing tools and techniques deployed to evaluate the following data formats or types:

- **Structured** - tabular data is the most common data format used in data science, be it in finance, health, biotechnology, cybersecurity, etc. It is very important to understand structured data contents before starting any data selection operations on it.
- **Unstructured** data:
    - Images - Images are used in computer vision algorithms; this use requires image annotation be performed before feeding the data into the model.
    - Text data - used in NLP models, be it for classification or sentiment analysis. The format it can come in varies, it can be a part of a tabular data file, or just a .txt file. Depending on the delivered data format, EDA steps vary. In case of .txt files being delivered; the data can contain a lot of noise and needs to be cleaned prior to analysis.
- **Synthetic Data** - Synthetic data evaluation is a critical step of the synthetic data generation pipeline. Validating the synthetic data training set to be used in a Machine Learning algorithm ensures model performance will not be impacted using synthetic data, and it, most importantly, answers the question of how representative the synthetic dataset is [20]. In order to evaluate synthetic data, you also need a portion of real data to compare it to. Depending on the format of synthetic data, same methods explained in DQW phases can be used.

Table 1 carries information on the Python packages and modules that will be used as part of the DQW for automatic EDA and data selection of different data formats, ensuring automatic and efficient way of cleaning various input data formats, describing them and selecting training data for the ML model.

Data Format	Preparation	Description	Visualisation	Selection	Python Package	Reference
Tabular	✔	✔		✔	Pandas	[21]
Tabular		✔			Numpy	[22]
Tabular			✔		Seaborn	[23]
Tabular		✔			SciPy	[24]
Tabular		✔		✔	PyOD	[25]
Tabular		✔	✔		Pandas Profiling	[26]
Tabular Image	✔				Matplotlib	[27]
Image	✔				Pillow	[28]
Image	✔				OpenCV	[29]
Image		✔		✔	Alibi Detect	[30]
Image	✔				basic-image-eda	[31]
Text	✔				html.parser	[32]
Text	✔				Codecs	[33]
Text	✔				NLTK	[34]
Text	✔				spaCy	[35]
Text	✔	✔		✔	TextBlob	[36]
Text	✔				CountVectorizer	[37]
Text		✔	✔		WordCloud	[38]
Text		✔		✔	TextStat	[39]
Synthetic tabular		✔	✔		SDMetrics	[40]

Table 1. Existing Python packages and modules used in DQW.

Data Preparation

In order to perform EDA, we need to prepare incoming data. This phase is important as it gets the data from a given source and pre-processes it for python functions deployed in the DQW. This step is crucial for unstructured data, as it may be noisy coming from a data source and, thus, difficult to analyse without adequate preparation.

Based on the data format at hand, we separate the following Data Preparation techniques:

1. Structured data – tabular data:
    a. Load and prepare data using Pandas.
2. Unstructured data - Image content specific EDA and data preparation - reading images, converting, and scaling images, computing derivatives, plotting or saving results.
    a. Use Image module from of the package Pillow or Python Imaging Library (PIL) with numerous modules and functions to prepare image data.
    b. Image De-Noising [41], the process of removing image noise while at the same time trying to preserve details and structures. For this, use OpenCV package functions.
3. Unstructured data – text:

a. Clean up data, get the clean corpus (collection of linguistic data) by escaping HTML characters with html.parser Python package, decode data with Codecs package, remove stop words and punctuation. Furthermore, normalize text through stemming and lemmatization with NLTK.
b. Create a Document Term Matrix (DTM) with Sklearn package module CountVectorizer in Python. DTM provides the frequency of a word in a corpus. It helps in analysing the occurrence of words in different documents by converting text data into structured format.

Data Description

In this phase we use statistical analysis of data to gain insights into data assumptions. Statistical methods explained below will be used to understand the incoming data, highlight missing entries and isolate possible outliers.

1. Structured data - Explore features of a data frame loaded using phase 1.
    a. Quantitative measures – number of rows and columns.
    b. Qualitative measures – column types.
    c. Descriptive statistics with NumPy for numeric columns, for example, count, mean, percentiles and standard deviation. For discrete columns, count, unique, top and frequency.
    d. Explore missing data with Pandas functions is.na, is.null, etc.
    e. Examine outliers [42]:
        i. Mathematically determine with SciPy stats.zscore, which shows the signed number of standard deviations by which the value of an observation or data point is above the mean value of what is being observed or measured.
        ii. Mathematically determine using NumPy function quantile, which computes the q-th quantile of the data along the specified axis. We can also calculate the inter-quantile range (IQR), the first quartile subtracted from the third quartile.
        iii. The PyOD package is quite useful here and employs multiple algorithms for automatic outlier detection.
2. Unstructured data.
    a. Image format metrics are metrics associated to unstructured data itself. A proposed package in Python for describing images would be basic-image-eda, a simple multiprocessing EDA tool to check basic information of images under a directory. This package allows for very simple EDA of images in a directory and displays number of images, channels, extensions, mean values, etc.
    b. Annotation Metrics - image annotation is the process of building datasets for computer vision models [43]. This enables machines to learn how to automatically assign metadata into a digital image using captioning or keywords. This technique is used in image retrieval systems to organize and easily locate images from a database. The EDA at this step would include tabular data format since image annotations are structured.
3. Unstructured data – text. Use clean text files and assess:

a. Frequency - Count most common words with WordCloud package in Python. This is the quickest way of seeing what the handled data contents are, in addition, it provides visualisation in form of word clouds.
b. Analyse sentiment with TextBlob in case of classification tasks. We can investigate the polarity of the text and represent it in form of bar graphs.
c. Investigate readability of data with Textstat, typically used for determining readability, complexity, and grade level of a corpus.

Data Visualisation

This step includes visualisation of data description results we obtain in previous step. The structured data visualisation techniques can also be deployed for TDM and image annotations as they are also a form of structured data.

1. Structured data - Examine data distributions with Seaborn functions boxplot, scatter, etc.
2. Unstructured data – images.
    a. Visualizing image contours and plotting image histograms with Matplotlib.
    b. Annotation metrics, distribution and correlation of variables.
3. Unstructured data – text. Visualise word clouds with WordCloud package.

Data Selection

The final phase of EDA is using the insights gathered in previous phases and selecting representative data. This phase is closely connected with the Data Description phase as we will be using the results from statistical analysis done during that phase to select the data.

1. Structured data.
    b. Handling outliers is one of the most important steps in statistical analysis of data. A method of detecting outliers has been explained in Data Description phase. This phase of the DQW removes the outliers.
    c. Handling invalid and inconsistent data. These can lead to issues down the line and may need to be removed.
    d. Handling missing data, which can lead to issues in the training phase, can be done by removing the missing rows, dropping columns heavy in missing data and filling in missing values with mean.
2. Unstructured data – images.
    a. Handling outliers with a Python open-source package Alibi Detect, which offers multiple deep learning algorithms for outlier detection in image data.
3. Unstructured data – text.
    a. Evaluate most common words and remove data sources that give too many positive and positively correlated words.
    b. Evaluate data complexity and remove sources with insufficient data quality.

4.1.3.2 Automatic selection of training examples

Curating and labelling real-life data for supervised machine learning tasks requires human labour and is therefore expensive. Suppose we are given a small initial set of reliable, labelled training data, and a larger set of less carefully labelled and potentially unreliable data. The goal of this work is to automatically choose which examples from the larger set we should include in our training data. We have the option to use a human expert to make judgments, but the objective is to minimize the expert work needed.

In parallel, we seek to understand that if we, in a continuously learning system, perform a self-supervised retraining step, can we automatically detect anomalous training examples and omit them as well as choose the most useful set of data for retraining.

This work is aiming to serve the Philips Finland use case and is done in collaboration with Philips and VTT. The work has produced some early results so far in simple examples; in the next phase of IVVES, we seek partners' problems and data sets where we could investigate this.

4.1.3.3 Quality Assurance of Semi-Natural Language Data

With a lot of research been done in the field of natural language, there are various methods to assess the quality of text data. To check the quality of text data in English language, stop-words, such as 'a, an, the' etc. can be used to identify the articles, words can be stem to their basic form, punctuations can be detected, and the overall quality of text is verified. When we move from English to other languages such as German, Spanish, French, there are various tools and libraries, which can capture the topical signals of the textual data. With the advancements in AI, and increase in the popularity of NLP, tools and libraries for processing and verifying the quality of language data will always be emerging. On the other hand, data of a semi-natural language is yet under study.

One such data is Command-Line commands. Since this data is not a standard natural language, any standard natural language processing tool, will not be a likely option to work with. Until we build a tool or library for the semi-natural textual data, we will always be looking for meta-data or some reference data, such as manual pages for the commands. The two most used Operating Systems (OS), are Windows based and Unix based. Since the Windows based OS is a proprietary OS, there are officially maintained documentations of the Windows commands available. On the other hand, Unix based OS, such as Linux, is an open-source OS, Linux commands' manual pages are maintained as open-source web projects. To assess the quality of commands data, these documentation or manual pages are the most important sources.

To understand the domains or subfields where semi-natural text data is used, we can take the examples of cyber security applications. Most of the organizations with their expertise in the field of cyber security get commands data continuously. To detect the usual commands and threats, to understand the user's behaviours while working with commands, to learn the prevailing commands and parameters, these organizations need to analyse this semi-natural textual data. As mentioned in the above paragraph, one main source of commands data quality is the manual pages of the commands.

By web scrapping the commands documentation, we can collect the description of the commands, syntax of the commands, and parameters of the commands. Then the commands are compared against each other, to find the similar commands, to find the common parameters, and to create clusters of the commands, which are from the same domain. To understand it better, we can take the example of Windows commands, 'erase' and 'del'. By comparing the description of these two commands, it can be verified that these two commands are used for the same purpose, to delete files or directories. By comparing their parameters, it can be observed that they share the same parameters. Now without this reference data, when a cybersecurity application receives these two commands with the same parameters, it can identify one of them (let's say 'erase') as an unusual command which is following the same pattern and parameters of another command (which is 'del'). However, with the pre-analysed results, these two commands can be tagged as normal commands.

Similarly, to build a machine learning algorithm which learns the commands, their parameters, their syntax, these manual pages are useful. It can be seen that some of the commands belong to a bigger cluster of the commands which are of the same domain, such as 'bitsamdin *' commands, 'manage bde *' commands, 'logman *' commands for Windows OS, and 'lttng-*' commands, 'semanage *' commands, and 'systemd-*' commands from Linux OS. These commands share the same pattern, mostly the same set of parameters, so for any cybersecurity application to differentiate between these commands, the analysis of these manual pages is a pre-requisite, to not only comply with the quality of the data but also verify the syntax and semantics of the incoming data.

Another scenario of the semi-natural textual data is for the programs or applications which need to put checks, rules and conditions on the incoming data quality. With the web scrapped manual pages, similar commands are clustered already. These clusters can be used to save time and resources for building the quality assessment tool, such as for dozens of 'bitsadmin' commands, there is no need to put a set of rules and conditions for each command separately. Similarly, for the commands, which share the same parameters, an ML algorithm can be used to identify the valid commands and unusual commands based on their parameters to verify the quality of the incoming data.

This discussion was just for the quality assessment of incoming data of semi-natural language. There are a lot of possibilities and wide-open research areas in the field of semi-natural language, which will be identified with the passage of time. The more we build mature and sophisticated tools for natural language processing, the more we will extend our research towards semi-natural language.

4.1.3.4 QA for text driven ESG investment systems

Introduction

The Natural Language Process (NLP) is changing the way companies understand text. The irruption of game-changing innovations and open-source technologies has boosted the state of the art in the NLP field. Unstructured text is being used as input data for many industrial domains (i.e. predicting market trends, changes based on sentiment analysis, impacting production lines

and logistics). Its combination with other AI techniques applied to numerical data sources is the next breakthrough and will foster the integration of NLP into regular Data Analysis. Data Analytics companies are curating and collating information from diverse sources (including publicly available sources through the internet) to feed AI models [44] [45] [46]. In this context, the investment research market is facing seismic shifts, with 70% percent of companies using alternative data [47]. Given the vast amount of data and information available, that analysis can only be reliably carried out with artificial intelligence.

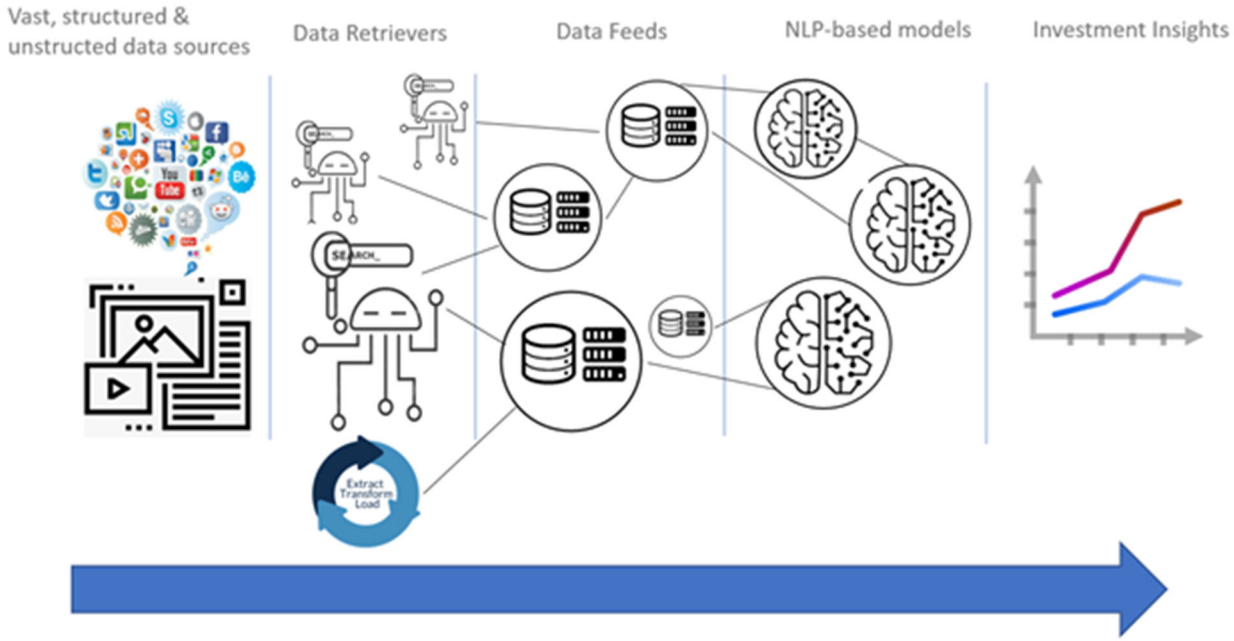

Figure 10: AI-based process for Investment Insights

However, the growth of AI-based analysis of sources for non-financial (particularly ESG-related) information, has also impacted the way that companies communicate with the external audience, being savvier with their wording [48]. This is causing the appearance of biased content, that must be taken into account before applying NLP-based techniques -heavily relying on sentiment analysis- to get insights and trends. Hence, a systematic and continuous analysis of source credibility and content credibility must be implemented.

Current Status Techniques for QA in ESG-related solutions

Until recently, the most extended approaches were based on creating a set of weights for key, curated sources (reports, corporate websites, NGOs news and social posts, vertical news and mainstream news…). However, given that ESG investment is solid trend that is increasingly impacting the market, the sources to analyze are growing fast, together with an increasing risk of wrong ESG scoring due to weak credibility assessment.

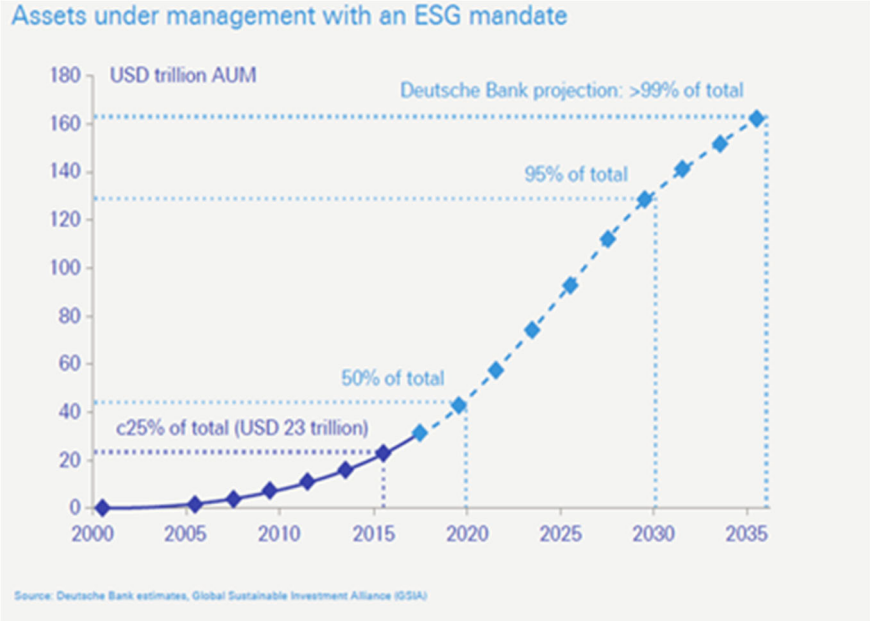

Figure 11: Assets under management with an ESG mandate

In a more general way, this problem is quite related to fact-checking, that is being supported by a dense network of volunteers. This work is currently supported by Fact Check Tools [49], as the ones provided by Google. However, there is a lack of tools and techniques to properly analyze new sources and new content for ESG investment.

Roadmap for QA in text-based models

SII CONCATEL & NETCHECK (CCTL/NC) are exploring the analysis of ESG-related content and sources' credibility. For this, CCTL/NC are designing a structure that will be implemented in a Knowledge Graph (KG). This KG is being fed with:

- Reports.
- Corporate websites.
- NGOs news and social posts.
- Vertical news and mainstream news.

Linking sources, content and authors, and also containing the content (text) as a property of "content" nodes. This KG will have manually generated labels related to credibility of sources and content.

Four main approaches are being implemented to provide a classification of new sources and content, with respect to credibility:

- Basic Content-based NLP approaches: based on title or headline, content and domain.
- Graph Embeddings [50] [51] [52]: as an output, we will provide a solution that, given an article, the level of credibility will be shown.
- Structure Analysis: Based on the analysis of the structure of links (source, content, author) and analyzing differences between credible vs non credible structures.
- Text/Graph to Image: To complement these approaches, we want to explore how image data QA could perform with text2image approaches. This is currently in a conceptual stage, but some approaches [53] [54] are being considered. This could help both for data quality and explainability.

The novelty is relying on two main concepts: first, the generation of techniques for credibility assessment based on knowledge graphs is a very recent topic, and all the tools, techniques and resources are mainly focused in the political domain. In IVVES, a generalization of methods will be performed, implementing afterwards a specific workflow for ESG-related environment. Second, this workflow will be based on a set of tools and solutions that will be combined (Basic Content-based NLP approaches, Graph Embeddings, Structure Analysis & Text/Graph to Image approaches) to provide enhanced credibility assessment tools and detect any bias in content and sources. These tools will be linked to the tools and workflows generated in WP3 and WP4, putting all together in WP5; generating a framework that may also integrate solutions from third parties.

4.2 Model Quality

The following sections describe novel XAI approaches that can be used to provide both interpretability and explainability required to meet the regulatory technology requirements when operating in the financial domain. Additionally, a use-case from the industrial sector is presented where the application of XAI techniques will be investigated to improve the interpretability of the output generated by the ML-based system.

The following sections also describe key building blocks of an MLOps pipeline and how MLOPs practices can be used to improve the workflow in situations where concerns such as reproducibility and workflow automation are relevant. The application of these approaches will be investigated by using the requirements of a use-case from the cybersecurity domain as reference. Additionally, custom triggering mechanisms related to concept drift detection will be investigated.

4.2.1 Explainable AI in Financial Investments

The field of Financial Investments faces the same technological challenges regarding system reliability, resilience, robustness, and security as other domains. In addition, the systems in place need to meet regulatory requirements concerning the auditability, transparency, and explainability of ML/AI components. This, combined with the growing trend regarding ESG-focused (Environmental, Social, and Governance) investment means the explainability and compliance aspects need to be extended to these emerging areas. Investors are not only focused on the profitability of their investments, but also on "how" returns are obtained.

The ESG boom has not been accompanied by an official roadmap on what is considered responsible investment and what is not, and although the European Commission has published a report on Taxonomy, there are differences of criteria between the different providers evaluating ESG investment scores.

It is here, where Evolving Systems are making the difference, establishing a continuous analysis on different data sources, including non-standard sources within the financial field (social networks, corporate data, etc.), for the inference of changes in the context. As previously explained, this analysis is heavily relying on NLP-based approaches. These approaches may provide not only good scoring for ESG-investment, but accurately forecast trends in different domains [44][45][46]. This will literally change the investment market [47][48].

It is also mandatory to provide explainability at three main levels: customers, experts, and regulators. Home Offices, Securities Agencies and Fintech companies are demanding effective tools to assess and validate how the ES provides a recommendation or insight. Hence, it is required to interpret the predictions of the system in understandable terms to humans. The lack of auditability, transparency, and explainability of ML-enabled systems' results are limiting compliance with regtech (regulatory technology) constraints and a bigger impact on the sector.

At this stage, we are currently working on two main concepts:

- LIME for NLP: as a first step to provide a basic explanation of the insights and conclusions generated by the system. The goal is to explain, given a security and a forecasted ESG score, why and how this score has been concluded. This is related to two main processes:

credibility assessment and ESG value derived from NLP analysis (heavily relying on sentiment analysis). This should allow an expert and regulator to understand why a set of news and reports are impacting a change in the ESG score of a security.

- Knowledge Graph-based XAI: KG have the potential to provide context, bridging the gap between automated logical reasoning and machine learning [55]. As a first step SII CONCATEL is working on the analysis on the approach defined by LU, Yi-Ju et al. [56], and willing to extend the concept beyond social media, to be applied to the incoming data, once the QA has been performed. It is expected also to take advantage of the context to generate reports, based on the latest approaches [57] [58]. The novelty relies in generating a system that can not only be validated at a theoretical level, but one that also complies with the requirements to be provided as an industrial solution, therefore, taking into account reliability and performance.

In addition to these concepts, other XAI approaches using a hierarchical explainable AI to achieve model transparency similar to decision based neural networks will be explored as discussed in the following chapters.

4.2.1.1 Hierarchical Explainable AI

There are several ways to provide insights into how otherwise a seemingly black-box model sees the input data as discussed above. Perhaps the most typical explanations would be *global explanations* where the relevance of each input feature is visualized as an ordered bar chart. Such chart provides a quick outlook on what features the model utilizes in its predictions, and how relevant the features are. This kind of plot gives significant insight into what is important for a model to perform its predictions but does not explain *how,* on average, the model utilizes the features.

In order to scrutinize how the features contribute, on average, to the predictions, *partial dependency plots* are of assistance. Partial dependence plots depict how individual features and in which feature range contribute. This is typically conducted by plotting each feature separately and having feature range as the x-axis and the contribution on the y-axis. The partial dependency plots provide insights on how the model maps the independent variables to the dependent variable(s) and provide an opportunity for model sanity checks: feature contributions being highly volatile would give a hint about the (lack of) stability and robustness of the model. Additionally, partial dependency plots enable to reflect if the model makes sense if one knows the problem domain. Hence, partial dependency plots can be considered essential when analysing the model and, sometimes, to provide insights about the process to be modelled.

In addition to the global explanations, local explanations can be utilized to better understand how the model has treated samples individually. This is, typically, performed by depicting the contributions of each feature to the dependent variable as a bar chart. Thus, there would be as many bars in the chart as there would be features. For time-series, rolling time-series contributions can be utilized, and there would be as many lines as there would be features.

Even though there are XAI methods for depicting both global and local explanations, the problem is that the widely available XAI means tend to depict the explanations as a *flat vector* without structure. As discussed in Chapter “Model transparency with decision based neural networks”, in images one typically utilizes a *saliency map* to illuminate parts of images to show what sections seem to attract the model, but that cannot be considered as an explanation on *what* special there

is in the illuminated area. It is a similar case in Natural Language Processing (NLP) where parts of text are typically highlighted to show what parts were relevant and how the words and sentences contributed to the predictions. Even although this kind of approach provides the first steps for sanity checks i.e., to see the model is not attracted to something clearly irrelevant, explanations remain lacking.

Images and NLP have in common that the feature vectors tend to be big. In images, the features are pixels, and in NLP, the features can be originated from one-hot-encoded words for instance. Additionally, wide feature vectors are prevalent in numerous other domains in addition to images and NLP. Therefore, *flat* explanations are ubiquitous. One cannot consider tens, hundreds or even more bars in a bar chart or hundreds of graphs in partial dependency plots as an explanation neither illuminated parts of images: the information is there, but it is hiding in plain sight.

As discussed in Chapter “Model transparency with decision based neural networks”, Neural-Backed Decision Trees (NBDT) has been proposed. The idea is to depict the explanations hierarchically, which is easier for humans to understand than the *flat* representation format. However, even though NBDT is an interesting approach, it is utterly dependent on the underlying model: the artificial neural network (ANN). Although ANNs can be considered as universal approximators, ANNs cannot be considered as the most suitable models for every problem: one can utilize ANNs for all cases, but one perhaps should not. Thus, there should be a model-agnostic method for hierarchically depicting the explanations.

4.2.1.2 Towards Structure Imposed Explainable Artificial Intelligence

The objective is to transform *flat* explanations into a hierarchical structure like as discussed in Chapter “Model transparency with decision based neural networks” and in [59] but instead of resorting to neural networks or something alike to Naïve Bayes, we would implement a model-agnostic method to impose structure to the explanations. Additionally, we would want to interfere with the prediction model as little as possible so that the method can be applied even to the existing models. Thus, one should utilize model-agnostic method for explanations and later utilize another method to impose the structure to the explanations.

Now, the hierarchical structure can be imposed either driven by

1. data solely,
2. domain-knowledge, or
3. hybrid of the aforementioned.

Data-driven structure would be about scrutinizing the explanations and forming the structure solely from that basis. Utilizing domain knowledge is about using prior information as conducted in [59] in order to have a hierarchical structure which is specific to the domain in question. The hybrid would then be about, for example, imposing principal structure based on the domain knowledge and leave the lesser nodes to be driven by the data, or by adjusting the whole structure like in Bayesian reasoning where a priori information is adjusted by data. In this section, we limit the discussion only to the first approach.

The tentative idea is to utilize the existing well-known algorithms for the task, namely Shap [60] for explanations, and agglomerative clustering for hierarchical structuring the explanations. The tentative algorithm fitting procedure idea is as follows:

1. Fit the primary prediction model by utilizing the training-set. The model can be any model explainable by Shap.
2. Utilize Shap to explain the training-set by utilizing the primary model.
3. Hierarchically cluster the Shap explanations by utilizing agglomerative clustering. The result is a dendrogram of the explanation structure.
4. Fit a logistic regression model per each branch in the dendrogram with the data which fell into the branch in order to later utilize the classifiers during the prediction-time. We call these models as Branch Classifiers (BC). The result is a set of BCs which are specific to each branch.

Now, as the clustering is conducted in an unsupervised fashion, the meaning of the branches is to be deduced. The meaning of the branches requires either manual inspections of the samples which fell into those branches or utilizing advanced means to assist in tagging of the branches. Discussion of these advanced methods is omitted in this section.

The reason to utilize a mere logistic regression as a BC model is not only due to the performance reasons but for simplicity, too. The explanation model cannot be more complex than the primary model as otherwise the BRs should also be explained. Additionally, as the Shap values of different features are on the same scale, the magnitude is reflected in the coefficients of the linear model. Inspecting the coefficients of such a linear model assists in deducing the meaning of the branches if no advanced means to assist branch tagging is available.

The following is the procedure during the prediction-time:

1. Utilize the fitted primary model to predict the dependent variable.
2. Get the Shap explanations of the predictions.
3. Utilize the pre-fitted BCs to form the hierarchical structure. Take the most abstract BC, predict the branch of the explanation, take the next BR specific to that branch and work downwards until no more BCs are left. The result is the hierarchy through which the explanation went through. The hierarchy is the final explanation aside to the raw Shap values, which can also be utilized aside to the hierarchical representation.

As a result, there should be a hierarchical structure for the explanations. The branches should have some meaning, but as the structure is learned in an unsupervised fashion, there is no guarantee that the structure has semantic importance. For guaranteed semantic meaning, Domain-Specific Structure Imposed eXplainable Artificial Intelligence should be utilized but further discussion of such discussion is omitted in this section.

4.2.1.3 Model transparency with decision based neural networks

As machine learning and AI applications sees growing adoption in industries that have many sensitive, ethical and/or social implications i.e. finance and medicine, justifiable and transparent predictions become increasingly important in machine learning implementation. The main reasons for model interpretability are to:

1. Convince the domain expert and/or end-user that the prediction is trustworthy
2. Discover undesirable model biases and optimise model quality and fairness to mitigate those biases
3. Minimise negatively influenced decisions from a faulty model or invalid justification

Many computer vision applications in these domains (e.g. medical imaging) require high accuracy as well as insight into the model decision making process in order to uncover any ethical or negative social violations. However, there is an accuracy-interpretability dichotomy for classification problems. Classic decision trees with deep learning usually sacrifices interpretability to maintain accuracy or vice versa - underperforming accuracy to maintain interpretability [61].

Explainable AI (XAI) methods attempts to bridge the gap between accuracy and interpretability by justifying predictions but **without interpreting the model directly**. Generally, XAI can be grouped into two types of justifications – saliency maps and sequential decision processes. Saliency maps aim to explain features that impacted the model's prediction by identifying the pixels that most affected the prediction; however, the maps focus on the input and do not necessarily explain the model's decision-making process.

Alternatively, we can use sequential decision methods like classic decision tree models, to break up predictions into a sequence of smaller semantically meaningful decisions, offering insight into the model's decision process. However, fusing deep learning and decision trees can suffer from significant accuracy loss to maintain interpretability (the accuracy-interpretability dichotomy).

Thus, to produce high-accuracy, interpretable models that explain high-accuracy neural networks, we combine neural networks with decision trees to form **Neural-Backed Decision Trees** (NBDTs). NBDTs are interpretable as decision trees and can output **intermediate decisions** for a prediction. This hybrid design addresses the failures of traditional neural networks to provide justification and interpretability, while preserving high accuracy. [62]

A NBDT is a hierarchical classifier that [63]:

1. Uses a hierarchy derived from model parameters, this is to avoid overfitting
2. Can be created from any existing classification neural network without architectural modifications
3. Retains interpretability by using a single model, sequential discrete decisions, and pure leaves

NBDTs are as interpretable as decision trees. For example, given an image, a neural network may output *Dog*. However, an NBDT can output both Dog, Animal, Chordate, and Carnivore. Giving insight into how the model came to its decision. See **Figure 12** for example.

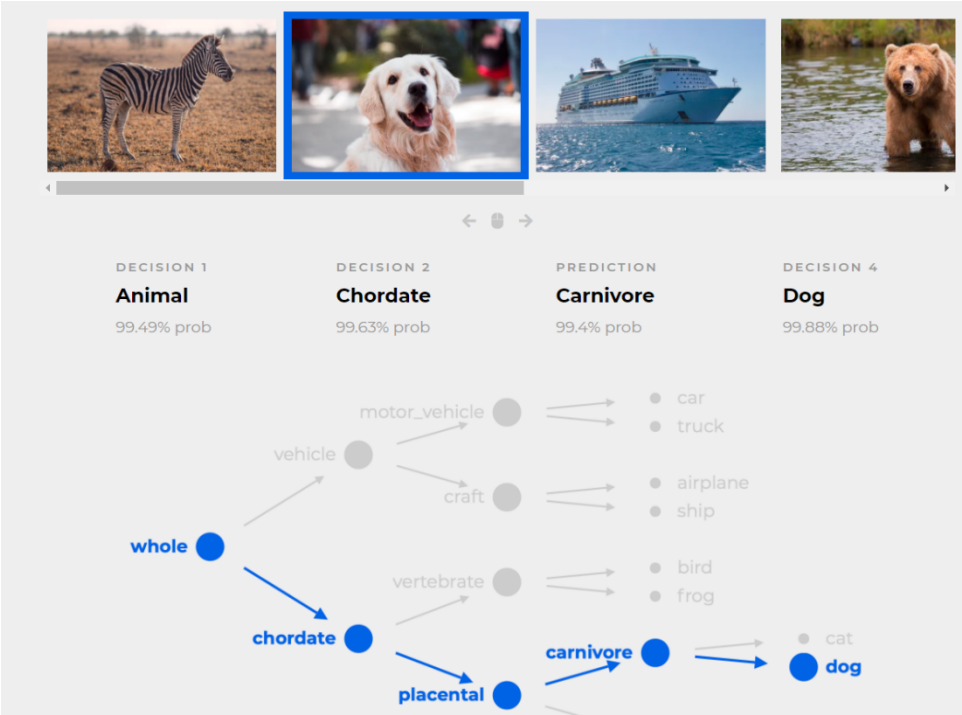

Figure 12: NBDT Output Example Image *[62]*

In this example, each node in the decision tree is a neural network making low-level decisions. The low-level decision made by the neural network above is “is a carnivore” or “is a chordate”. This type of transparency for unseen object classification is critical for sensitive machine learning applications (**Figure 13**).

For low-dimensional tabular data, the decision rules that the model processes in a decision tree, are even more straightforward and simple to interpret than decision rules for high-dimensional data like images.

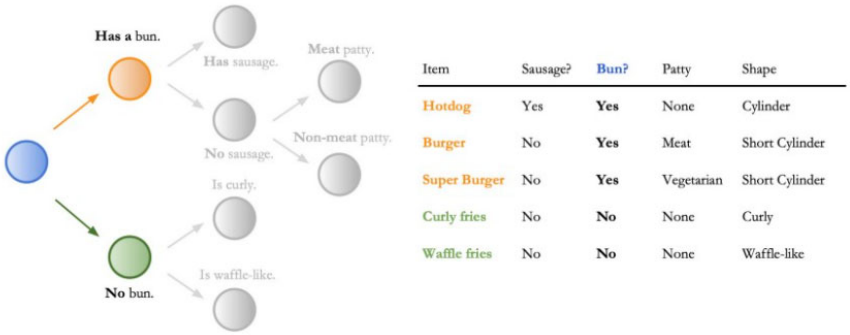

Figure 13: NBDT Output Example Tabular *[62]*

The figure above shows easy interpretable decision rules with low-dimensional tabular data. The right shows a sample of the dataset and the left shows the decision tree trained on the data. In this example the decision or classifier is ‘has a bun’ or ‘no bun’.

The process of training and running inference on an NBDT can be followed in four steps [**Figure 14]** [63]:

- Construct a hierarchy for the decision tree. This hierarchy determines which sets of classes the NBDT must decide between. We refer to this hierarchy as an **Induced Hierarchy**.
- This hierarchy yields a particular loss function, that we call the **Tree Supervision Loss**. Train the original neural network, *without any modifications*, using this new loss.
- Start inference by passing the sample through the neural network backbone. The backbone is all neural network layers before the final fully-connected layer.
- Finish inference by running the final fully-connected layer as a sequence of decision rules, which we call **Embedded Decision Rules**. These decisions culminate in the final prediction.

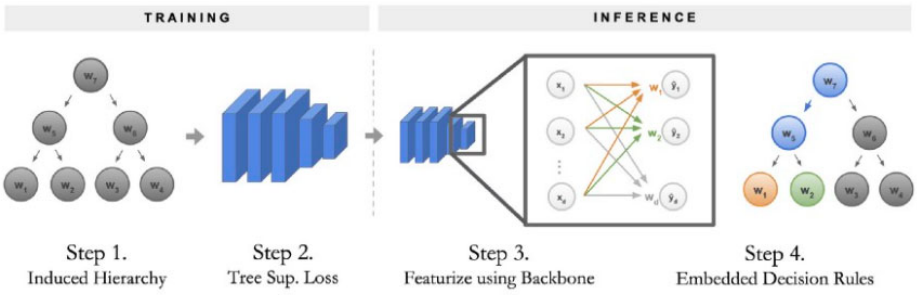

***Figure 14: Construction of an NBDT** [63]*

In a highly regulated industry with sensitive AI applications like the medical field, the NBDT method is necessary in order to yield both high accuracy and model interpretability. This ensures the AI is high performing and trustworthy. For example, in a medical imaging use case that uses AI to aid in diagnostic conclusions, the quality and reliability of the AI is utmost priority to ensure the patient is diagnosed accurately. In this example, we can utilise NBDTs as an image classifier, with the diagnostic images as the training set. When we run inference on the trained classifier as embedded decision rules, the final prediction will output the predicted class, as well as the model's decisions.

The end-user i.e. the health expert or physician will then be able to easily interpret the model's prediction as well as the process. This level of transparency will yield in better decision making for the end-user, as the user has more information to base their decision. It also provides a justifiable decision in the case of ethical implications or healthcare regulations.

Similarly, in a financial services industry example; NBDT's can be used to explain how a model makes decisions for regulated applications such as a loan application predictor, a fraud detection model or a fake news classifier. If the AI application is using neural networks, we can convert the NN into an NBDT. We can do this by training the original NN with an NBDT loss. To run inference, we can wrap our original classification NN with and NBDT wrapper and build our own induced hierarchy. Although NBDTs are primarily used in image applications, text data is an ongoing avenue of research for the IVVEs project.

4.2.2 Explainable AI in Industrial Environment for Automatic Defect Inspection

Grupo Antolin holds a 50% shareholding in Keyland. Grupo Antolin is one of the largest manufacturers of vehicle interiors in the world, achieved sales amounting to €5,214 million in 2019.

Car Interiors design and manufacture has become a lot more complex, with surfaces made up of many more layers with a lot more choice in colours, targeting a greater demand for personalization from end users. Optical quality inspection is still one of the common methods to ensure quality control. So far, this is preventing full automation and integration towards the Industry 4.0 concept. The increasing variability in colours, shapes, textures, positions… has led this sector to look for automating defect inspection with artificial intelligence (AI). Deep learning (DL), especially convolutional neural networks (CNN), has proven to be very effective for image detection and classification, and is now being adopted to solve industrial inspection tasks. The challenge is mainly that inspections must constantly adapt to different quality requirements for different products. Therefore, a better understanding of the outcomes of the ML-based system must be present.

Figure 15. Grupo Antolin Inspection Operator [64]

Figure 16. Grupo Antolin QA Operator [64]

At this stage, KEYLAND is focused on providing some interpretations of the predictions done by the model. As a first step, we propose a SUT based on surface inspection with a U-NET architecture.

4.2.3 MLOps and Reproducibility

MLOps is an engineering practice that is mainly intended to apply DevOps principles to development of machine learning systems and unify the ML development (Dev) and ML system operation (Ops). It involves automation of all steps of ML development including integration, testing, deployment and infrastructure resource management.

The ongoing work related to MLOps seeks to understand the different ways to introduce continuous deployment in the world of ML [65]. To this end, we have performed a literature survey to understand which factors need to be considered in the construction of the deployment pipeline. So far, we have been able to identify several dominant tools that we plan to investigate in detail to better understand their benefits and downsides.

The focus of this work is very hands-on in nature, and hence special attention needs to be placed on creating tool chains that are generic for different use cases. Therefore, we need to experiment with different tool combinations as well as consider different kinds of data sets in training and operational use. In the long run, we will also study monitoring mechanisms for ML features, including, for instance, monitoring and detection of potential bias in ML operations. In addition, the inclusion of high-performance computing architectures used for learning is foreseen.

Once the deployment pipeline is completed, we plan to apply the approach to the use cases where data from partners is available and where partners seek to continuously deploy ML systems. Currently the plan is to apply these practices to a use case provided by F-Secure.

4.2.3.1 Continuous monitoring of ML models in practice

In our use case the goal is to group actors based on the typical actions they perform. The way we model these relationships is a sparse matrix. Different matrix factorization methods can be used to cluster actors into similarly behaving groups. The challenge here is that for many clustering models the model retraining leads to non-deterministic results. Models obtained in different training sessions are different and thus the classification results are permutated and to some degree possibly mixed. This behaviour is characteristic of matrix factorization and similar methods (LDA, SVD, etc.). A possible way to improve clustering performance and stability is to make the matrix even more sparse by filtering the input data. One of the goals is to improve such data pre-processing to not require manual work and expert knowledge while still avoiding the loss of important features.

We are especially interested in clustering models, where there is no ground truth available and quality assessment is typically made by an expert manually and hence subjective. Traditionally an expert investigation is required after each training run to interpret the uncovered profiles. Forcing stable clusters and/or automated clusters interpretation would be useful for testing and automation. One possible approach to achieve this is to use some semi-supervised learning models to identify clusters, where expert knowledge is used to label some samples and that information is used in the learning process. Here we would obviously want to select optimal samples to show for the human expert to maximize the benefit of the expert work.

We want to develop continuous monitoring of ML models in production and trigger retraining based on changes in data and/or model performance. For effective monitoring, good metrics for concept drift and model performance are required. In addition to performance metrics, we are

also interested in recognizing possible model poisoning attacks from the input data as an extreme case of concept drift.

F-Secure can provide real data for this use case for clustering computer usage based on the typical programs run on the machines.

4.2.3.2 MLOps Pipeline

This chapter describes the building blocks of an MLOps pipeline and lists the tools considered to be used in the implementation stage.

MLOps is a superset of DevOps. Introducing machine learning to a software system increases the complexity of the system, as well as its continuous integration and delivery pipeline. More elements need to be in version control, and a single training process can take even weeks to complete and require special hardware. Computationally expensive model training processes require CI/CD pipelines to have access to GPU hardware resources. On **Figure 17** Azure presents performance differences of different models training processes with varying hardware, showing that using GPUs dramatically increases performance, regardless of model and framework [66].

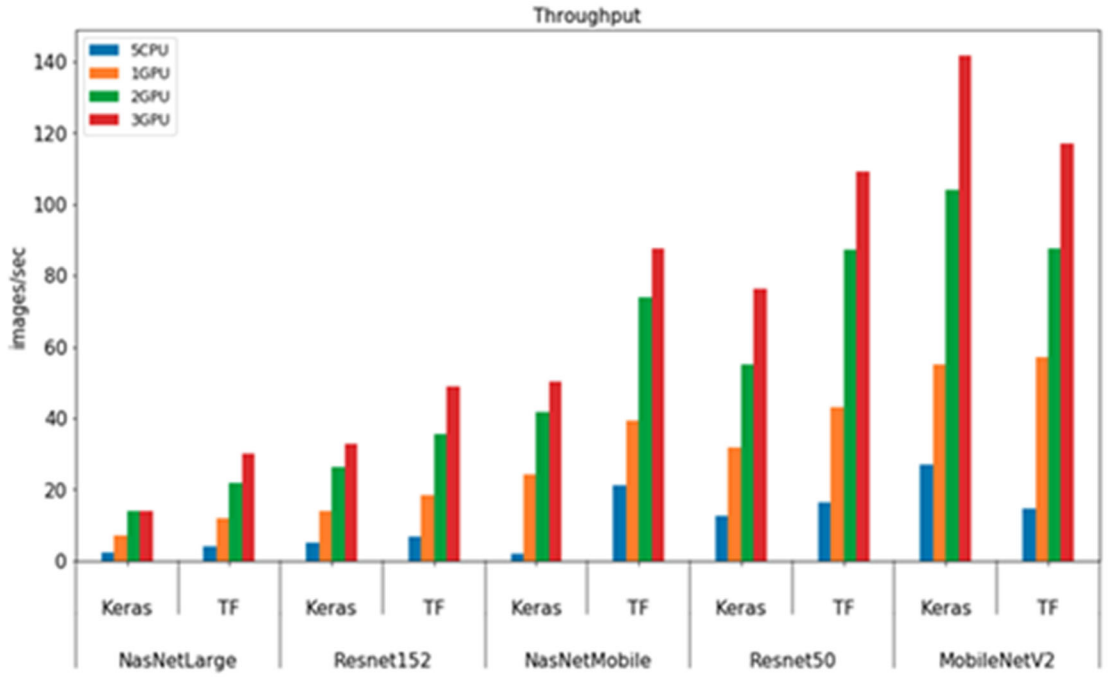

Figure 17: Hardware performance [66]

The CI/CD pipeline needs to always know where the production data is stored and have authorized access to it. Data can be vast and updating even daily or in real-time. The data is often transferred through the internet, from storage to the pipeline, making processes longer and bringing weight to even geographical distances between the data and the pipeline system. For traceability and governance of the trained model, the dataset needs version control. This is done by storing snapshots or data differences of the dataset.

In traditional software systems, the product artefact is built from a specific snapshot of code. However, in machine learning systems, the artefact is a product of the source code and the data used to train, visualized in **Figure 18**. Often the training source code also takes model hyperparameters as parameters. In a Machine Learning system, to reproduce an artefact, its source code, hyperparameters and training data needs to be version controlled.

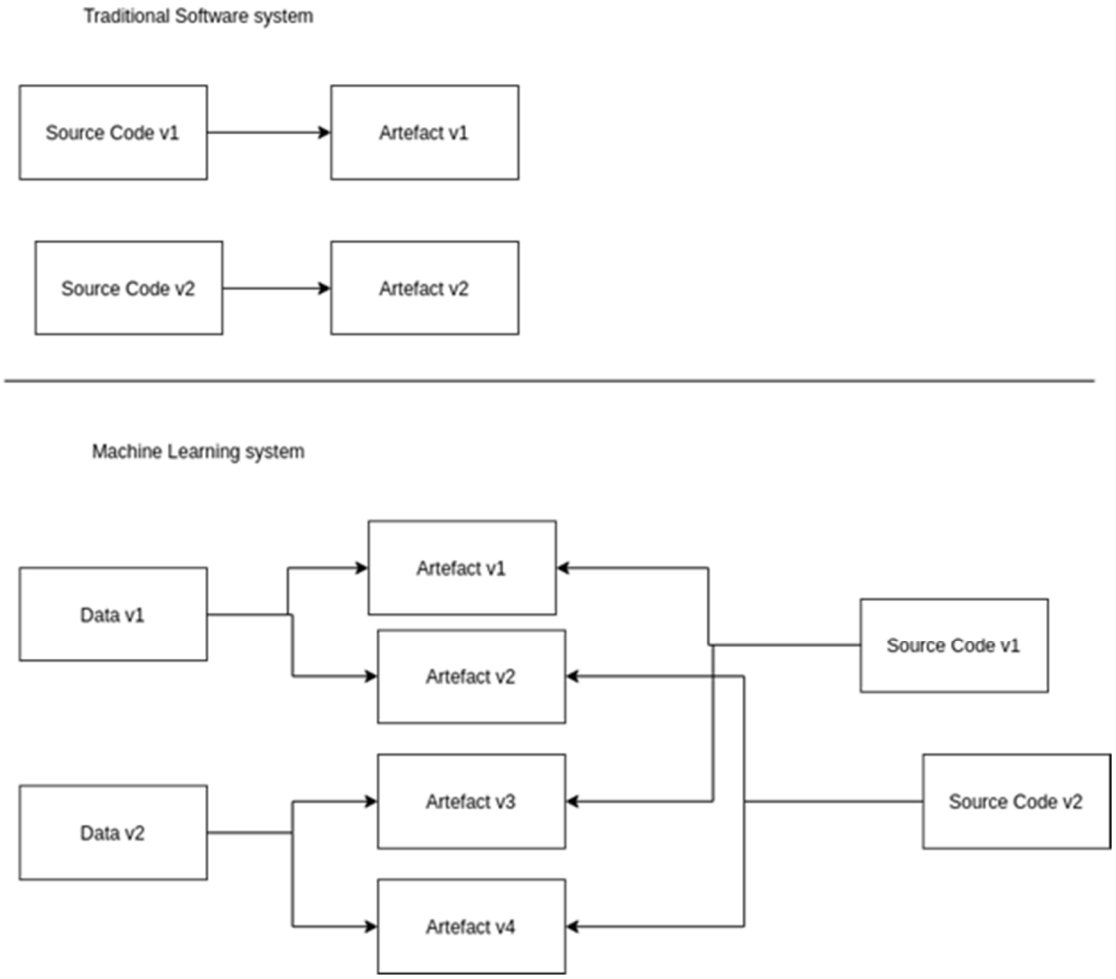

Figure 18. Traditional Software artefact vs. ML artefact

Machine learning systems require more phases of testing. While traditional software needs unit, integration, security and end-to-end system testing, an ML system requires all of this plus data and model testing. An unregulated predictive model in a production setting can introduce biased, discriminative and illegal inferences.

A Machine Learning continuous delivery pipeline consists of ETL, training and serving pipelines. ETL pipelines ensure that the data is sound and accessible from the right locations and the training pipeline ensures that a model is created, tested and evaluated. Serving a pipeline, transfers the trained model for the end-users and sets up monitoring systems for it.

As the training can take hours to several days to complete, the training pipeline must be capable of restoring progress from a checkpoint in case of fault or suspension of the system. Because of this same time constraint, it is also essential that the pipeline can run multiple training sessions in parallel, independently from each other.

After a specific version of the model is created, it needs to be evaluated and tested in the pipeline before green-lighting it as production-ready. A model is tagged with a version number and metadata considering its hyperparameters, as well as its dataset, training, and evaluation code versions. With the necessary metadata, the model can be reproduced in the future, and its performance traced back to its data, training and evaluation. This reproducible environment is paramount for the model development, for example, to investigate why faults like biased inference occur and how to eliminate it.

The serving pipeline takes care of the deployment strategy of the model. When a new model is trained - or in traditional software, a new version of the code is created, it is common to deploy it into a staging environment before going into production. After staging is green-lighted the new artefacts might be fully deployed by various deployment strategies, for example as a rolling A/B release, where a steadily increasing portion of the production traffic is split to the new version ultimately replacing the older. This strategy is used to monitor and benchmark the new version's performance with a smaller portion of end-users to detect faults and rollback before any more damage is done.

The serving pipeline automatically integrates the served models into various monitoring systems to track performance and detect faults like model drifting, where the models domain has changed radically since its training. A common way to fix models concept drift is to re-train the model with an updated dataset. A monitoring system should detect the drift and trigger a re-training workflow, automatically training a new model and serving it once its ready.

Choosing an action policy for faulty models is not trivial. The policy could be different for different alerts. For example, when drift is detected, it is not clear if the current model should be allowed to stay online. It is domain-specific. In some cases, the harm of wrong inference can be such that the model should be taken down immediately on detection or a new model is deployed as a hot-fix as soon as its finished training. In contrast, in other cases, the drift is not that critical, and the model should stay online until the re-trained model is tested, evaluated and safely elevated to the production.

The action policy should be changeable for each release, for example by stating that a specific release cannot be rolled back and any detected faults should not be acted upon at all or even as a nuclear option, take down the whole system. For example, when detecting a high error rate, the action could be to roll back to the previous version, but what if the older version had a significant issue with making discriminatory and illegal decisions? A naive solution would be to alert developers on fault detections and rely on the developers to solve all issues, but in most cases, this is not feasible or effective.

The toolchain considered for the implementation of MLOps pipeline consists of various open-source cloud-native tools. At the heart of it all is **Kubernetes**, which is the de-facto industry standard for orchestrating lifecycles of distributed containerized workloads. Toolchain includes:

- For data & model storage we use **AWS S3** [67] as remote storage and **MinIO** [68] inside the Kubernetes cluster.
- **Argo Workflows** [69] as a workflow engine orchestrating containerized sequential and parallel workloads inside the cluster. Workloads orchestrated consists of different ETL, model training, testing and evaluation processes.
- **Seldon core** [70] for model serving.
- **Knative eventing** [71] for communicating between models inside the cluster, e.g. monitoring models and the inference API, backed up with a **Kafka** or **NATS** backchannel.
- **Prometheus** [72] **& Grafana** [73] for real-time metrics, alerts and dashboards.
- **Istio** [74] for networking and gateway solutions.
- **Flux v2** [75] as a GitOps tool.
- **SealedSecrets** [76] for secret handling.

4.3 Testing Techniques for Machine Learning

In the following sections we describe SoTA concepts and 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 as mentioned above and 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 for a traditional approach to testing such as using a reinforcement learning agent to generate unit and performance test cases based on the test conditions and objectives. This aims to accelerate test case generation and efficiency for the ML system under test.

We propose an oracle-centred approach to evaluate learning algorithms to optimise hyper-parameter tuning of a decision-tree ML model and essentially optimise the performance of the model. This is done by generating data from reference trees and comparing them with learned trees. Additionally, we use metamorphic testing techniques to automate black-box testing by generating label preserving perturbations to inputs to scale test creation. Furthermore, we investigate the generation of adversarial attacks based on the metamorphic transformations which can be combined with reinforcement learning test case prioritization to optimise test selection. Finally, we investigate the possibility of using AutoML to generate ML models and compare the performance of these autonomously generated models to manually created ML models. This is to test and evaluate model robustness in an effort to optimise hypermeter tuning and model configuration for utmost performance and quality assurance. The partners in collaboration of the aforementioned AI-driven testing initiatives include; RISE, CONCATEL & NetCheck, CRIM, University of Helsinki and Techila Technologies.

4.3.1 Machine Learning-Assisted Testing

Nowadays, regarding the wide use of machine learning components in many software intensive systems, testing ML-enabled systems is of great importance. Generating effective test inputs which could lead to malfunctions or improper functionality is challenging. Deep Neural Network (DNN)-based systems are currently one of the common categories of ML-enabled systems used in many industrial domains such as aerospace and automotive ones. Common existing approaches for testing and verification of DNN systems could be generally categorized into three classes of automated approaches which are as follows [77]:

- Search-based (using evolutionary algorithms) for generating adversarial examples (or test cases). These approaches mainly search for adversarial test inputs to show the lack of sufficient robustness/resilience of the system against the adversarial perturbation [78] [79] [80]
- Automated formal verification by different types of techniques like game-based approximate verification approaches. These approaches are mainly used to provide a sort of formal guarantee on the robustness of the system. They mainly aim at providing a guarantee on the robustness of the system within a maximal size of the perturbation which does not cause a malfunction [81] [82]
- Probabilistic verification of DNNs, specifically for Bayesian Neural Network (BNN) [83].

The mentioned techniques are mainly used at the level of unit testing, and moreover the ML-enabled systems are often a complex of ML-components together with other components. 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.

The current ML-enabled system under test (SUT) in our research is a Pedestrian Detection System (PDS), as an ADAS (advanced driver-assistance system), which involves cameras and an ML detection module. The testing environment is an industry-grade automotive simulator. Currently, in industry, the system level testing of ADAS is often performed through on-road testing or ad hoc field tests. However, these types of testing are expensive and inefficient. Meanwhile, simulation-based testing is an efficient, cost effective and scalable complementary approach for the system level testing.

Generating effective test cases, i.e., the ones which could lead to improper functionality of the system, is a challenging task within the system-level testing of these systems. Search-based technique such as using evolutionary algorithms, e.g., NSGA II, is one of the core techniques among the existing testing approaches [84] [85]

In our work, we propose a Reinforcement Learning-assisted test agent which learns an optimal policy (way) to generate effective test cases through exploring the space of the test conditions. The idea of building smart test agent using RL paradigm has been also applied to other contexts in software testing such as performance testing of software programs. In [86] [87] [88] [89] we have presented a smart performance testing framework consisting of two RL-assisted test agents that learn the efficient generation of performance test cases to meet the testing objective and replay the learned policy in further testing situations which leads to higher efficiency in test case generation.

Reinforcement Learning (RL) [90] is a fundamental category of machine learning mainly intended to find the optimal way to make decisions in decision making problems. It is inspired by human's learning and work differently from supervised and unsupervised learning paradigms. The learning is done based on a continuous interaction between a smart agent and the problem environment which is system under test (SUT) in our case. The smart test agent explores the effects of different test scenarios on the behaviour of the system during the steps of the interaction with the SUT. At each step of the interaction, it observes the status and makes a decision to set a test scenario. Then the SUT is tested under the recommended test scenario, and the test agent receives a reward signal indicating the effectiveness of the recommended test scenario. Some of the main differences between RL and other learning paradigms are that there is no supervisor in RL, and the agent just receives a reward signal from the environment, the agent goes through the environment based on a sequential decision-making process.

In this work we use a model-free temporal difference learning, i.e., DQN [91], for training the smart test agent. It learns how to generate the effective test scenarios which result in an improper function. For further direction, we plan to investigate the possibility for transfer learning in this context. It involves training a smart test agent in a simulation environment and using the trained agent for generating test scenarios in realistic (on-field) test environment.

4.3.2 Testing Learning Algorithms

4.3.2.1 An Oracle-centered Approach to evaluate Decision Tree Learning Algorithms

Rhea Inc and several other IVVES partners want to enhance the security of computer systems. In particular they want to improve the classification of network traffic by early identifying malicious traffic (system attacks). It is our goal within IVVES to develop machine learning models for enhancing the classification of the traffic. Traditionally, the models are produced by machine learning algorithms after some tuning activities performed by experts. Nevertheless, the less trustable are the learning algorithms, the less trustable are the outputted models, no matter how good the tuning is.

The state of the art [93] indicates that decision trees can be used to classify network traffic and many learning algorithms have been applied to generate such trees. The ability of the generated trees to classify the traffic is evaluated with domain specific test data by computing some metrics such as precision and F1 score. We believe that domain-specific data agnostic evaluation technique can be useful to select a “best” learning algorithm for tuning and decision tree generation, especially for the network traffic classification. This kind of evaluation techniques is rare in the literature.

We propose [92] a novel oracle-centred approach to evaluate (the learning ability of) learning algorithms for decision trees. It consists of generating data from reference trees playing the role of oracles, producing learned trees with existing learning algorithms, and determining the degree of correctness (DOE) of the learned trees by comparing them with the oracles. Such an approach is inspired by our work on testing finite state machines [97] [98]. The average DOE is used to estimate the quality of the learning algorithms, i.e., their learning ability.

We assess five decision tree-learning algorithms with the proposed approach. The decision tree learning algorithms include four heuristic-based algorithms namely ID3 [99], J48 (a WEKA implementation of C4.5 [100]), simpleCART (a WEKA implementation of CART [95]), and RandomTree [97], and an exact algorithm which infers optimal decision trees InferDT [94]. The evaluation result is independent of a specific dataset. The evaluation results show that, when training on deterministic datasets with no noise, InferDT produces the most accurate model. In the family of heuristic-based decision trees, ID3 and RandomTree have the best performance, where ID3 performs slightly better than RandomTree. The results also show the effectiveness of the proposed evaluation method. By using DOE as the metric, it successfully distinguished the performance difference between learning algorithms. The detailed contribution can be found in [92].

For further contributions, we plan to develop decision trees for traffic analysis applying ID3 and Infer DT on specific datasets from the partners, once datasets are made available. It is also planned investigating oracle-based evaluation approaches for other ML models such as feed forward neural networks or recurrent neural networks.

4.3.3 Metamorphic Testing for NLP-Based ESG Investment Solutions

At this stage, SII CONCATEL is working in the generation of label-preserving perturbations to inputs. The first approach is based on basic perturbations (changing location names for NER capabilities) and introducing typos to test robustness. The generation of test cases is based on abstractions [101] to scale up test creation and ease the generation of perturbations. This is the foundations for future work. The novelty is based on generating, based on context provided by the knowledge graph, a set masks that will be used by the human tester (**Figure 19**).

Figure 19. Templating with masked language models. [102]

Adversarial Attack based on Metamorphic Relations

As a second step, CCTL/NC is focusing in the generation of adversarial attack based on Metamorphic Transformations. For this, the adaptation of Pick-n-Plug and Pick-Permute-Plug approaches [102] are being used, generating a PoC to check the robustness of the Incoming data QA techniques, and the ESG-scoring system. The novelty is the combination of these approaches with Knowledge Graphs. The solution will take a natural language sentence, a chosen sensitive attribute A and also a knowledge graph G as inputs to draw adversarial sentences that will be eventually ingested by the “Plug” operators. Eventually, this will be combined with Reinforcement Learning-based test case prioritization (WP3), that we are expecting to allow to optimize the selection of useful relations for metamorphic testing.

4.3.4 Testing Model Robustness

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 [103] for this task.

In particular, the goal of this work is to explore different AutoML systems to generate classification and time-series models. The particular focus is on how robust these models are towards outliers and other faults in their input. In later project years, we expect to extend the analysis focused on model accuracy to the entire system behaviour in case of faulty input data.

So far, we have used data sets that are openly available on the internet. However, as the project progresses, we seek to examine the partners' data sets. AutoML requires a lot of computing resources. There we will investigate with Techila Technologies, a project partner in IVVES, how the time needed for massive computation can be reduced as well as other performance-related topics.

5. Conclusions

Since this is a first version of validation methods and techniques for ML, the final version will be deliverable 2.4., in which we will continue development and finalize the results and outcomes. It should therefore be noticed that the cases and research questions presented in this deliverable are subject to change or evolve in the next deliverable. The next phase at the WP2 is deliverable 2.3. where we develop the actual tools for validating ML components, data quality and model quality. Deliverable 2.3. will be confidential and will be published only among the consortium members.

6. References

[1] http://braintumorsegmentation.org

[2] S. Bakas, 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)

[3] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. Kirby, et al., "Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-GBM collection", The Cancer Imaging Archive, 2017. DOI: 10.7937/K9/TCIA.2017.KLXWJJ1Q

[4] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. Kirby, et al., "Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-LGG collection", The Cancer Imaging Archive, 2017. DOI: 10.7937/K9/TCIA.2017.GJQ7R0EF

[5] S. Bakas, 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] B. H. Menze, 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

[7] Yu, Biting, Luping Zhou, Lei Wang, Yinghuan Shi, Jurgen Fripp, and Pierrick Bourgeat. 2019. “Ea-GANs: Edge-Aware Generative Adversarial Networks for Cross-Modality MR Image Synthesis.” IEEE Transactions on Medical Imaging 38 (7): 1750–62. https://doi.org/10.1109/TMI.2019.2895894.

[8] Pan, Yongsheng, Mingxia Liu, Chunfeng Lian, Tao Zhou, Yong Xia, and Dinggang Shen. 2018. “Synthesizing Missing PET from MRI with Cycle-Consistent Generative Adversarial Networks for Alzheimer’s Disease Diagnosis.” In Lecture Notes in Computer Science, 11072 LNCS:455–63. Springer Verlag. https://doi.org/10.1007/978-3-030-00931-1_52.

[9] Mondal, Arnab Kumar, Jose Dolz, and Christian Desrosiers. 2018. “Few-Shot 3D Multi-Modal Medical Image Segmentation Using Generative Adversarial Learning,” October. http://arxiv.org/abs/1810.12241.

[10] Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2016. “Image-to-Image Translation with Conditional Adversarial Networks.” Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 2017-January (November): 5967–76. http://arxiv.org/abs/1611.07004

[11] Cirillo, Marco Domenico, David Abramian, and Anders Eklund. 2020. “Vox2Vox: 3D-GAN for Brain Tumour Segmentation.” arXiv preprint arXiv:2003.13653, 2020

[12] Holmes, C. J., R. Hoge, L. Collins, R. Woods, A. W. Toga, and A. C. Evans. 1998.

"Enhancement of MR Images Using Registration for Signal Averaging." Journal of Computer Assisted Tomography 22 (2): 324–33. https://doi.org/10.1097/00004728-199803000-00032.

[13] Eklund, Anders. 2019. "Feeding the Zombies: Synthesizing Brain Volumes Using a 3D Progressive Growing GAN," December. http://arxiv.org/abs/1912.05357.

[14] Foroozandeh, Mehdi, and Anders Eklund. 2020. "Synthesizing Brain Tumor Images and Annotations by Combining Progressive Growing GAN and SPADE," September. http://arxiv.org/abs/2009.05946.

[15] Jenkinson, Mark, Christian F Beckmann, Timothy E J Behrens, Mark W Woolrich, and Stephen M Smith. 2012. "FSL." NeuroImage 62 (2): 782–90. https://doi.org/10.1016/j.neuroimage.2011.09.015.

[16] Karras, Tero, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. "Progressive Growing of GANs for Improved Quality, Stability, and Variation." In 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings. https://youtu.be/G06dEcZ-QTg.

[17] Park, Taesung, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. "Semantic Image Synthesis with Spatially-Adaptive Normalization." Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2019-June (March): 2332–41. http://arxiv.org/abs/1903.07291.

[18] Marko Koskinen, et. al , "– State of the Art on Validation Techniques for ML," ITEA, 30-Jun-2020. [Online]. Available: https://itea3.org/project/ivves.html. [Accessed: 2020].

[19] "An Artificial Intelligence Quality Framework | Sogeti." [Online]. Available: https://www.sogeti.nl/nieuws/artificial-intelligence/blogs/artificial-intelligence-quality-framework. [Accessed: 25-Jun-2020].

[20] M. Hittmeir, "On the Utility of Synthetic Data: An Empirical Evaluation ...," www.myhealthmydata.e, 2019. [Online]. Available: http://www.myhealthmydata.eu/wp-content/uploads/2019/10/paper_142.pdf. [Accessed: 03-Dec-2020].

[21] Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 2019, 51-56.

[22] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke & Travis E. Oliphant. Array programming with NumPy, Nature, 585, 357–362 (2020), DOI:10.1038/s41586-020-2649-2

[23] Waskom, M. et al., mwaskom/seaborn: v0.8.1 (September 2017), 2017, Zenodo. Available at: https://doi.org/10.5281/zenodo.883859.

[24] P Virtanen et al, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. 2020, Nature Methods, 17(3), 261-272.

[25] Zhao, Y., Nasrullah, Z. and Li, Z., 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. Journal of machine learning research (JMLR), 20(96), pp.1-7.

[26] S Brugman, Pandas Profiling, GitHub repository, 2020, https://github.com/pandas-profiling/pandas-profiling

[27] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, 2007, vol. 9, no. 3, pp. 90-95.

[28] Clark, A. (2015). Pillow (PIL Fork) Documentation. readthedocs. Retrieved from https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

[29] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

[30] Van Looveren, Arnaud and Vacanti, Giovanni and Klaise, Janis and Coca, Alexandru (2019). Algorithms for outlier and adversarial instance detection, concept drift and metrics. GitHub repository 2020, https://github.com/SeldonIO/alibi-detect.

[31] S Kim, Basic Image EDA, GitHub repository, 2020, https://github.com/Soongja/basic-image-eda

[32] T. Qiufeng, "taoqf/node-html-parser," GitHub, 2019. [Online]. Available: https://github.com/taoqf/node-html-parser. [Accessed: 07-Dec-2020].

[33] "codecs - Codec registry and base classes," codecs - Codec registry and base classes - Python 3.9.1rc1 documentation. [Online]. Available: https://docs.python.org/3/library/codecs.html. [Accessed: 07-Dec-2020].

[34] Bird, Steven, Edward Loper and Ewan Klein (2009). Natural Language Processing with Python. O'Reilly Media Inc.

[35] Honnibal, Matthew and Montani, Ines (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

[36] Loria, S. (2018). textblob Documentation (2020). Release 0.15, 2. https://textblob.readthedocs.io/en/dev/.

[37] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. (2011) Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research.

[38] A. Mueller, “wordcloud,” PyPI. [Online]. Available: https://pypi.org/project/wordcloud/. [Accessed: 07-Dec-2020]

[39] S. Bansal and C. Aggarwal, “textstat,” PyPI. [Online]. Available: https://pypi.org/project/textstat/. [Accessed: 07-Dec-2020].

[40] C Sala, et al., SD Metrics, GitHub repository (2020). https://github.com/sdv-dev/SDMetrics

[41] Fan, L., Zhang, F., Fan, H. et al. Brief review of image denoising techniques. Vis. Comput. Ind. Biomed. Art 2, 7 (2019). https://doi.org/10.1186/s42492-019-0016-7

[42] N. Sharma, “Ways to Detect and Remove the Outliers,” Medium, 23-May-2018. [Online]. Available: https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba. [Accessed: 07-Dec-2020].

[43] J. Low, “What is Image Annotation?,” Medium, 10-Jan-2020. [Online]. Available: https://medium.com/supahands-techblog/what-is-image-annotation-caf4107601b7. [Accessed: 07-Dec-2020].

[44] KARALEVICIUS, Vytautas; DEGRANDE, Niels; DE WEERDT, Jochen. Using sentiment analysis to predict interday Bitcoin price movements. *The Journal of Risk Finance*, 2018.

[45] VALENCIA, Franco; GÓMEZ-ESPINOSA, Alfonso; VALDÉS-AGUIRRE, Benjamín. Price movement prediction of cryptocurrencies using sentiment analysis and machine learning. *Entropy*, 2019, vol. 21, no 6, p. 589.

[46] KRAAIJEVELD, Olivier, et al. The predictive power of public Twitter sentiment for forecasting cryptocurrency prices. *Journal of International Financial Markets, Institutions and Money*, 2020, vol. 65, no C.

[47] What’s the future of investment research?; refinitiv. https://www.refinitiv.com/perspectives/future-of-investing-trading/future-investment-research/ [Accessed December 2020]

[48] Deutsche Bank Research.(2018). Big Data Shakes Up ESG Investing (konzept). [Accessed December 2020]

[49] Fact Check Tools, https://toolbox.google.com/factcheck/about. [Accessed December 2020]

[50] Translating Embeddings (TransE), http://pyvandenbussche.info/2017/translating-embeddings-transe/. [Accessed December 2020]

[51] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multirelational data. In Advances in neural information processing systems, pages 2787{2795, 2013.

[52] PAN, Jeff Z., et al. Content based fake news detection using knowledge graphs. En International semantic web conference. Springer, Cham, 2018. p. 669-683.

[53] Workshop report: Building Linked Data heatmaps with Clojurescript & thi.ng, https://medium.com/@thi.ng/workshop-report-building-linked-data-heatmaps-with-clojurescript-thi-ng-102e0581225c. [Accessed December 2020]

[54] Text2Image: A new way to NLP?, https://towardsdatascience.com/text2image-a-new-way-to-nlp-cbf63376aa0d. [Accessed December 2020]

[55] Knowledge Graphs For eXplainable AI, https://towardsdatascience.com/knowledge-graphs-for-explainable-ai-dcd73c5c016. [Accessed December 2020]

[56] LU, Yi-Ju; LI, Cheng-Te. GCAN: Graph-aware Co-Attention Networks for Explainable Fake News Detection on Social Media. arXiv preprint arXiv:2004.11648, 2020. [Accessed December 2020]

[57] Text Generation from Knowledge Graphs with Graph Transformers. https://towardsdatascience.com/text-generation-from-knowledge-graphs-with-graph-transformers-c84156ddd446. [Accessed December 2020]

[58] LIU, Ye, et al. Commonsense Evidence Generation and Injection in Reading Comprehension. arXiv preprint arXiv:2005.05240, 2020. [Accessed December 2020]

[59] Mattila, J 2013, 'Disease State Index and Disease State Fingerprint: Supervised learning applied to clinical decision support in Alzheimer's disease: Dissertation', Doctor Degree, Tampere University of Technology (TUT), Espoo. http://www.vtt.fi/inf/pdf/science/2013/S51.pdf

[60] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in *Advances in Neural Information Processing Systems 30*, 2017.

[61] A. Wan, “Neural-Backed Decision Trees,” *NBDT: Neural-Backed Decision Trees | Alvin Wan | Efficient Machine Learning, XAI Researcher at UC Berkeley*, 2020. [Online]. Available: https://research.alvinwan.com/neural-backed-decision-trees/. [Accessed: 01-Dec-2020].

[62] A. Wan, “Making Decision Trees Accurate Again: Explaining what Explainable AI did not,” *Medium*, 20-Apr-2020. [Online]. Available: https://medium.com/riselab/making-decision-trees-accurate-again-explaining-what-explainable-ai-did-not-abb73e285f22. [Accessed: 01-Dec-2020].

[63] A. Wan, L. Dunlap, and D. Ho, “NBDT: Neural-Backed Decision Tree,” *Arvix.org*, 2020. [Online]. Available: https://arxiv.org/abs/2004.0022. [Accessed: 2020].

[64] http://www.grupoantolin.com/sites/default/files/201057es.pdf

[65] Lim, J., Lee, H., Won, Y., & Yeon, H. (2019). MLOP lifecycle scheme for vision-based inspection process in manufacturing. In *2019 {USENIX} Conference on Operational Machine Learning (OpML 19)* (pp. 9-11).

[66] GPUs vs CPUs for deployment of deep learning models. en.url:https://azure.microsoft . com / en - us / blog / gpus - vs - cpus - for - deployment - of - deep -learning-models/ (visited on 11/27/2020)

[67] 'Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service (S3)', *Amazon Web Services, Inc.* https://aws.amazon.com/s3/ (accessed Dec. 08, 2020).

[68] M. Inc, 'MinIO | High Performance, Kubernetes Native Object Storage', *MinIO*. https://min.io (accessed Dec. 08, 2020). [Accessed December 2020]

[69] 'Workflows & Pipelines | Argo'. https://argoproj.github.io/projects/argo/ (accessed Dec. 08, 2020). [Accessed December 2020]

[70] 'Seldon Core - Open Source Machine Learning Deployment for Kubernetes', *Seldon*. https://www.seldon.io/tech/products/core/ (accessed Dec. 08, 2020). [Accessed December 2020]

[71] 'Knative Eventing', *Knative*. https://knative.dev/docs/eventing/ (accessed Dec. 08, 2020). [Accessed December 2020]

[72] 'Prometheus - Monitoring system & time series database'. https://prometheus.io/ (accessed Dec. 08, 2020). [Accessed December 2020]

[73] 'Grafana: The open observability platform', *Grafana Labs*. https://grafana.com/ (accessed Dec. 08, 2020). [Accessed December 2020]

[74] 'Istio', *Istio*. https://istio.io/ (accessed Dec. 08, 2020). [Accessed December 2020]

[75] fluxcd/fluxv2 [Internet] Flux project; 2020 [cited 2020 Dec 8]. Available from: https://github.com/fluxcd/flux2. [Accessed December 2020]

[76] bitnami-labs/sealed-secrets [Internet]. Bitnami Labs; 2020 [cited 2020 Dec 8]. Available from: https://github.com/bitnami-labs/sealed-secrets. [Accessed December 2020]

[77] Kwiatkowska, M. Z. (2019). Safety verification for deep neural networks with provable guarantees.

[78] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. In ICLR, 2014

[79] 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.

[80] 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.

[81] Wicker, M., Huang, X., & Kwiatkowska, M. (2018, April). Feature-guided black-box safety testing of deep neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (pp. 408-426). Springer, Cham.

[82] Wu, M., Wicker, M., Ruan, W., Huang, X., & Kwiatkowska, M. (2020). A game-based approximate verification of deep neural networks with provable guarantees. *Theoretical Computer Science*, *807*, 298-329.

[83] Cardelli, L., Kwiatkowska, M., Laurenti, L., Paoletti, N., Patane, A., & Wicker, M. (2019). Statistical guarantees for the robustness of Bayesian neural networks. In IJCAI 2019

[84] Ben Abdessalem, 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).

[85] Ben Abdessalem, R., Panichella, A., Nejati, S., Briand, L. C., & Stifter, T. (2018, September). Testing autonomous cars for feature interaction failures using many-objective search. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 143-154). IEEE.

[86] Moghadam, H. M., Machine Learning-Assisted Performance Assurance, Licentiate Thesis, Mälardalen University, 2020.

[87] Moghadam, M. H., Saadatmand, M., Borg, M., Bohlin, M., & Lisper, B. (2019). An Autonomous Performance Testing Framework using Self-Adaptive Fuzzy Reinforcement Learning, Software Quality Journal. Springer

[88] Moghadam, M. H., Saadatmand, M., Borg, M., Bohlin, M., & Lisper, B. (2020, October). Poster: Performance Testing Driven by Reinforcement Learning. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)* (pp. 402-405). IEEE.

[89] Moghadam, M. H. (2019). Machine learning-assisted performance testing. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 1187-11*89) (Winner of Silver Prize at ACM SRC)*

[90] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[91] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529-533.

[92] Tianqi Xiao, Omer Nguena Timo, Florent Avellaneda, Yasir Malik, Stefan Bruda. An Approach to Evaluating Learning Algorithms for Decision Trees. Arxiv, 2020

[93] Tarek Abbes, Adel Bouhoula, Michaël Rusinowitch. Protocol Analysis in Intrusion Detection Using Decision Tree. Proceedings of the International Conference on Information Technology: Coding and Computing, IEEE, 2004

[94] Florent Avellaneda. Efficient inference of optimal decision trees. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI, pages 3195–3202. AAAI Press, 2020.

[95] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. Classification and Regression Trees. Wadsworth International Group, Belmont, California, 1984.

[96] Eibe Frank, Mark Hall, Geoffrey Holmes, Richard Kirkby, Bernhard Pfahringer, Ian H Witten,and Len Trigg. Weka-a machine learning workbench for data mining. In Data mining and knowledge discovery handbook, pages 1269–1277. Springer, 2009.

[97] Omer Nguena Timo, Alexandre Petrenko, and S. Ramesh. Fault model-driven testing from FSM with symbolic inputs. Softw. Qual. J., 27(2):501–527, 2019.

[98] Omer Nguena Timo, Alexandre Petrenko, and S Ramesh. Using imprecise test oracles modelled by fsm. In Proceedings of IEEE International Conference on Software Testing, verification and Validation Workshops (ICSTW), pages 32–39. IEEE, 2019.

[99] J. Ross Quinlan. Induction of decision trees. Machine learning, 1(1):81–106, 1986.

[100] Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[101] RIBEIRO, Marco Tulio, et al. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. *arXiv preprint arXiv:2005.04118*, 2020.

[102] CHAN, Alvin, et al. Metamorphic relation based adversarial attacks on differentiable neural computer. arXiv preprint arXiv:1809.02444, 2018.

[103] Nurminen, J. K., Halvari, T., Harviainen, J., Mylläri, J., Röyskö, A., Silvennoinen, J., & Mikkonen, T. (2019, October). Software Framework for Data Fault Injection to Test Machine Learning Systems. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (pp. 294-299). IEEE.