IVVES

Industrial-grade Verification and Validation of Evolving Systems

Labeled in ITEA3, a EUREKA cluster, Call 5

ITEA3 Project Number 18022

D2.1 – State of the Art on Validation Techniques for ML

Due date of deliverable: June 30, 2020
Actual date of submission: June 30, 2020

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

Organisation name of lead contractor for this deliverable: Techila Technologies

Author(s): Heba Sourkatti, Janne Merilinna, Harri Pölönen, VTT; Zafar Hussain, Lalli Myllyaho, Jukka K. Nurminen, Tommi Mikkonen, University of Helsinki; Tia Nikolic, Almira Pillay, Sogeti; Toni Kajantola, Futurice; Juan Sanchez, Aunia; Mahshid Helali Moghadam, RISE; Marko Koskinen, Rainer Wehkamp, Techila Technologies, Elio Saltalamacchia, David Diaz, Ivan Samaniego, Noemi Merino, Concatel; Jesús Arce, Tomas Miguel Calvo, Jesus López, Keyland; Juan Corral, Manuel Rodicio, Netcheck

Status: Final

Version number: 1.0

Submission Date: 30-June-2020

Doc reference: IVVES_Deliverable_D2.1_V1.0.docx

Work Pack./ Task: WP 2 / T2.1

Description: ***(max 5 lines)*** This document describes the state of the art on Validation Techniques for ML and focuses on Explainable Artificial Intelligence, testing approaches for ML and AI systems and data augmentation to facilitate model development.

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	19/06/2020	First draft	Draft	All
V0.2	25/06/2020	Merge material from contributors	Draft	WP2 + PMT
V0.3	29/06/2020	Final merge and style updates	Concept	Submitted to PMT
V1.0	30/06/2020	Approved by PMT, to be submitted to ITEA3	Final	Uploaded to ITEA

Table of Contents

Glossary

Abbreviation / acronym	Description
AN	Agglomerative Nesting
AI	Artificial Intelligence
ANN	Artificial Neural Networks
BDL	Bayesian Deep Learning
CMO	Chande Momentum Oscillator
DL	Deep learning
DS	Density-based Spatial Clustering
DevOPS	Development and operations
DCT	Differential Combination Testing
DIP-VAE	Disentangled Inferred Prior Variational Autoencoder
EM	Expected Maximization
XAI	Explainable Artificial Intelligence
EDA	Explorative Data Analysis
FF	Farthest-First Traversal
GAN	Generative Adversarial Network
IUT	Implementation Under Test
ICE	Individual Conditional Expectation
KDE	Kernel Density Estimation
KNN	K-Nearest Neighbour
LRP	Layer-wise relevance propagation
LIME	Local Interpretable Model-Agnostic Explanations
ML	Machine Learning
MLOps	Machine Learning Operations
MRIP	Metamorphic Relation Input Pattern
MRPs	Metamorphic Relation Patterns
NB	Naive Bayes
NLP	Natural language Processing
PDP	Partial Dependence Plot
ProtoDash	Prototypes with Importance Weights
QAIF	Quality Artificial Intelligence Framework
SA	Sensitivity Analysis
SHAP	SHapley Additive exPlanations
TA	Technical Analysis
XAI	Explainable Artificial Intelligence

1. Executive Summary

This document (D2.1) describes the state of the art validation methods and techniques for ML. The structure of this document is based on the Quality Artificial Intelligence Framework (QAIF), developed by partner Sogeti NL.

QAIF, inspired by the CRISP-DM Framework, [1] [2] is a cohesive, generic framework that can be tailored to a specific AI solution in a given business context. The framework is comprised of six gates that follow the process flow of the AI project development cycle (CRISP-DM), Business Understanding, Data Understanding, Data Preparation, Model Development, Model Evaluation and Model Deployment. The gates can be broken down into project phase, processes, outcomes, governance and people. In each gate, there are specific tasks that need to be complete for the gate to be passed through in order to enter the next gate. This ensures that each phase of the AI development cycle is validated thoroughly.

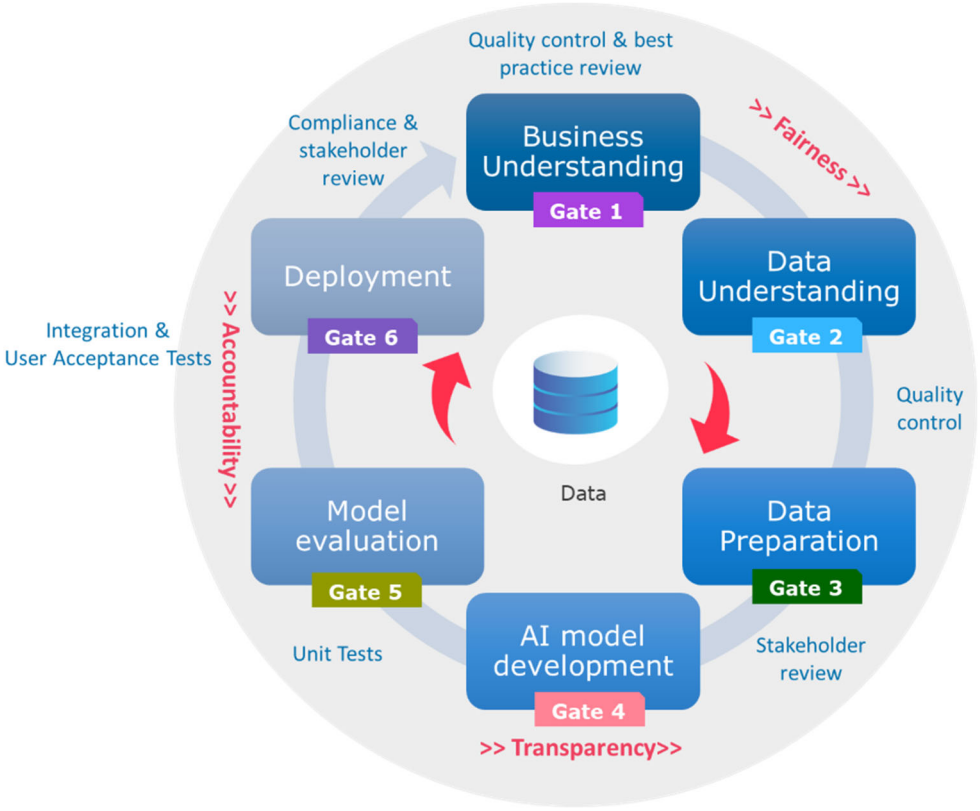

Figure 1. Sogeti Quality AI Framework

This document describes State of the Art Validation methods by each phase or gate in QAIF. Specifically following gate 1, 3, 4, 5 and 6. The gates 2 and 3 are further described in D4.1, which is in the scope of ITEA IVVES project, under work package 4.

The QAIF phases are described in the following chapters of this document:

1. Business understanding (Gate 1) – Business Understanding (2.1)
2. Data preparation (Gate 3) – Data Augmentation (3.1)
3. AI Model development (Gate 4) – Testing and Validating Machine Learning Systems (3.2)
4. Model evaluation (Gate 5) – Explainable Artificial Intelligence (3.3)
5. Deployment (Gate 6) – MLOps - Continuous Deployment and Delivery (3.4)

2. Introduction

Due to the popularity of applying machine learning and artificial intelligence to solve an ever-increasing variety of challenges, the need for verifying and validating models is also increasing accordingly. In order to meet this growing need, different approaches focusing on model interpretability, data augmentation and various testing approaches will be discussed in this document.

Explainable Artificial Intelligence (XAI) refers to a field of study that focuses on methods and techniques that can be used to make machine learning models more easily interpretable and generally speaking tries to answer questions such as “Why does my model do this?”, “Why doesn’t my model do this?” and “What should I do to fix it?”. Answering these questions is becoming increasingly relevant as the use of ML and AI is being adopted in business-critical application and in situations where government regulations apply. When working with some of the more classical machine learning models, such as decision trees, model interpretability may be more inherent simply due to the nature of the model. However, when we apply the same questions to a deep neural network containing millions of parameters, the questions become much harder to answer.

The research field related to XAI is very broad, meaning the state of the art study included in this document only covers a very small subset of the field, focusing on the financial sector. However, the XAI tools and methodologies are of course applicable in other sectors, as they all deal with the same principle components: models and data.

Data is the foundation of any machine learning model. Depending on the industry sector, data properties can cause additional requirements, especially when working with data containing Personally Identifiable Data (PII), especially in situations where the data is also covered by the European Union General Data Protection Regulation (GDPR). To ensure that sufficient and compliant data is available for model training, data augmentation can be a significant asset. The state of the art study about data augmentation included in this document gives a description of traditionally used methods and state of the art methods deep learning methods.

2.1. Business Understanding

According to user and case studies from real projects it is important to understand where and how AI or ML methods and algorithms could be used and how they would bring value, compared to traditional software approaches. While traditional software testing methods may not be enough to validate and verify complicated AI&ML systems it is also known that understanding how the ML system works and what should be the output may be very difficult or even impossible without ways to explain what is the reason behind the decision the system has made based on the data. Using methods to explain the behaviour of the service is democratizing the decision-making being it based on machine or human actions.

Raising question like “How the quality is defined for the service?” and “What is right and wrong and sustainable?” are rather ethical questions where the answers are needed to remain responsible in the eyes of end users. Good collaboration and process of sharing the learnings is a key to build quality and trust in in the first place.

To achieve that, start finding the problem worth solving and define quality right in the beginning of project together with all parties which in turn creates the right, positive, atmosphere for the work and guides for the right tools and methods to implement and to validate.

This means explainability and ethics needs to be taken into the consideration right in the beginning of the project.

How to define quality?

Business Understanding is a critical measure to validate the correctness and generated value of the service from the business point of view. It includes understanding of the customers / end users' needs, overall business goals, setting the right quality targets and definitions, identify and agree needed development models and processes during the service lifespan.

Business goals:

- Vision of Success
- Business goal KPIs
- Data Science goals

User needs:

- End-user needs
- Validation

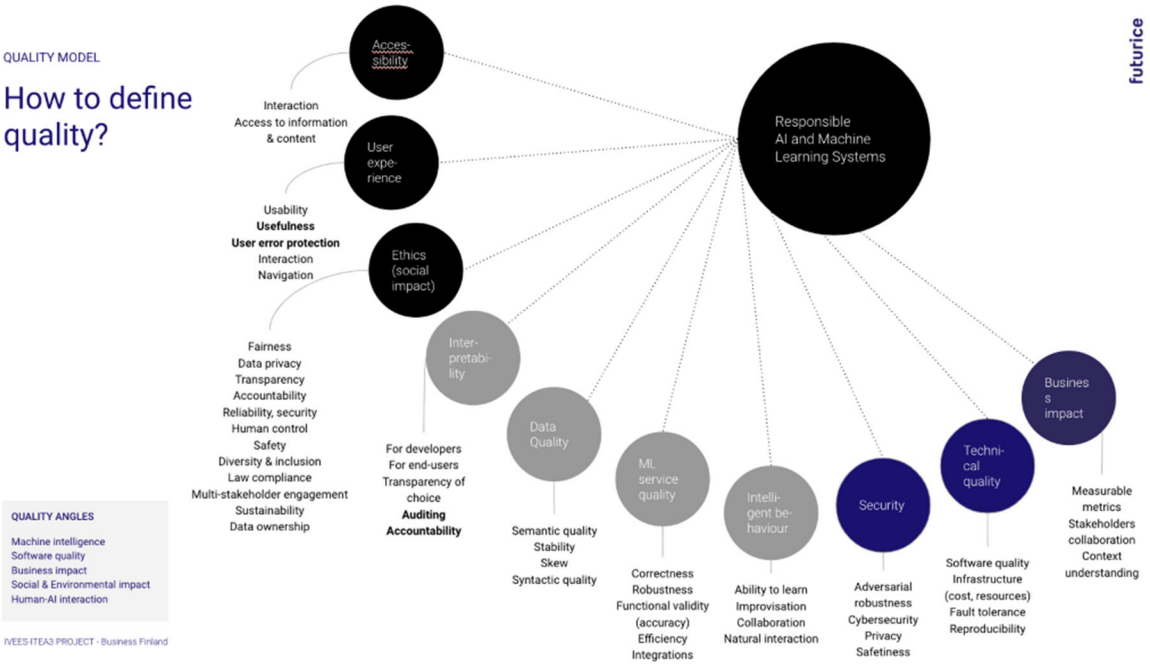

Figure 2. Defining quality.

Basic methods to gain business understanding are stakeholder interviews and close collaboration. This translates in starting the project with a short sprint/phase for research and understanding, where you have interviews, knowledge transfer and workshops for alignment.

In some occasions to clarify the use cases, data availability / quality and business case in general the project could be started by creating a Proof of Concept and utilize Lean Service creation [CC license] methods to collect feedback and learnings back to the service under development and altogether have better business understanding and understanding what is the problem project is going to solve and find suitable tools, methods, algorithms and team setup to implement and verify it .

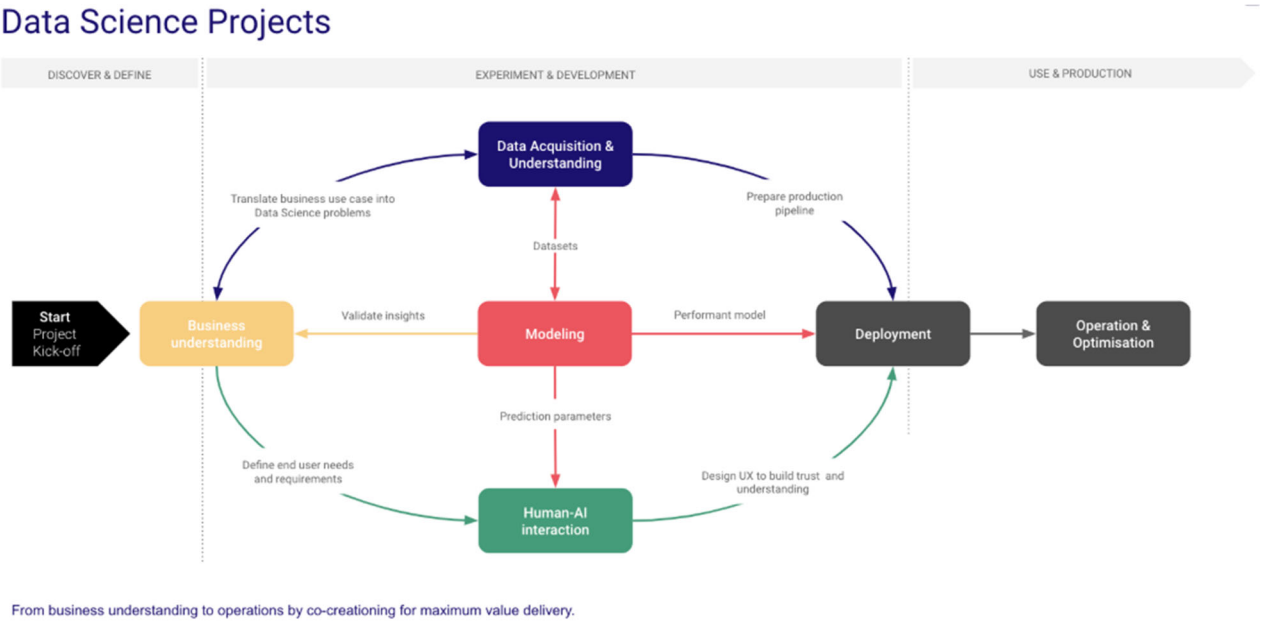

Figure 3. Data Science project flow.

Co-operation

Having access to key stakeholders is crucial for to make right decisions and good collaboration. For example, the Product Owner or main contact point might not be the domain expert but they should help the team to find them and create communication practises between the specialist teams.

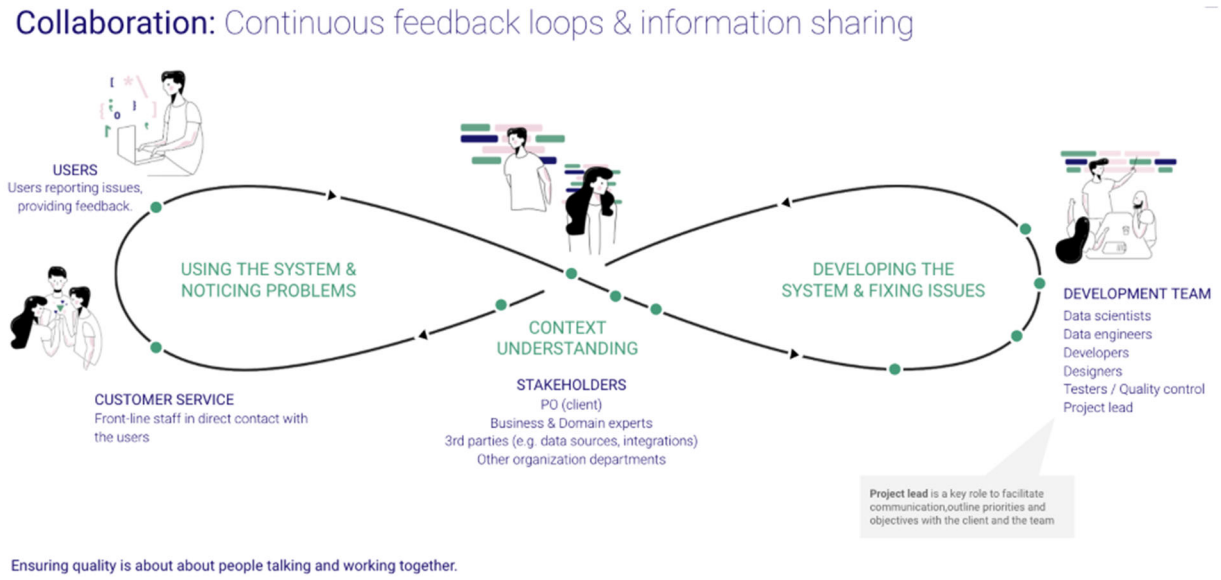

Figure 4. Maintaining and developing systems based on continuous feedback.

The business understanding creation should include the data scientist but facilitated by either a service designer or business designer (or someone who has facilitation skills and knows how to navigate the technical and business side of things). This collaboration and building common understanding among the cross-competence teams and key stakeholders is the main point to achieve good quality and learning sharing. Good collaboration ensures that constant feedback loops finds the correct parties while corrective actions and learning is happening without a heavy process and hierarchy.

The following chapters will give a more detailed description on the methods and techniques that can be used to test and validate machine learning models to ensure model and data quality meets the business requirements.

3. State of the Art Topics

3.1. Data Augmentation

Deep learning has rapidly become the state-of-the-art in medical image analysis tasks, such as segmentation, classification, and outcome prediction, displacing conventional methodologies in which manually designed feature vectors are analyzed with chosen classifiers. Theoretically, the performance of a deep learning model is directly linked to the number of nodes used in the model so that model with higher number of nodes is expected to describe the training data more closely. This leads to a need for more data so that over-fitting to the training data can be avoided and the model stays generally usable.

In many medical applications it is not practically feasible to solve the need for additional data by simply acquiring more images. For example, the disease under investigation may be so rare that additional patients are not available, imaging costs may be too high or imaging is invasive by nature. Therefore, creating additional data synthetically has attracted lots of scientific interest in recent years.

Here we concentrate mainly on improving deep learning model accuracy through data augmentation, although another valid aspect would be model's robustness or resistance to adversarial noise. Su et al. have shown that 70 percent of images in their dataset could be misclassified by changing a single pixel value. Adversarial search to add noise has been shown to improve performance on these kinds of adversarial examples, but it is unclear if this is useful for improving model accuracy on "normal" samples.

3.1.1. Augmentation methods

A good review of the current deep learning augmentation methods is given in [3], but there are not that many papers that compare the performance differences of augmentations methods using deep learning. In one such study [4] they found that cropping, flipping, WGAN [5], and rotation generally performed better than others. The combinations of flipping+cropping and flipping+WGAN were the best overall, improving classification performance on CIFAR-10 dataset by +3% and +3.5%, respectively.

Basic image manipulation

The easiest way to create new data samples is using basic geometric transformations such as flipping, cropping, rotation, skewing and translation. The transformations need to be selected so that the created data is realistic and useful for the task. For example, flipping human head MRI image in left-right axis generates probably very realistic new data whereas rotation of 90 degrees probably does not generate useful data.

In non-medical image analysis, such as machine vision for self-driving cars, it is common to use various color space transformations to simulate different lighting conditions etc. In many medical applications, images are not based on multiple colors, but different imaging modalities can be used analogously as different channels instead. For example, different MRI sequences (T1 + T2) or different modalities (CT + PET) can be combined. Many color space transformations can directly be applied to these kinds of settings.

Kernel filters, such as sharpening and blurring, can also be easily used to create slightly modified data. An obvious disadvantage of this technique is that it resembles closely to the internal mechanisms of CNNs and might actually teach the network how to perform these operations.

In a technique called PatchShuffle Regularization new synthetic images are created by swapping rectangular parts from different positions of the original image. Although improvements have been reported in other image analysis fields, the approach will probably produce very unrealistic images in medical

applications. For example, swapping patient's ear with patient's eye in an MRI image does not produce realistic data. However, in terms of deep learning model accuracy, this approach could be worth testing.

Another simplistic way to create non-realistic data would be mixing images together by averaging their pixel values. Although the images produced this way have been shown to improve CNN performance a bit in other applications, the approach may not be very useful with medical data. However, a similar but more intuitive and feasible approach would be data augmentation through elastic/non-rigid transformation. This was found out to outperform feature space augmentation by [6] in classification of hand-written numbers. With medical imaging this could be used by mapping one patient to another and taking a half-way transformation as a new synthetic data. Or, if data from multiple time-points is available, more data from a single patient could be generated by taking a half of the transformation between the data from different time points.

In [7] deformations used in augmentation are determined based on statistical models and the method was shown to improve the classification results. The method was used to segment 2-dimensional cardiac MRI images.1

Random erasing is of image data can be seen as analogous to dropout layer in the model except in the input data space rather than embedded into the network architecture. Obviously, erasing does not impute new information into the training dataset but is more designed for improving model robustness.

Instead of modifying actual image data directly, new image samples can be created by augmenting feature space data. For example, DeVries and Taylor [8] used SMOTE to create new feature vectors by combining nearest neighbor feature vectors. A disadvantage of feature space augmentation is that it is very difficult to interpret the created vector data directly and the decoded images may not be realistic at all. If the whole encoding part of the CNN being trained is used in augmentation, massive auto-encoders will be created which are very difficult and time-consuming to train. In [6] they found out that when it is possible to transform images in the data-space, data-space augmentation will outperform feature space augmentation.

Some articles have indicated that augmenting not only training data but testing data as well may improve the model's results. This can be seen as analogous to ensemble learning techniques in the data space. This may be more related to algorithm robustness than accuracy, though. Small test data augmentations can also be used to test the confidence of a given prediction.

Deep learning methods

Generative adversarial networks (GANs) are behind many examples that have received lots of public attention. For example, GANs can be used to create synthetic faces that look very realistic but do not belong to any actual human being. Thus, it is very tempting to apply GAN models also for data augmentation purposes with medical data and various methods have already been developed.

As exciting as the potential of GANs is, it is very difficult to get high-resolution outputs from the current cutting-edge architectures. Increasing the output size of the images produced by the generator will likely cause training instability and non-convergence. Another drawback of GANs is that they require a substantial amount of data to train. Thus, depending on how limited the initial dataset is, GANs may not be a practical solution. Salimans [9] provide a more complete description of the problems with training GANs.

Standard GAN network has been used in creating synthetic MRI data e.g. in [10].

ANT-GAN [11] was recently introduced to generate abnormal-to-normal translation using purpose-built GAN network. The framework is able to produce healthy-looking medical images based on its abnormal-looking counterpart i.e. brain image with a lesion. No paired training data is needed. ANT-GAN is also able to work other way around: to produce brain MRI image with artificial lesion corresponding to a healthy brain MRI image.

StyleGAN [12] is one of the latest data-driven unconditional generative image modeling approaches developed by Nvidia Corporation. There's also a further development of the algorithm called StyleGAN2 with source code available at https://github.com/NVlabs/stylegan2. Unfortunately, the source code uses TensorFlow framework, but does not support more recent TensorFlow 2.0 framework. StyleGAN2 features redesigned generator normalization, progressive growing and regularization which improves image quality and provides some other advantages as well.

WGAN [5] is a GAN architecture which is known to achieve convergence quickly without suffering the issues of vanishing or exploding gradients. The training process is known to be stable and it has also proven to generate highly diverse data samples with low noise. WGAN should be able to achieve high quality results with little hyper-parameter tuning. WGAN has been used in synthetic brain MRI image generation[10],

DCGAN [13], which was further developed into RDCGAN [14], is a direct extension of the GAN with the exception that DCGAN uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively. The discriminator consists of convolution layers with striding, batch normalization layers and LeakyReLU activation layers. The generator consists of transposed convolutional layers, batch normalization layers, and ReLU activation layers. The random input vector is drawn from a standard normal distribution. DCGAN has been used to create synthetic PET data [15], liver lesions in CT data [16] as well as brain MRI data in [17], [18].

CycleGAN [19] is one of unpaired image-to-image translation algorithm that try to learning a mapping from one image to another without perfect alignment. It has already been used in creating synthetic CT image data [20], generating accompanying synthetic CT [21] or PET [22] data based on existing MRI data,

Conditional GANs[23] are used e.g. in generating T2-weighted MRI sequences from T1-weighted sequences [24] and generating diffusion weighted MRI data from computed tomography perfusion (CTP) input [25].

AutoAugment [26] was developed to search for best data augmentation method automatically so that validation accuracy is maximized. The augmented data consists of image batches with each batch augmented individually using basic image processing functions such as translation, rotation, or shearing. Note that because optimal augmentation has to be searched for every image batch, AutoAugment is a very expensive algorithm requiring huge amount of training. AutoAugment has been used in grading glioma MRI images in [27].

3.1.2. Generation of Synthetic MRI Data

In [17] brain MRI images were generated using standard DCGAN [13] network with only slight modifications. The population consisted of both healthy patients and patients with a history of cerebrovascular accident. Unfortunately, the source codes or the trained model has not been shared publicly. But some implementation of standard DCGAN are available on the internet and could be worth testing.

In [10] Han et al generated brain MRI images that an expert was not able to distinguish from the real ones in the Visual Turing Test. However, the images were only 128×128 pixel slices and not three-dimensional data. Combining individually synthetic slices into a 3D volume probably results in a non-realistic result.

The only GAN method producing 3D brain MRI data from a random noise vector we have found is presented in [18].

Residual cyclic unpaired encoder-decoder network (RescueNet [28]) was recently developed for more accurate brain tumor segmentation. The problematic need for large scale labeled data for training of deep

segmentation network is solved in this approach by using unpaired training. The approach seems to outperform other compared methods and achieves Dice index 0.94 when compared to the manually determined ground truth.

For liver tumor diagnosis Zhao et al [29] developed a method called Tripartite-GAN which generates contrast-enhanced magnetic resonance images (CEMRI). However, this method requires standard MRI image as an input, so the method does not suit very well for data augmentation purposes.

One interesting aspect comes from a cyber security conference paper [30] in which they tried to “attack” the medical data system by altering images with lung cancer. They were able to both inject lung cancer into images of a healthy person as well as remove lung cancer from images of a patient without radiologist being able to spot the modification.

3.2. Testing and Validating Machine Learning Systems

The techniques and methods discussed below are based on the scientific literature and some of the most commonly used approaches. Since the focus point here is the state of the art methods, which are presumably evolving and emerging with time, these discussed methods and approaches just give an overview. For practical purposes, these methods can be used as baseline depending on the problem in hand. These approaches also open doors for further research in the field of validation and verification of machine learning systems.

With the tremendous advancements in the field of machine learning in the last few years, some serious impediments are raised to their widespread adoption. The challenges arise because of the inherent uncertainty of the behaviour of the algorithms. Current machine learning model assessment techniques, such as cross-validation, do not provide any insight about the possible aberrations that can occur in large, complex, and ever evolving systems.

Since machine learning models are largely dependent on data, and based on the configuration settings, these models can affect various characteristics of the software systems such as security, interpretability, and scalability. In the absence of any test oracle, performing any tests on these models is a very challenging task. To troubleshoot and debug, three key components need to be considered when testing machine learning systems. These components are data, model, and infrastructure.

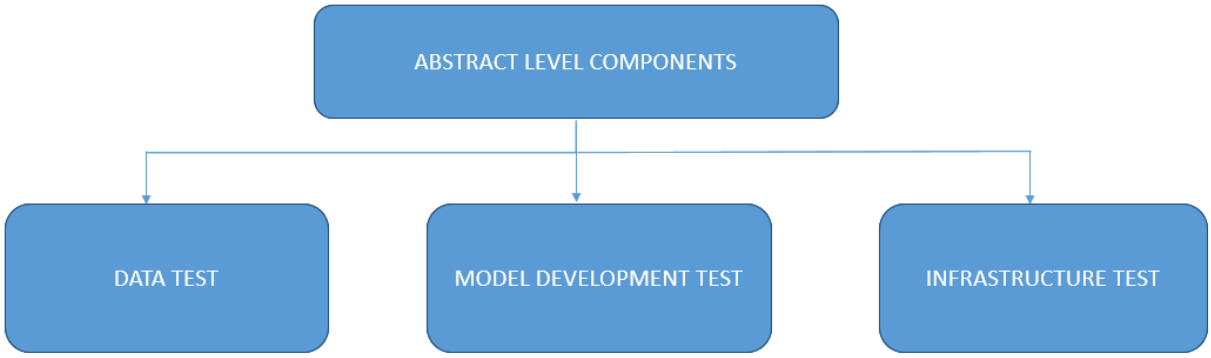

Figure 5. Basic Components of Machine Learning Model Testing.

From the perspective of testing the properties of machine learning models, security, efficiency, fairness, transparency, robustness etc. are being studied by the research community. A recent survey [31] gives a

brief description of the methods being used to test components and the properties of machine learning systems. It covers over hundred papers on:

- testing machine learning components (data, model etc.),
- properties (security, scalability),
- testing workflows (test case generation, evaluation), and
- application scenarios (machine translation, autonomous driving etc.).

This survey covers state of the art methods and systems to test classical machine learning models as well as more complex deep learning systems.

DeepXplore and DLFuzz, which are the deep learning system testing techniques based on neuron coverage to generate test input and adversarial examples respectively, are two of the methods to test deep learning systems. Similarly, for classical machine learning methods such as Support Vector Machines, metamorphic relations can be used to uncover the implementation bugs by mutating the features or instance orders. For data testing, a machine learning tool to inspect dataset, *data linter* can be used. It inspects miscoded data, outliers and scaling errors in the data.

These were just brief descriptions of the some of the methods and systems discussed in some detail in the survey. Following are detailed analysis of the techniques and state of the art systems for testing machine learning systems.

3.2.1.Data Testing

The fundamental parts of any machine learning system are the data and the algorithm. When validating and verifying machine learning systems, focus is mostly on the accuracy of the model. Data, which is as important as algorithm itself, often gets neglected in validation and verification procedures even though the whole machine learning system is built on it.

Following are the key components to consider while testing data.

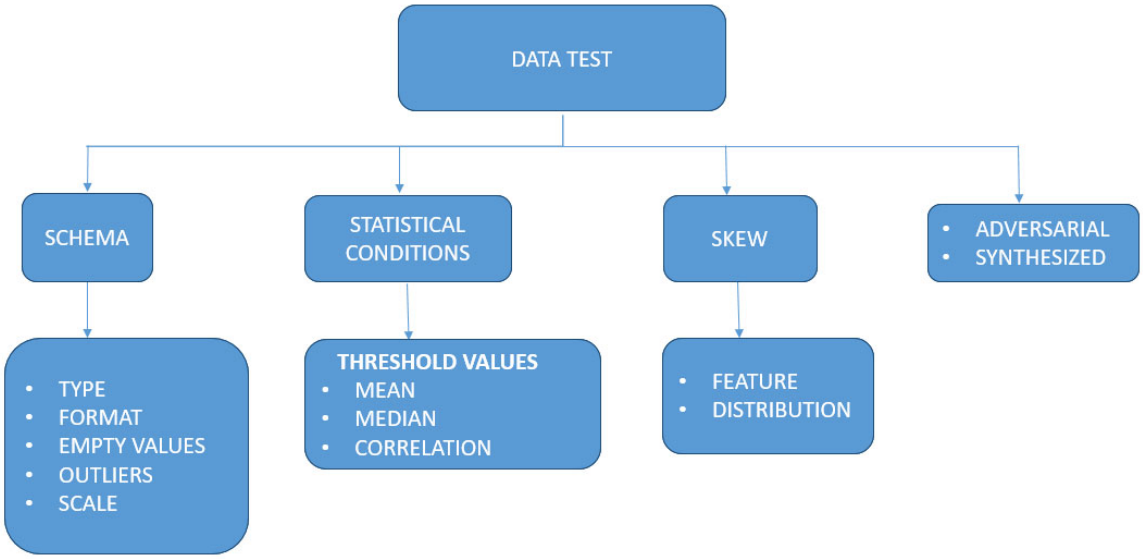

Figure 6. Data Testing Approaches.

A data validation system is proposed in [32]. The system consists of three components: *Data Analyzer, Data Validator,* and *Model Unit Tester*. Data Validator checks the properties of data against a provided

schema. The schema can have data types, data formats and any other conditions on the features such as ‘not-null’ or ‘not-duplicate’ etc. This component produces a baseline schema, which attempts to capture significant properties of data. With the new data ingestions, Data Validator recommends updates in the initial schema.

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

For the data validation, [33] has also focused on schema testing, suggesting some properties to be encoded in the schema. These properties are:

- types of features,
- expected presence of each feature in terms of minimum count and examples,
- minimum and maximum values of a feature such as numeric fields, and
- lengths of feature values such as string length.

The authors suggested to flag any deviations of data from schema as anomalies. These methods proved to be helpful in understanding the evolution of data Schema and also serve as a stable description of the data, which can drive other components such as feature-engineering.

The above-discussed schema testing approach depends on a user’s/developer’s domain knowledge but most of the time, especially in production environment, the schema and format of data is not known in advance. To tackle this issue, a system is proposed to automate large-scale data quality verification [34]. Though the proposed system considers different approaches to test data quality such as data consistency, accuracy and completeness, the focus is on constraint suggestions using a heuristic approach. This system assumes that the data is provided without any schema or data type information. The system first computes data size, then runs data type detection on each column, and computes the number of distinct values for each column. Next, it computes the summary statistics such as mean, standard deviation, approximate quartiles of each numeric column. Then it computes the frequency distribution of each column. Using the results of above steps, the system recommends constraints for the data set, such as: a value should not be null, the range of values, the data type, etc.

Statistical conditions are another layer of validation on a data set. For this purpose, a Data Analyzer approach is proposed in [32]. The Data Analyzer calculates a predefined set of statistical conditions such as mean, median, mode etc. The data is validated against these properties based on threshold values. The threshold values and the range of the values are domain specific. This method not only provides scalability but also helps in detecting anomalies as early as possible.

An important aspect of data validation in production is the discrepancies between training and real data. Discrepancies occur when a particular feature gets a different value than the training value or when the distribution of feature values is different from the one the model is trained on. Discrepancies can occur for many reasons, for example, a feature is added or removed at development side but not on production, or sampling data scheme has some flaws which results in training data distributions that look different from real data

The Data Validator [32], as discussed above for schema testing, also monitors skew between training and serving data continuously. Data Validator compares incoming data batches with training batches continuously and detects if there are disparities. It does a key-join between training and serving data batches followed by feature wise comparison. To quantify the skew in the distribution, the authors in [32] argued that general approaches such as cosine similarity or KL divergence might not be sufficient. They

proposed to use the largest change in probability as a value in the two distributions as a measurement of their distance.

In machine learning systems testing, the most important and least implemented test is the training/serving skew as these errors are responsible of production issues. Skew between different data sources can be tested by computing distribution statistics [35]. The distribution statistics includes min, max, average values, fraction of missing values etc. To get an alert on a metric, a threshold value should be tuned carefully to ensure a low enough false positive rate for a response.

To evaluate the skew between training and serving data, [36] proposed two measurements focusing on Deep Learning systems. Kernel Density Estimation (KDE) to approximate the likelihood of the system having seen a similar input during training. Another is based on the distance between vectors representing the neuron activation traces of the given input and the training data (e.g., Euclidean distance).

3.2.2.Model Testing

Generally speaking, traditional software systems are tested against a given set of possible outcomes. The procedures are known in advance and the output is compared with the expected output. This knowledge of expected output makes it easier to test a traditional system.

However, when testing machine learning systems, the absence of any test oracle makes testing very hard. Various possible methods are studied but the foundation was laid in [37] with a proposal of *Metamorphic Testing Technique.* Before discussing Metamorphic Testing in detail, following are the key components and approaches to test machine learning models.

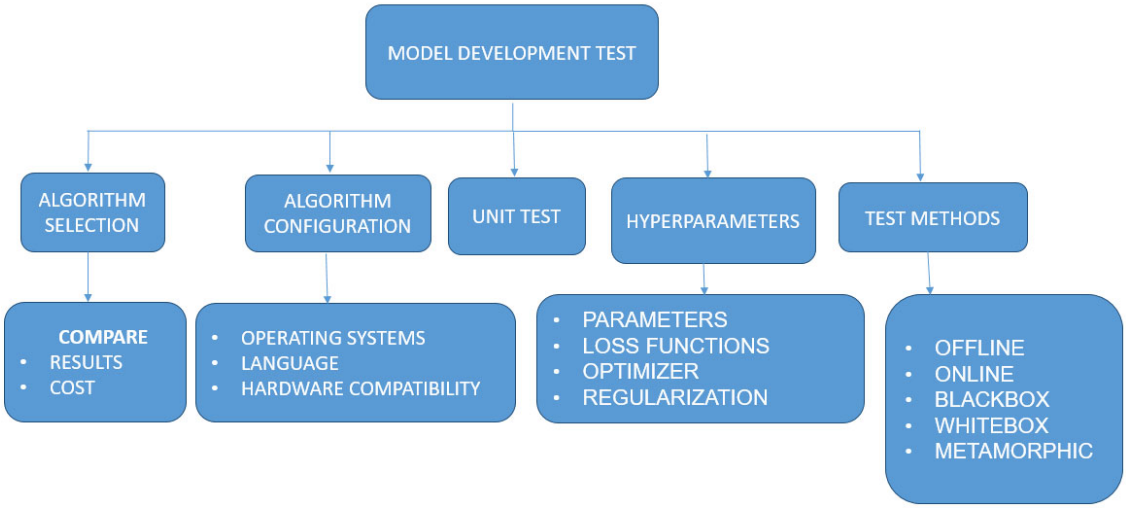

Figure 7. Machine Learning Model Testing Approaches.

In general cases, metamorphic testing 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.

Applying this general approach in machine learning systems, authors in [37] first tested it on a Ranking system. They argued that instead of original values of attributes, their relative values determine the model. The assumption from this argument is that adding any constant value to every feature, or multiplying each

feature by a constant should not affect the model performance. Depending on the domain knowledge, metamorphic testing can also validate the permutation property of the model. If the model is built without any specific order of attributes, changing the attributes' order should not affect the model. Similarly, by taking the opposite of an input, the output can be predicted, which is the *invertive* property of metamorphic testing.

Two more properties of metamorphic testing, inclusive and exclusive, can also be tested based on the domain knowledge and model's behaviour. 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.

The above described concept of metamorphic testing is extender further [38] for various machine learning systems by adding detailed metamorphic relations based on the basic six properties. The added properties are permutation of class labels, permutation of attributes, addition of informative and uninformative attributes, addition of training samples, removal of classes and samples. This approach is evaluated for the *k-nearest neighbour* and *naïve bayes classifiers.*

The experimental study uncovered some violations in metamorphic relations for both of the classifiers, though some violations were described as unnecessary properties for the algorithms. The violations, which were considered as necessary properties of the models, indicate defects in the implementation of the models. These violations fall under the verification category. Some violations, which were found in the metamorphic relations, were considered as not necessary properties of the algorithm. These violations do not indicate the defects in the implementations of the algorithms but shed light to support the validation of the model. This research work focused on two specific classifiers though, but the metamorphic relations experimented here can be studied for any machine learning system.

A lot of research in the field of ML testing is focusing on metamorphic testing, but this is not the only studied approach. Multiple implementations is another approach to test machine learning systems. In this approach, the same algorithm is implemented multiple times. The testing of multiple implementations works under the observation that running the test input on multiple implementations of the same supervised learning algorithm, and then using the common test output produced by a majority (determined by a predefined percentage threshold) of these n implementations. *This method is evaluated for supervised learning systems in [39] by iterating four steps:*

1. Determine parameters for the Implementation Under Test (IUT),
2. Collect multiple implementations,
3. Collect data sets and create test inputs, and
4. Run test inputs on the multiple implementations to detect deviation.

The authors used various open source ML projects such as Weka, RapidMiner, and KNIME and tested 19 implementations of K-Nearest Neighbour (KNN), and 7 implementations of Naïve Bayes (NB). Authors focused on three research questions:

- How effectively can Majority-voted Oracle help detect faults?
- How much more effectively can Major-Oracle help detect faults compared to Benchmark-listed Oracle (BenchOracle)?
- How does the choice of data sets impact fault detection effectiveness when using Major-Oracle or Bench-Oracle?

The results show that their proposed approach of the majority-voted oracle, produced by multiple implementation testing, is an effective proxy of a test oracle. The results show that the majority-voted oracle has low false positives and can detect more real faults than the benchmark-listed oracle. In particular, the approach detects 13 real faults and 1 potential fault from 19 KNN implementations and 16 real faults from 7 NB implementations. Their proposed method detected 7 real faults and 1 potential fault among the three

popularly used open-source ML projects. This method is a good baseline for algorithm evaluation in the absence of test oracle.

Even though testing a machine learning system is a difficult task since it comes without any oracle, it becomes even more difficult when testing an *unsupervised* machine learning system. Of unsupervised machine learning systems, clustering is the widely use technique. Generally speaking, there are two possible techniques to validate clustering results, external validation and internal validation.

In external validation, the clustering results are compared against an external measure or benchmark. These benchmarks focus on homogeneity or compactness of the clusters. F-score or f-measure is one such benchmark, which validates clusters based on Recall and Precision. This was an example, but in general, good benchmarks are hard to find because of no knowledge of data structure.

In internal validation, clusters are validated by adopting features inherent to the data, such as inter-cluster compactness and intra-cluster separation. These techniques can be applied without any oracle, but they are not robust as they are associated with the features. Any change in the features may have a very significant effect on the results. As both of these validation techniques have their own limitations, a method based on metamorphic testing is proposed in [40].

The proposed methods have several metamorphic relations to test clustering algorithms. These relations are:

- changing object order,
- changing object orders but keeping the centroids,
- shrinking clusters towards their centroids,
- mirroring the data,
- adding sample objects to cluster's centroid and near a cluster's boundary,
- adding informative attributes and removing redundant attributes,
- rotating and scaling the coordinate systems, and
- inserting outliers.

By focusing on the user's perspective, the METTLE method proposed by authors assesses and validates clustering systems with respect to a set of system characteristics defined by the user.

Two main research questions were studied, performance of clustering system against proposed metamorphic relations and underlying behaviour causing the violations in these metamorphic relations. Six clustering algorithms were selected for the experimental study. These algorithms include:

- K-means,
- X-means,
- Expected Maximization (EM),
- Agglomerative Nesting (AN),
- Farthest-First Traversal (FF), and
- Density-based Spatial Clustering (DS).

Authors conducted 100 trials with different datasets to find the answers of two research questions discussed above. To evaluate performance of clustering systems, results show that k-means violated nine metamorphic relations, whereas FF and DS violated seven metamorphic relations each. X-means and EM violated five and four metamorphic relations respectively. AN performs best against these metamorphic relations by violating only three. Since the metamorphic relations are involved in data transformation, so K-means is more sensitive to data transformation whereas AN is the least sensitive. These violations are tested against the parameter, reclustering percentage, where a sample was reclustered. A reclustering percentage of greater than zero indicates a violation of metamorphic relation.

To find the answer of the second research question, uncovering the underlying behaviour which causes the violation of metamorphic relations, five clustering patterns were visualized. These patterns were:

- Border (a sample on border is assigned to another cluster in the follow-up dataset),
- Merge & Split (two clusters are merged in the follow-up and another split into more clusters),
- Split (one or more clusters are split in follow-up),
- Noise (reclustering occurred for those samples which were considered as noise),
- NUM (number of clusters are different after the follow-up).

All six clustering algorithms were found to be related to Border pattern while only K-means and FF were related to Merge & Split. EM and AN had metamorphic relation violations with Split pattern and DS metamorphic relations were violated by Noise pattern. DS also had violations of metamorphic relations because of NUM pattern. To sum up the results of the testing clustering algorithm, reordering data and adding noise will reassign the samples near boundary. K-means tends to local optima when the clusters are well separated which will result in reclustering, while X-means is less sensitive to local optima. EM is the most robust among six algorithms, whereas AN is more robust than FF with data transformations. FF is more sensitive to reordering the data and noise. All algorithms were found to violate metamorphic relation of changing coordinate system, which was caused by min max normalization. This method not only provides a detailed clustering validation technique but also helps the end users to gain a better understanding of the program under test.

Deep learning (DL) systems are increasingly used in various fields. However, the DL models are difficult to test and existing DL testing relies heavily on manually labelled data and often fails to expose erroneous behaviour for the inputs on the border. To unravel this problem, Differential Combination Testing (DCT), an automated DL testing tool for systematically detecting the erroneous behaviour of more corner cases without relying on manually labelled input data or manually checking the correctness of output behaviour is proposed in [41]. This system is divided into three sections:

- differential testing,
- neuron coverage, and
- image combination transformations.

Authors applied differential testing using multiple DNNs with similar functions as cross-references, so that the test inputs are not necessarily labelled images and DCT can automatically check the correctness of output behaviours. To measure the test degree of the internal logic of the DNN, the authors used the neuron coverage method. In the next step, combination transformations are applied to generate synthetic realistic images and then the feedback is predicted.

The system is evaluated using MNIST and ImageNet datasets with six different image transformations. These transformations are translation, scale, rotation, contrast adjustment, brightness adjustment and blur. The results show that by randomly combining six different transforms, thousands of valid synthetic images that represent the erroneous behaviour of DNN models were generated. The results indicate that as the kind of random image combination transformations increases, the synthetic images that detect the erroneous behaviour of the DNN models also increase significantly.

To evaluate the performance authors used two metrics, Neuron Coverage and Execution Time and used four different methods such as DCT, DeepXplore, adversarial testing, and random selection testing. The results show that the DCT covers an average of 8.25% and 9.3% more neuron coverage than adversarial testing and random testing, while DCT and DeepXplore are comparable in neuron coverage and can be both used for more thorough testing on neurons.

When comparing the execution time, the results prove that no matter what kind of image combination transformations the DCT runs, it is much shorter than the execution time of DeepXplore. This is a very good approach to test the deep learning system but the system is only evaluated with image dataset. It still needs further extensions to be used as a generic system for differential testing.

With the advancement in machine learning, integrated and embedded systems are emerging rapidly. These systems perform sophisticated tasks by executing analytical pipelines of components. These systems are gaining popularity worldwide, but still lack the ability to diagnose and fix the errors. The problem of understanding and troubleshooting failures of machine learning systems is of particular interest in the research community. Since the components are integrated, identifying and blaming a specific component in case of a failure is very hard as failure can occur at multiple points in the execution workflow.

To tackle this problem a troubleshooting methodology that relies on crowdworkers to identify and fix faults in the system is presented in [42]. The crucial part of this approach is human intervention. Human fixes simulate improved component output that cannot be produced otherwise without significant system development efforts. First, workers evaluate the system output without any fix to analyse the current system state. To simulate a component fix, the input and output of the component accompanied with the fix description are sent to a crowdsourcing platform as microtasks. Once workers apply the targeted fixes for a component, the fixed output is integrated back into the running system, which thereby generates an improved version of the component. The system is executed as a simulation with the injected fix and the output is evaluated again via crowdworkers. The overall process collects a valuable set of log data on system failures, human fixes, and their impact on the final output. This data can then be analysed to identify the most effective combination of component improvements to guide future development efforts.

The above proposed method answers two main questions: How does the system fail and how to improve the system. The evaluation of machine learning systems goes beyond the accuracy and requires a deeper understanding of the behaviour. The workflow of this approach to understand the behaviour is based on continuous evaluation. To troubleshoot, designer chooses which fix workflows to execute and evaluate. The proposed method has four steps:

1. Workers evaluate the final output of the current system on different quality measures.
2. Workers complete the respective micro-task for examining and correcting the component output for each fix in the workflow.
3. Executing a fix workflow involves integrating the corrected outputs of each component into the system execution.
4. Workers re-evaluate the new system output after incorporating component fixes.

These four steps are iterative, which helps in not only fixing the problems, but also in understanding the behaviour of the system under various combinations of fixes. The authors evaluated the method on image dataset, but it can be applied to a broad range of component-based systems that are designed to be modular and their component input / output is human-interpretable.

3.2.3. Testing AI Systems

As stated earlier that the survey [31] provides a brief description of the methods being used to test properties and components of machine learning systems, but it does not focus on the testing of complete applications. A recent, still ongoing study on the state-of-the-art validation methods for complete Artificial Intelligence (AI) systems and application fills the void left in [31]. This study gives a systematic literature review of the methods being used to validate AI systems and applications. It covers the most common applicability areas (domains) of the AI systems (such as Wearable AI, Robots, Smart Environment etc.), and what are the actual objectives of the AI systems (such as Maintenance, Security, Recognition and Classification etc.). It also takes into account what AI systems are considered in these state-of-the-art methods. These systems are *Model*, a simple and complete system with AI model and a user interface, *Systems* consists of two components working together, and a *Multi-Component* system, which consists of multiple components and possibly multiple AI models.

This study indicates that the most common validation methods are Trial, Simulation, and Statistical proof. It shows that Robot is the most frequent used domains in these systems and recognition and classification are the most common objectives of these AI systems. The result shows that Systems were the mostly

studied complexity levels in these papers followed by more complex *Multi-Component* systems. This study provides a holistic view of the methods which are seen to be suitable to answer the challenges of vagueness in AI. It also indicates that there is room and desire in the research community to study continuous validation methods for AI systems and applications.

3.2.4. Testing Machine Learning Properties

The testing infrastructure of a complete deployed system is a vital part in traditional software development. Since machine learning models can be a sub-part of a complete system, testing the models' properties is the next essential phase. Machine Learning properties concern the conditions we should care about for ML testing and are usually associated with the behaviour of an ML model after training.

These properties can be functional, such as Correctness and Over/Under-fitting or non-functional such as security, interpretability, scalability etc. The survey [31], contains the literature review of testing these properties. Generally, correctness of an ML model concerns the accuracy of the model for an unseen data, but the drawback of this method is that it cannot differentiate between the error types (False Positive or False Negative).

The commonly used correctness measure metrics are cross-validation, bootstrap, precision, recall etc. These approaches have their limitations as they make some assumption about the state of the data. For example, Precision and Recall may be misled when data is unbalanced. Some other approaches are being studied in the research community such as creating a mirror program from the training data, and then use the behaviour of the mirror program to serve as a correctness oracle. The mirror program is expected to have similar behaviour as the test data. Other approaches focus on the variability between training and testing data, detecting bugs that lead to low correctness etc.

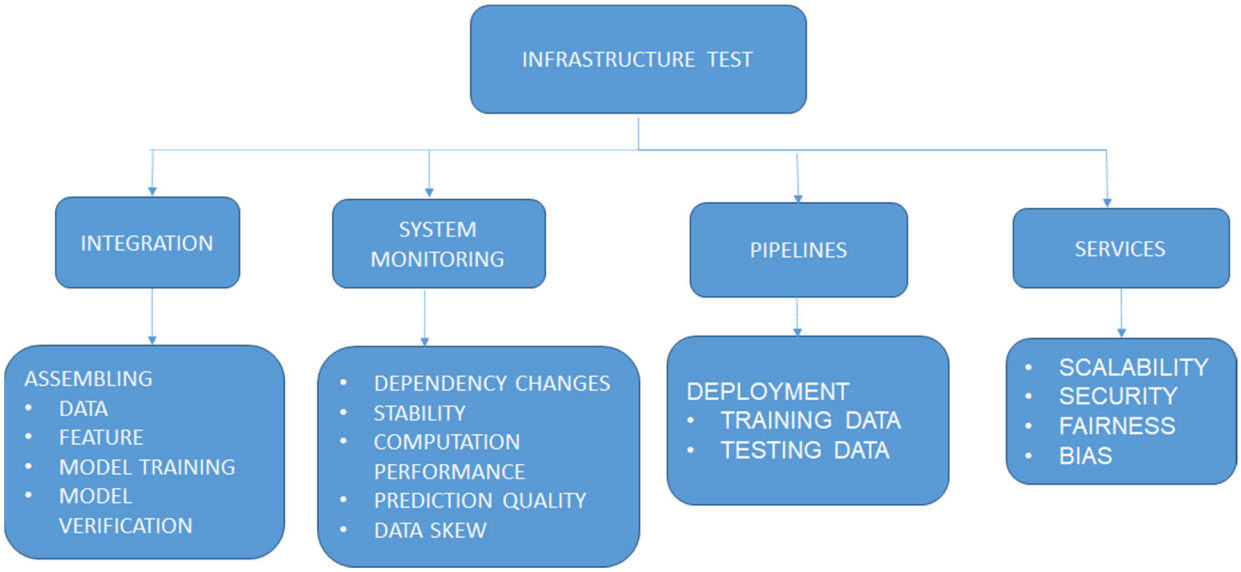

Figure 8. Infrastructure Testing of Machine Learning Systems.

Another property to test is the relevance of an ML model. A poor model relevance is usually associated with overfitting or underfitting and cross-validation is the most widely used method to evaluate model relevance. If the test data is unrepresentative of potential unseen data, cross-validation may not detect overfitting.

One of the alternate methods to evaluate model relevance is to inject noise to the training data, re-train the model against the perturbed data, and then use the training accuracy decrease rate to detect overfitting/underfitting. This method is based on the intuition that an overfitted model tends to fit noise in the training sample, while an underfitted model will have low training accuracy regardless of the presence

of injected noise. So, both overfitting and underfitting tend to be less insensitive to noise and exhibit a small accuracy decrease rate against noise degree on perturbed data. Various alternate approaches are discussed in [31], such as, generating of adversarial examples, re-sampling the training data, and using time complexity of an ML model to select an equal correctness with less training time.

Robustness and Security are non-functional properties of machine learning models and a very wide research domain in itself. At its core, robustness is the correctness of a system with the existence of noise [31]. Robustness can be measured by three basic metrics:

- Pointwise robustness, which indicates the minimum input change where the classifier fails to be robust,
- adversarial frequency, which shows how often changing an input changes a model's output, and
- adversarial severity, finding the distance between an input and its nearest adversarial example.

There are several approaches discussed in [31] to calculate robustness, one such approach is creating a set of attacks to construct an upper bound on the robustness of an ML model. The existence of adversarial examples can cause serious consequences in safety-critical applications such as self-driving cars. The study of these adversarial examples for critical applications needs a complete and thorough review of literature but the methods that could be useful areas for future research at the intersection of traditional software testing and machine learning are discussed briefly in this section.

To test the security of an ML model adversarial examples can be generated using distance metrics to quantify similarity [31]. Another approach is to standardize the implementation of adversarial example construction and create a benchmark to distinguish whether a good result is caused by a high level of robustness or by the differences in the adversarial example construction procedure.

Usually the interpretability of machine learning models includes humans in the loop. This manual approach is the primary approach of interpreting an ML model [31]. The taxonomy of interpretability consists of three approaches human-grounded, application grounded and functionally grounded.

The first two approaches involve humans in the experiment to evaluate the interpretability whereas the functionally grounded approach uses a quantitative metric as a proxy for explanation quality, for example, a proxy for the explanation of a decision tree model may be the depth of the tree. In the automatic assessment of interpretability, one approach is to use a metric, which measures whether the model has learned the object in object identification scenario via occluding the surroundings of the objects. Alternatively, the concepts of Metamorphic Relation Patterns (MRPs) and Metamorphic Relation Input Patterns (MRIPs) can be adopted to help end users understand how an ML system is working. In some applications such as medical, interpretability can be achieved and improved by transforming classifier scores to a probability of disease scale. This can be a useful approach if classifier scores on arbitrary scales can be calibrated to the probability scale without affecting their discrimination performance.

3.2.5. Model Performance Evaluation

Model evaluation is a key step in testing how the model generalises on unseen data, this determines the accuracy of the model. There are two main ways to determine the model performance using a test set that has not been seen by the model [43].

1. **Holdout:** this method tests the model on different data to which it was trained on, to avoid overfitting. The dataset is randomly divided into three sets; training – a set to train the model, validation – a set to asses performance during training in order to hyper-parameter tune and a test set – to test the model on unseen data for future use.
2. **Cross-Validation**: this method is a resampling technique that subsets the data into a training set and a validation set where K is the number of subsets to be created and K-1 is the number of training subsets and the remaining subset is used to test.

When evaluating and tuning the performance of computationally intensive machine learning models, the amount of computational work required can easily be non-trivial and the amount of available processing capacity can become a limiting factor in the tuning process. In these types of situations, parallel processing techniques can be used to greatly improve the performance of the hyper-parameter tuning process and have the potential to greatly help researchers and business practitioners alike in the execution of analysis tasks. [44]

Model performance metrics are used to determine accuracy. The metrics are reliant on the context of the machine learning model and task (regression, classification, clustering etc.). Some metrics to determine supervised tasks include [43]:

1. Classification task metrics:
    a. **Classification accuracy** – a ratio of the number of correct predictions made to all the predictions made
    b. **Confusion matrix** – correct versus incorrect classifications for each class
    c. **Logarithm loss** – predicted probability is between 0 and 1. The log loss increases as the predicted probability diverges from the actual labelled class, so the goal is to decrease log loss.
    d. **AUC** - the ability of the binary classifier to discriminate between positive and negative classes
    e. **F-score** - a measure of a test's accuracy that considers both precision (number of correct positive results as a ratio of the total predicted positive samples) and recall (number of correct positive results as a ratio of total actual positive samples) of the test to compute the score.

2. Regression task metrics:
    a. **Root mean-squared error (RSME)** - the average magnitude of the error by taking the square root of the average of squared differences between prediction and actual observation
    b. **Mean absolute error (MSE)** - the sum of the absolute differences between predictions and actual values

3.3. Explainable Artificial Intelligence

3.3.1. Introduction

Over the last decade, Artificial Intelligence (AI) has dramatically entered our everyday life due to its capabilities and the emerging need for automating decision-making. Undeniably, AI systems are already in use in various domains for decision-making ranging from simple decisions like recommender systems in online trading and personal assistants to more critical decisions such as insurance risk, healthcare, and law. AI methods such as Machine Learning (ML) algorithms have produced remarkable development in decision-making systems in terms of prediction performance. However, a clear justification for such decisions is needed, especially for the critical ones made by AI systems without human involvement. Although ML models have provided high accuracy results, they tend to be opaque, which makes it challenging to comprehend their internal mechanisms or to explain their general behaviour. [45]

Artificial intelligence, predictive machine learning in the context of this report, is quite often seen opaque method, which takes data in and outputs some values. Between these two phases, there is a *black box* that solves the mapping, the equations. This is partially true, especially when utilizing non-linear models state of the art, which can be notoriously difficult to interpret if compared to the more traditional linear counterparts. However, the algorithms, and the models, which are manifestations of the algorithms, can be analysed, interpreted, and sometimes *explained* even in human-friendly fashion.

Since diverse areas employ AI systems, researches must tackle the explainability challenge from different angles as each application domain has its own needs and goals. Furthermore, different audiences require different kinds of explanation. For instance, some of the standard terms used in the literature to refer to XAI goals are *trustworthiness*, *fairness*, *causality*, *transparency*, and *justification*. All these indicate the importance of explainability in practice. Developers can utilize explainability not only to understand the *black box* model but also to debug the system by detecting unexpected model behaviour or any bias in the training dataset or in the model. Hence, AI explainability can also facilitate in improving the model and retrieve new hidden characteristics. [46]

In this report, a brief state of the art study is documented, which shows how to shed light on, sometimes, opaque machine learning models. Instead of providing a comprehensive state of the art study on the whole vast field of explaining features, samples, models, and predictions, we will very briefly discuss and list some of the familiar means for explaining machine learning models. This report heavily leans towards discussing Explainable AI (XAI) from a practitioner's perspective; otherwise the methods can remain at a bit theoretic level, or their scope can remain vague. This is achieved by approaching the XAI from a demonstration viewpoint. It depicts how some of the XAI methods can be utilized in common situations one faces in many machine learning projects.

The demonstration is about a naïve stock trading algorithm. It functions as a part of an overall trading system that is excluded from the discussion. The demonstration depicts 1) how to select relevant features for the model, 2) what is their order of importance, 3) how they contribute and interact globally, and also 4) how to explain the predictions sample-by-sample so that one can study every single prediction separately which facilitates further analysis. These steps are the ones that are the most likely steps in every solemn machine learning project.

The rest of this report is organized as follows. In Section 2, XAI related terminologies, taxonomy, and some of the available XAI techniques for explaining machine learning models, and their implementations are presented. Section 3 introduces the demonstration of some of the widely used XAI methods and their use in a practical manner. Discussion and summary close this report.

3.3.2. Explainable Artificial Intelligence

XAI, eXplainable Artificial Intelligence, refers to techniques that interpret the predictions of machine learning systems in understandable terms to humans. However, there is no standard definition for XAI stated by researchers. Moreover, different researching groups approach the explainability challenge from various perspectives. As a result, various exchangeable terms are utilized to refer to XAI such as *Interpretable ML*, *Explainable AI*, and *Transparent AI*. However, those mentioned above have a common goal, which is to improve the trustworthiness of AI-systems and provide clear explanations of AI decisions to a non-technical audience.

In Machine Learning, some models are interpretable models by design, which might have different levels of transparency. Such as algorithmic transparency, simulatability, and decomposability. This category includes Linear/Logistic regression, Decision trees, K-Nearest Neighbours, Rule-based learning, General Additive Models, and Bayesian models. On the other hand, there are algorithms such as Artificial Neural Networks (ANN), which cumbersome to analyze and interpret due to often consisting of millions of parameters. Thus, XAI is a multifaceted challenge consisting of numerous sub-fields and challenges, as depicted in the XAI Taxonomy [46] [47].

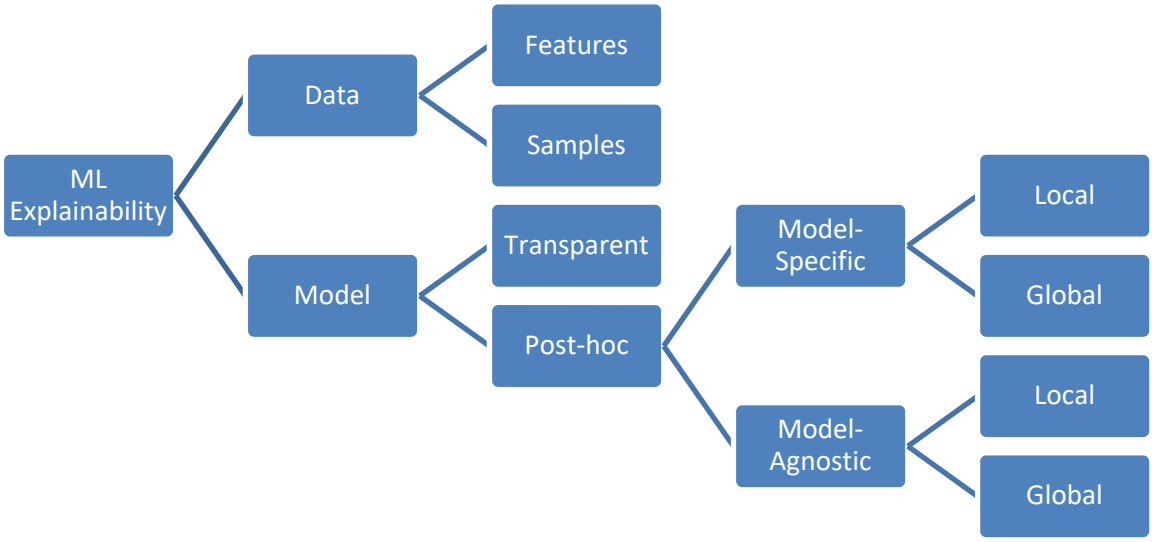

Figure 9. The XAI taxonomy.

Taxonomy

In general, explainable ML can be divided into explaining 1) the data itself, and 2) the model. As the model is based on the data, these two sections overlap and intersect. Explaining the data is about explorative data analysis (EDA). It consists of numerous techniques, including statistical analysis and unsupervised machine learning methods such as clustering, latent variable analysis, and anomaly detection. In addition to explaining the data in isolation, the data can reveal more from a defined perspective when utilizing means of model interpretation. Additionally, IBM AI Explainability 360 toolkit [47] provides explicit data explanation algorithms, namely ProtoDash, to explain samples and DIP-VAE to explain features. Both will be discussed later in this document.

The list of XAI techniques that can be applied is extensive and the techniques and methods can be divided into several different categories based on different criteria. Broadly speaking, XAI techniques can be divided into **ante-hoc techniques** and **post-hoc techniques**. Ante-hoc techniques ensure clarity of the model from the start, and post-hoc techniques ensure clarity and understanding during the training and testing phase of the model development cycle [48].

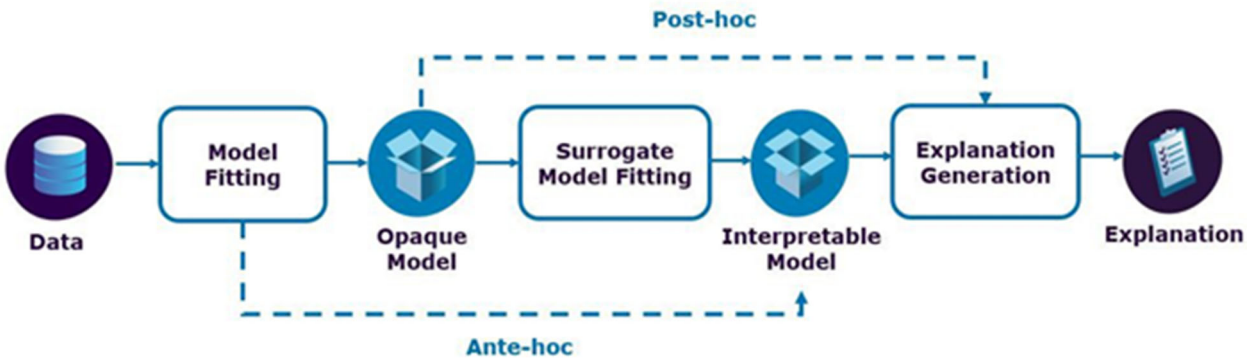

Figure 10. Model Interpretation Process [48].

1. ***Ante-hoc technique:*** *An example of an ante-hoc technique is Bayesian Deep Learning (BDL). With BDL the degree of certainty of predictions by the neural network is displayed. Weight distributions are assigned to the different predictions and classes. This provides insight into which characteristics lead to which decisions and their relative importance.*

2. ***Post-hoc technique:*** *An example of a post-hoc technique is Local Interpretable Model-Agnostic Explanations (LIME). LIME is mainly used for image classification tasks and works with the help of a linear interpretable model. To find out which parts of the input contribute to the prediction, we divide the input into pieces. When interpreting a photo, the photo is divided into pieces and these "disturbed" pieces are exposed to the model again. The output of the model indicates which parts of the photo influence the correct interpretation of the photo. In addition, insight is gained into the reasoning behind the algorithm decisions.*

   Post-hoc methods and tools can also be classified into model-agnostic and model-specific methods. Whereas the first-mentioned refers to techniques that can be applied to any kind of ML model, the latter is for specific ML models.

Additionally, the explanation scope can further be divided into local and global interpretations. While local interpretation is for explaining a single prediction or output decision, global interpretation is for explaining the behaviour of the whole model on average. Both local and global model interpretation methods can also be utilized to understand the data itself better, thus providing a loop-back to the data interpretation.

Algorithms for XAI

XAI is a young yet rapidly growing discipline that has attracted the attention of numerous parties. Further, XAI began to frequently appear in many conferences as well as its devoted events such as the ICML Workshop on Human Interpretability in Machine Learning (WHI) [49] and Fairness, Accountability, and Transparency in Machine Learning (FAT-ML) [50]. Consequently, several ML explainability contributions are continuously being added to the field with many novel approaches.

The more significant part of the XAI methods focuses on interpreting complex ML models by simplification. Either by approximating the model partially by a simpler linear model. Also known as surrogate models that represent complex models by interpretable ones. Alternatively, by extracting rules that lead to a particular result (e.g., G-REX and Local Interpretable Model-agnostic Explanations LIME). Other techniques utilize visualization as their main tool for explainability to represent the hidden patterns inside the black box. For example, some methods have extended Sensitivity Analysis (SA) visualization capabilities [51] [52], also [53] discusses Individual Conditional Expectation (ICE) plots that enhances the classical Partial Dependence Plot (PDP) to visualize model estimated by any supervised learning algorithm. However, visualization tools usually appear as a complementary part of a feature relevance technique.

Feature relevance methods explain the black box by ranking each feature's contribution to the final prediction obtained by the model (e.g., Quantitative Input Influence QII, Automatic STRucture IDentification ASTRID, SHapley Additive exPlanations SHAP). Another interesting approach for ML explainability is Example-Based Explanations. It picks a specific instance of the data to explain the model (e.g., Counterfactual explanations [54]) or explain the underlying data distribution (e.g., Prototypes and criticisms [55]). For instance, Counterfactual explanations outline the minimum changes that would have led to the desired decision without clarifying how the decision was made internally [56]. Whereas, Prototypes and criticisms provide a representation for the data distribution by selecting instances from the data called prototypes and outliers that are not well denoted by the prototypes named criticisms [57].

Additionally, Attention mechanisms are still being evaluated to provide some insights on explanation, mainly focused in NLP. In a nutshell, Attention mechanisms provide weights related to the importance of features for a given data set, with respect to specific outcomes of the model. Its application to XAI field, is based on the intuition that taking into account the set of the input in which the model is mainly focused on, some information related to reasoning may be generated. This is currently generating an intense debate, as it is shown in [58] which is a reply to the [59].

[60] provided an approach that identifies the relative importance of input features in relation to the predictions based on the behaviour of an adversarial attacks on the DNN. The method, called Adversarial Explanation for Artificial Intelligence systems –AXAI- inherits from the nature of adversarial attacks to automatically find and select important features affecting the model's prediction to produce explanations. The underlying logic is simple: since attacks tend to manipulate important features in the input to deceive a DNN, rather than trying to build a model that learns to explain the DNN's behaviour, they propose to use the attacks to learn the behaviour.

There are other key approaches to be taken into account for XAI applied to NLP. Layer-wise relevance propagation (LRP) have been demonstrated by [61] as a useful tool, for fine-grained analysis at document level and as a dataset-wide introspection across documents, to identify words that are important to a classifier's decision. This approach may have impact on other applications based on other NN architectures or on other types of classification problems (as sentiment analysis). Marshall, and Wallace developed a model (RA-CNN) for text classification that extends the CNN architecture to directly exploit rationales when available. As a conclusions, they stated that RA-CNN automatically provides explanations for classifications made at test time, thus providing interpretability.

As discussed, XAI is an evolving field with diverse methodologies that are challenging to be classified into exclusive categories. Instead, the methods often fit into several categories concurrently. There are many attempts for classifying XAI methods and their sub-fields, where each has its own field of study covering a vast amount of literature and approaches. Those will not be discussed here in detail. However, instead, the reader is guided to [62] for feature analysis and engineer. Also, to get an overall picture of the field, refer to [46] and especially [63] for model interpretation and methods used with accompanying examples. Nevertheless, in Table 1, a set of algorithms for explaining models is presented. The list if far from comprehensive but can be used as a quick reference for the available implemented and frequently utilized XAI techniques.

Table 1 Example algorithms for XAI

Algorithm	Domain	Scope	Data Type	Implementation
G-REX [64]	Model-agnostic	Global/Local	Tabular	NA
QII [65]	Model-agnostic	Global/Local	Tabular	QII
ELI5	Model-agnostic	Global/Local	Any	ELI5
LIME [66]	Model-agnostic	Local	Any	LIME
Anchors [67]	Model-agnostic	Local	Tabular, Text	Anchor
Real-Time Image Saliency [68]	Model-agnostic	Local	Image	pytorch-saliency, SaliencyMapper
ASTRID [69]	Model-agnostic	Global	Any	astrid-r
KernelSHAP [70]	Model-agnostic	Global/Local	Any	KernelExplainer
TreeSHAP [71]	Tree ensemble	Global/Local	Tabular	TreeExplainer
DeepLIFT [72]	Deep learning	Local	Any	DeepLIFT
DeepSHAP [73]	Deep learning	Global/Local	Any	DeepExplainer
Grad-CAM [74]	Deep learning	Local	Image	ELI5, Captum
Deep Visualization [75]	Deep learning	Local	Image	DeepVis

In addition to separate methods and tools, large organizations have developed more general AI interpretability toolkits that support multiple algorithms. Table 2 lists some of the available open sources toolkits.

Table 2 XAI Toolkits

Toolkit	Organization	Supported algorithms	Implementation
ELI5	MIT	LIME, Grad-CAM, Permutation Feature Importance Explainer (PFI)	ELI5
Captum [76]	Facebook	IntegratedGradients, DeepLIFT, SHAP, Grad-CAM, PFI	Captum
AI Explainability 360 [47]	IBM	LIME, SHAP, Contrastive Explanations Method, ProtoDash, DIP-VAE, ProfWeight	AIX360
InterpretML	Microsoft	SHAP, Global Surrogate, PFI	InterpretML
Tensorboard	Google's PAIR	SHAP, Integrated Gradients, SmoothGrad	What-if-Tool
XAI	The Institute for Ethical AI & ML		XAI

Next, five model-agnostic algorithms are briefly discussed to provide an overall comprehension of how some of the advanced methods function.

Disentangled Inferred Prior Variational Autoencoder (DIP-VAE)

DIP-VAE algorithm [77] is an unsupervised representation learning algorithm that will take the given features and find a new representation that is disentangled (i.e., entangled: multiple meaningful attributes are combined into a single feature.). In other words, DIP-VAE infers high-level independent features from unlabelled observations in such a way that the resulting features are understandable or have a semantic interpretation. For example, consider a shirt image from the Fashion MNIST dataset. DIP-VAE will detect the sleeve length as a feature, and the sleeve length will increase and decrease when varying this feature. Thus, the algorithm associates a dimension to a meaningful generative factor.

To demonstrate the applicability of DIP-VAE, AIX350 presents a dermoscopy tutorial [78] that illustrates a skin cancer diagnoses use-case. In the study, they have stated that dermatologists use rules of thumb known as the ABCD signs (i.e., asymmetry, border, color, and diameter) to examine skin cancer. Accordingly, learning the representations that capture the high-level concepts is crucial to building the trust of machine learning among dermatologists.

Prototypes with Importance Weights (ProtoDash)

ProtoDash algorithm [57] is an example-based data explanation method that understands the data through extracting prototypes (i.e., samples that relay the essence of a dataset) and criticisms (i.e., samples that are outliers). ProtoDash provides a fast prototype selection method with assigned non-negative weights to rank the importance of the selected representatives resulting in a summarized representation of the dataset distribution. ProtoDash is an extension of the MMD-critic [55], which is an unsupervised learning algorithm that can automatically identify prototypes and critics from unlabelled data to characterize the data distribution fully.

Data Shapley

Data Shapley algorithm [79] was not originally intended for data explainability but to address data valuation in supervised machine learning instead. It quantifies the value of each data sample on the final prediction performance. Hence, it gives insights about the data quality as well as explains the importance of each sample. Data Shapley 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. In contrast, low Shapley value data will detect outliers, and reflect whether their contribution deteriorates the model performance or makes no difference.

Local Interpretable Model-agnostic Explanation (LIME)

LIME [66] is a novel interpretation method that explains the ML-system by learning a local linear model to approximate each sample. In other words, LIME's objective is to set a minimization function that fits a linear model locally to the original model that is wanted to explain. The parameters that needed to be chosen are 1) loss function 2) regularizer 3) local kernel (i.e., to define locality), and all these parameters are forced under local accuracy and consistency.

In order to explain one prediction point. LIME samples instances around the point of interest by perturbing the sample point, where perturbed instances are the minimum cluster of points that can achieve the maximum probability. Afterward, it gets predictions from the black-box model and then weighs the samples by how close they are to the original point. Finally, it learns a simple model by those weighted samples that will approximate the black-box model locally. As a result, LIME is a local explainer that interprets the model behaviour for each prediction separately but not the whole model behaviour.

SHapley Additive exPlanations (SHAP)

Many explanation methods, including LIME and others, utilize additive feature attributions (i.e., base their explanation on the sum of individual feature attribution values) to represent the black box output. Algorithms based on game theory present a unique solution to a set of properties in terms of fairness.

SHAP [70] integrates LIME with Game Theory capabilities to provide a unique solution with fast computation. SHAP defines feature attributions as follows. The algorithm starts with a basic expectation, i.e., named the base rate, which represents the expected value of the output. It then starts conditioning one feature at a time until all model features have been conditioned, i.e., full joint distribution of the input features.

In the contexts of game theory, Shapley values are utilized to allocate contributions among a set of inputs to the output of the function. Shapley values represent how the contribution of each *player (i.e., a feature)* to the game is divided fairly. Three axioms describe the fairness properties:

1. Local accuracy; which explains the additivity property where the sum of local feature attributions equals the difference between the base rate and the model output.
2. Missingness; this property will assign a 0 Shapley value for the missing feature.
3. Consistency; ensures the monotonicity by increasing the Shapley values if the model changes so that the marginal contribution of a feature value increases.

Shapley values result from averaging over all N! possible combinations making it a computationally challenging solution. Thus, SHAP combines insights from different additive features attribution methods and approximates the Shapely values to make it computationally feasible.

Many SHAP implementations are already available as an open source. For instance, Kernel SHAP is a model-agnostic method generated by combing linear LIME and Shapley values. While Deep SHAP is a model-specific method for deep learning models that approximate Shapley values using the Deep Learning Important FeaTures (DeepLIFT) algorithm. TreeSHAP is available for tree-based models.

3.3.3. Demonstration of XAI Methods

Explainable AI covers a wide range of topics which might seem to be somewhat disconnected without concrete examples of their use is real-life settings. Thus, in this section, a use case is discussed in order to demonstrate some of the XAI methods.

Naïve Technical Indicator Based Long-only Stock Position Trading Algorithm

Publicly traded companies are traded in marketplaces where buyers and sellers sell pieces of companies to each other. The price of stocks is driven by fundamentals and forces of speculation which as a combination form a stochastic non-stationary process where SNR (signal to noise ratio) is very low.

Efficient market hypothesis claim that all information is already in the price in every time tick. Therefore, there should be no real signal left to take an advantage of. Nevertheless, this is still under debate especially when knowing that there are still some [80] anomalies that pass rigorous scrutiny to be taken advantage of. Additionally, there are multitudes of hedge funds, funds and even retail investors who have been successful in beating the market for decades. Alas, stock trading is far from a solved problem, and it truly is one of the most difficult problems there is to solve if even solvable.

In this demonstration, a naïve technical-indicator based long-only stock position trading algorithm is discussed from a viewpoint of what XAI methods can be, and which were, utilized and how to facilitate developing a software component to take its part in a swing/position trading system. The algorithm itself is about predicting a proxy of relative risk adjusted returns. Some details of the features, feature engineering, risk-models, and other information based on proprietary solutions are either excluded or obfuscated without mentioning. Nevertheless, the XAI methods presented are real. The purpose is to show how the prediction model can be interpreted and further explained to a degree which enables performing a sanity check, which is crucial in all real-world mission-critical systems.

The Problem Statement

The goal of the algorithm is to answer a question often asked by retail investors, especially by speculators: “Should I buy a piece of action of this particular company?” The answer should be along the lines “With respect to this particular investment universe, the company in question is expected to perform in top 2 quantiles out of 10 within the next 20 trading days when its risk adjusted returns are considered”. Considering the problem statement, the problem is, thus, about predicting the *rank* of risk-adjusted returns of companies w.r.t. to the rest of the investment universe.

Data

OHLCV Price Data

In this demonstration, OHLCV (Open-High-Low-Close-Volume) daily price data is utilized. No fundamental or alternative data sources are taken advantage of. The data spans the past 20 years of daily data of S&P500 and OMXH investment universes when available. The list of tickers is based on S&P500 and OMXH companies at the end of the year 2019, where there are at least five (5) years of consecutive data available from the current date to the past. No control for survival bias is conducted. Due to anomalies in the data, a few, non-named, tickers are excluded from the investment universe. The total dataset consists of circa 2.5M samples.

Features

OHLCV data cannot be directly utilized as such as the price data is not stationary, and there are hundreds of stocks, each having their own price scales. Thus, the OHLCV data is transformed into conventional technical analysis (TA) indicators with various time-windows. Typical departures from (exponential) moving averages are also included in addition to some mathematical transformation of the signals.

In addition to the common TA-indicators, fractionally differentiated price information is also included to enable some differentiation from very common feature engineering when it comes to naïve stock trading systems. Fractionally differentiating a time-series, in this case, guarantees (2% p-value) unit-root (by

Augmented Dickey Fuller Test) stationarity of the time-series while preserving some memory of the price information compared to integer differentiation thus providing an additional view to e.g. price *mean-reversion* aside to the typical heuristic-driven approaches provided by TA-indicators. The purpose is to provide mathematically sound (some) memory-preserving means to encode the price information in hope for smaller codependency with the traditional TA-indicators.

As features, various a-typical risk models are also included in order to provide some differentiation from the typical Sharpe[1] and Sortino[2] based models. Here, risk models adapted from [81] are utilized.

Sometimes it is the anomalies and special situations that drive the market. Therefore, anomalies in the calculated features are also computed and added to the feature-set. Anomaly detection is performed by utilizing Isolation Forest[3] anomaly scores.

The Dependent Variable

Instead of predicting the *rate of change* in stock price within certain time-period, risk adjusted returns are ought to be predicted. The main reason for doing so is that one does not typically want to construct a high volatile portfolio but a portfolio which maximizes some risk-adjusted returns metric such as Sharpe. Instead of utilizing typical Sharpe or Sortino, or even the aforementioned risk models, *Serenity Ratio* in particular, a differentiation of upside and downside average and tail-risks is utilized. Details of the aforementioned risk-models are excluded from discussion. Utilizing upside and downside risks provide smooth functions as depicted in Figure 11 below, which could be considered more robust than utilizing point-wise yields in the choppy markets we have nowadays. Additionally, Sharpe and Serenity, suffer in their interpretation when the returns fall negative.

In Figure 11, upside, downside, and total risk are depicted. Positive values for upside and downside risks represent the average and the tail-risk of the current asset price. The total risk is upside risk subtracted by downside risk; therefore, positive values represent *positive surprises* and negative values tell about *negative surprises*. To be noted, the risk models do not strive for predicting the future, but they model the currently observable risk within a certain time-window.

[1] https://www.investopedia.com/terms/s/sharperatio.asp
[2] https://www.investopedia.com/terms/s/sortinoratio.asp
[3] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

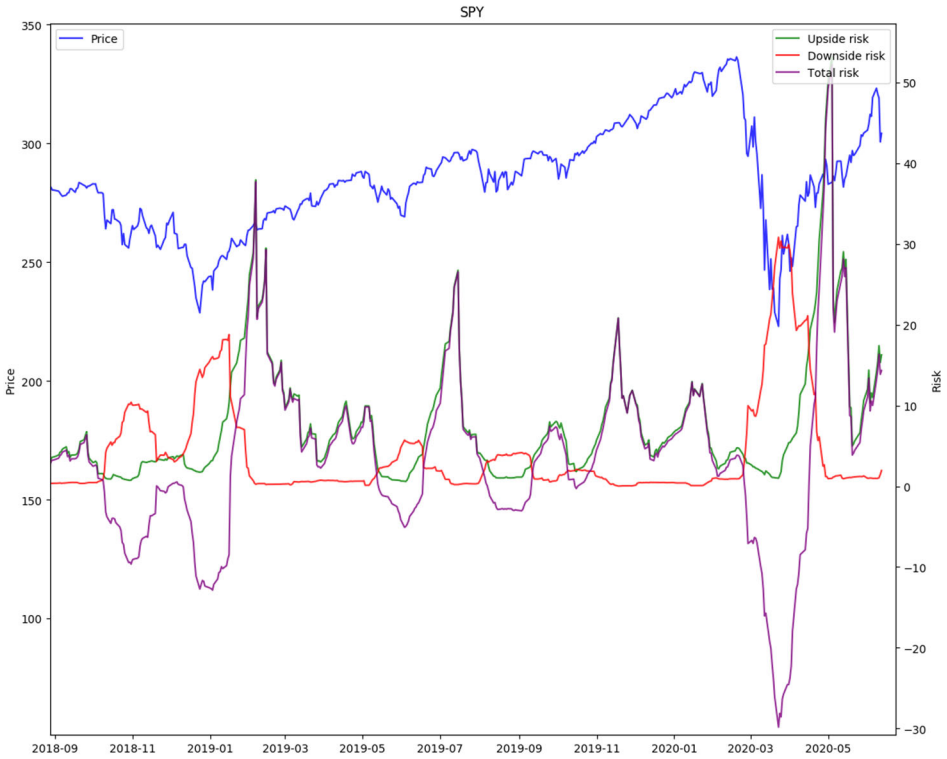

Figure 11. S&P500 upside, downside and total risk.

As the problem statement is not exactly about predicting the risk adjusted returns individually but the *rank* of each stock instead, a quantile-based *ranking* is formed based on the above-mentioned upside and downside risks, and the *total risk*. In Figure 12, the said, true, *rank* performance is depicted. The *rank* is the dependent variable it is wanted to predict for each stock in the investment universe.

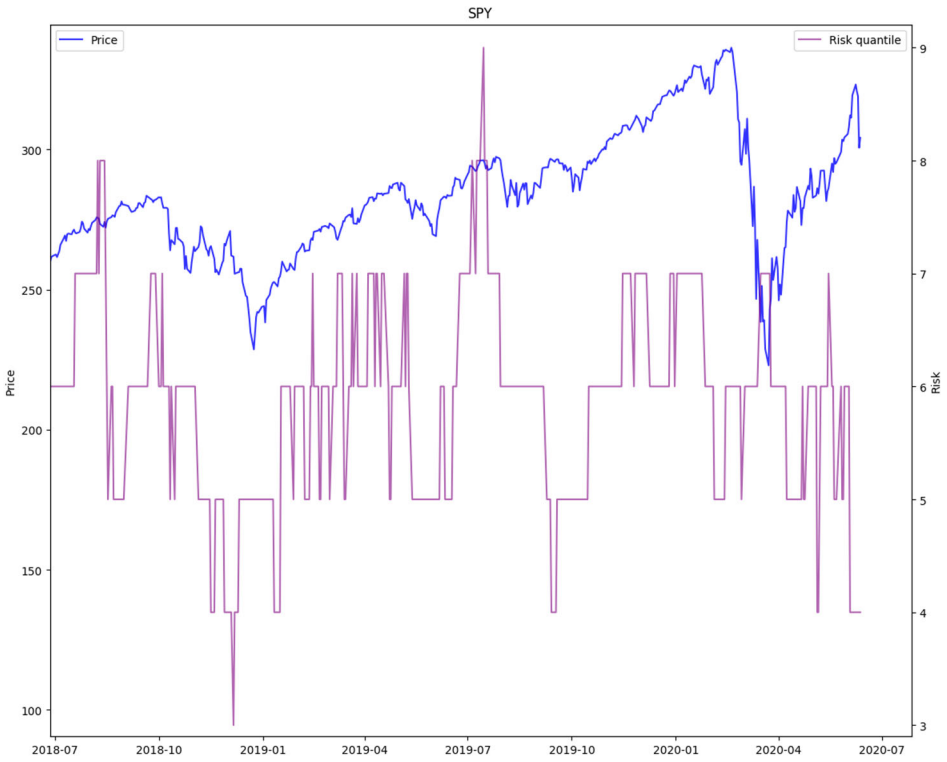

Figure 12. S&P 500 risk performance.

In the context of XAI and this demonstration, it is wanted to know how the model makes its predictions after it has been fitted to predict such a dependent variable. It is wanted to scrutinize whether the model has been fitted to random price fluctuation or can deduce some sound theory which would explain the behaviour of the model. If there is already a theory, which the model seems to be adhering to, then there is a smaller chance of the model being fitted to mere random noise even although the performance metrics might be good enough. In finance, and trading in particular, overfitting is a real concern. One must be very careful when utilizing powerful modern machine learning algorithms, which will find a *signal* even in white noise.

Modelling and Explaining the Model

In this demonstration, a single machine learning algorithm is utilized. LightGBM [82] is a state-of-the-art tree-based algorithm for classification, regression, and ranking. LightGBM is selected due to it being a fast, easily customizable algorithm which also integrates smoothly with SHAP algorithm for explainability. In this demonstration, *point-wise ranking* is conducted mainly due to facilitating explainability even although *list-wise ranking* is more natural to essentially a *ranking* problem at hand. Gradient-boosting is utilized as a core algorithm.

Feature Selection

There are multitudes of means for feature selection. Here, as a core feature selection algorithm, SHAP is utilized instead of e.g., feature importance provided by the LightGBM algorithm itself or any other means available. The idea is to take the mean absolute contribution of each feature and filter out the features which do not, on average, contribute more than a fixed value of 1% to the predictions. In this way, one can significantly reduce the number of features to favour further explainability of the model and to also fight against overfitting.

As in this demonstration, there are multiple time-horizons which are wanted to predict, feature selection is conducted for all time-steps of interest separately. In Figure 13, the selected features for each horizon are depicted. The x-axis represents the features to be considered for inclusion, whereas the y-axis is the prediction time-horizon in days. Red (1) values represent that a feature is selected whereas blue (0) depicts that a feature has been eliminated (contribution <1%).

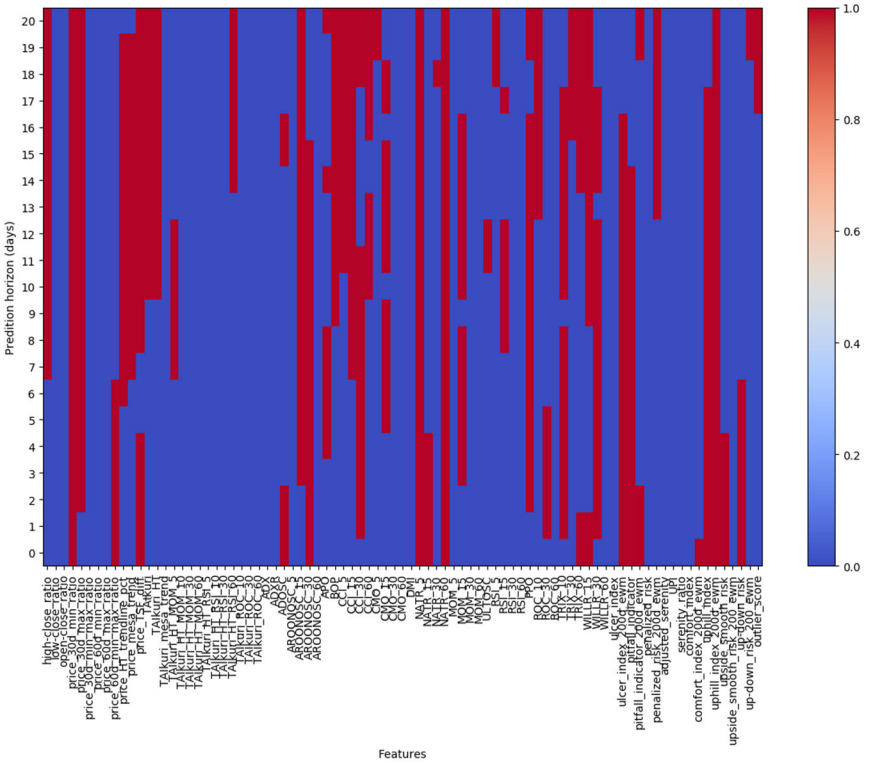

Figure 13. Selected features for each prediction time-horizon.

As depicted in Figure 13, multiple features were eliminated from the feature-space altogether due to not being important enough in the sense of SHAP contributions (>=1% in average). To be noted, some of the excluded features might be relevant for certain time-periods or maybe even for specific stocks. However, when considered all stocks in the investment universe together, those features are not relevant. The selected features are relevant, on average, for all stocks. Also, be noted that the feature selection method does suffer from (possible) codependency. Accordingly, some of the excluded features can be relevant but are masked due to codependency.

As depicted, the importance of some of the features emerge and disappear, depending on the horizon. This makes sense as stocks do tend to go through e.g., *mean-reversion* that takes some time to affect. Additionally, the impact of some features tapers away and become non-relevant as they impact only to very near-term predictions due to autocorrelation.

Global Feature Interpretation

In Figure 13, all the relevant features are depicted, but the order of importance is not visualized. In order to scrutinize the order of importance better, one can e.g., take the mean absolute value of the SHAP values and visualize the values as a bar chart, as depicted below in Figure 14.

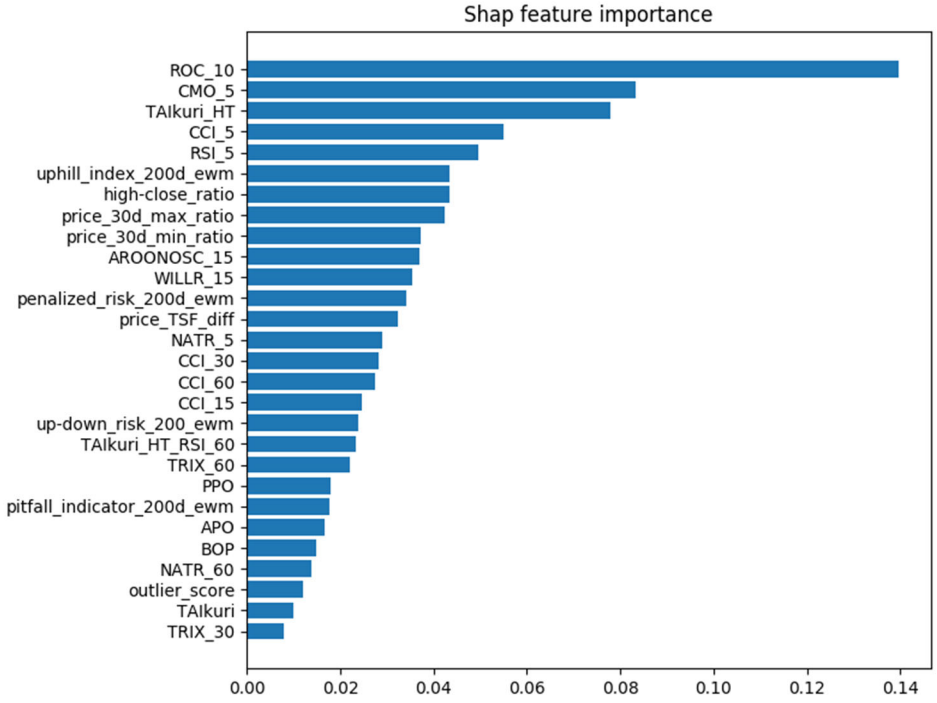

Figure 14. Feature importance.

Figure 14 shows the mean absolute SHAP values of predicting 20 trading days forward of the relevant features. ROC (rate-of-change) within the past ten trading days seems to be the most important feature in predicting the dependent variable, followed by Chande Momentum Oscillator (CMO), which models a very near-term momentum of the stock price. TAIkuri_HT is an Instantaneous Trendline of Hilbert transformed unit-root stationary manifestation of the price information. That is, the price information is transformed with memory-preserving fractional differentiation to a form where the signal has zero mean and is normally distributed, and also smoother with Hilbert transformation. The z-score of the signal indicates how expensive a stock is at any point in time, given its history. Thus, it indicates the probability of *mean-reversal*.

Three of these features together might indicate that the model concentrates on *momentum* of the asset and the probability of *mean-reversal* as the priority. The rest of the features are about refining the predictions.

In order to better understand how the features contribute in predicting the dependent variable, a partial dependence plot can be utilized, which depicts how the features contribute and in which direction. The partial dependence plot is based on SHAP values. Figure 15 depicts such a plot.

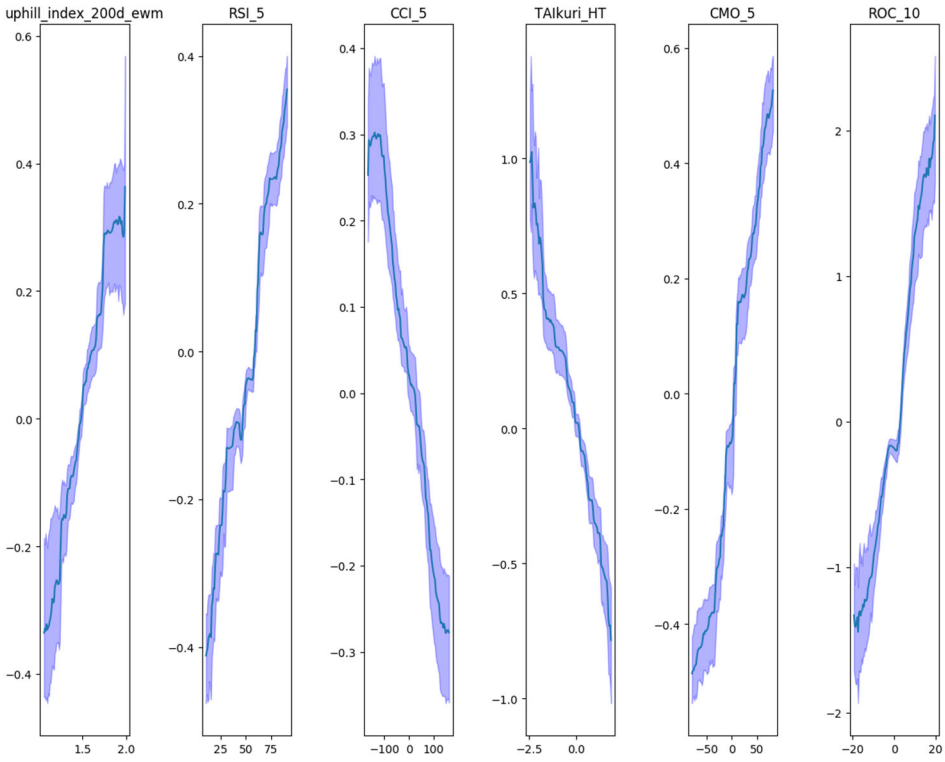

Figure 15. Partial dependence plot of the six most important features.

The partial dependence plot represented in Figure 15 shows how six the most significant features contribute toward predicting the dependent variable. The mean SHAP value within the feature value range is shown with a dark blue curve, and 5-95% confidence intervals are visualized with a light blue shadow curve. Whereas feature importance plot shows only how important, on average, a feature is, the partial dependence plot depicts how the features contribute, and when the confidence intervals are also visualized, one gets a sense of variation in the contributions and perhaps something about the possible interactions of the features.

In case of this experiment, ROC contributes linearly to the dependent variable. That means when there is a rapid positive change in the stock price, it is more likely that after 20 trading days the dependent variable is higher. In this case e.g., 20% change already at the time of observation, the impact to the dependent variable will be +2 (the dependent variable is between 1-10 where 10 means the best possible risk when one wants to go *long*). It is a similar case when it comes to CMO, although its impact is significantly smaller.

To be noted, the dependent variable's timespan covers the current values of the indicators as its value depends on a few weeks of data. Nevertheless, even although significant up momentum already started, on average, the impact on the dependent variable in the next 20 days is still positive. Thus, although one could think there might be (downward) volatility after a significant bump in the price, on average, the already started momentum still has a positive impact on the dependent variable.

What is to be noted is that usually CMO values above 50 are considered *overbought,* meaning that one probably should consider selling the stock rather than go *long,* but that does not seem to be the case here. On the contrary, CMO is not as common indicator as RSI (Relative Strength Index) which is also visible in the plot but it seems to be the same case with RSI, too. More *over-bought* the stock is, more likely it is for a stock to perform better in the future too, at least in these short time-windows. This goes directly against the TA-literature. On the other hand, CCI (Commodity Channel Index) reacts as described in the literature. What possibly seems to be happening is that CCI covers some over-optimistic areas of

ROC, CMO, and CMI as they all look at the price information from slightly different angles. One should study how the aforementioned indicators *interact* or just correlate with each other to understand them better.

TAIkuri_HT is the least known feature, or perhaps even a unique one being proprietary, among the depicted indicators. As discussed, the z-score of TAIkuri_HT tells how far the price is from its mean. As one could assume, values above zero (0) means likely *mean reversal* down. The contrary is true for negative values. This is exactly what TAIkuri_HT is designed to achieve, and the data and the model seem to validate the indicator.

Considering the above, together, the model and the features seem to reveal and confirm the commonly known strategy for swing and position trading: “trend-following”. That is, the safest possible bet in speculative stock trading is to just merely follow the trend. Adding TAIkuri_HT into the mix tells that one perhaps should buy into the trend only when the *mean-reversal* to up is in progress already. Thus, buying into the trend when the trend is overdue is not necessarily a good idea. Buying into the trend when the stock price has been hammered down is much safer but only when the stock is trending already. That is, the general rule of thumb is to buy the deep dip, but only after the direction has changed already. Again, a commonly shared strategy in swing and position trading but now shown also here with the help of XAI and a powerful machine learning algorithm.

Interaction of the Features

SHAP can also be utilized to scrutinize the interaction of the features, but let us here study the features by utilizing Friedman H-statistic [83] in order to get an overall understanding of which features might interact. To be noted, only features that have a significant role in predictions should be considered as less meaningful features might have just spurious interactions. In this case ROC_10, CMO_5, TAIkuri_HT are of interest. Figure 16 outlines the global interactions.

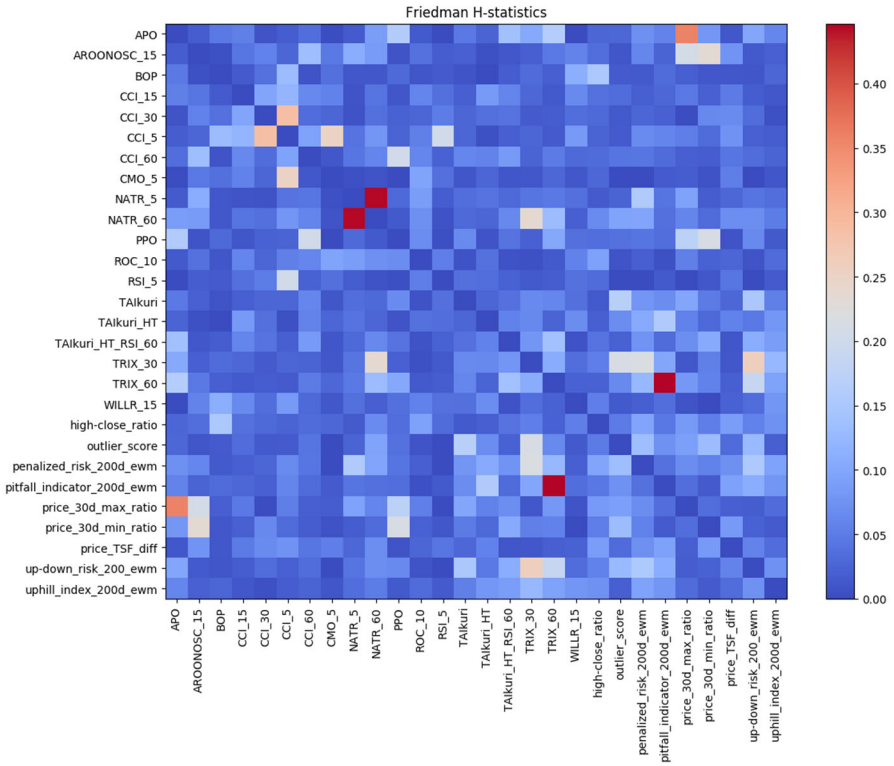

Figure 16. Friedman’s H-statistics of the features.

As shown in Figure 16, none of the aforementioned features have strong interactions with each other. However, there are strong interactions between some other features, but those might be just spurious

interactions as the features have not been identified to be very strong predictors, even although being still relevant. Thus, we will omit further analysis of them.

In addition to scrutinizing H-statistics, one can plot partial dependence plots with two features plotted in the same figure simultaneously to see how the features contribute together. Figure 16 represents such a plot.

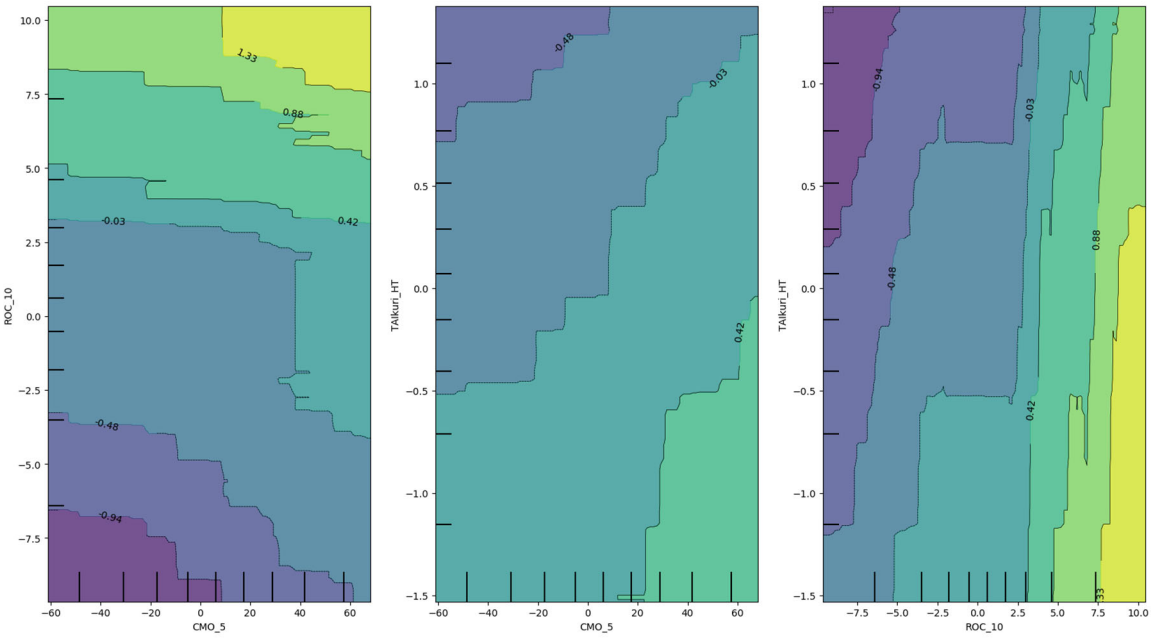

Figure 17. 2D partial dependence plots of the most significant features.

Scrutinizing Figure 17, one can get an understanding of the codependency of the features. For instance, CMO_5 and ROC_10 seem to correlate linearly, which hints one should likely eliminate either one of them. Collinearity is not necessarily a big problem when utilizing tree-based models, but it harms the model's interpretability.

TAIkuri_HT seems to have an interesting relationship between CMO_5 and ROC_10. As discussed above, TAIkuri_HT is about *mean-reversion*, which can also be understood by inspecting the figure above. When TAIkuri_HT reaches low negative values (z-score), and CMO_5 or ROC_10 gets high values, there is likely a positive impact. When TAIkuri_HT reaches high z-score (an asset is above its mean), even although CMO_5 hits high, the chances for high prediction value are not as significant as it would have been when TAIkuri_HT would be low. It is the same with ROC_10. Again, this tells about buying a dip (indicated by TAIkuri_HT) and when there is a strong positive momentum already.

In Figure 18, the relationship between TAIkuri_HT and 200d exponentially weighted mean of the dependent variable (up-side – down-side risk) is depicted. Again, if TAIkuri_HT reaches low values, it does not matter significantly what the current long-term risk situation is, although positive risk has a positive impact on the dependent variable. On the other hand, when the price of an asset is already high (TAIkuri_HT reaches high values), the negative risks loom already in the near-term future. What can also be seen is that when the risk is already close to zero or even negative, and when TAIkuri_HT is high, the situation seems to worsen even more. Thus, when the price of an asset is high already (TAIkuri_HT), and the downside risk is very much present already, the situation will likely continue getting worse in the near future. The trend following strategy seems to work for the down direction, too.

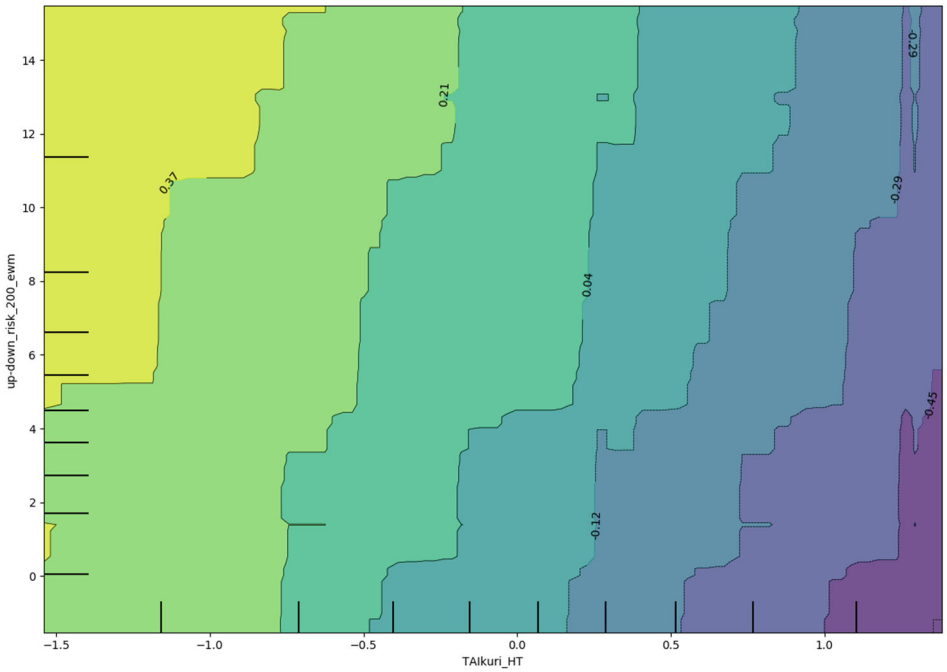

Figure 18. Impact of TAlkuri_HT and long-term risk on the dependent variable.

SHAP Timeseries Analysis

A model can also be interpreted sample-by-sample basis to understand better how it performs at different times. In Figure 19, the Covid-19 crash of 2020 in S&P500 is depicted from the model's perspective, where the cumulative SHAP values of the three most significant features are visualized.

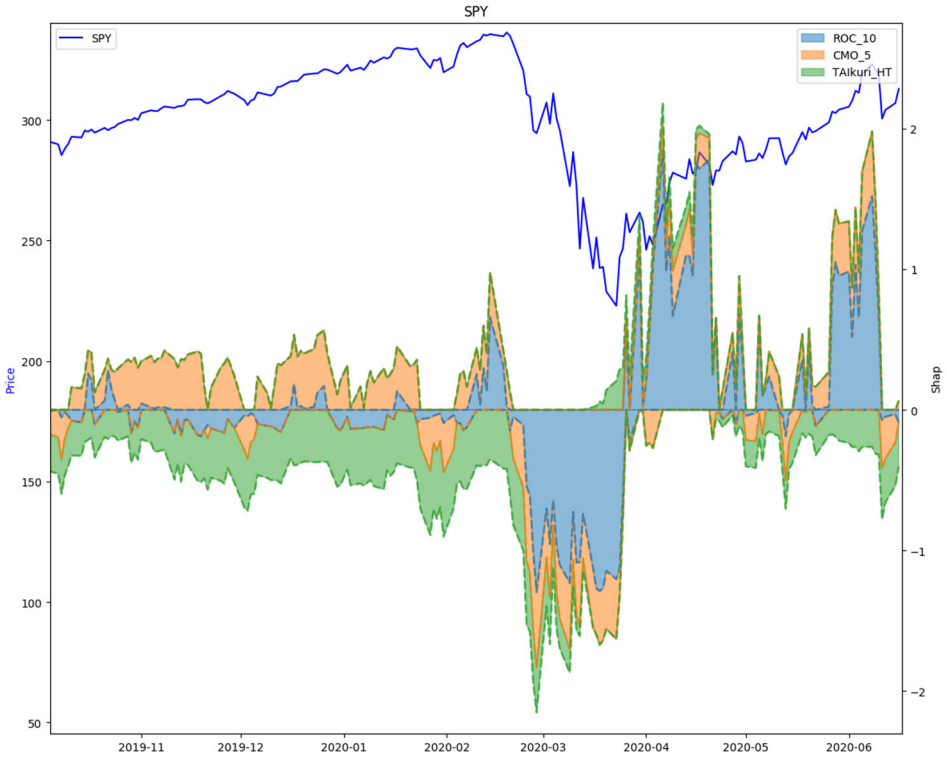

Figure 19. Three the most significant features and their SHAP contributions to the recent Covid-19 crash.

In late 2019, SHAP of ROC_10 was quite flat, meaning that there was no pull from the rapid price appreciation, but as the price of S&P500 slowly crawled up, CMO was still heavily influencing in positive pull it being a momentum indicator. TAIkuri_HT had significantly contributed negatively to the predictions for some time already as the market had been heating up slowly but surely. The market was above its mean, and although there would not have been Covid-19 crash, the market would have very likely pulled back, according to TAIkuri_HT.

In late February 2020, the crash started and continued till 23.3.2020. ROC_10 immediately reacted by pulling the predictions down, followed by CMO_5, whereas TAIkuri_HT already had a negative outlook on the market. Just before reaching the bottom of the market, TAIkuri_HT suddenly reverted to positive to pull the predictions up, whereas the rest of the three were very negative. TAIkuri_HT considered the market to be oversold already.

From the bottom, the price jumped fiercely up to which ROC reacted sharply. CMO remained neutral. After the initial market pump, the indicators started to fluctuate around the mean except for TAIkuti_HT, which started to pull back due to the overbought market situation. The momentum was gone until the next pump to which ROC and CMO reacted sharply again until the market had its small crash again. During the whole time, TAIkuri_HT became increasingly pessimistic.

3.3.4. Ethical & Legal Compliance

Ethical and legal consideration is highly important when building and deploying a machine learning model. by adhering to ethical and legal best practices, we are mitigating any risks of unfairness and non-compliance. Generally, each corporate company or institution will have their own company-wide ethics framework that should underly every action. This framework serves as a general guide; however, since AI is a new and evolving area, an individual AI Principles Framework should be created and followed when initiating, building and deploying any AI project. The framework should be dynamic, as this technical field continues to evolve and adhere to any regulation that exists or comes about in the future.

The AI Principles Framework that is company/institution mandated should include the following aspects when evaluating any AI application [84], [85]:

1. **Have an impact** – the purpose of the model should always be validated and revisited by stakeholders throughout the development cycle, and the benefits of the model outcome and who will be affected by it, should outweigh any risks.
2. **Fairness** – the model should avoid any biases, whether that may be creating or reinforcing bias, and should be built with AI best practices (i.e. QAIF – Data Understanding and Data Preparation phases).
3. **Safe** – the model should be extensively tested through performance metrics as well as unit, system and user acceptance tests to ensure the utmost safety of the model.
4. **Accountable & transparent** – the model should be transparent in the way that it was built and the outcomes should be trustworthy and justified. (i.e. XAI for a technical evaluation of this principle)
5. **Scientifically advanced** – the model should be scientifically and mathematically advanced according to the project context, to ensure the highest level of quality and accuracy.
6. **Compliant** – the model should comply with any policies set by GDPR regulations, such as data privacy and the ‘right to explanation’.

3.3.5. Explainable AI Summary

Predictive machine learning is about taking input and finding a function that maps the input to the output. In the case of ordinary least squares, the mapping is explainable by scrutinizing the found coefficients and the intercept, along with the statistical characteristics of the features and the model itself. On the other hand, non-linear models are not necessarily as easily explainable in ease if the algorithm itself is not self-explanatory. Thus, in order to explain and to facilitate the understanding of how the model behaves, additional means must be taken.

In this report, a brief state of the art study on widely available and known XAI methods are reported. The purpose of the study is to have an overview of the available methods but not to provide a comprehensive systematic review.

In addition to the state-of-the-art study, a demonstration is presented in order to serve AI and XAI practitioners better, as otherwise the XAI methods can remain theoretic and separate. The demonstration is about a stock trading algorithm where the typical XAI methods are taken advantage of in order to select the relevant features and later to explain how the features behave from the model's perspective. The explanations truly open up how the model sees the data and also sheds away empirical evidence on a widely known trading strategy.

3.4. MLOps - Continuous Deployment and Delivery

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. Building an ML is not a challenge itself, the main challenge is building an integrated ML system and operating it continuously in production. In addition to ML code, the main involved activities in the process of ML system development are data collection, data verification, testing and debugging, resource management, model analysis, process and metadata management, serving infrastructure, and monitoring [86],[87],[88].

Model monitoring: The performance of the model is monitored continuously and in case of performance degradation a new iteration of training might be invoked.

The following chapter introduces a selection of tools and frameworks used in MLOps. For a more detailed overview about the general MLOps properties can be found in D4.1 document of the IVVES project.

3.4.1. MLOps Tools

The rise of the MLOps concept has brought up a race to provide commercial tools and pipelines. The faster ML is infiltrating in commercial environments, the sooner the hidden costs of ML have come to light. Developing a ML system is not just developing a model and it is mandatory for ML Dev and Ops Teams to have optimized tools for every process in the pipeline. The goal is to reduce what is defined as the technical debt.

Technical debt is traditionally described as Technical debt is the ongoing cost of expedient decisions made during code implementation. It is caused by shortcuts taken to give short-term benefit for earlier software releases and faster time-to-market. Technical debt tends to compound. Deferring the work to pay it off results in increasing costs, system brittleness, and reduced rates of innovation. [89] [90] stated that ML systems have a special capacity for incurring technical debt, as they have all the maintenance problems of traditional software system, plus ML-specific issues. The commercial MLOps landscape is its infancy. Besides some ad-hoc tool chains based on different providers, there are cloud providers offering MLOps services, and some not cloud-specific solutions. IVVES should take into account these solutions, in order to boost the results of market-oriented research activities along the project. Some key actors in the MLOps landscape are listed in the following chapters.

MLOps for AWS SageMaker

Amazon SageMaker is a cloud machine-learning platform that was launched in November 2017. It enables developers to create, train, and deploy machine-learning (ML) models, including embedded systems and

edge-devices. Awlabs provide an exhaustive workshop to guide users through the MLops lifecycle applied in SageMaker [91]

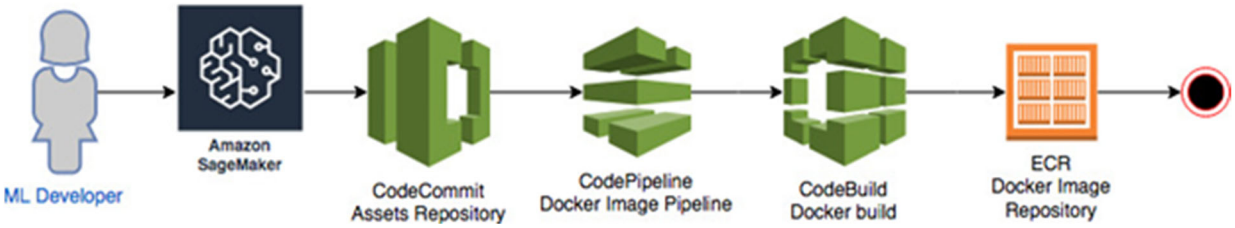

Figure 20. Sagemaker development flow.

The process is based on an Industry process for Data Mining and Machine Learning called CRISP-DM [92]. CRISP-DM stands for “Cross Industry Standard Process – Data Mining” and is an excellent skeleton to build a data science project around.

MLOps for Azure

Azure Machine Learning is the central piece of Azure-ML related components, be it creating new models, deploying models, managing a model repository, or automating the entire CI/CD pipeline for machine learning. Azure describes [93] a model lifecycle management by combining Azure DevOps and GitHub.

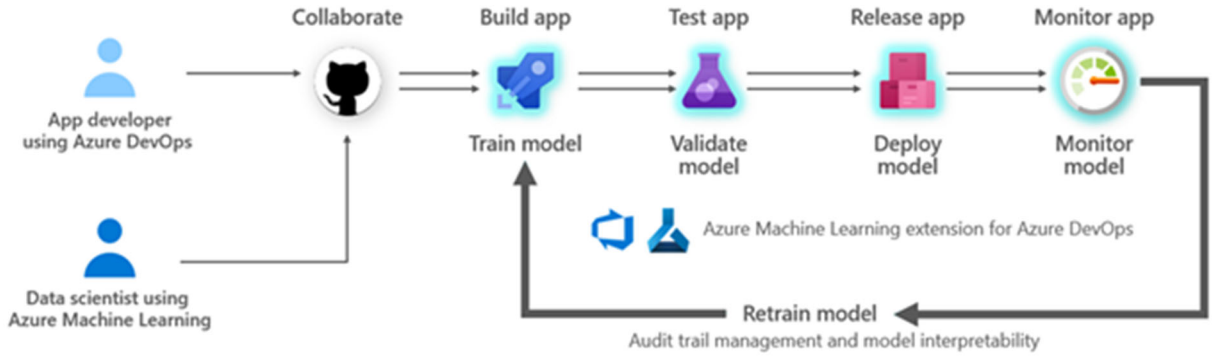

Figure 21. MLOps development flow.

Databricks MLflow

It is an Open Source Machine Learning Platform. It has an Open interface (it is designed to work with any ML library, algorithm, deployment tool or language. It is based on REST APIs and it is Open source. Currently, MLflow consist of three main modules:

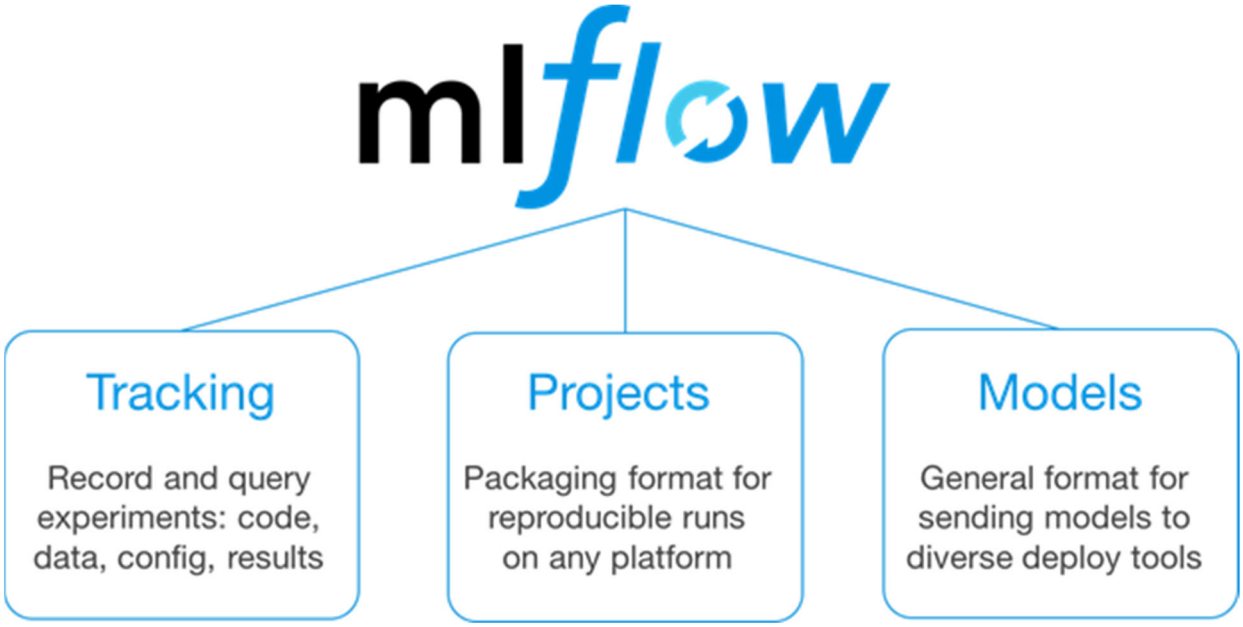

Figure 22. Databricks development flow.

An additional module (MLflow Registry): a centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of an MLflow Model. It provides model lineage (which MLflow experiment and run produced the model), model versioning, stage transitions (for example from staging to production), and annotations.[94]

Kubeflow

Kubeflow is a Cloud Native platform for machine learning based on Google's internal machine learning pipelines. Kubeflow is a free and open-source machine learning platform designed to enable using machine learning pipelines to orchestrate complicated workflows running on Kubernetes (e.g. doing data processing then using TensorFlow or PyTorch to train a model, and deploying to TensorFlow Serving). Kubeflow was based on Google's internal method to deploy TensorFlow models to Kubernetes called TensorFlow Extended.[95] In a nutshell, Kubeflow is integrating the most popular data science tools in one place.

4. Conclusions

XAI covers a wide range of topics where this document touches only a minuscule portion of the whole field. Nevertheless, XAI can be divided into an XAI taxonomy where the first branch separates data interpretation and model interpretation from each other, although these two are still intimately related when it comes to explaining the overall AI solution. Data interpretation can be further divided into scrutinizing features of the samples and studying the samples themselves. Model interpretation can be divided into transparent models that are somewhat interpretable alone. In contrast, post-hoc methods are required for other types of models in order to understand better how the problem has been modelled.

As XAI covers such a wide scope of methods, algorithms, and disciplines, a demonstration of utilizing typical means for model explanation is presented. As a demonstration, a component playing its part in an overall stock trading system is utilized.

In the demonstration, explaining the solution begins from feature selection, which is a crucial step in any machine learning project where overfitting is a real concern, and especially when it is striven to understand the underlying phenomena better. For feature selection, SHAP is utilized, one of the many methods for discovering how the features alone and together contribute to predicting the target variable.

Feature selection alone gives some level of understanding when one understands what the features try to accomplish. However, the explainability can further be improved via means of partial dependence plots, which show how the features globally contribute by revealing the feature value ranges and also the impact of each range on the dependent variable. Partial dependence plots can be plotted in 2D too in order to capture interactions between the features. Alternatively, one can utilize e.g., Friedman H-statistics to reveal the interactions.

In addition to explaining the features from a global perspective, with some methods (i.e., in this case Shap) one can study how the model behaves sample-by-sample. Sample-by-sample explanations enable us to scrutinize how the model behaves in e.g., special situations. Sometimes one can also discover if the model is not reactive at all.

XAI methods not only enable to understand better how the features and models behave thus facilitating model validation. Nonetheless, XAI methods also enable to turn machine learning upside down and utilize it as a research tool for discovering theories based on the data which are not necessarily visible just by studying the data alone without advanced machine learning algorithms for explainability. Discovering the real phenomena is the ultimate goal for model explainability.

Respectively, data augmentation methods help with several different aspects of developing ML and AI systems. In situations where the amount of training data is not sufficient, additional data can be generated via deep learning methods involving different flavours of GANs. This additional training data can help in avoiding overfitting, which is extremely useful in situations where it is not possible to obtain any additional real training data. Finally, due to the artificial nature of the synthetic training data, regulation such as GDPR no longer apply in the same manner. This makes sharing data possible between relevant parties, allowing model validation and other collaboration to take place even in situations where the original data set cannot be shared.

Given the large selection of different GAN variants architectures, finding a generally applicable, objective way to measure the performance of different variants would allow for evaluating and comparing the performance of different architectures. Additionally, as training GANs and optimizing hyperparameters can be computationally intensive, ensuring that the cost of the training process does not become a limiting factor is also very important aspect. The advent of cloud computing will help ensure that all interested parties have access to sufficient amounts of hardware. However, to take full advantage of this, researchers and developers will need to have access to suitable distributed computing development platforms, that will

enable both interactive, fast model development during the development phase and robust pipelines when migrating the model to production environments.

References

[1] A. Qureshi, “Managing Data Science Projects Using CRISP-DM,” *Medium*, 28-Nov-2019. [Online]. Available:https://medium.com/@aqureshi/managing-data-science-projects-using-crisp-dm-2b0682c0d894.

[2] A. Pillay, “An Artificial Intelligence Quality Framework,” *Sogeti.nl*, 01-Oct-2019. [Online]. Available: https://www.sogeti.nl/nieuws/artificial-intelligence/blogs/artificial-intelligence-quality-framework.

[3] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.

[4] J. Shijie, W. Ping, J. Peiyi, and H. Siping, “Research on data augmentation for image classification based on convolution neural networks,” in *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, 2017, vol. 2017-January, pp. 4165–4170.

[5] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” Jan. 2017.

[6] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, “Understanding data augmentation for classification: when to warp?,” *2016 Int. Conf. Digit. Image Comput. Tech. Appl. DICTA 2016*, Sep. 2016.

[7] “SMOD-Data augmentation based on Statistical Models of Deformation to enhance segmentation in 2D cine cardiac MRI.”

[8] T. DeVries and G. W. Taylor, “Improved Regularization of Convolutional Neural Networks with Cutout,” Aug. 2017.

[9] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved Techniques for Training GANs,” *Adv. Neural Inf. Process. Syst.*, pp. 2234–2242, Jun. 2016.

[10] C. Han *et al.*, “GAN-based synthetic brain MR image generation,” in *Proceedings - International Symposium on Biomedical Imaging*, 2018, vol. 2018-April, pp. 734–738.

[11] L. Sun, J. Wang, Y. Huang, X. Ding, H. Greenspan, and J. Paisley, “An Adversarial Learning Approach to Medical Image Synthesis for Lesion Detection,” *IEEE J. Biomed. Heal. Informatics*, pp. 1–1, Jan. 2020.

[12] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and Improving the Image Quality of StyleGAN,” 2019.

[13] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[14] M. Mehralian and B. Karasfi, “RDCGAN: Unsupervised representation learning with regularized deep convolutional generative adversarial networks,” in *2018 9th Conference on Artificial Intelligence and Robotics and 2nd Asia-Pacific International Symposium, AIAR 2018*, 2018, pp. 31–38.

[15] J. Islam and Y. Zhang, “GAN-based synthetic brain PET image generation,” *Brain Informatics*, vol. 7, no. 1, pp. 1–12, Dec. 2020.

[16] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, “GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification,” *Neurocomputing*, vol. 321, pp. 321–331, Mar. 2018.

[17] K. Kazuhiro *et al.*, “Generative Adversarial Networks for the Creation of Realistic Artificial Brain Magnetic Resonance Images,” *Tomogr. (Ann Arbor, Mich.)*, vol. 4, no. 4, pp. 159–163, Dec. 2018.

[18] G. Kwon, C. Han, and D. shik Kim, “Generation of 3D Brain MRI Using Auto-Encoding Generative Adversarial Networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, vol. 11766 LNCS, pp. 118–126.

[19] J.-Y. Zhu, T. Park, P. Isola, A. A. Efros, and B. A. Research, “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks Monet Photos.”

[20] V. Sandfort, K. Yan, P. J. Pickhardt, and R. M. Summers, “Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks,” *Sci. Rep.*, vol. 9, no. 1, pp. 1–9, Dec. 2019.

[21] Y. Lei *et al.*, “MRI-only based synthetic CT generation using dense cycle consistent generative adversarial networks,” *Med. Phys.*, vol. 46, no. 8, p. mp.13617, Jun. 2019.

[22] Y. Pan, M. Liu, C. Lian, T. Zhou, Y. Xia, and D. Shen, “Synthesizing missing PET from MRI with cycle-consistent generative adversarial networks for Alzheimer’s disease diagnosis,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11072 LNCS, pp. 455–463.

[23] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets.”

[24] S. Ul *et al.*, “Image Synthesis in Multi-Contrast MRI with Conditional Generative Adversarial Networks.”

[25] J. Rubin and S. M. Abulnaga, “CT-To-MR Conditional Generative Adversarial Networks for Improved Stroke Lesion Segmentation,” 2019.

[26] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation strategies from data,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, vol. 2019-June, pp. 113–123.

[27] C. M. Lo, Y. C. Chen, R. C. Weng, and K. L. C. Hsieh, “Intelligent glioma grading based on deep transfer learning of MRI radiomic features,” *Appl. Sci.*, vol. 9, no. 22, p. 4926, Nov. 2019.

[28] S. Nema, A. Dudhane, S. Murala, and S. Naidu, “RescueNet: An unpaired GAN for brain tumor segmentation,” *Biomed. Signal Process. Control*, vol. 55, p. 101641, Jan. 2020.

[29] J. Zhao *et al.*, “Tripartite-GAN: Synthesizing liver contrast-enhanced MRI to improve tumor detection,” *Med. Image Anal.*, vol. 63, p. 101667, Jul. 2020.

[30] Y. Mirsky, T. Mahler, I. Shelef, and Y. Elovici, *CT-GAN: Malicious Tampering of 3D Medical Imagery using Deep Learning*. .

[31] J. M, Zhang, M. Harman, L. Ma, and Y. Liu, “Machine learning testing: Survey, landscapes and horizons.,” IEEE Transactions on Software Engineering., 2020 (Published online).

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

[33] D. Baylor, E. Breck, H. Cheng, N. Fiedel, Y. F, Chuan, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, and et al., “Tfx: A tensorflow-based production scale machine learning platform.,” In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,, pp. 1387–1395, ACM, 2017.

[34] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Biess-mann, and A. Grafberger, “Automating large-scale data quality verification.,” Proceedings of the VLDB Endowment11, 12, pp. 1781–1794, 2018.

[35] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, “The ml test score: A rubric for ml production readiness and technical debt reduction,” Big Data, pp. 1123– 1132, 2017.

[36] J. Kim, R. Feldt, and S. Yoo, “Guiding deep learning system testing using surprise adequacy” preprint arXiv:1808.08444, 2018.

[37] C. Murphy, G. E, Kaiser, L. Hu, and L. Wu, “Properties of machine learning applications for use in metamorphic testing.,” In SEKE, 2008.

[38] X. Xie, J. Ho, C. Murphy, G. E, Kaiser, B. Xu, and T. Y, Chen, “Application of metamorphic testing to supervised classifiers.,” 2009 Ninth International Conference on Quality Software, Jeju, pp. 135–144, 2009.

[39] S. Srisakaokul, Z. Wu, A. Astorga, O. Alebiosu, and X. Tao, “Multipleimplementation testing of supervised learning software,” In Proc. AAAI-18 Workshop on Engineering Dependable and Secure Machine Learning Systems (EDSMLS), 2018.

[40] X. Xie, Z. Zhang, T. Y, Chen, Y. Liu, P. Poon, and B. Xu, “Mettle: A metamorphic testing approach to validating unsupervised machine learning methods,” 2018.

[41] W. Chunyan, G. Weimin, L. Xiaohong, and F. Zhiyong, “Differential combination testing of deep learning systems.,” Artificial Neural Networks and Machine Learning-ICANN 2019: Image Processing. ICANN 2019. Lecture Notes in Computer Science, vol. 11729, 2019.

[42] B. Nushi, E. Kamar, E. Horvitz, and D. Kossmann, “On human intellect and machine failures: Troubleshooting integrative machine learning systems„” in 31st AAAI Conference on Artificial Intelligence, AAAI 2017, p. 1017-1025, 2017

[43] S. Mutuvi, “Introduction to Machine Learning Model Evaluation,” *Medium*, 05-Feb-2020. [Online]. Available: https://heartbeat.fritz.ai/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f.

[44] Karel Dejaeger, Seppe vanden Broucke, Tuomas Eerola, Rainer Wehkamp, Lieve Goedhuys, Mikkel Riis, Bart Baesens, "Beyond the hype: cloud computing in analytics" 2012 .[Online]. Available: http://www.techila.fi/wp-content/uploads/2012/08/KU-Leuven-Beyond-the-hype-Cloud-computing-in-analytics2.pdf

[45] A. Adadi and M. Berrada, “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI),” *IEEE Access,* vol. 6, pp. 52138-52160, 2018.

[46] A. B. Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila and F. Herrera, “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,,” *Information Fusion,* vol. 58, no. 1566-2535, pp. 82-115, 2019.

[47] V. Arya, R. K. E. Bellamy, P.-Y. Chen, A. Dhurandhar, M. H. S. C. Hoffman, S. Houde, Q. V. Liao, R. Luss, A. Mojsilović, S. Mourad, P. Pedemonte, R. Raghavendra and J. Richards, “One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques,” Sep 2019.

[48] A. Leemans, “Make your Artificial Intelligence more trustworthy with eXplainable AI,” *SogetiLabs*, 06-May-2019. [Online]. Available: https://labs.sogeti.com/make-ai-trustworthy-with-explainable-ai/.

[49] U. Bhatt, A. Dhurandhar, B. Kim, K. R. Varshney, D. Wei, A. Weller and A. Xiang, “2020 Workshop on Human Interpretability in Machine Learning (WHI),” 2016-2020. [VTT_AOnline]. Available: https://sites.google.com/view/whi2020. [Accessed 6 2020].

[50] S. Barocas, S. Friedler, M. Hardt, J. Kroll, S. Venkatasubramanian and H. Wallach, “The FAT-ML Workshop Series on Fairness, Accountability, and Transparency in Machine Learning.,” 2014-2018. [VTT_AOnline]. Available: https://www.fatml.org/organization. [Accessed 6 2020].

[51] P. Cortez and M. J. Embrechts, “Opening black box Data Mining models using Sensitivity Analysis,” in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, Paris, 2011.

[52] P. Cortez and M. J. Embrechts, “Using sensitivity analysis and visualization techniques to open black box data mining models,” *Information Sciences,* vol. 225, pp. 1-17, 2013.

[53] A. Goldstein, A. Kapelner and E. Pitkin, “Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation,” *Journal of Computational and Graphical Statistics,* vol. 24, no. 1, pp. 44-65, 2015.

[54] S. Sharma, J. Henderson and J. Ghosh, “CERTIFAI: Counterfactual Explanations for Robustness, Transparency, Interpretability, and Fairness of Artificial Intelligence models,” 2019.

[55] K. Been, R. Khanna and O. O. Koyejo, “Examples are not enough, learn to criticize! Criticism for Interpretability,” in *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, 2016.

[56] S. Wachter, B. Mittelstadt and C. Russell, “Counterfactual explanations without opening the black box: automated decisions and the GDPR.,” *Harvard Journal of Law & Technology,* vol. 31, no. 2, pp. 841-887, 2018.

[57] K. S. Gurumoorthy, A. Dhurandhar, G. Cecchi and C. Aggarwal, “Efficient Data Representation by Selecting Prototypes with Importance Weights,” in *International Conference on Data Mining (ICDM)*, 2019.

[58] Sarah Wiegreffe, Yuval Pinter. 2019 Attention is not not Explanation. arXiv:1908.04626

[59] Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. arXiv:1902.10186

[60] Rahnama, Arash; Tseng, Andrew. An Adversarial Approach for Explaining the Predictions of Deep Neural Networks. arXiv preprint arXiv:2005.10284, 2020.

[61] Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, Wojciech Samek. 2016. Explaining Predictions of Non-Linear Classifiers in NLP. arXiv:1606.07298

[62] M. Kuhn and K. Johnson, Feature Engineering and Selection: A Practical Approach for Predictive Models, 2019.

[63] C. Molnar, Interpretable Machine Learning, 2020.

[64] U. Johansson, L. Niklasson and R. König, “Accuracy vs. comprehensibility in data mining models,” in *Proceedings of the seventh international conference on information fusion*, 2004.

[65] A. Datta, S. Sen and Y. Zick, “Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems.,” 2016.

[66] M. T. Ribeiro, S. Singh and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier,” in *the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’16)*, New York, 2016.

[67] M. T. Ribeiro, S. Singh and C. Guestrin, “Anchors: High-Precision Model-Agnostic Explanations,” in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[68] P. Dabkowski and Y. Gal, “Real Time Image Saliency for Black Box Classifiers,” in *In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS’17)*, California, 2017.

[69] A. Henelius, K. Puolamäki and A. Ukkonen, “Interpreting Classifiers through Attribute Interactions in Datasets.,” in *Proceedings of the 2017 ICML Workshop on Human Interpretability in Machine Learning (WHI 2017)*, 2017.

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

[71] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal and S.-I. Lee, “From local explanations to global understanding with explainable AI for trees,” *Nature Machine Intelligence,* vol. 2, p. 56–67, 2020.

[72] A. Shrikumar, P. Greenside and A. Kundaje, “Learning Important Features Through Propagating Activation Differences,” in *In Proceedings of the 34th International Conference on Machine Learning (ICML’17)*, 2017.

[73] H. Chen, S. Lundberg and S.-I. L. Chen, “Explaining Models by Propagating Shapley Values of Local Components,” in *ArXiv*, 2019.

[74] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, 2017.

[75] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs and H. Lipson, “Understanding Neural Networks Through Deep Visualization,” in *ICML Deep Learning Workshop*, 2015.

[76] “captum,” Facebook, [Online]. Available: https://captum.ai/.

[77] A. Kumar, P. Sattigeri and A. Balakrishnan, “Variational Inference of Disentangled Latent Concepts from Unlabeled Observations,” in *international conference on learning representations*, 2018.

[78] P. Sattigeri and V. Arya, “dermoscopy,” IBM, 25 January 2020. [Online]. Available: https://github.com/IBM/AIX360/blob/master/examples/tutorials/dermoscopy.ipynb.

[79] A. Ghorbani and J. Zou, “Data Shapley: Equitable Valuation of Data for Machine Learning,” in *International Conference on Machine Learning*, 2019.

[80] K. Hou, C. Xue and L. Zhang, “Replicating Anomalies,” *The Review of Financial Studies,* vol. 33, no. 5, pp. 2019-2133, 2018.

[81] R. B. de Viéville, R. Gelrubin, E. Lindet and C. Chevalier, “An Alternative Portfolio Theory,” KeyQuant, Paris.

[82] “LightGBM,” [Online]. Available: https://lightgbm.readthedocs.io/en/latest/.

[83] J. H. Friedman and B. E. Popescu, “Predictive learning via rule ensembles.,” *Annals of Applied Statistics,* vol. 2, no. 3, pp. 916-954, 2008.

[84] S. Pichai, “AI at Google: our principles,” *Google*, 07-Jun-2018. [Online]. Available: https://www.blog.google/technology/ai/ai-principles/.

[85] B. Casey, R. Vogl, and A. Farhangi, “Rethinking Explainable Machines: The GDPR's 'Right to Explanation' Debate and the Rise of Algorithmic Audits in Enterprise ,” *Berkeley Technology Law Journal*, vol. 34, Feb. 2019.

[86] GigaOm-Delivering on the Vision of MLOps, https://azure.microsoft.com/en-gb/resources/gigaom-delivering-on-the-vision-of-mlops/, retrieved June 2020

[87] MLOps with a Feature Store, https://towardsdatascience.com/mlops-with-a-feature-store-816cfa5966e9, retrieved June 2020

[88] MLOps: CI/CD for Machine Learning Pipelines & Model Deployment with Kubeflow, https://growingdata.com.au/mlops-ci-cd-for-machine-learning-pipelines-model-deployment-with-kubeflow/, retrieved June 2020

[89] https://towardsdatascience.com/intro-to-mlops-ml-technical-debt-9d3d6107cd95

[90] SCULLEY, David, et al. Hidden technical debt in machine learning systems. En Advances in neural information processing systems. 2015. p. 2503-2511.

[91] https://github.com/awslabs/amazon-sagemaker-mlops-workshop

[92] https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining

[93] https://azure.microsoft.com/es-es/blog/automated-machine-learning-and-mlops-with-azure-machine-learning/

[94] https://www.mlflow.org/docs/latest/model-registry.html

[95] https://en.wikipedia.org/wiki/Kubeflow