BANK OF ENGLAND

Staff Working Paper No. 947

Software validation and artificial intelligence in finance – a primer

Imane Bakkar, Chiranjit Chakraborty, Carsten Jung, Marta Kwiatkowska and Carl Taylor

October 2021

BANK OF ENGLAND

Staff Working Paper No. 947
Software validation and artificial intelligence in finance – a primer

Imane Bakkar,[1] Chiranjit Chakraborty,[2] Carsten Jung,[3]
Marta Kwiatkowska[4] and Carl Taylor[5]

Abstract

The use of machine learning and artificial intelligence (AI) in finance poses growing risks for software validation to financial institutions, markets, and decision makers, making it a key priority for regulators. This paper discusses accepted software validation practices; highlights challenges to those practices introduced by AI; and suggests areas of focus for developers when creating AI-based solutions for the finance industry. We discuss how practices may need to evolve to respond to these new challenges. Our objective is to inform policymakers and governance bodies, and to raise awareness among decision makers in financial institutions. We do not make policy recommendations at this stage.

Key words: Artificial intelligence, machine learning, finance, financial regulation, software validation, regulatory technology, supervisory technology.

JEL classification: C80, C87, C88, G18, G28, C40, L50, L86, N70.

(1) Bank of England. Email: imane.bakkar@bankofengland.co.uk
(2) Bank of England. Email: chiranjit.chakraborty@bankofengland.co.uk
(3) Institute for Public Policy Research. Email: c.jung@ippr.og
(4) University of Oxford. Email: marta.kwiatkowska@cs.ox.ac.uk
(5) Bank of England. Email: carl.taylor@bankofengland.co.uk

The views expressed in this paper are those of the authors, and not necessarily those of the Bank of England or its committees. The authors would like to thank especially Daniel Kroening for his contributions to this paper, and Andreas Joseph and Misa Tanaka for their comments and guidance. In addition, the authors would like to thank Louise Eggett, Arthur Turrell, John Lewis, Andrew Nye, Khim Murphy, Alina Barnett, Gerardo Ferrara, David Bholat, Marcus Buckmann, Tangy Morgan and participants of a research seminar at the Bank of England for their valuable comments and supports. Any errors and omissions, of course, remain the fault of the authors.

The Bank's working paper series can be found at www.bankofengland.co.uk/working-paper/staff-working-papers

Bank of England, Threadneedle Street, London, EC2R 8AH
Email enquiries@bankofengland.co.uk

ISSN 1749-9135 (on-line)

I Introduction

The term artificial intelligence (AI) refers to the tools and techniques that enable a machine to make decisions without human intervention. The concept of AI, machine dependency of human society and potential risks have been captured in science fiction stories for more than a century. In the year 1909, E.M. Forster wrote a science fiction short story, titled "The Machine Stops" [Forster, 1909]. The story described a world where the human population could not live on the surface of the earth, as the air was not breathable and nature was decimated. Humans lived underground, each one alone in a standard room, and could communicate via a form of video conferencing. All humanity relied on a giant machine to provide for its needs. A "Mending Apparatus" was a system charged with repairing defects that appear in the Machine, and those who questioned it or its possibility for failure were dismissed. The story ends with the Machine collapsing, bringing down most of humanity with it.

We are nowhere near the ultimate information technology (IT) system failure imagined in this novel. However, software [1] and an increasing number of systems essential to the well-functioning of cities, countries and global institutions rely on the decisions made by lines of code, is now ubiquitous and forms part of what may be deemed critical infrastructure. Several critical activities of a variety of software engineering processes, where human intervention was required, are now being automated by machine learning (ML) and AI methods. Financial institutions (FI) are no exception and are increasingly relying on software to make critical business decisions. Whether this relates to automated trading, making payments, risk management or simple reporting: software permeates all the areas of activity in the financial system. With this, software validation—the theory and practice of making sure software works robustly and as intended—is becoming of critical significance.

Executives and policy makers will have to understand the big issues in this field together with the accompanying growth of AI, as first highlighted in the Bank of England and Financial Conduct Authority (FCA)'s 2019 report on Machine Learning in UK Finance (see [Bank of England and FCA, 2019]). Poorly validated software—including that using AI—can lead to biases, unintended risk-taking and operational risks that can transmit to the safety and soundness of firms (see [Proudman, 2019]). The aim of this paper is to help inform the debate around these issues, by asking two questions: What are the current practices for software testing and validation? And what changes to established practices does the introduction of AI require, particularly within the financial industry?

To answer these questions, it is important to view software as the output of an engineering process. Just as when we construct a building, control and robustness can be built into the design of the software and IT system that the software forms part of. That means testing should consider a number of plausible software failure cases, build controls to narrow down, or constrain,

[1] The most common definition of software is a collection of instructions and operating information used by a computer (See for instance the IEEE SWEBOK [Bourque and Fairley, 2014])

how things could go wrong, and provide safeguards for such scenarios. This paper considers conventional software validation techniques as a starting point and outlines the differential introduced by AI software, hence bridging the software validation gap from non-AI to AI-specific challenges, particularly in the financial system. Whilst we do not aim to present any policy conclusions for regulators, we provide suggestions and considerations for policymakers and firms' decision-making bodies.

In section II, we explain what software risk is using a proposed framework and provide illustrative examples to demonstrate the importance of software validation. We also outline why validation of AI driven systems is of importance to regulators in the same section. In section III, we provide an overview of conventional software testing and validation techniques. In section IV, we explain how AI is bringing different challenges and what new considerations are needed for AI-based software validation, including where it interacts with conventional software. We suggest some considerations and propose ideas to contribute to the debate around AI driven software governance. Finally, in section V, we wrap up this primer with potential next steps.

In this paper, we primarily intend to lay out the elements we judged important for a discussion on AI-software validation. As software becomes more able to make decisions independently from human intervention, given the emergence of AI/ML, assessing its risks should be elevated in firms' and regulators' priorities. So this paper aims to start a debate on how the introduction of AI may necessitate changes, or revisions, to accepted software validation practices that form a key element of established IT management processes.

II Software risk, cost of failure and the importance of a regulatory framework

Software forms a central component of the core IT systems of financial institutions, supporting key business components and hence the financial services that they provide. A failure of software to ensure the confidentiality, availability and integrity of those IT systems, and associated data, could impact the operational resilience of individual and interconnected FIs; customers; the banking system and financial stability more broadly.

The most common meaning of software is defined as a collection of instructions and operating information used by a computer. An increasing number of IT systems essential to the functioning of cities, countries, global institutions, national and international banking systems rely on the decisions made by lines of code (See IEEE SWEBOK 2014 [Bourque and Fairley, 2014]). Software related risks can be broadly defined as falling into three categories: model risk; technology risk; and data risk. We define these as [2] :

[2]Model risk and Technology risk are widely defined in financial regulations, and these categories are proposing our own interpretation of these definitions as market practitioners and academics. Data risk is a nascent field with less regulatory coverage

- Model risk is the risk that models – a collection of algorithms calculating an output given certain inputs/parameters and mathematical assumptions – are either incorrectly implemented (with errors) or make use of questionable assumptions, or assumptions that do not hold true in a particular context; and
- Technology risk is the risk that the underlying technology (eg. software program code, processing capacity, networking) used to support, distribute and implement a given model does not operate as intended; and
- Data Risk is the risk that the data used as an input into a model is incorrect, incomplete, biased, corrupt, or unprotected from undue access and manipulation.

Model risk is separate from technology risk. Whilst the latter is the risk that software does not work as intended, model risk includes the risk that a model, as designed, does not produce the intended outcome. Whereas the first one may come for example from issues with a mathematical set of formulas or assumptions, the second one may come from the software interface including incompatibility between software and hardware.
There is a lot of literature on model risk and model validation in the financial system (see, for example, [Scandizzo, 2016] on risk models, [Jongh et al., 2017] on recommendations for a model validation framework, and the European Banking Authority (EBA) requirements for an example on regulations [EBA, 2014]). Global financial regulation generally incorporates model validation requirements (primarily on risk modelling). However, there is much less available research and regulation on how technology and components (both software and hardware) perform and are tested by FIs.

Financial regulators usually address these risks as separate areas and are generally neutral regarding the technology solutions FIs implement to deliver the regulated services they are authorised to provide. Regulators are equally non-prescriptive regarding how FIs should validate, or test, their software. Under the current regulatory framework (e.g. the provisions of Capital Requirements Regulation (CRR) and Capital Requirements Directive (CRD) IV published by Prudential Regulation Authority (PRA) [PRA, 2020, PRA, 2021]) software risk as we define it in this paper is mostly intended to be captured under the operational risk elements of the current financial services and banking regulatory framework. However, the emergence of AI in the financial system (see [Leo et al., 2019]) means that it will be increasingly difficult to separate model risk from technology and data risk, particularly when there can be end-to-end automation using AI. We will discuss in this paper that one of the key differences between non-AI and AI software is that in the latter the three categories of risk discussed above are co-mingled, and thus impossible to separate and treat independently. As such, a holistic approach to AI software validation will be fundamental to effectively manage the risks from its usage.

In addition, model, technology and data risks have already crystallised for some software currently being used in the financial system, leading to high

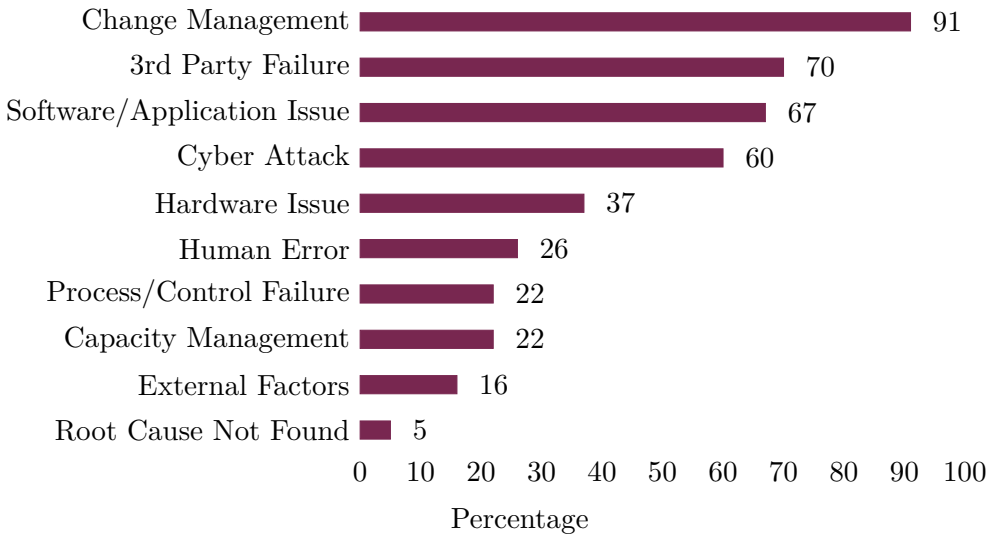

Figure 1: More than half of financial firms' IT incidents in 2018 were due to IT change. Source: [FCA, 2018]

profile IT incidents, and in the following section we focus on illustrations of such events.

IT incidents

According to Tricentis [3], there were 48 publicly known instances of software failures globally in 2017 alone, 19 of which were in banking [4]. These failures could translate into lost revenue due to customers being unable to use a firm's product; reputational costs that can be reflected in a hit to the share price once a failure is made public; regulatory penalties such as fines; and the costs of fixing the software defect that caused the incident, and associated data.

Changes to IT systems are the largest source of IT incidents in the UK according to the FCA (Figure 1), with many IT incidents arising due to shortcomings in software validation practices. Definitions of these categories in the figure are generally self explanatory from the names. We will discuss change management, the dominant category, more in the next section (section III). We will also set out an overview of established software validation practices there. In section IV we highlight how the introduction of AI could give rise to new risks, necessitating changes to established software validation practices. Strikingly, in many cases, the technical approaches needed to address AI related risks have not yet been fully developed; and, accordingly, best practices have yet to be established.

[3]Please see Tricentis 5th Edition of the Software Fail Watch `https://www.tricentis.com/resources/software-fail-watch-5th-edition/`

[4]To note that Tricentis methodology, which is based on a Google news alert on specific words (eg. Glitch or Bug) estimates that cumulatively IT system incidents have led to losses of about $1.7 trillion worldwide. The same edition identified 606 recorded software failures

For FIs, an IT outage becomes particularly costly if it: (1) disrupts business as usual, such as asset trading; or (ii) impacts customers, which could result in regulatory censure (e.g. fines) or customer redress costs.

To illustrate what form IT incidents can take and their consequences, we consider the following case studies from the last ten years:

Case 1 On Wednesday 20 June 2012, customers of a UK bank found they could not use online banking facilities to access their accounts or obtain accurate balances, discovering that they were unable to drawdown loans, transfer payments or transfer monies using SWIFT. The problems affected customers in the UK and abroad, with some unable to access cash while overseas. The incident affected around 6.5 million customers with disruption to the majority of the bank's systems lasting until 26 June 2012, and disruption to the systems of one of the bank's regional subsidiaries continuing until 10 July 2012. The cause was a software compatibility issue between upgraded software and the previous version of the software, which occurred when an upgrade was backed out on Sunday, 17 June 2012. [5].

Case 2 Over the weekend of the 20-22 April 2018, a UK bank transferred c. 5 million customers to a new IT platform, with the new platform going live on the evening of Sunday 22nd April. While the customer migration proceeded as planned post migration, many customers experienced significant problems accessing and using their accounts via the bank's online and mobile app channels, and communicating with the bank over the telephone and in branches. [6].

Case 3 In August 2012, a global financial institution specialising in market making experienced over $400 millions of trading losses, because of the unexpected behaviour of new software code over a 45min period. The firm operated an algorithm that executed orders automatically on trading venues and the software error resulted in a large number of erroneously generated orders reaching exchanges. The loss arose from unwanted large positions accumulating rapidly due to the software defect [7].

The above examples demonstrate that software failure could impact the provision of financial services, affecting significant numbers of customers or market participants, and could have potentially large financial cost.

There are other examples from recent years that include issues related to algorithmic trading leading to flash crashes. Some examples of such crashes are, UST flash crash during October 2014 , flash events for CHF de-peg in January 2015, and GBP - USD currency pair crash in October 2016 etc. as documented

[5] See https://www.fca.org.uk/publication/final-notices/rbs-natwest-ulster-final-notice.pdf
[6] See https://www.tsb.co.uk/news-releases/slaughter-and-may
[7] https://www.sec.gov/litigation/admin/2013/34-70694.pdf

in various regulators' reports eg. [Bouveret et al., 2015], [Breedon et al., 2018] [Cielinska et al., 2017], [Schroeder et al., 2018].

This specific field of algorithmic trading is possibly the main area to date where some regulators and academics have started to link the model, technology and data risks under the same assessment framework. In 2015, a group of institutions issued a briefing note highlighting key control principles and sound practices (see the Senior Supervisor Group briefing note April 2015 [US Department of the Treasury et al., 2015]), outlining the need for testing during all phases of an algorithm development life-cycle. More recently, the PRA in the UK issued a supervisory statement on algorithmic trading (see PRA supervisory statement SS5/18 [PRA, 2018]) which discusses both the validation of algorithms and testing of their deployment. The European Union (EU)'s Markets in Financial Instruments Directive (MIFID) [European Commission, 2014] regulation has also aimed at reducing algorithmic trading risks. However, and as we specify later in this paper, the introduction of AI could trigger the need of a strengthened software validation approach.

Recent regulatory initiatives

In April, 2021, the European Commission brought in a proposal for a regulation laying down harmonized rules on applying artificial intelligence and amending certain union legislative acts [European Commision, 2021]. The proposed legislation will be applicable to any AI system (or AI outputs) used in the EU (except some military applications), not just the financial sector. It can take couple of years before it becomes law and several modifications are expected in the meantime. Let us discuss the proposal at a very high level before we deep dive into the connected technical areas in the next sections.

In the proposal, four levels of AI risk and corresponding systems are identified (based on their application) subject to different regulatory approaches. These four levels of systems are: prohibited AI system, high risk AI system, lower-risk AI system and any other AI system. Prohibited AI systems includes real-time facial recognition and social scoring to understand someone's trustworthiness or personality from their social media etc. Application of AI for such use cases are prohibited. High risk AI systems include applications where there is potential for harm to health and safety or for an adverse impact on fundamental rights, e.g. bio-metric identification systems, systems used in appraisals for education or vocational training, systems used for recruitment, systems used for assessing eligibility for public benefits and services, creditworthiness checks etc. AI systems that fall into the categories described in the proposed regulation must meet specific expectations, including areas of compliance covering data, transparency, safety and control. Low risk AI systems include applications where there is very low potential for harm to health and safety, fundamental rights etc. Here the main obligation that is proposed is transparency, i.e., the human must know, unless it is obvious from the context, that they are engaging with the output of an AI system (e.g., chatbots). For AI systems that do not fall within any of the above categories under formal regulation, the proposals suggest some codes of conduct to encourage voluntary adherence to the expectations set out

	Lines of code
Car in 1981	50,000
Apollo 11	145,000
Boeing 787	7,000,000
Microsoft Windows	50,000,000
Modern car	10,000,000 to 100,000,000
Google's infrastructure	2,000,000,000

Table 1: Lines of code underlying products. Source: [Algaze, 2017]

in the proposed regulations.
In section IV, our paper addresses a subset of the same issues that this proposed regulation is trying to tackle and, in particular, aims to illuminate some of the technical challenges involved in resolving them. Given this recent progress by the regulators, our paper can make valuable contribution in any debate related to such initiative by outlining how software validation will need to change following the introduction of AI.

III Conventional IT Change Management

This section discusses accepted IT change management processes in more details.

III.1 FI's IT systems and what we mean by "change"

Modern FIs depend on large IT systems. The numbers of lines of programing code in software included in commercial products has increased exponentially over the last ten years (Table 1). A modern car, for instance, incorporates software consisting of between 10 million and 100 million lines of programme (software) code. Such software is generally the product of many years of development, incorporating incremental additions and revisions with each new version. The software system within a financial institute is equally big and complicated.

Software often consists of a multitude of integrated components that individually solve specific problems, or perform specific functions, and which then pass outputs on to other software components, supporting an end-to-end process. Such software components are the building blocks of an integrated IT system, which when integrated together solves higher-order problems. As we will show below, software validation can take place both at the level of individual software components but also at IT system level.

Different software components of an integrated IT system can individually perform widely differing tasks. Figure 2 illustrates this, using the example of a modern bank. A simple payment involves the interplay of a multitude of software components: some of which may be relatively new, such as a mobile application allowing customers to query account information or make payments

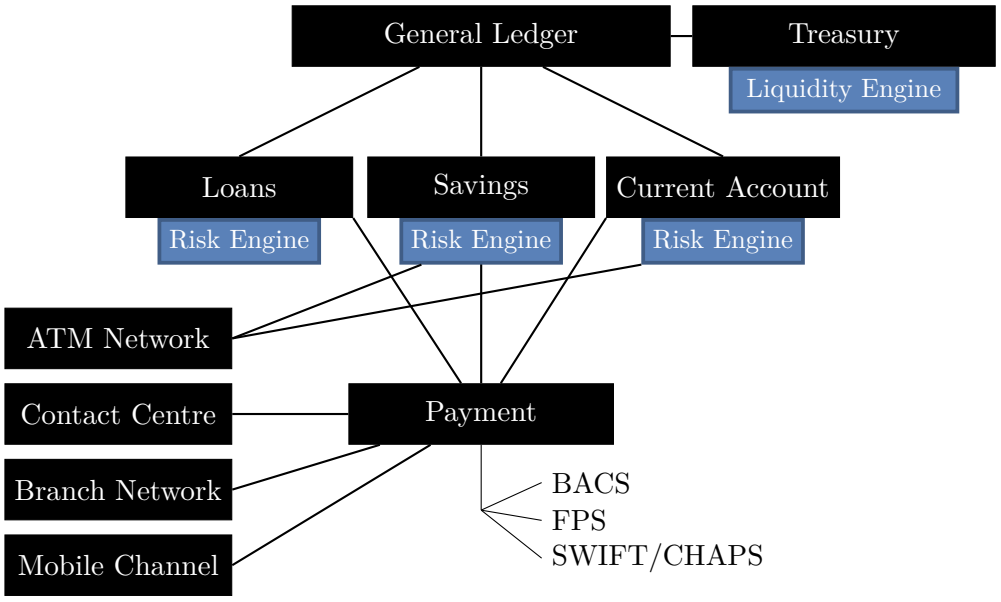

Figure 2: Integrated software components in a large bank. Source: Authors

via mobile phones; and some of which may be 20 years old, such as those used for current or savings accounts. All of these software components need to work together to support, for example, the processing of a single payment.

The different software components in turn depend on a wider IT architecture, shown in Figure 3. Different software components may run on different integrated IT architectures, referred to as hardware. Typically a large bank's IT architecture could encompass large powerful mainframe computers, smaller yet powerful computers known as Servers (running UNIX or Microsoft Windows operating system software), and more recently third party Cloud hosting, all connected by a network.

This diversity of IT software and hardware components acquired over time must be taken into account when change is introduced with implications for software validation and testing more broadly.

III.2 Risks introduced by change

Changes to IT systems (IT change) can take many forms. An IT change could be: the application of a software patch, the installation of a small piece of software to correct a known software defect; a change to configuration settings associated with existing software or IT hardware; or the introduction of new software components and IT hardware, a more extensive and therefore higher risk of change to an existing IT system.

Although the scale of the change is an important metric, this is not always an effective guide to the risk that an IT incident could result from the change, or its

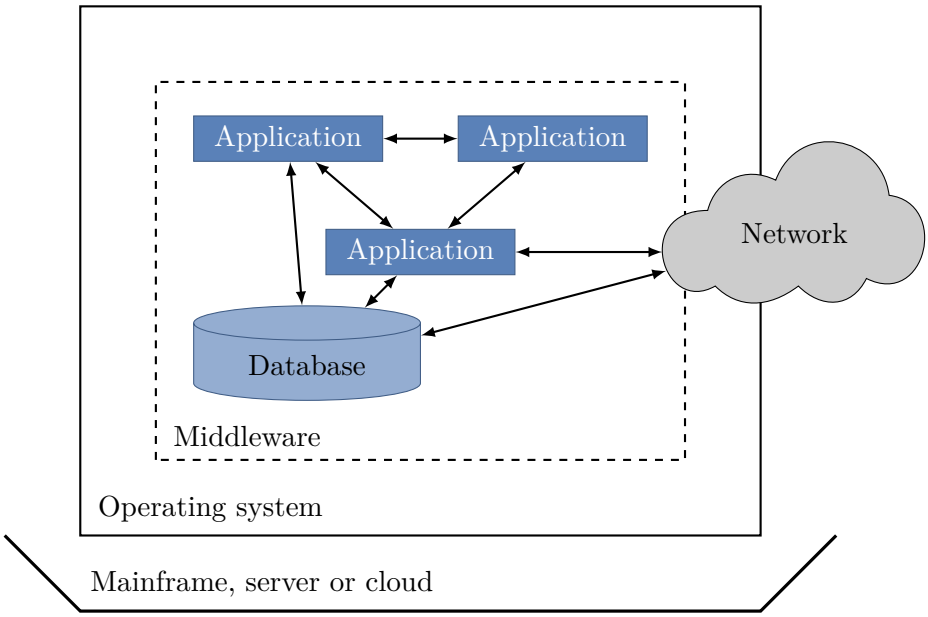

Figure 3: Modern IT architectures consist of a multitude of software and hardware components which can be changed in various ways. Source: Authors

potential impact. More generally, IT changes, including those coming from the introduction of AI, introducing new IT systems or changing existing IT systems, need to be managed using rigorous processes. This would help to safeguard their confidentiality, availability and integrity, and prevent IT incidents that could have a range of detrimental impacts on individual FIs or the banking system more broadly.

Software engineering and computer science have developed multiple ways to ensure that complex IT systems comprised of multiple software and IT hardware components work well together. However, the IT systems of FIs are already complex, relying on several—and heavily interconnected—software components. This complexity is compounded by how FIs outsource several of their software needs to third parties, including some of their critical IT systems. In addition to that, any trading activity that the FIs may be engaged with on externally hosted exchanges or clearing houses, will have their own IT systems.

There are various established practices to manage IT change, which include a range of IT system testing techniques. A key mitigant of the risks arising from IT change is to robustly test software that has been changed based on an accurate risk assessment, in an environment that is representative of the Live environment where the software is in use, so that unexpected outcomes can be identified and rectified before such IT changes are introduced into the Live system.

Currently, most FIs have testing processes in place with the testing under-

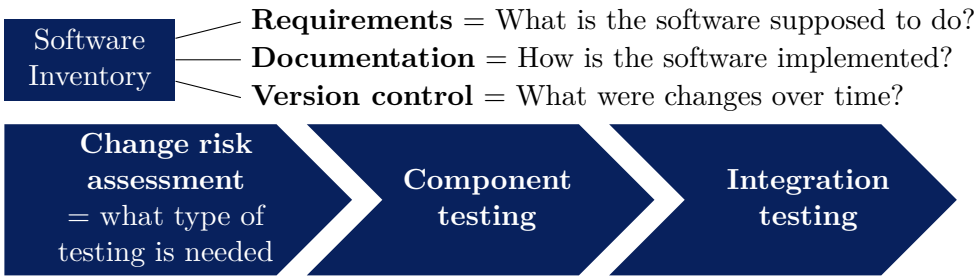

Figure 4: Four key aspects of IT change management: IT asset inventory, change risk assessment, component testing and integration testing. Source: Authors

taken to validate each IT change dependent on the nature of the change and which parts of the FI's IT systems the change will affect. However, broadly speaking, a basic suite of testing policies would in general include the following component and architecture level procedures:

Requirements, documentation, and version control: This entails documenting the various components of the system and outlining what a piece of software, and the IT system that it forms part of, is supposed to do. It also includes having a robust way to maintain various versions of code the underlying software used, and a traceable way to log all the changes made to it.

Change risk assessment: This refers to an assessment that takes place before any testing is done, to assess what risks are presented by a given IT change and specifically what would happen if the change went wrong, i.e. its potential impact. This part of the process requires holistic thinking about the software and IT hardware environment, and determines what types of testing should be conducted prior to any IT change being introduced into the Live environment.

Component (unit) testing: Unit testing focuses on verifying that individual software components which will be integrated with other software components to create the IT system function as specified in the technical specification. The technical specification generally defines the range of inputs that the software component (or unit) is required to successfully accommodate, and the desired outputs. Unit testing will also include negative testing to verify that the software will reject input data that does not conform to the technical specification to ensure that unexpected processing and outputs do not happen.

System (integration) testing: When a new software or IT hardware component is introduced into an IT system it is important not just to test the component (unit) itself, but also how it interacts with other components of the IT system. It is significantly different than just doing unit testing and hence, system or integration testing is done to achieve this. System testing

involves testing more “high-dimensional” problems on how the software and IT hardware components work together to support the functionality required for the IT system.

There have been a few studies (e.g. [Kinch et al., 2007]) observing that there is a wide range of attitudes towards the value of the testing, and that larger organisations tend to emphasise the process more than smaller organisations. A more recent survey [Hynninen et al., 2018] identified that the top two key pre-occupations of the testing community were: (i) the need to raise awareness of the importance of testing; (ii) and issues around automation such as the inability to automate tests due to time demands and lack of prioritisation of automation. Another survey—one of the longest running surveys of the testing community[8]—identified that the size of testing teams keep getting smaller, and that testing in the Live environment is an increasing trend.

Finally, it is unclear currently how much testing FIs perform on their data warehouses with archived data, or their transactional and non-transactional data stores used to model or support their businesses. It is clear that testing and maintaining "traditional" software is a complex task, adding AI into this at large scale will add its own challenges.

IV How does AI require changes to established software validation (testing) practices?

IV.1 What are AI, ML and related algorithms

Let us try to first define more precisely what we mean by AI and ML, as well as some related concepts. AI is a set of techniques or technologies that enable a computer system to make decisions without any input from humans. ML is a subset of AI, it is basically a technique for realizing AI. So ML can be considered to be a set of methods or algorithms to train computer systems such that they can learn how to make decisions without human intervention. There are three broad types of ML algorithms, supervised, unsupervised and reinforcement.

For supervised learning, the algorithm learns from a full set of labeled data while it is trained on it. For example, during credit decisions, labels can be just good or bad corresponding to each loan entry in the historical data. From that labeled training data the algorithm tries to learn how to predict labels while the input variables corresponding to any unknown customer are available (e.g., information collected during loan application). The label can be both categorical (good or bad customer) or continuous with a numerical value (e.g., optimal loan amount to be granted for each historical loan entry). These are called classification and regression type algorithms respectively. For unsupervised, there are no such labels available in the data. Rather, the algorithm tries to learn relationships among different entries based on how similar or different they are and group them accordingly. There are also semi-supervised algorithms which

[8]See state of testing report 2019 by InfoQ here

is a mixture of both. The artificial neural network (ANN) is one of the most popular ML algorithms where there are layers of computational units between input and output variables. Each such computational unit is relatively simple, but the overall algorithm can predict very complex non-linear relationships. When there are too many such layers between input and output layers (termed as hidden layers), it is often termed as a deep neural network (DNN) and algorithm as a deep learning algorithm. This type of network architecture can be used to solve both supervised and unsupervised problems.

Whereas the supervised learning algorithm tries to learn from examples, the reinforcement learning (RL) algorithm tries to learn from experiments. RL algorithms develop models learning from mistakes, by trial and error. Just like humans when playing a game, they make mistakes, learn from them and adapt. Similarly, in RL, an agent (in a chess game, a player), takes actions (moves pieces) decided by its policy (strategy in the human brain) in an environment (the board, pieces, rules, and objectives), by moving between successive states (positions), each of which has a reward associated with it. A deep neural network can also be used here to take actions choosing from an optimal strategy, this is called deep reinforcement learning.

IV.2 AI is already used in the financial system

AI - and in particular ML - is already in use in the financial system, as highlighted by the Turing institute [Buchanan, 2019], noting that AI is used in: surveillance and compliance systems; algorithmic trading; robo-advisors which are able to make portfolio asset allocation decisions; and providing loans and insurance. ML can also be used for macro-economic forecasting, horizon scanning to understand the change in financial landscape as well as other financial policy decisions (either for internal policies by an individual financial institution or by regulators [Chakraborty and Joseph, 2017]).

In 2019, the Bank of England published the results of a survey which showed that in the UK, ML is increasingly being used in UK financial services. Two-thirds of respondents reported that they already use AI/ML in some form. The median firms typically use live ML applications in two business areas and respondents expected this to more than double within the next three years. Similar to the Turing institute report, the survey shows that the most common usage of AI/ML is in fraud detection and customer services. In addition, several firms mentioned using ML techniques in their risk-management processes (see Machine Learning in UK financial services, Bank of England and FCA joint survey 2019 [Bank of England and FCA, 2019])

In June 2020, the International Organisation of Securities Commissions (IOSCO) issued a consultation on the use of AI by intermediaries and asset managers, which proposed guidance on a regulatory framework to its members [IOSCO, 2020]. The consultation states that asset managers and advisory firms are starting to use AI and ML to support their advisory and support services, risk management, client identification and monitoring, selection of trading algorithms and portfolio management, as well as - for some asset managers - order execution. In June 2020 also, Banque de France (BdF) and The French

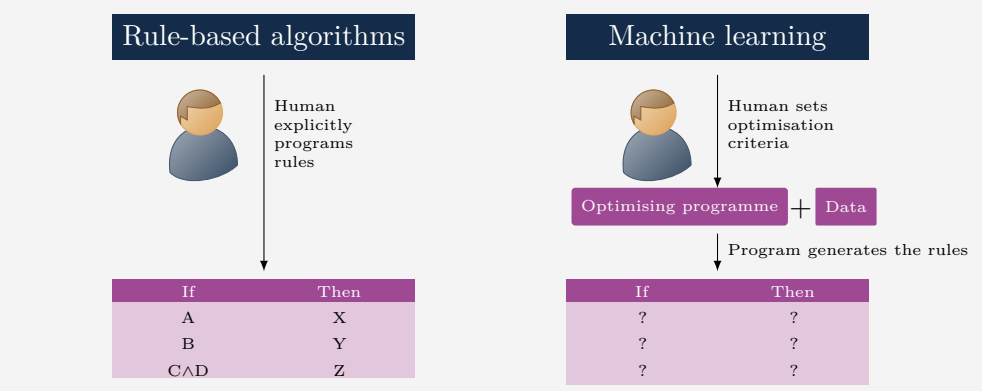

Figure 5: Machine learning algorithms make decisions without being explicitly programmed. Source: Box 1, Machine Learning in UK Financial Services, [Bank of England and FCA, 2019]

Prudential Supervision and Resolution Authority (ACPR) published a discussion paper focusing on example usages of AI in credit scoring, surveillance (eg. anti-money laundering) and customer protection [Dupont et al., 2020].

IV.3 How is AI different from conventional software?

We argue that the processes for validating conventional software explained in Section III are not wholly applicable to systems that implement artificial intelligence. As mentioned in IV.1, AI is the theory and development of computer systems able to perform tasks that previously required human intelligence. AI is a broader field, of which machine learning or ML is a sub-category which enables the system to learn such tasks automatically. This contrasts with more traditional so-called "rules-based software algorithms" where human experts explicitly decide what decisions will be taken by the software and under which circumstances (Figure 5).

There are three important aspects to this making AI based systems different from traditional systems:

1. First, as we argue below, the use of data to train computers to make decisions, makes ML based software development very different from other software development approaches that generally build and implement IT system solutions in response to formally documented requirements.

2. Second, ML is getting more attractive for the use-cases dependent on data that is otherwise difficult to analyse, e.g. unstructured data. For instance, this could be data from news sources, satellite images or social media. "Unstructured data" has very few pre-defined meanings. Modern ML models have proven very effective at extracting meaning from such data (e.g. classifying images), but in many applications, it can be harder to validate. Many such use cases are not really addressed by traditional

software and hence, there is no use case specific validation approach that is well-established.

3. Third, following from the points above, ML is also attractive for highly complex applications. It has been shown to be very successful at natural language processing, image recognition and optimisation in high-dimensional spaces (including financial applications). An example of this could be a situation where a bank aims to identify what is the lowest cost to execute a trade and corresponding strategy, given a number of constraints (e.g. order size and timeframe). There are multiple dimensions to this, such as exact timing, splitting up the trade into smaller parts, and multiple trading venues where it could be executed. In such a context, where there are multiple "moving parts", finding the optimal strategy becomes a complex problem to solve. Software used for such complex applications is inherently harder to validate than software solving simple problems.

Many of the challenges raised through ML approaches are novel to the software engineering community. Methods to address them are not yet fully available whilst the deployment of ML is rapid and extensive. In the absence of tangible solutions to ensure ML techniques have the appropriate controls, this deployment could give rise to concerns especially when it affects critical applications, such as the regulated financial services sector.

In the following sections, we discuss the challenges AI introduces for software validation in more detail.

IV.4 Potential challenges for AI software

IV.4.1 Issue 1: Lack of clear requirements for AI

The traditional Software Engineering approach is to devise very high-level requirements, and then refine them with an increasing level of detail. Software outputs can then be checked against such detailed (or "low-level") requirements (Figure 6).

Requirements highlight the inputs and outputs of individual software components, and what correct execution looks like. This implies also specifying what the software should not do. They constitute a set of rules for the IT system, against which it can be tested. Software that has been successfully tested against its requirements can be said to be verified. This means it has been established that the system works as intended for the specified cases.

Example of a credit assessment IT system with and without ML
For instance, a high level system requirement may be: the IT system has to determine the creditworthiness of a customer. The high level requirements would be to capture details of the customer and process them to arrive at a credit score. The lower level requirements would be to (i) capture relevant details including identification details (e.g. National Insurance Number, name and

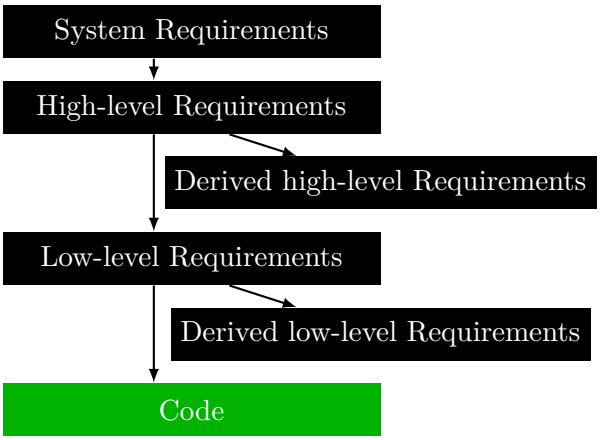

Figure 6: Requirements for software engineering. Source: Authors

address), income and credit reference agency inputs; (ii) calculate a probability of default using a pre-determined methodology using static factors as input; and (iii) generate a score that can be used to determine whether to lend to the customer and what terms and conditions to apply.

To test for example step (ii) in this case, one would back-test (to verify whether the pre-determined methodology would have captured the right credit score for past clients) or check sensitivity (e.g. plot how the probability of default would change vs. different statistical factors). Another test would be to check that for a set of factors on the extreme ends of the low and bad probability of default, the probability of default output is at the right end of the spectrum. An ML based software will calculate the probability of default using statistical regression based on a large database of clients and their credit scores, as well as any other data deemed necessary for its learning by the developer. It would evolve when the data is updated and calculate the probability of default based on an increasingly complex decision tree which is likely to be non-linear. It is clear therefore that to test whether ML based software produces the desired outcomes one would have to design more advanced tests. During such testing, what should be the acceptable range of performance for each evaluation criteria (e.g., minimum percentage of bad customers to be identified correctly by the model on a benchmark data set) to get the ML model approved, are often inconsistent across different firms. Although for the credit risk assessment there are some well established guidelines by regulators (e.g., policy statement PS11/20 on Credit risk: Probability of Default and Loss Given Default estimation [PRA, 2020]), for other similar applications, for example, pricing, there are very few.

Machine learning systems are hard to test Despite the widespread adoption of ML systems, they often exhibit unexpected behaviours. For instance, image recognition systems that work well in most cases may grossly mis-classify certain objects. For example, for an AI system within a self-driving car, obstacles

can be mistaken for a marking on the road [9]. Usually such an error should be ruled out through a requirement that is clearly tested against.

But for many other ML cases, such exception cases can be very hard to identify and test for. ML models are tested on test sets, often very similar to the ones that they have been trained on. However, special or low probability cases would need to be part of the test set in order for an artificial intelligence to know how to address them and finding such test cases can be challenging. That means verifying an ML model against requirements is fundamentally constrained by the nature of the test data set. On the other hand, many rules learned by the ML model may not be verified in the test set and thus their behaviours remain unknown throughout the model validation process.

In theory, synthetic data could be generated to test for such corner cases that lead to unexpected behaviour. In practice, for large-scale systems with millions of parameters it may not be possible to comprehensively test for such cases (see [Pei et al., 2019]). Hence, the key challenges in automated systematic testing of large-scale ML systems are mainly (i) how to trigger different parts of an ML system's logic to uncover different types of erroneous behaviours, and (ii) how to identify erroneous behaviours of such a system without checking for specific corner cases.

The problem gets exacerbated with "end-to-end" ML systems Additional complexity is introduced in machine learning systems that are more "end-to-end". This refers to an architecture in which a single trained component is implemented to perform an activity that would usually be done by multiple, interacting components. Particularly in a reinforcement learning framework, as in figure 7, where the agent is learning by experimenting with the environment, the whole sequence of actions and strategy corresponding to that could be made end to end by training a big complex deep neural network. For instance, an AI system can try to learn optimized trading decisions for a high-frequency trading environment with a complex information set, by trying different strategies in a simulated but realistic trading environment.

The implications of this choice of architecture for software validation are stark. Because, there are often no meaningful low-level requirements to capture in such framework and hence, support validation. Everything is learned and implemented end to end and often too complex to understand which section is doing what and why. Validation for a trained system like this can be done by means of a test environment with simulated inputs (separate from the training environment) and trialing in a real environment without making any realistic action (e.g., check what trading decision the AI system will make without actually executing the trade). This type of validation in a formal set up (with proper train, test and validation environment) is generally still very novel for the financial sector.

In such complex use cases with highly specialised AI systems, existing procedural, organisational (incl. managerial) and technological frameworks for

[9] see for example study by labsix (MIT students lab) on adversarial objects fooling AI here or research referred to here.

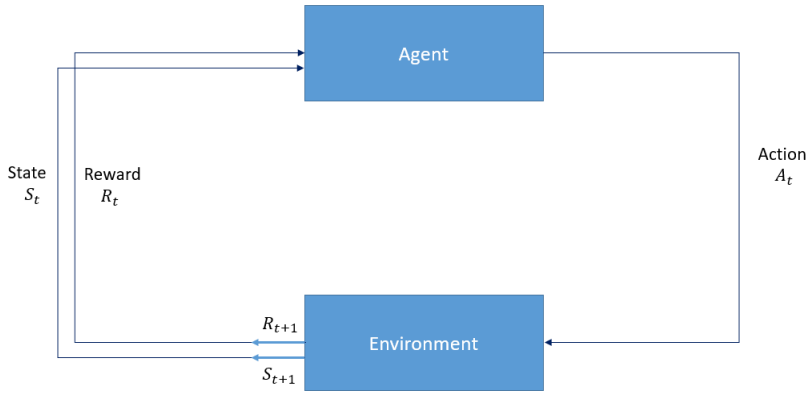

Figure 7: The agent–environment interaction in reinforcement learning. Source: Drawn by Authors from standard reinforcement learning literature [Sutton and Barto, 2018]

iterating software requirements are not quite applicable. In particular, static design methodologies might not be compatible with such non-deterministic ML techniques. This means that building the right test case at inception would become critical and that a wide range of test data (or inputs for simulated test environments) would need to be selected and maintained, in line with the self-learning evolution of the algorithm. But in practice, this is often too difficult to make them representative for the required production use-case. As a result, for many such ML applications, the desired behaviour of a component cannot be effectively expressed and tested as we can do with traditional software logic (see [Sculley et al., 2015]).

Potential solutions There is an emerging view in the literature of developing methods that are able to find corner cases through an informed search. That means, rather than iterating through all possible inputs, the methods "strategically" look for areas that have not been covered by the training set. They then employ methods that generate novel data points that "fool" one system but can be checked by other "checker" ML systems. However, our view is that such approaches are still in their infancy, and further work is necessary. In general, stressed testing with extreme inputs has been quite popularly used in traditional statistical models for a long time, which can help for some ML models as well.

Another potential solution is, trialing the new complex ML systems for a fairly long period to identify any such real-life corner cases. During the trialing period, the existing non-ML rule based system can be used for primary decision making, but the new system can still be used as a challenger system to compare both the outcome. This "parallel running" of the ML system can be used to validate its outputs, help to build the end-user's trust and ability to use it, and iron out any potential issues related to extreme or unknown scenarios.

Finally, to ensure that the ML system is not making drastic decisions from unexpected or unanticipated inputs, a proper ongoing monitoring system should

be in place for live usage. In general, during the actual ML modeling, the model developer should potentially provide the range of inputs within which the system can perform more reliably. If more extreme inputs are experience during live use, the ongoing monitoring system should flag that immediately and let some domain experts look into the situation to override the decision made by the ML system if needed.

The next two sections focus on data validation and explainability.

IV.4.2 Issue 2: Lack of explainability and diagnostic tools for AI

Financial institutions must comply with regulatory expectations and specific regulations when performing regulated activities. For example, assessing the appropriateness of specific financial services or products for individual customers must be performed in compliance with MiFiD II. Financial institutions therefore must be able to formally explain the basis of such decisions, being required to provide evidence on how they comply with regulatory expectations and specific regulations. An inability to do so could expose financial institutions to regulatory fines or some other form of sanction.

As a result, the explainability of any model used to support the provision of financial services, or products, needs to be transparent to regulatory scrutiny so that the financial institutions can assure themselves that they are compliant with regulatory expectations and relevant regulations. Additionally, regulators can also satisfy themselves that the financial institutions that they regulate are compliant with relevant regulations.

This expectation extends to AI and ML models that support the provision of financial services, or products. It may also extend to AI or ML models used by financial institutions to manage their operations and finances given that regulators have expectations regarding their financial and operational resilience. Similar to any statistical model, if the AI or ML model is making judgments on the basis of a misinterpretation of data used to train it, it would be reasonable to expect that a financial institution needs to be able to identify this deficiency and rectify it as soon as possible. And during real world usage, the model's output should be complying with regulatory requirements.

In rule-based software systems, existing diagnostic tools detect aspects or specific components of the system that may lead to similar problems coming from misinterpretation. For instance, if software produces an undesired result, the exact component that was faulty can be identified, and the code changed to address the issue (Figure 8). More generally, it is possible to rule out classes of undesired behaviours for traditionally engineered software.

Building an ML system equivalent to that where each section or component is clearly understood, is more difficult. As presented by [Carvalho et al., 2019], these algorithms can become so complex as to be a black box in terms of the decisions they make. This can prevent a human, expert, or nonexpert from being able to verify, interpret, and understand the reasoning of the system and how particular decisions are made.

AI systems often include large, non-parametric ML techniques which make them very hard to clearly disentangle and understand which sections of them are

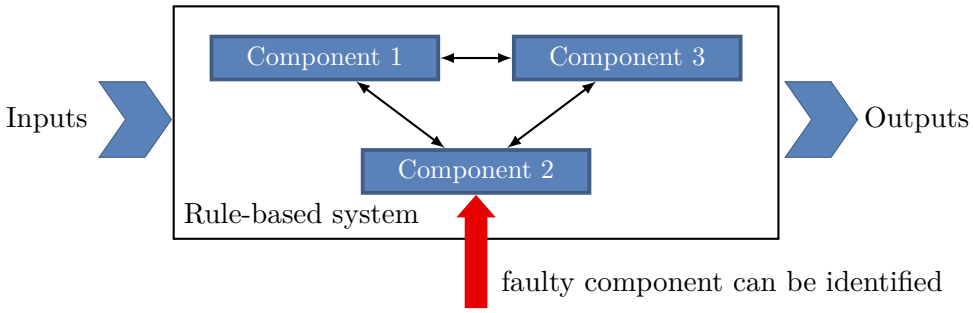

Figure 8: Stylised representation of a rule-based system. Source: Authors

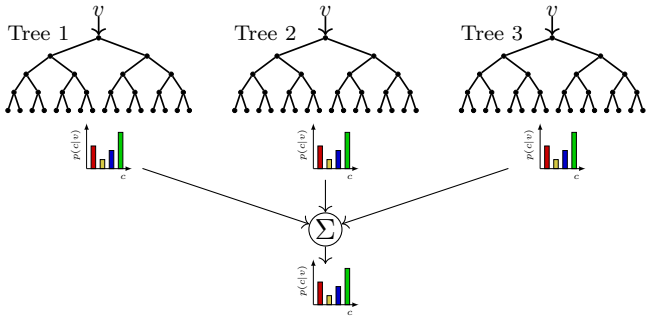

Figure 9: A small tree-based ensemble model already can be hard to explain. Source: Authors visualisation based on existing literature of decision tree based models

responsible for certain outputs. For instance, Figure 9 shows a relatively small tree-based ensemble model, which is at the less complex end of the machine learning spectrum. Such models can contain hundreds of trees in practice, whose results get averaged statistically. However, even with a small number of trees, it is difficult to fully explain the model and the importance of different trees and their sub-components in plain English. Due to this type of complexity of the models, their predictions are often difficult to explain and reverse engineer. It is thus not possible to know why exactly something has gone wrong. This, in turn, means it cannot be easily fixed.

This is sometimes referred to as machine learning's "black box" problem. It is important to note that even if ML models are available for inspection, their size, interconnected nature and complexity make it difficult to explain their operation to humans. For example, an ML model used to predict mortgage defaults may consist of hundreds of large decision trees deployed in parallel, making it difficult to summarize how the model works intuitively. Regulatory authorities have also identified this issue. For instance, the Financial Stability Board (FSB) highlighted in its 2017 report [FSB, 2017] that the lack of interpretability or "auditability" of AI and machine learning methods could become a macro-level risk. Similarly, a widespread use of opaque models may result in unintended consequences. IOSCO (see [IOSCO, 2020]) raised some of these concerns in its June 2020 consultation paper, in particular, on the usage of deep unsupervised

learning algorithms which are at the more complex end of the ML algorithms spectra.
Beyond the algorithms themselves, the operating system and process of an AI-driven software needs to be transparent to ensure that the AI is both trustworthy and can be used responsibly. A more recent publication by the Alan Turing Institute has also raised [Ostmann and Dorobantu, 2021].

The explainability of a model can help us to examine whether it has correctly interpreted inputs to make decisions or learned from some co-occurred information that exist widely among biased training data [Du et al., 2020]. To illustrate this specific issue, we will use an example from AI that recognises images. For instance, in an algorithm that identifies cats in images, the developer may not be able to know if the algorithm truly learned the features of a cat, or whether it learned a feature that co-occurred in all pictures in the training set. For instance, if all pictures of cat in the training set were taken on or near a couch, that may be the distinguishing feature that the algorithm uses. In this case, the developer might not be able to know that the algorithm was right, but "for the wrong reasons". This is often termed as "shortcut learning" and difficult to avoid.

Potential solutions Explainable AI is an active area of research and some recent developments [Lundberg and Lee, 2017] are already showing promise. Such techniques help to understand which factors were most important in a decision-making process. They can indicate when models ignore important parts of the training data set or are learning the "wrong solution". As compared in the paper mentioned above, Shapley values (SHAP) and LIME are the two popular measures used in this context. The Shapley value, the idea borrowed from game theory, is a solution concept to fairly distribute both positive and negative impacts of different factors to predict the target variable when those factors are working in coalition. The LIME technique attempts to understand the impacts of different factors in the model by perturbing the input of data samples and understanding how the predictions change. In their paper [Lundberg and Lee, 2017], the authors show that Shapley values provide the guarantee of accuracy and consistency whereas LIME is actually a subset of SHAP but lacks the same properties. Hence, in situations where the regulation requires explainability, e.g., "right to explanations" related to credit scoring, the Shapley value could be the only legally compliant method, because it is based on a solid theory and distributes the effects fairly among input features. However, further exploration and discussion with the regulators will be needed in this regard.

It is worth noting that there are a few disadvantages of the aforementioned techniques. Firstly, it can be computationally very expensive to implement these quantification measures. Whilst it is possible to narrow down to a few inputs on the basis of human decisions, explanations quantified using Shapley values always use all the input features, and therefore prove costly. Secondly, it might be difficult to explain these techniques themselves.
There is existing research work on this topic [Lundberg and Lee, 2016], that has tried to improve current practice, and this remains an active area of research.

If it has become clear that the algorithm was right "for the wrong reasons" it may still not be clear or straightforward how to fix the issue. For instance, as described in the above example, if a problem with the training data is identified, currently, re-training with new datasets can be the only way of addressing the bugs in trained models.

In their work on Machine Learning Interpretability: A Survey on Methods and Metrics [Carvalho et al., 2019], the authors compared a wide set of interpretability literature and methodologies, and concluded that one of the key challenges around it is its subjectivity as a concept. The authors argue that interpretability is a very subjective concept and, therefore, hard to formalize or quantify. They also refer to [Mills and Keil, 2004] to state that explanations may highlight an incompleteness, and that interpretability can and should be used to confirm the key criteria that one may wish to optimize. Five main areas that could be improved by good interpretability are, Fairness (to ensure that predictions are unbiased and do not implicitly or explicitly discriminate against protected groups), Privacy (to ensure that sensitive information in the data is protected), Reliability/Robustness (small changes in inputs do not cause large changes in outputs), Causality (by focusing on silo causal relationships e.g. human-AI), and Trust (it is easier for humans to trust a system that explains its decisions rather than just providing the decision). A complete black-box model can not be considered very good unless it achieves robustness in those five areas.

In general, the importance of interpretability should be taken into consideration during the AI system development phase while the model choices are made. Some models are more opaque than others, and an understanding of priority for specific use cases can lead to a more rational use of AI. Finally, even if the model is opaque or "black box", it can still be used as a challenger model until its actions are better understood as mentioned before.

Beyond interpretability: the issue of entanglement In order to understand the issue properly, we will try to better understand the concept of entanglement as explained by [Sculley et al., 2015]. Let us consider a system that uses input features $x_1, \ldots, x_n$ in a model. If we change the input distribution of values in x_1, even slightly, the importance, weights or influence of the remaining $n-1$ features may all change. Adding a new feature x_{n+1} can cause similar changes, as can removing any feature x_j. So in that sense, no inputs are ever really independent. The paper refers to this as the CACE principle, i.e., Changing Anything Changes Everything. CACE applies not only to input signals, but also to hyper-parameters, learning settings, sampling methods, convergence thresholds, data selection, and essentially every other possible tweak related to a robust ML model design.

The problem gets exacerbated further, if the individual ML-based components are put together, constituting a system of interacting ML components. In such systems, components can have complex influences on each other. When the quality of a component's outcomes depend on the output of previous components, blame cannot easily be assigned to individual components without decoupling

imperfection problems in component inputs [Nushi et al., 2017].

Potential solutions One possible mitigation strategy is to isolate models and decompose them in smaller sets to explain a decision. But this approach is only useful in situations in which sub-problems could be split into several sub-sets and where the errors in the component models are uncorrelated [Sculley et al., 2015]. In general, it could also be mitigated in the ML system design stage. The main problem in hand can be segmented in mutually exclusive and collectively exhaustive problems solved by separate but comparable ML components, where dependency among components can be minimized. However, for all problems such segmentation may not be possible without sacrificing model performance or accuracy.

Accuracy or robustness? More generally, we view that in most of the cases there will be a trade-off between accuracy and robustness in AI based software, which will make it more difficult to test and explain. Accuracy measures how correctly the model can predict the target variable. On the other hand, robustness measures how much an algorithm is able to cope with input errors or extreme values during execution. Artificial intelligence algorithms usually make decisions via optimising across a number of variables to achieve a given objective. It is generally not feasible to get an optimal output across a whole input space, without compromising robustness [Su et al., 2018, Zhang et al., 2019]. That means, if a model (which is more complex or powerful) has high accuracy, it can capture signals from different input variables quite nicely. But it also means that the model prediction can simply blow-up (i.e., give some extreme impractical predictions) if some of the inputs receive extreme values, potentially outside the input range the model is trained on. So the model is not very robust to handle noisy or extreme values. On the other hand, if another relatively simple model is robust enough to handle such extreme values, it is less able to capture all the subtle input signals needed to make a very accurate prediction.

As an example, a deep neural network that is trained to classify images might use low-level details (say a texture) to identify an object, delivering high accuracy. However, this means that a semantically irrelevant permutation of this low-level texture will result in misclassification, which means that the classifier is not robust. Given the increasing use of image processing for KYC by FinTechs and neo-banks, we need to be aware of the challenges. At the time of writing this paper, the research into the trade-off between accuracy and robustness is still a nascent field, and we still do not have good answers on how this trade-off could be improved, but we note that this problem was examined by an EU commission technical report. The report notably recommends adopting transparency at the conception stage and emphasizes the need for an explainability by design approach [Hamon et al., 2020]. This accuracy vs robustness trade-off becomes more complicated to identify or understand in some of the modern ML techniques, e.g., deep learning and First and Frugal tree [Luan et al., 2011]. The same model could actually be highly accurate for a given set of inputs and highly robust for another . Hence, someone can argue that such trade-off mentioned above does

not actually exist or at least, is not model dependent. For most of the use cases and popularly used models, we believe that it is important for the ML model developers to be careful about this trade-off. And more generally, it is useful to be aware of different schools of thoughts related to this.

Potential solutions In general, it is important to adopt a definition for robustness and accuracy[10], given the importance of both of these measures for AI-based software. Robustness will be taken to refer to the capacity of software to remain unaffected by small variations in test or live conditions, and accuracy to refer to how closely the software's output aligns with the expected output or the realised output (the latter being output generated by back-testing). Such validations and testing are generally difficult to accomplish by using a simple tick-box approach. Hence, an agreed framework to measure them can be useful. Use of a cross-validation approach together with stress testing the models with extreme inputs can often be very effective.

IV.4.3 Issue 3: Difficulty of data validation when using ML

To understand this issue better, let us take the example of complex derivative pricing software for a given derivative. The model would use a snapshot of everyday data to calibrate and calculate a price. There will be model risk associated with the way the price is calculated, and proper validation of the input data used to perform the calibration will be necessary to ensure that it is reliable and within the right range (usually performed by an independent reviewer). That price will then be used by a human to quote transactions, or by algorithms to manage risks. In a machine learning based software scenario this process would be quite different. The machine learning algorithm would look at a broader dataset of information including: historical prices ranges, any input model settings, and may also include relevant news and timing information. This data will be used by the algorithm to learn how to price the derivatives and risk manage it dynamically. It is no longer sufficient to verify that a given data snapshot of a few variables is reliable. Now, a whole evolving dataset needs to be validated to ensure that the machine learning algorithm does not learn in a skewed way (specially when the training data is imbalanced, i.e., more or less data for certain price ranges it is predicting). Data therefore plays a critical role in ML. Every ML model is trained and evaluated using datasets, and the characteristics of these datasets will fundamentally influence a model's behaviour. A model is unlikely to perform well in a live environment if its deployment context does not match its training or evaluation datasets, or if these datasets reflect unwanted biases [Gebru et al., 2018].

In recent examples concerns were raised when it was shown that ML models can reproduce or amplify unwanted societal biases reflected in datasets. Much like

[10]The trade-off between robustness and accuracy in machine learning techniques is a nascent field of research. See for example "Is Robustness the Cost of Accuracy? – A Comprehensive Study on the Robustness of 18 Deep Image Classification Models" , 2019 (Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, Yupeng Gao) here

a faulty capacitor in a circuit, the effects of these biases can propagate throughout an entire ML system. For example, the authors in [Buolamwini and Gebru, 2018] found that three commercial gender classifiers had near-perfect performance for lighter-skinned men while error rates for darker-skinned women were as high as 33%; Amazon cancelled the development of an automated hiring system because the system amplified gender biases in the tech industry [Dastin, 2018]. [Holstein et al., 2019] found that many industry practitioners turn first to datasets when problems have been identified, but that the sources of such issues can be difficult to identify.

In summary, datasets become an important modeling factor that needs validation as a key model input, rather than just traditional data cleaning. Standard data validation approaches focusing on data correctness, outlier detection and faulty labels identification may not be enough.

Let us try to understand how the datasets are treated for ML modeling. Generally, input datasets are split into three parts: (i) training data, used to train the model; (ii) validation data, used to calibrate the model settings; and (iii) testing data, used to check its accuracy with unseen data. Given they are cuts of the same larger dataset, the biases in that original data also become part of the test data. And there are relatively few approaches available to validate input datasets to resolve such issues that are very crucial for an ML system validation, e.g., biases, test-set dependency etc.

As such, we view that non-validated data can lead to three types of biases:

1. Mathematical bias that comes from the data not being appropriate for the adopted statistical measure to analyse them or from the data being erroneous (for example, mathematical/statistical technique assumes certain patterns or distribution in the input variable which if incorrect or imprecise leads to calibration errors).

2. Ethical bias where the disproportional representation within data discriminates against part of a given population on certain (potentially sensitive) characteristics such as race, religion, gender etc. Both IOSCO and the FSB for example view this as an area of concern (Cf. [FSB, 2017] and [IOSCO, 2020]).

3. Decision bias arising due to controls/optimisation constraints from the algorithm settings, governance or controls (this is often related to data snooping, when the model gives the result what we want rather than what it should). This can lead to risks of taking biased decisions that impact the firm/the market.

There is an active discussion around these biases. For example, the areas where AI usage might grow rapidly could be lending, insurance and mortgage attribution. If an ML model is trained on say, 20 years of historic data that, classifies certain attributes as high risk - eg. postcode (see [Larson et al., 2015] and [Zarsky, 2014]), it could lead to a form of ethical bias. This is a known problem in AI usage across a number of variables that would inform its decision-making objective (see [Buolamwini and Gebru, 2018], [Learned-Miller et al., 2020], and

[Lee et al., 2019] for example). Even if such variables are explicitly removed (e.g., postcode, gender etc.) from the input variables, their correlation with other included variables (i.e., occupation, which could be correlated to postcode) could still lead to such bias.

Potential solutions A few solutions are starting to be developed to protect an ML model from poor quality input data. For example, repeating data selection, where we can randomly sample from given data with replacement to create many training data sets and compute variance to understand its suitability for modeling. The final trained model is generally more robust when trained repeatedly with all those samples created.

Mathematical and decision bias are relatively easier to avoid with good ML modeling practice. But ethical bias is often difficult to identify or overcome. Potential solutions mentioned to overcome model interpretability or explainability challenges (issue 2) can help here as well to understand the influence of different factors. To better flag all kinds of bias in datasets, [Gebru et al., 2018] have developed a method "datasheets for datasets" approach, which proposes that every dataset be accompanied with a datasheet that documents its motivation, composition, collection process, recommended uses, biases. Several big tech firms have built-in methodologies, often based on testing data samples to detect faulty or skewed data before it affects performance [Breck et al., 2019]. In their paper [Hall and Gill, 2018], the authors [11] proposed correlation network graphs as a way to promote understanding by displaying important and complex relationships in a dataset. They suggested that, correlation network graphs can enhance trust in a model if variables with thick connections to the target are important variables in the model. Also, common sense relationships displayed in the correlation graph (also, coming from domain knowledge) should be reflected in a trustworthy model.

IV.4.4 Issue 4: Parallelisation raises sequencing risks

Parallelisation refers to a type of computation where processes are performed simultaneously, rather than being performed sequentially. AI systems that are implemented using neural networks rely on numerical computation, and often run in a highly parallelised fashion, say on Graphic Processing Units (GPUs), with tens of thousands of computational units. This means that the computation is split into independent parts, which are processed simultaneously on separate units. The results are subsequently collated. Such numerical optimization tasks are generally sequential in nature, and if we try to make them parallelisable using some kind of approximation algorithms or heuristics, the numerical precision of the computations could be reduced heavily.

Experience from numerical analysis gives strong evidence that low-precision computations lack robustness with respect to the ordering of the operations. In order to deliver higher speed, GPUs and clusters do not guarantee a deterministic

[11] Relying on the combination creates a strong entanglement: improving an individual component model may actually make the system accuracy worse if the remaining errors are more strongly correlated with the other components [Sculley et al., 2015].

ordering of the computations they perform. Algorithms need to be crafted very carefully to mitigate these effects [Stoer and Bulirsch, 2002].

In effect this means that, in a parallelised ML model training it can be the case that the same job is repeated but a different result is obtained, because of different random initialisation. This may be exacerbated when switching to different hardware, compiler, GPU driver or computational library if not careful.

Possible solutions To avoid or minimize this issue, the practitioners need to take care of it in the ML model design stage, they need to make sure that the modeling framework is robust enough. During the choice of heuristic, it is important that the model developers are aware of the upper bound of the error they can potentially make because of parallelisation. There are plenty of post-modeling tests the developers can perform to understand it better. It may increase computation cost, but in general, writing code with a robust numerical algorithm might be preferable compared to expanding the code just to increase precision.Testing and training algorithms on small chunks of data in a parallelised context, rather than try to make algorithms trained and tested in silo operated in parallel, might lead to better results. Some papers have begun examining this issue [Kamp et al., 2017]).

IV.4.5 Issue 5: Novel issues arising from integrating AI in systems

In traditional software validation approaches, the interaction among interfaces is tested by building an abstracted model of a system of interacting components and running through a large number of test cases. Some simplifications have to be made of the workings of individual components to keep the task computationally manageable. The more components a system has, the bigger the computational challenge becomes to test the system. When ML components are introduced, for example to improve accuracy, this can make validation significantly more challenging. This is in part as it increases the non-linearities and increases the risk of other issues such as over-fitting. Some papers examined some "rectified" algorithms to account for that (see [Kulathunga et al., 2020]) whilst research teams have examined continuous integration for ML software (see [Karlaš et al., 2020])

Moreover, and because of the highly non-linear nature of ML algorithms, there is a “potential for competing or mismatched nonlinearities in model components” (see [Goldstein and Coco, 2015]), e.g. if there is a mismatch between ML derived and theoretical components in a numerical model (see [Goldstein et al., 2014]). Such mismatches can inhibit the developers’ ability to understand sensitivity over a broad range of input parameter values.

A separate way in which system-level validation can become more difficult when ML is introduced, can be termed as “correction cascades” as mentioned in [Sculley et al., 2015].

As mentioned in the paper, there are often situations in which a model m_a for a problem A exists. But, we need to find solution for a slightly different problem A'. In this case, one can try to learn a model m'_a that takes outcome from m_a as input and learns how to make a small correction, as it can be a

faster way to solve the problem. However, this correction model has created a new system dependency on m_a, making it significantly more challenging and expensive to maintain and implement improvements to that model in ongoing basis. The cost increases when correction models are cascaded down the line. For example, suppose there is another model for problem A'' learned on top of m'_a, and so on, with several slightly different test distributions. Once in place, a correction cascade can create an improvement deadlock, as improving the accuracy of any individual component actually leads to system-level updates. It can become more complicated if there is any circular dependency because of poor design principles. Yet another way in which this can occur is, when an ML output changes qualitatively over time, due to re-calibration or data drift, affecting other parts of the system. Note that there is also a systemic dimension to the data validation point (issue 2).

Possible solutions The model-cascading issue mentioned above is often observed in the financial industry. For example, in credit risk management, model developers often build an Internal Ratings Based (IRB) model first [12]. After that, the outcomes from these models are used for developing other models (eg. accounting models used by auditors to calculate impairment provisions). Maintaining and updating two different suites of models can be challenging as the requirements for them can change independently. One potential solution to avoid such an issue is that one can add the input mode features from m_a (as mentioned in previous section) directly while creating a separate model for A'. In this case the models can be maintained independently. However, additional cost for a fresh model development should also be considered. If such model cascading can not be avoided, a rigorous and transparent ongoing model motoring system should be in place, with clear allocations of responsibility among different stakeholders.

IV.4.6 Issue 6: AI algorithms can systematically learn to play sophisticated collusive strategies

In a corrupted market environment, companies or market participants, which should ideally compete against each other, may conspire to work together in a non-competitive, secret or often illegal manner. This is termed as collusion. AI systems - through their learning techniques - can automatically learn to adopt collusive strategies. In the event of large scale ML model or AI system adoption to make pricing decisions, for example, this could have a material impact on markets. In an OECD (Organisation for Economic Co-operation and Development) paper [OECD, 2017] the authors raised concerns related to this potential issue, and discussed some of the challenges such algorithms present for competition, law enforcement and market regulation (the latter very relevant to financial markets).

In their experiments [Calvano et al., 2020], the authors constructed AI pricing agents and let them interact repeatedly in controlled environments to learn

[12] IRB models are used for regulatory purposes to understand capital requirements

about optimal pricing decisions. They have found that even relatively simple pricing algorithms systematically learn to play sophisticated collusive strategies. Because of the black-box nature of such algorithms as mentioned before, particularly for such reinforcement learning algorithms, it is very difficult to trace how they have learned to take such strategies. They learn to collude purely by trial and error, with no prior knowledge about the environment and even without communicating with other pricing agents. They don't need to be specifically designed or instructed to collude. For fair competition policy, this is a serious challenge and proper validation to avoid such situation is very difficult.

Possible solutions There is not yet an established technical solution to address such issues. However, more awareness can help to identify such issues sooner. One may even suggest to ban usage of such smart AI altogether in pricing decisions. Even if one argues that this move is not regressive, implementing such restrictions could be quite difficult in practice. [Beneke and Mackenrodt, 2021] have explored how fines, structural and behavioural remedies can serve to discourage collusive results while preserving the incentives to use efficiency-enhancing algorithms. Theoretically, the market participants can build other ML algorithms to identify such collusive strategies as soon as they are made by the pricing algorithms and then explicitly program them to abandon that strategy (or flag the identification to a human). However, more research is needed on this and good collaboration will be required among computer scientists, economists, and legal experts before making a concrete policy move.

V Conclusion: Key take-aways and next steps

In conclusion, the below key points should be borne in mind, in our view, by financial institutions when developing AI/ML solutions to support the provision of financial services:

1. ML software development is data driven rather than hand-crafted rule based making the technology hard to test conventionally, further exacerbated by end-to-end ML systems.

   Software validation may need to move from testing based on requirements to validation based on representative test data sets. These should include "corner cases" or "tail event" cases, representing scenarios not catered for by training data sets.

2. ML's "black box" nature can make it impossible to interpret how decisions are made. "Explainability" techniques may help attribute which factors are most important in a decision making process, but this may not enable identifying which part of a big ML model framework is responsible for any undesirable model behaviour. Entanglement can also mean inputs are not independent with complex interdependencies between ML components. Decomposing ML models into smaller parts to generate decisions can add clarity although such opportunities may diminish as ML architectures become more end-to-end.

3. The characteristics of training data sets fundamentally influence ML model behaviour, potentially replicating or amplifying data set bias. Training data sets must be validated to ensure they are correct and representative, addressing outlier data elements and faulty labels. Data sets used for different purposes such as ML training, calibrating ML models or checking the accuracy of ML models should be free of common biases or flaws to ensure that they are fit for the specific use case that they are applied to.

   Emergent solutions exist to ensure that data sets are fit for purpose including: repeating data selection in a random way; formally documenting the composition, collection process, recommended uses, and inherent biases of data sets; development of methodologies to detect faulty or skewed data sets; and network graphs to visualise data sets and highlight data relationships graphically.

4. Using parallel processing to support ML models can result in unintended or inconsistent outputs disruption if the ordering of computational steps and processing takes place out of sequence because of poor overall modeling framework. It's important that the ML models, particularly when there are inter-dependencies among components and different sub-models, have robust controls over the ordering of computation steps.

5. ML models are non-deterministic in nature. Some commentators have observed challenges associated with integrating non-deterministic ML models with software components that are deterministic, i.e. procedural in nature, when for example the output of an ML model changes qualitatively over time, due to re-calibration, impacting integrated software components.

As mentioned in the beginning of this paper, AI is increasingly becoming an area of focus of various regulators and governance bodies. Some guidance and discussions around testing and validation of AI have been published (see for example the BdF and ACPR discussion paper [Dupont et al., 2020]). However, the interaction of AI with existing software is seldom discussed. Based on our identified features of AI/ML, and as a starting point for debate, we suggest that policy makers and firms' governance bodies consider the following check-list for AI validation:

- Given that AI software is primarily used for decision making (either directly or indirectly), consideration should be given to an accountability regime around it. An accountable person could be, for example, responsible for delegating the decision making process to AI system for certain operations, and another person could be held accountable for how the AI itself makes decision.

- Given AI will generally be integrated into existing software, this might cause some unexpected behaviours both from the AI and existing software components. Therefore, it will be important to identify if any retesting of the existing traditional software is needed when integrating an AI component into it.

- There needs to be a process by which the integrity of data and models used for AI software operations is verified. In particular, our three identified possible sources of bias: Mathematical, Ethical and Decisional need to be considered at inception. This should also include checking and validating training data sets.

- Any software validation may need to consider the three risks we defined in section II, Model risk, Data risk and Technology risk and whether the inter-dependencies between these are understood and documented.

- An explainability strategy should be defined for each AI process, and assessed by an independent governance body. Traceability of decisions via automated logs is highly recommended.

- Regular stress testing of AI processes under extreme conditions and scenarios should be considered.

  More generally, we would suggest that as AI usage grows, the delegation of decision making to automated processes should always be accompanied by humans continuing to take these decisions at a smaller scale, at least in the earlier days of the AI system. Regular comparison between human and AI decision would be also important for ongoing monitoring. These are all our subjective views aiming to start a needed discussion on this field. Other literature went beyond our scope and considered possible risks from the usage of AI in regulation and risk management practices (see [Danielsson et al., 2021]).

We have tried to highlight in this paper why existing software validation processes will need to be revised to accommodate the inclusion of AI components in IT systems. We have also stressed on the importance of data in the context of AI (including ML), both from the perspective of training and test sets. As the application of human-designed hand-crafted logic is less in AI software compared to traditional software, required logic is directly learned from the data, it is very important to be careful about this input data. Finally, integrating ML into existing rules-based software comes with its own distinct challenges. The non-deterministic nature of AI and ML, often coupled with large-scale parallelisation, means that traditional regression testing (i.e., re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change made) cannot guarantee robustness. Again, currently, no water-tight approaches exist to address these challenges.

All areas of the financial system may not apply AI or ML to an equal degree, across organisations and even across business units within the same firm. As a result, the extent of the challenges posed by AI and the applicability of solutions will also vary. We have not focused much on application-specific challenges in this paper. One can consider that as a next step where more opinions from subject matter experts can be incorporated. Similarly, as a next step, one can try to work on more application-specific financial regulations highlighting any gap in existing regulations in a more explicit manner.

ML, and AI in general, can also help to automate a lot of the existing testing processes themselves and may improve the capacity we have now to test software, improving resilience. Therefore, creating the right framework for AI software testing could yield wide-reaching benefits, with the appropriate regulatory focus.

References

[Algaze, 2017] Algaze, B. (2017). Software is increasingly complex. That can be dangerous. https://www.extremetech.com/computing/259977-software-increasingly-complex-thats-dangerous. visited on 2021-09-13.

[Bank of England and FCA, 2019] Bank of England and FCA (2019). Machine learning in UK financial services. https://www.bankofengland.co.uk/-/media/boe/files/report/2019/machine-learning-in-uk-financial-services.pdf. visited on 2021-09-17.

[Beneke and Mackenrodt, 2021] Beneke, F. and Mackenrodt, M. O. (2021). Remedies for algorithmic tacit collusion. *journal of Antitrust Enforcement*, 9(1):152–176. https://doi.org/10.1093/jaenfo/jnaa040.

[Bourque and Fairley, 2014] Bourque, P. and Fairley, R. E. (2014). Guide to the software engineering body of knowledge, version 3.0. www.swebok.org. visited on 2021-09-13.

[Bouveret et al., 2015] Bouveret, A., Breuer, P., Chen, Y., Jones, D., and Sasaki, T. (2015). Fragilities in the U.S. treasury market: Lessons from the "Flash Rally" of October 15, 2014. https://www.imf.org/external/pubs/ft/wp/2015/wp15222.pdf. visited on 2021-09-17. IMF Working papers WP/15/222.

[Breck et al., 2019] Breck, E., Zinkevich, M., Polyzotis, N., Whang, S., and Roy, S. (2019). Data validation for machine learning. https://mlsys.org/Conferences/2019/doc/2019/167.pdf. visited on 2021-09-17. Proceedings of SysML.

[Breedon et al., 2018] Breedon, F., Chen, L., Ranaldo, A., and Vause, N. (2018). Judgement day: algorithmic trading around the swiss franc cap removal. In *Staff Working Paper No. 711*. Bank of England.

[Buchanan, 2019] Buchanan, B. G. (2019). Artificial intelligence in finance. http://doi.org/10.5281/zenodo.2612537. The Alan Turing Institute.

[Buolamwini and Gebru, 2018] Buolamwini, J. and Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. http://proceedings.mlr.press/v81/buolamwini18a.html. visited on 2021-09-17. Conference on Fairness, Accountability and Transparency, Proceedings of Machine Learning Research.

[Calvano et al., 2020] Calvano, E., Calzolari, G., Denicolò, V., and Pastorello, S. (2020). Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review*, 110(10):3267–97. https://doi.org/10.1257/aer.20190623.

[Carvalho et al., 2019] Carvalho, D. V., Pereira, E. M., and Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics journal, MDPI*, 8(8):832. https://doi.org/10.3390/electronics8080832.

[Chakraborty and Joseph, 2017] Chakraborty, C. and Joseph, A. (2017). Machine learning at central banks. *Bank of England Working Paper No. 674. SSRN*. https://dx.doi.org/10.2139/ssrn.3031796.

[Cielinska et al., 2017] Cielinska, O., Joseph, A., Shreyas, U., Tanner, J., and Vasios, M. (2017). Gauging market dynamics using trade repository data: the case of the swiss franc de-pegging. In *Financial Stability Paper No. 41*. Bank of England. https://www.bankofengland.co.uk/financial-stability-paper/2017/gauging-market-dynamics-using-trade-repository-data-the-case-of-the-swiss-franc-de-pegging, visited on 2021-09-17.

[Danielsson et al., 2021] Danielsson, J., Macrae, R., and Uthemann, A. (2021). Artificial intelligence and systemic risk. *Journal of Banking and Finance, Forthcoming*. http://dx.doi.org/10.2139/ssrn.3410948.

[Dastin, 2018] Dastin, J. (2018). Amazon scraps secret AI recruiting tool that showed bias against women. https://uk.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUKKCN1MK08G. visited on 2021-09-17, Reuters.

[Du et al., 2020] Du, M., Liu, N., and Hu, X. (2020). Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77. https://doi.org/10.1145/3359786.

[Dupont et al., 2020] Dupont, L., Fliche, O., and Yang, S. (2020). Governance of artificial intelligence in finance, discussion document. https://acpr.banque-france.fr/en/governance-artificial-intelligence-finance. visited on 2021-09-17.

[EBA, 2014] EBA (2014). Model validation. *European Banking Authority Regulation and policy*. visited on 2021-09-17.

[European Commision, 2021] European Commision (2021). Proposal for a regulation laying down harmonised rules on artificial intelligence (artificial intelligence act). https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence-artificial-intelligence. visited on 2021-09-17.

[European Commission, 2014] European Commission (2014). Markets in financial instruments (MiFID II) directive 2014/65/eu. https://ec.europa.eu/info/law/markets-financial-instruments-mifid-ii-directive-2014-65-eu_en. visited on 2021-09-17.

[FCA, 2018] FCA (2018). Cyber and technology resilience: Themes from cross-sector survey 2017/2018. *Financial Conduct Authority Survey*. https://www.fca.org.uk/publication/research/technology-cyber-resilience-questionnaire-cross-sector-report.pdf, visited on 2021-09-17.

[Forster, 1909] Forster, E. M. (1909). The Machine Stops. In *The Eternal Moment and Other Stories*. The Oxford and Cambridge Review.

[FSB, 2017] FSB (2017). Artificial intelligence and machine learning in financial service. https://www.fsb.org/wp-content/uploads/P011117.pdf. visited on 2021-09-17.

[Gebru et al., 2018] Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H. M., Daumé, H., and Crawford, K. (2018). Datasheets for datasets. *CoRR*, abs/1803.09010. http://arxiv.org/abs/1803.09010, visited on 2021-09-17.

[Goldstein and Coco, 2015] Goldstein, E. B. and Coco, G. (2015). Machine learning components in deterministic models: hybrid synergy in the age of data. *Frontiers in Environmental Science*. https://doi.org/10.3389/fenvs.2015.00033.

[Goldstein et al., 2014] Goldstein, E. B., Coco, G., Murray, A. B., and Green, M. O. (2014). Data-driven components in a model of inner-shelf sorted bedforms: a new hybrid model. *Earth Surface Dynamics*, 2(1):67–82. https://doi.org/10.5194/esurf-2-67-2014.

[Hall and Gill, 2018] Hall, P. and Gill, N. (2018). *An Introduction to Machine Learning Interpretability*. O'Reilly. ISBN: 9781492033141.

[Hamon et al., 2020] Hamon, R., Junklewitz, H., and Sanchez, I. (2020). Robustness and explainability of artificial intelligence. *EU Commission Joint Research centre*. ISBN 978-92-76-14660-5. JRC119336. https://doi.org/10.2760/57493.

[Holstein et al., 2019] Holstein, K., Vaughan, J. W., Daumé, H., Dudík, M., and Wallach, H. M. (2019). Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 600. ACM. https://doi.org/10.1145/3290605.3300830.

[Hynninen et al., 2018] Hynninen, T., Kasurinen, J., Knutas, A., and Taipale, O. (2018). Software testing: Survey of the industry practices. *41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1449–1454. https://doi.org/10.23919/MIPRO.2018.8400261.

[IOSCO, 2020] IOSCO (2020). The use of artificial intelligence and machine learning by market intermediaries and asset managers, consultation report june 2020. https://www.iosco.org/library/pubdocs/pdf/IOSCOPD658.pdf. visited on 2021-09-17.

[Jongh et al., 2017] Jongh, P. J. R. D., Larney, J., Mare, E., van Vuuren, G. W., and Verster, T. (2017). A proposed best practice model validation framework for banks. *South African Journal of Economic and Management Sciences*, 20(1). https://doi.org/10.4102/sajems.v20i1.1490.

[Kamp et al., 2017] Kamp, M., Boley, M., Missura, O., and Gartner, T. (2017). Effective parallelisation for machine learning. *31st Conference on Neural Information Processing Systems*, volume 30. https://papers.nips.cc/paper/2017/file/38811c5285e34e2e3319ab7d9f2cfa5b-Paper.pdf. visited on 2021-09-17.

[Karlaš et al., 2020] Karlaš, B., Interlandi, M., Renggli, C., Wu, W., Zhang, C., Mukunthu, D., Babu, I., Edwards, J., Lauren, C., Xu, A., and Weimer, M. (2020). Building continuous integration services for machine learning. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 2407–2415. https://doi.org/10.1145/3394486.3403290.

[Kinch et al., 2007] Kinch, J., Dey, P. K., and Ogunlana, S. O. (2007). Managing risk in software development projects: a case study. *Industrial Management and Data Systems*, 107(2):284–303. https://doi.org/10.1108/02635570710723859.

[Kulathunga et al., 2020] Kulathunga, N., Ranasinghe, N. R., Vrinceanu, D., Kinsman, Z., Huang, L., and Wang, Y. (2020). Effects of the nonlinearity in activation functions on the performance of deep learning models. *Cornell University Database*. https://arxiv.org/abs/2010.07359v1. visited on 2021-09-17.

[Larson et al., 2015] Larson, J., Mattu, S., and Angwin, J. (2015). Unintended consequences of geographic targeting. *Technology Science Dataverse (Harvard University)*. https://doi.org/10.7910/DVN/VEBPCZ/.

[Learned-Miller et al., 2020] Learned-Miller, E., Ordonez, V., Morgenstern, J., and Buolamwini, J. (2020). Facial recognition technologies in the wild: a call for federal office. https://circls.org/primers/facial-recognition-technologies-in-the-wild-a-call-for-a-federal-office, visited on 2021-09-17.

[Lee et al., 2019] Lee, N. T., Resnick, P., and Barton, G. (2019). Algorithmic bias detection and mitigation: Best practices and policies to reduce consumer harms. *The Brookings Institution*. https://www.brookings.edu/research/algorithmic-bias-detection-and-mitigation-best-practices-and-policies-to-reduce-consumer-harms/. visited on 2021-09-17.

[Leo et al., 2019] Leo, M., Sharma, S., and Maddulety, K. (2019). Machine learning in banking risk management: A literature review. *Risks journal, MDPI*, 7(1):29. https://doi.org/10.3390/risks7010029.

[Luan et al., 2011] Luan, S., Schooler, L. J., and Gigerenzer, G. (2011). A signal-detection analysis of fast-and-frugal trees. *Psychological Review*, 118(2):316–338. https://doi.org/10.1037/a0022684.

[Lundberg and Lee, 2016] Lundberg, S. and Lee, S. (2016). An unexpected unity among methods for interpreting model predictions. *CoRR*, abs/1611.07478. http://arxiv.org/abs/1611.07478, visited on 2021-09-17.

[Lundberg and Lee, 2017] Lundberg, S. M. and Lee, S. (2017). A unified approach to interpreting model predictions. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 30:4768–4777. https://arxiv.org/pdf/1705.07874.pdf, visited on 2021-09-17.

[Mills and Keil, 2004] Mills, C. M. and Keil, F. C. (2004). What lies beneath? understanding the limits of understanding. thinking and seeing: Visual metacognition in adults and children. *MIT Press: Cambridge, MA, USA*, pages 227–249. ISBN: 9781417560646.

[Nushi et al., 2017] Nushi, B., Kamar, E., Horvitz, E., and Kossmann, D. (2017). On human intellect and machine failures: Troubleshooting integrative machine learning systems. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1017–1025. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/15032, visited on 2021-09-17.

[OECD, 2017] OECD (2017). Algorithms and collusion: Competition policy in the digital age. https://www.oecd.org/competition/algorithms-collusion-competition-policy-in-the-digital-age.htm, visited on 2021-09-17.

[Ostmann and Dorobantu, 2021] Ostmann, F. and Dorobantu, C. (2021). AI in financial services. *The Alan Turing Institute.* https://doi.org/10.5281/zenodo.4916041.

[Pei et al., 2019] Pei, K., Cao, Y., Yang, J., and Jana, S. (2019). Deepxplore: Automated whitebox testing of deep learning systems. *Commun. ACM*, 62(11):137–145. https://doi.org/10.1145/3361566.

[PRA, 2018] PRA (2018). Supervisory statement ss5/18: Algorithmic trading. https://www.bankofengland.co.uk/-/media/boe/files/prudential-regulation/supervisory-statement/2018/ss518.pdf. visited on 2021-09-17.

[PRA, 2020] PRA (2020). Capital requirements directive. https://www.bankofengland.co.uk/-/media/boe/files/prudential-regulation/policy-statement/2020/ps1120.pdf. visited on 2021-09-17.

[PRA, 2021] PRA (2021). Policy statement ss11/20: Credit risk: Probability of default and loss given default estimation. https://www.bankofengland.co.uk/prudential-regulation/key-initiatives/capital-requirements-directive-iv. visited on 2021-09-17.

[Proudman, 2019] Proudman, J. (2019). Managing machines: the governance of artificial intelligence. https://www.bankofengland.co.uk/-/media/boe/files/speech/2019/managing-machines-the-governance-of-artificial-intelligence-speech-by-james-proudman.pdf. visited on 2021-09-13.

[Scandizzo, 2016] Scandizzo, S. (2016). *The validation of Risk Models*. Springer. ISBN: 978-1-137-43696-2.

[Schroeder et al., 2018] Schroeder, F., Allan, M., Lepone, A., Leung, H., and Satchell, S. (2018). Flash crash in an OTC market. *FCA, Occasional paper 37*. https://www.fca.org.uk/publication/occasional-papers/occasional-paper-37.pdf, visited on 2021-09-17.

[Sculley et al., 2015] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J., and Dennison, D. (2015). Hidden technical debt in Machine Learning systems. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 2503–2511. http://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems, , visited on 2021-09-17.

[Stoer and Bulirsch, 2002] Stoer, J. and Bulirsch, R. (2002). *Introduction to Numerical Analysis*. Springer. ISBN: 9780387954523.

[Su et al., 2018] Su, D., Zhang, H., Chen, H., Yi, J., Chen, P., and Gao, Y. (2018). Is robustness the cost of accuracy? - A comprehensive study on the robustness of 18 deep image classification models. *CoRR*, abs/1808.01688. http://arxiv.org/abs/1808.01688, visited on 2021-09-17.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). Reinforcement learning: An introduction. *The MIT Press*. http://incompleteideas.net/book/the-book-2nd.html, visited on 2021-09-17.

[US Department of the Treasury et al., 2015] US Department of the Treasury, the Board of the Governors of the Federal Reserve System, the Federal Reserve Bank of New-York, the US Securities Exchange Commission, and the US Commodities and Futures Trading Commission (2015). The US treasury market on October 15, 2014. https://home.treasury.gov/system/files/276/joint-staff-report-the-us-treasury-market-on-10-15-2014.pdf, visited on 2021-09-17.

[Zarsky, 2014] Zarsky, T. (2014). Understanding discrimination in the scored society. *Washington Law Review*, 89(4):1375–1412. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2550248, visited on 2021-09-17.

[Zhang et al., 2019] Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. (2019). Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. http://proceedings.mlr.press/v97/zhang19p/zhang19p.pdf, visited on 2021-09-17.