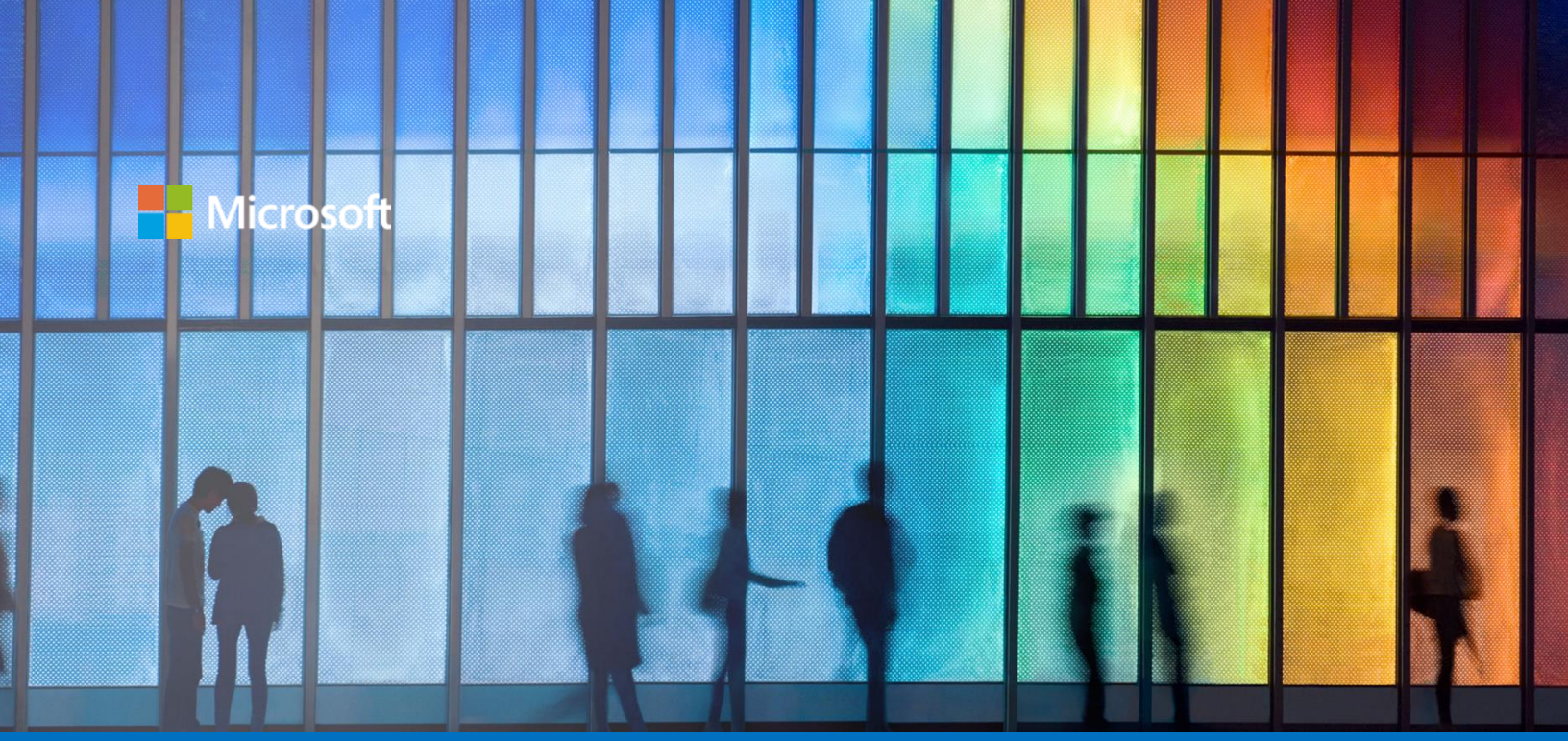


Putting Responsible AI into practice for your AI-based solutions

A starter guide for data scientists, AI developers, and other AI practitioners

July 2021

Author: Abderrahmane Lazraq, Microsoft France

Reviewer/Contributor: Philippe Beraud, Microsoft France

For the latest information about establishing a Responsible AI strategy, please see https://aka.ms/**AIBS**

For the latest information about designing, building, and managing your Responsible AI solution, please see https://aka.ms/**RAIresources**

This page is intentionally left blank.

Table of contents

Notice

MICROSOFT DISCLAIMS ALL WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, IN RELATION WITH THE INFORMATION CONTAINED IN THIS WHITE PAPER. The white paper is provided "AS IS" without warranty of any kind and is not to be construed as a commitment on the part of Microsoft.

Microsoft cannot guarantee the veracity of the information presented. The information in this guide, including but not limited to internet website and URL references, is subject to change at any time without notice. Furthermore, the opinions expressed in this guide represent the current vision of Microsoft France on the issues cited at the date of publication of this guide and are subject to change at any time without notice.

Publication: July 2021
Version 1.0

About this guide

Welcome to this **Putting Responsible AI into practice for your AI-based solutions** starter guide for data scientists, AI developers, and other AI practitioners.

The tech industry is being called upon to develop and deploy Artificial Intelligence (AI) technologies and Machine Learning (ML)-powered systems (products or services) and/or features more responsibly – we will further refer to these as **AI systems** -. Yet many organizations implementing such AI systems report being unprepared to address AI risks and failures.

To meet these challenges, Microsoft is striving to adopt a human-centered approach to AI, designing and building technologies that benefit people and society while also mitigating potential harms. This includes understanding human needs and using these insights to drive development decisions from beginning to end.

This guide consists of a series of modules for data scientists, AI developers and other AI practitioners, as well as potentially anyone interested considering the wide range of socio-technical aspects involved in the subject.

As such, it is intended to:

- Share and comment from the intended audience perspective our ongoing journey towards Responsible AI (RAI), starting from core principles from which this effort is deeply anchored to the practices we adopt, enforce, and evolve company-wide in terms of end-to-end lifecycle for the design, the development, the deployment, and the monitoring of these AI systems,
- Give you an overview of the most prominent RAI tools we open sourced as (standalone) libraries and dashboards, or integrated into Azure Machine Learning (Azure ML) and its MLOps capabilities, and where to leverage them in your own product development lifecycle.

As far as the latter is concerned, this guide will more particularly explore two categories of these practical tools that help putting RAI principles into practice:

1. Tooling to (better) understand the behavior of AI systems,
2. Tooling to help protecting AI systems' models and the data assets,

As well as widgets which group these tools under a single roof and allow access to them through a unified set of dashboards.

The document is organized as follows. **Module 1** introduces Responsible AI. One should note that the definition of Responsible AI varies from organization to organization. Here we will focus on Microsoft's definition of Responsible AI, and the related approach, and provide you for that purpose with an overview of Microsoft's Responsible AI journey starting with our core RAI principles, the standards in place, the related activities and practices in terms of AI systems development lifecycle, along with the tooling to sustain the implementation of the requirements and to help fulfill the various RAI objectives.

Then **Module 2** and **Module 3** each explore a series of Responsible AI tools that can be used to understand and protect AI systems and the data used for training or inference, respectively. For each Responsible AI tool, we will provide a description of the concept behind that tool, a description of how it works and a hands-on tutorial to walk you through each outlined tool.

The goal is indeed to allow you to jump straight into using each of these techniques by providing you with minimal example code to get you started as well as pointers towards more comprehensive resources if you wish to expand your knowledge of a particular technique.

By the end of the guide, you will be able to:

- Have an overview of the why, the what and the how regarding the adoption of a Responsible AI strategy and approach for your AI-based solutions, illustrated through the lenses of the Microsoft journey, from **Microsoft's core Responsible AI principles** to the way these principles translate into a framework of requirements, guidance, and governance through the Responsible AI standard and related practices.
- Assess and mitigate your AI system's unfairness issues using **Fairlearn**.
- Understand your model's global behavior or understand the reasons behind individual predictions using the unified **InterpretML** API.
- Use **Responsible-AI-Widgets** interpretability, error analysis and fairness dashboards to develop and monitor your own AI systems, and related solutions more responsibly.
- Protect personal data for your ML-powered solutions and applications using Differential Privacy (DP) through the **SmartNoise** system.
- Identify and anonymize Personally Identifiable Information (PII) in your data using **Presidio** data protection and anonymization SDK.

Guide elements

For each Responsible AI tool explored in this guide in Modules 2 and 3, we provide the following elements:

- **Description of the tool** and the underlying concept.
- **Hands-on tutorial** containing the most important steps and their outputs. These are meant to show you the core elements and do not contain for example code data loading and processing which we preferred to omit here for conciseness, please refer to the Jupyter notebooks for a comprehensive run through each tutorial.
- **Samples' interactive Jupyter notebooks** which you can access by downloading or cloning the following GitHub repository: **https://github.com/alazraq/responsible-ai-tools-tutorials**.

Guide prerequisites

To successfully leverage the code in the Jupyter notebooks accompanying Module 2 and Module 3 in this guide, you have multiple options with the four main ones being:

1. **Running the samples' Jupyter notebooks in JupyterLab.** JupyterLab is a web-based interactive development environment for creating such Jupyter notebooks.

   JupyterLab provides flexible building blocks and an interface for interactive, exploratory computing with, at its core, the ability to configure and arrange the user interface (UI), manipulate data with Python and display inline graphs, e.g., diagrams and/or dashboards from the results, etc. to support a wide range of workflows in data science and ML.

   Official instructions on how to install JupyterLab are provided here. For Windows 10 users, JupyterLab can be installed for example with the Windows Subsystem for Linux (WSL) 2 or through the Anaconda open-source distribution.

2. **Running the samples' Jupyter notebooks in Visual Studio Code**. Visual Studio Code (VS Code) is a free code editor and development platform that you can use locally on your Linux, MacOS or Windows environments, or connected to remote compute.

   Combined with the Jupyter extension (and the Python extension), it offers a full environment for Jupyter development that can be enhanced with additional language extensions. If you want a best-in-class, free Jupyter experience with the ability to leverage your compute of choice, this is a great option. Using VS Code, you can develop and run notebooks against remotes and containers.

   Instructions on how to get started with notebooks in VS code (optionally using container) are provided here.

   -or as an alternative -

3. **Running the samples' Jupyter notebooks in GitHub CodeSpaces**. GitHub CodeSpaces currently in beta provide you with cloud-hosted environments where you can edit your notebooks using VS Code or your web browser and store them on GitHub.

   GitHub CodeSpaces offers the same great Jupyter experience as VS Code above, but without needing to install anything on your local environment. GitHub CodeSpaces also allows you to use your cloud compute of choice. If you don't want to set up a local environment and prefer a cloud-backed solution, then creating a codespace is a great option.

   Instructions on how to get started with notebooks in your own codespace are provided here.

4. **Running the samples' Jupyter notebooks in Azure Machine Learning**. Azure Machine Learning (Azure ML) provides an end-to-end ML platform to enable you to build and deploy models faster on Azure.

   Azure ML allows you to run notebooks on a virtual machine (VM) or a shared cluster computing environment. If you are in need of a cloud-based solution for your ML workload with experiment tracking, dataset management, and more, we recommend Azure ML.

   To run the Jupyter notebooks in your Machine Learning workspace, please note that if you choose this option, there are two prerequisites:

   1. Having an Azure subscription. If you don't have an Azure subscription yet, you can create a free account here.
   2. Having a Machine Learning workspace. You will find instructions on how to setup one using the Azure portal here.

   Instructions on how to get started with Azure ML are provided here.

Other options to run Jupyter notebooks using products and services from Microsoft and GitHub exist. They can be found here.

Please note that whatever the option you choose, and provided you are using Python 3.7 or above, **the code within the Jupyter notebooks is self-sufficient**. You just need to uncomment the cells for libraries installation using the pip package manager when necessary and you should be good to go! Let's shortly see how to clone them.

Cloning the samples' Jupyter notebooks

The starter guide samples' repo on GitHub contains a series of samples' Jupyter notebooks.

Depending on your choice about the above prerequisites, you have various options available to clone this repo. For the sake of this starter guide, we assume here that you have a Windows 10 local environment with possibly WSL 2 installed with an Ubuntu distribution. (This also *de facto* covers both Linux and MacOS environment). Let's consider them in order.

Cloning the repo using GitHub Desktop

To clone the repo on your Windows 10 local machine, perform the following steps:

1. Download the GitHub Desktop installer and run it.
2. In the pop-up window, click **Install**. Follow the instructions.
3. Open a browser session and navigate on GitHub to the main page of the samples' repo at https://github.com/alazraq/responsible-ai-tools-tutorials.
4. Click **Clone or download**.
5. Click **Open in Desktop** to clone the repository. GitHub Desktop opens up and a dialog shows up.
6. Click **Choose...** and, using Windows Explorer, navigate to a local path where you want to clone that repo. Throughout this starter guide, we will use the `raisamples` folder as an illustration.
7. Click **Clone**.

See Cloning a repository from GitHub to GitHub Desktop.

You can instead install Git for Windows.

Cloning the repo using Git for Windows

To clone the repo on your Windows 10 local machine, perform the following steps:

1. Download the Git for Windows and run it.
2. In the pop-up window, click **Install**. Follow the instructions.
3. Open a PowerShell console, and run the following command:

```
PS C:\> md raisamples
PS C:\> cd C:\raisamples\
PS C:\> git clone https://github.com/alazraq/responsible-ai-tools-tutorials.git
```

Cloning the repo using Git on Ubuntu

Likewise, to clone the repo on your Ubuntu environment (WSL2), from a Bash terminal console, run the following commands:

```
$ cd $home
$ mkdir raisamples
$ cd raisamples
$ git clone https://github.com/alazraq/responsible-ai-tools-tutorials.git
```

So, at this stage, you're all set! It's high time to move to the first module.

If you are only interested in the tutorials themselves, please feel free to skip to modules 2 and 3. But we highly encourage you to follow the guide order as it defines what Responsible AI is from Microsoft's perspective and builds up from there.

Module 1: Building Responsible AI solutions

Responsible innovation is top of mind. Advancements in AI are indeed different than other technologies because of the pace of innovation – there has been hundreds of research papers published every year in the past few years -, but also because of its proximity to human intelligence. These advancements are:

- **Impacting us at a personal and societal level**. We refer to this as the sociotechnical impact of AI, which has given rise to an industry-wide debate about how the world should/shouldn't use these new capabilities. It is not because you can do something that you should necessarily do it.
- And also **changing the way companies approach AI**. The topic of Responsible AI is increasingly becoming an important theme as more companies struggle with challenges in terms of governance, security and compliance – you have hereafter some considerations and related percentages capturing interest of companies when investing in AI and ML technologies from a study conducted by the Capgemini Research Institute back in 2019 (See Organizations must address ethics in AI to gain public's trust and loyalty).

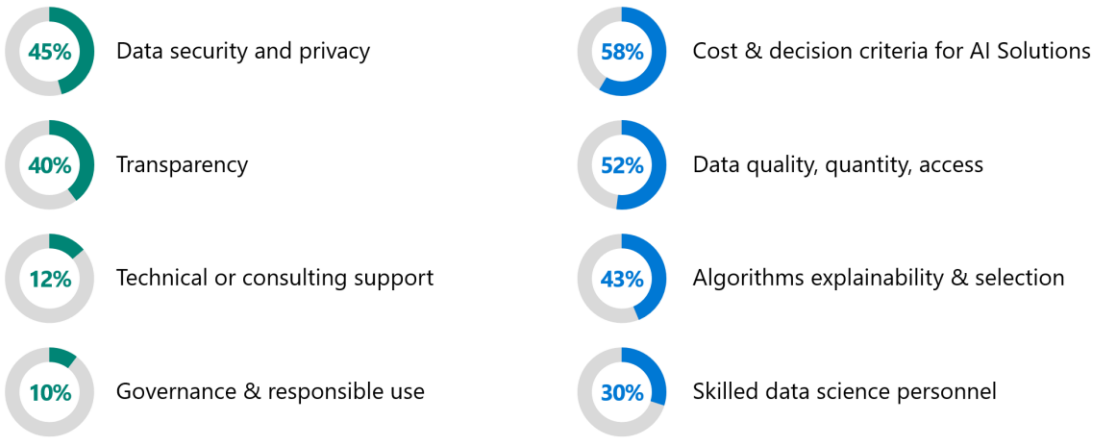


Figure 1. Most important considerations when investing in AI and ML technologies

The ability of AI systems to ensure data security and privacy, and the level of transparency of how systems work and are trained, are now the two most important requirements when investing in ML technologies. It's a marked shift from previous years, when scalability, performance, and ease-of-use topped investment priorities, in that order.

At the same time, studies have shown that explainability of algorithms are one of the key factors holding businesses back from implementing AI-powered solutions. Trust in an AI system is tied to its transparency, and without trust, people do not want to adopt AI or any other technology for that matter.

There are hundreds of ML algorithms and related techniques that data scientists can already access, and new ones emerge every day through various research papers, and all the AI innovation resulting from that is pushing the boundaries of science and technology – for example, the AI at Scale, on Microsoft Research (MSR) constitutes a new approach to AI that is fueling the next generation of AI innovation at scale with massive AI models trained with innovative tools and advanced infrastructure -.

If these AI systems are opaque and unable to explain how or why certain results are presented, this lack of transparency will undermine trust in the system and in any results they produce. The stance is clear: **people don't use what they don't trust**.

While it is true that sometimes the current state of the art of some technology doesn't allow to achieve something today, e.g., an intimate understanding of how a Deep Learning model comes up with a specific prediction, it doesn't mean we will remain in this situation forever; breakthroughs in innovation keep reminding us of how they possibly enable an alternative perspective and a new horizon for the future in various dimensions (See Innovation at Microsoft), and in the meantime we can already approximate the global vs. local behavior of such a models, see section **InterpretML** below).

"The more powerful the tool, the greater the benefit or damage it can cause… Technology innovation is not going to slow down. The work to manage it needs to speed up."

- *Brad Smith, President and Chief Legal Officer, Microsoft*

At the same time, one should also acknowledge that responsible innovation and consequently Responsible AI are an important yet underrated subject among data scientists, AI developers, and other AI practitioners in general. This is a current reality that undermines our collective efforts to gain control over AI systems and tackle all the concerns they induce, ranging from privacy concerns and information transparency to raising questions of who should be held accountable when AI systems behave unexpectedly or induce undesired side effects.

Addressing these ethical, transparency, and accountability concerns is no easy feat and requires coordinated multi-disciplinary efforts between not only data scientists, technical experts, but also ethics experts, law makers, etc. These efforts range from defining Responsible AI principles and outlining requirements to follow and goals to pursue in enacted policies and standards, to defining practices for solving some of these issues along with adapted guidelines and tooling.

This must then translate into the development lifecycle of these AI systems to **ensure that *what we (decide to) build* benefits people and society, and that *how we build* it begins and ends with people in mind. Responsible AI is Human-Centered.**

So, with all of that in mind, let's start by our own journey in this space to illustrate this before dedicating some time to explore the already available Responsible AI resources (that continuously evolve).

Understanding Microsoft's Responsible AI journey

"When your technology changes the world, you bear a responsibility to help address the world you have helped create."

- *Brad Smith, President and Chief Legal Officer, Microsoft*

The goal of this section is threefold:

1. Share our principles and how they translate into practices company-wide for our services and products.
2. Create awareness of the governance framework we have in place and the way we establish and evolve our standard to translate our responsibility in design, development, and deployment of AI systems.
3. Further present and discuss related practices and the already available tooling to sustain the implementation of the requirements outlined in the standard, and the related processes and activities as part of our ongoing Responsible AI journey.

Our journey towards Responsible AI begins nearly 5 years ago, with Satya Nadella penning an article in the Slate magazine titled The partnership of the future where our CEO explores how humans and AI can work together to solve society's greatest challenges. This article introduced concepts of transparency, efficiency but not at the expense of the dignity of people, intelligent privacy, algorithmic accountability, and protection against bias.

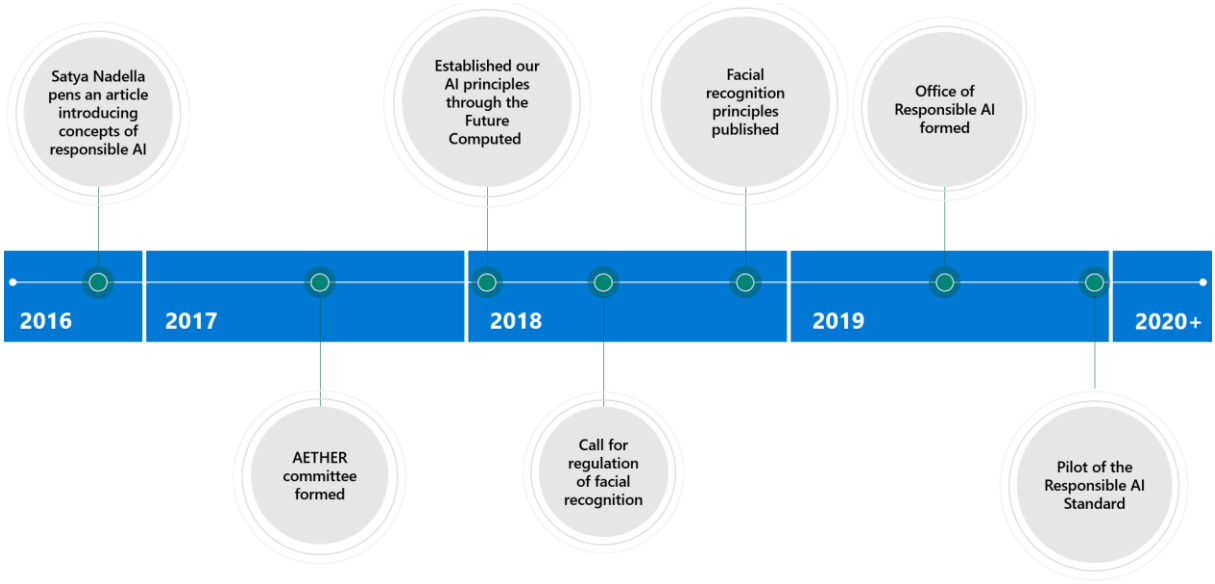


Figure 2. Microsoft's Responsible AI Journey

At the Microsoft Build conference in May 2017, Satya built on this with references to George Orwell and Aldus Huxley, and shortly after that, in July 2017, Microsoft formed our AI Ethics committee – AETHER (*AI, Ethics, and Effects in Engineering and Research*), an executive committee only, formed as a cross-company advisory group on AI ethics and effects in engineering and research for the Microsoft Senior Leadership Teams (SLT).

"AETHER will ensure our AI platform and experience efforts are deeply grounded within Microsoft's core values and principles and benefit the broader society. Among other steps, we are investing in strategies and tools for detecting and addressing bias in AI systems and implementing new requirements established by the GDPR. While there is great opportunity, ensuring we always act responsibly for our customers and partners will continue to be a hallmark of our work."

- Satya Nadella

AETHER spent time listening to our customers and internal experts, and then partnered with Legal Affairs to publish The Future Computed e-book in January 2018 that articulate six core principles that should guide our work and investments around AI.

As outlined above, Microsoft calls for Facial Recognition technology regulation in July 2018, and published later by the end of the year in December Facial Recognition principles.

Microsoft formed the Office of Responsible AI (ORA) in early 2019 and in the fall of this year, we published internally the first version of our Responsible AI Standard, a set of rules for how we enact our responsible AI principles underpinned by Microsoft's corporate policy. We are about releasing a version two of this Standard.

With that timeline in mind, lets now further consider the abovementioned core principles that pave our journey.

An introduction to Microsoft's AI principles

As Microsoft, we believe that the development and deployment of AI must be guided by the creation of an ethical framework. We set out our view back in 2018 in The Future Computed e-book with six core principles. These principles are the foundation for a responsible and trustworthy approach to AI at Microsoft. They act as a mental tool or framework in which to organize thinking about ethics at Microsoft.

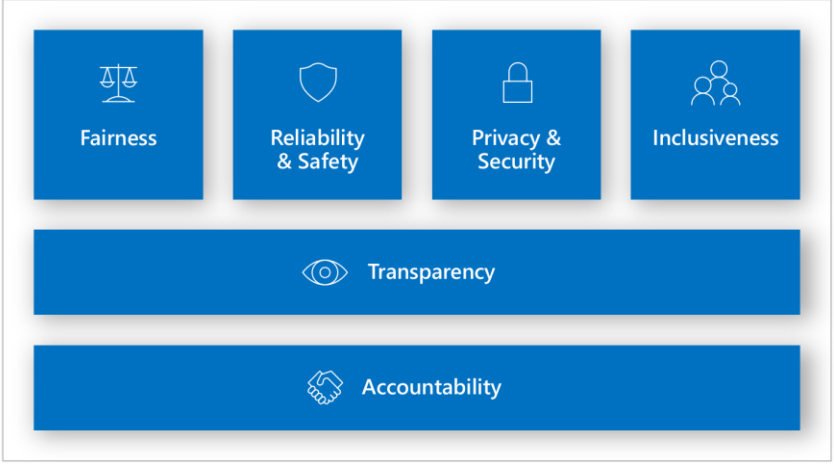


Figure 3. Microsoft Responsible AI Principles

A brief explanation of each principle is given below for your convenience:

1. **Fairness**. AI systems must be developed to **treat everyone fairly and avoid affecting similarly situated groups of people in different ways**. This principle acknowledges that defining and mitigating fairness issues for an AI system depends on understanding the system's purpose and context of use, and that a system's fairness reflects decision-making during both development and deployment.

   For example, when AI systems provide guidance on medical treatment, loan applications, or employment, they should make the same recommendations to everyone with similar symptoms, financial circumstances, or professional qualifications, regardless of other sensitive features like sex and race for example.

2. **Reliability & Safety**. To build trust, it's also important that AI systems operate reliably, safely, and consistently under normal circumstances and in unexpected conditions. How they behave and the variety of conditions they can handle reliably and safely largely reflects the range of situations and circumstances that developers can anticipate during design, build, prototyping, and testing.

   This principle encompasses consideration of the harms that might come from a technology, and ways employees can strive to minimize those risks, so technologies can give the greatest benefits to their users.

3. **Privacy and Security**. It's also crucial to develop AI systems that **can protect private information and resist attacks**. As AI becomes more prevalent, protecting privacy and securing important personal and business information is becoming more critical and complex. Privacy and data security issues require special close attention for AI because access to data is essential for AI systems to make accurate and informed predictions and decisions about people. Data must be secure at all stages, and to this end, actions must be taken to institutionalize privacy and security.

4. **Inclusiveness**. For the 1 billion people with disabilities around the world, AI technologies can be a game-changer. AI can improve access to education, government services, employment, information, and a wide range of other opportunities.

   Inclusive design practices can help system developers understand and address potential barriers in a product environment that could unintentionally exclude people. By addressing these barriers, we create opportunities to innovate and design better experiences that benefit everyone.

5. **Transparency**. When AI systems are used to help inform decisions that have tremendous impacts on people's lives, it's critical that people understand how those decisions were made. A crucial part of transparency is what we refer to as intelligibility or the useful explanation of the behavior of AI systems and their components.

   Improving intelligibility requires that stakeholders comprehend how and why they function so that they can identify potential performance issues, safety and privacy concerns, biases, exclusionary practices, or unintended outcomes. We also believe that those who use AI systems should be honest and forthcoming about when, why, and how they choose to deploy them.

6. **Accountability**. We believe the people who design and deploy AI systems **must be held accountable for how their systems operate** and for the impact of that operation on society. This includes considering the structures that can be implemented to ensure accountability at multiple levels, including design, development, sales, marketing, and use, as well as advocacy for the regulation of technologies when warranted.

Over the past few years, principles around developing AI responsibly have proliferated and, for the most part, there is overwhelming agreement on the need to prioritize issues like transparency, fairness, accountability, privacy, and security, see Principled Artificial Intelligence.

While principles are necessary, having them alone is not enough. The hard and essential work begins when you endeavor to turn those principles into practices. Merely having principles indeed does not imply a change in a company's culture unless those principles are made concrete through standards, practices and tools that help full time employees (FTEs) work through how to think responsibly when designing, developing, deploying, and monitoring AI systems.

How do these principles translate into practice?

To advance these principles and make sure they are enforced and implemented into the company's workflows, Microsoft developed several tools for incorporating applied ethics in technology. All these tools serve an ethical end; some are more procedural and are explored here, while others are more technical in nature and are explored in the next modules.

At Microsoft, we are moving from principles to practice since 2019 with the objectives to also empower our customers to do the same, as we did before for example with our Microsoft Security Development Lifecyle (SDL).

We (continue to) establish building blocks that will (or already are) the basis for our Responsible AI program at Microsoft:

- A governance structure and a system of governance to enable progress and accountability.
- Rules that standardize our responsible AI requirements, as well as goals as an opportunity to foster innovation and build better AI systems.
- Training and practices to help our FTEs act on our principles and think deeply about the sociotechnical impacts of our AI systems.
- And a set of tools, patterns and practices that help secure DevOps teams, Data Scientists and all other employees contributing to the implementation of AI systems in the implementation and integration of Responsible AI requirements into their everyday development practices.

We now discuss these building blocks that build upon the aforementioned principles in further details.

Figure 4. Microsoft's Building blocks for putting Responsible AI into practice

Practices

For us, as illustrated in the above figure, putting responsible AI into action starts with **practices that are grounded in human-centric design that spans roles**.

We've taken over 20 years of research and have applied it in the development of AI guidelines and standards that are meant to help others anticipate and address potential issues throughout the software development lifecycle and develop AI systems in a more responsible manner. Some of our guidelines include Guidelines for Human-AI Interaction, now as part of the newly released Human-AI eXperience (HAX) Toolkit (See New toolkit aims to help teams create responsible human-AI experiences), Conversational AI Guidelines, Inclusive Design Guidelines, an AI Fairness Checklist, and a Datasheets for Datasets.

These learnings helped inform new practices at Microsoft. For example, we developed Transparency Notes to help teams communicate the purposes, capabilities and limitations of an AI system so our customers can understand when and how to deploy our platform technologies. Transparency Notes fill the gap between marketing and technical documentation, proactively communicating information that our customers need to know to deploy AI responsibly. Our Face API Transparency Note was our first attempt at this new practice, and we now have a growing number of Transparency Notes being prepared across our platform offerings and already available outside - You can Bing to search and retrieve all of them -.

We also see synergies between our Transparency Notes and other industry efforts such as Model Cards, AI FactSheets, etc., and we're pleased to be playing an active role in the Partnership on AI' initiative to evolve the artifacts and processes for Responsible AI industry-wide.

Our AI Security Guidelines are another illustration of these practices. Established in collaboration with Harvard University (Berkman Klein Center for Internet and Society at Harvard University), they are a series of findings we share that can protect your AI systems with guidance materials for modeling, detecting, and mitigating security risks and ethics issues. See Failure Modes in Machine Learning.

Tools

In addition to the practices mentioned above, we've developed and continue to deploy a set of tools to help our engineering teams and others **understand, protect, and control their AI at every stage of innovation**. See section **Leveraging engineering systems and tools, patterns & practices** below.

Our tools are a result of collaboration across disciplines to strengthen and accelerate Responsible AI, spanning software engineering and development to social sciences, user research, law and policy. Theses tools range from a variety of asset types.

At the highest level, we've launched a series of research papers that set the context behind the responsible use of AI systems. To enable further collaboration, we also have open sourced many tools and datasets that others can use to contribute and build upon. We will cover some of them as part of the next 2 modules.

We've also democratized our Responsible AI tools through our managed services offered through Azure ML.

Governance framework

Critical to the Responsible AI discussion is the need for a governance framework that really starts from the very beginning in the business case and continues throughout the design, development, and ongoing management of AI. Our governing practices help to ensure and foster Responsible AI both within our company and beyond.

Our Responsible AI governance approach borrows from the "hub-and-spokes" model that has worked successfully to integrate security, privacy and accessibility into our products and services.

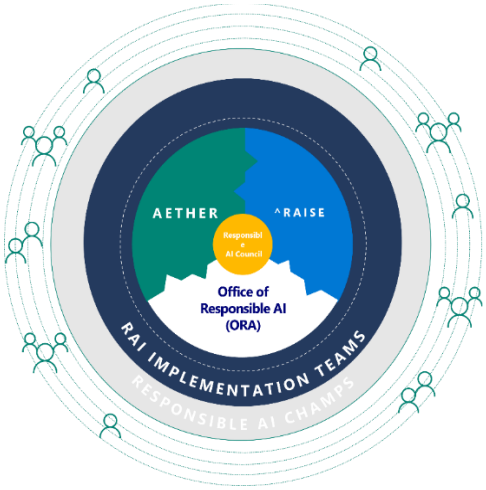


Figure 5. Microsoft's Responsible AI governance framework following the "hub and spokes" model

Our current model relies upon centralized and decentralized functions to put our Responsible AI principles into practice. This "hub-and-spokes" model provides the accountability and authority to drive initiatives while also enabling Responsible AI policies to be implemented at scale.

Our "hub" includes:

1. **The AETHER committee**, whose working groups leverage top scientific and engineering talent to provide subject-matter expertise on the state-of-the-art and emerging trends regarding the enactment of Microsoft's responsible AI principles.
2. **The Office of Responsible AI (ORA)**, which defines, enables, governs, and coordinates the company's approach to responsible AI. In other words, it sets our policies and governance processes, and as such, it enacts the already mentioned Responsible AI Standard.
3. And **the Responsible AI Strategy in Engineering (RAISE) group**, which enables our engineering groups to implement our responsible AI processes through systems and tools. This is both an engineering team and a strategic engineering initiative to enable the implementation of the above Standard across Microsoft's engineering groups: Cloud and AI, Experiences and Devices, Technology and Research, Gaming, and LinkedIn.

The three groups work together to set a consistent bar for Responsible AI across the company and they empower our "spokes" to drive initiatives and be accountable for them.

The spokes of our governance include **our Responsible AI Champs community**. The Champs are appointed by company leadership and sit in engineering and sales teams across the company. They raise awareness about Microsoft's approach to responsible AI and the tools and processes available, they spot issues and help teams assess ethical and societal considerations, and they cultivate a culture of responsible innovation in their teams.

A governance system should be agile to the changing nature of technology and the business. In that sense, our governance system continues to evolve to this day, as it should. As illustrated above, at the center of our governance framework "hub" is the Responsible AI Council that groups all the Corporate Vice Presidents (CVPs) of the Microsoft's engineering groups which are responsible for sponsoring this effort company-wide.

Introducing the (still internal) Responsible AI Standard

As stated above, Microsoft published internally the Responsible AI Standard to assist teams in implementing and deploying AI technologies in a responsible manner. Like other standards previously rolled-out and in application within the company for security, privacy, and accessibility, and by using our AI principles as a North star, we've created the (currently still internal) Microsoft Responsible AI standard.

This is a 3 parts-standard:

1. Starting with a Governance section, which covers roles and responsibilities of different parties involved in Responsible AI at Microsoft.
2. Continuing with a Requirements section where two sets of requirements are outlined:
    - Baseline requirements which apply to all projects.
    - Specific requirements which are tailored for specific projects where we've developed enough insight from sensitive uses cases and other policy efforts that we have defined additional and more detailed requirements for these specific use cases.

3. Ending with a final section devoted to Guidance, which aggregates insightful information on our principles and related goals to achieve with the above requirements that every FTE can align to for consistency across the company.

The Microsoft Responsible AI standard asks every FTE to:

- **Learn** about our Responsible AI assets to understand and raise awareness about the subject company wide.
- **Report sensitive uses** where each employee is invited to work with their teams to detect and assess sensitive use cases of AI and report these to the Office of Responsible AI for assistance.
- **Follow the Standard requirements** and recognize the occasions where particular domains or technologies might be impacted.
- **Ask for help** and reach out to their point of contact when they are uncertain or wish to report a sensitive use case. This is the role devoted to the RAI Champs.

About the Sensitive Uses

The Sensitive Uses triggers involve AI systems that may result in one of the following:

- **Denial of consequential services**. The scenario involves the use of AI in a way that may directly result in the denial of consequential services or support to an individual (e.g., financial, housing, insurance, education, recruiting, or healthcare services or support).
- **Risk of harm**. The scenario involves the use of AI in a way that may create a significant risk of physical or emotional harm to an individual (e.g., life-or-death decisions in military contexts, safety-critical manufacturing environments, clinical decision making in healthcare, or almost any case involving children or other vulnerable populations).
- **Infringement on human rights**. The scenario involves the use of AI in a way that may result in a significant infringement of one's human rights.

The above process of detecting sensitive use cases has helped us navigate the grey areas that are inevitably encountered and leads in some cases to new red lines. Outcomes of the process include giving up opportunities to build and deploy specific AI systems because we were not confident that we could do so in a way that upheld our principles.

In working through the complexities of several other use cases, we also came to appreciate the importance of three key learnings:

1. First, by digging into the details of use cases, we've been able to understand and articulate their different risk profiles, such as the impact on failure and misuse on stakeholders, and the readiness of the technology for the particular use case.
2. Second, we've learned the important role that benchmarking and operational testing play, helping to ensure that AI systems serve their stakeholders well and meet quality bars not just in labs, but also in the real world.
3. And third, we've learned how we need to communicate with our customers to empower them to deploy their systems responsibly.

Towards version two of the Responsible AI Standard

In the same spirit, we published the first version of the Responsible AI Standard with a will to learn more, and with a humble recognition that we were at the beginning of our effort to systematically move from principles to practices. Through a phased pilot across 10 engineering groups and two customer-facing teams, we learned

what worked and what did not. Our pilot teams appreciated the examples of how Responsible AI concerns can arise. They also struggled sometimes with the open-endedness of the considerations laid out in the Standard and expressed a desire for more concrete requirements and criteria. There was a thirst for more tools, templates, and systems, and for a closer integration with existing development practices.

Just over a year later, we're previewing version two of the Responsible AI Standard. We grounded this version two on all of these learnings and feedbacks. The revision reinforces a human-centered approach, building upon strong research and engineering foundations. It will mandate that teams building AI systems meet requirements that accrue to principle-specific goals. These goals help engage our engineering teams' problem-solving instincts and provide context for the requirements.

For each requirement in the Responsible AI Standard, we provide a set of implementation methods that teams can draw upon, including tools, patterns and practices crowdsourced from within and outside the company and refined through a maturity process. We expect this to be a cross-company, multi-year effort and one of the most critical elements for operationalizing Responsible AI across the company. We will continue to collect and integrate feedback as we move towards releasing this second version of the Standard and its global implementation.

What about a Responsible AI Lifecycle?

The set of needs in this field we're addressing within Microsoft and as part of the larger tech industry is quite unique:

- Identifying the potential benefits and harms to design a system responsibly requires attention to the specific context in which this system is deployed, rather than a "checklist" approach. There is no one, straightforward solution – we need to weigh the options of the best solution for a specific context.
- Many of the largest challenges come from intersections between two or more principles, for example preserving privacy while ensuring we know enough about sensitive attributes to evaluate and manage fairness or providing enough transparency about how systems function while making sure we're not opening security risks.
- Focus on identifying opportunities and challenges in context and managing tensions between benefits and harms, principles, and stakeholders – This is work that requires thoughtful collaboration and deliberation, often returning to the same discussion several times to develop a well-prioritized plan for addressing the challenges in a project.
- There is no perfect solution – the challenge is balancing the benefits of a system along with the potential harms and managing for future impact as well as we can, understanding that we are often just one part of a larger ecosystem accountable for the system.
- Recognize that issues can come up at any stage of the product development cycle and we need to keep looking for them, the upside is that there are mitigation techniques for different development stages from ML research to user experience (UX) design as part of the Human-AI (H-AI) experiences to provide.
- We can use these techniques to create new products and/or features, and to improve existing ones.

Thus, **in terms of an AI system's lifecycle, Responsible AI is not just about data science, it's about ALL disciplines working together and benefitting from each other**: program manager, data scientist, (AI) software engineer, (UX) designer, user researcher, content writer, marketing, and customer service.

Responsible AI is a human-centered approach to developing, designing, and deploying AI technologies and ML-powered systems (products or services) and features. It's a mindset and a toolset, and this applies both for new and existing services and products. The previously discussed building blocks help (secure DevOps) teams to

identify, evaluate, and mitigate possible harms in their technology stack and products throughout the entire product development lifecycle, internally called the **Responsible AI Lifecycle (RAIL)**.

To be able to leverage on that, Responsible AI practices need to fuel and be a part of every stage of the service or product AND ML models design process. **All the learnings that can be gathered from these stages constitute several opportunities to foresee how the considered AI systems should evolve to provide even better, more responsible solutions.**

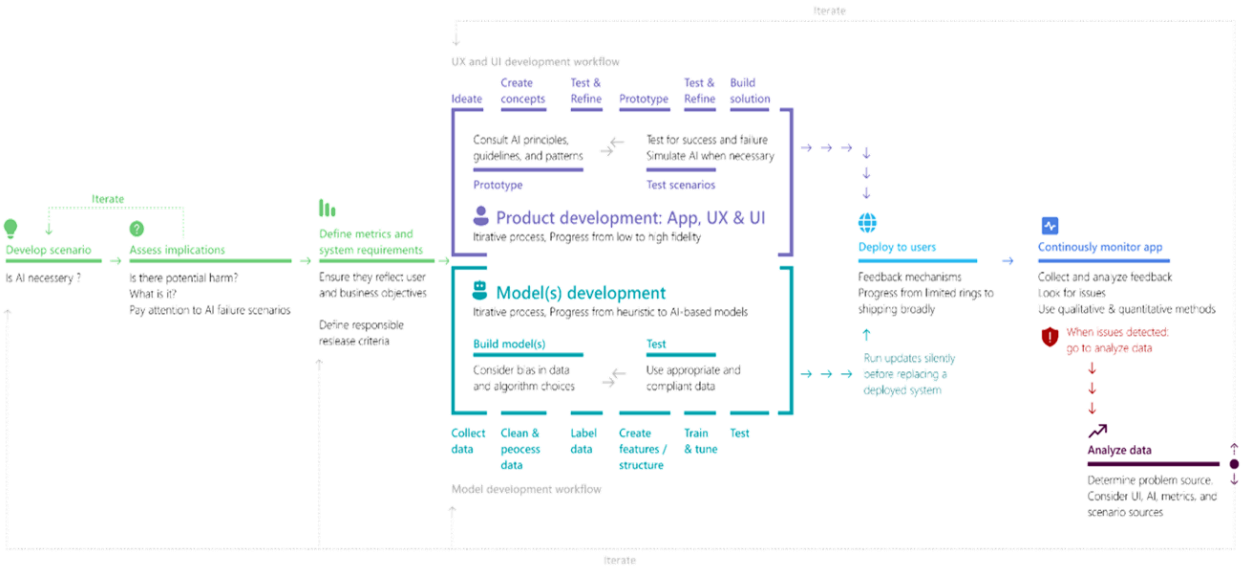


Figure 6. An overview of the Responsible AI Lifecycle (RAIL)

Putting our principles to work means operationalizing these stages with as a set of activities to conduct at each stage, the same way we've done with cybersecurity, privacy, and accessibility which are, as already abovementioned, systemically integrated into our services and products design, development and deployment practices. We endeavor to implement the same approach for our RAI principles, in the form of governance systems that shape the design, the development, the deployment, and the monitoring of these systems. We explore activities to be conducted at each stage of an AI system's lifecycle below.

ENVISION

- Understand end-users, use cases and scenarios of using AI.
- Impact assessment for feature or system
- Define responsible release criteria
- Create mitigation plan
- Training about how to do it
- Engage data scientists and engineers in evaluating tools
- The expectation is that the learnings from this stage will be used in the feature/system and ML development lifecycle stages and operations (MLOps)

DEFINE/PROTOTYPE/BUILD - MODELS

- Develop or evolve ML model based on responsible release criteria
- Understand how the ML model will be used
- Review often with feature/system vTeam (including PM, Dev, Test, UX Design)
- Apply fairness tools as appropriate to project/sprint
- Test for RAI issues and work on mitigating them repeatedly

DEFINE/PROTOTYPE/BUILD - APP

- Develop the user experience with Human-AI design guidelines & the Human-AI eXperience (HAX) toolkit
- Create RAI-related work items based on mitigation plan and H-AI guidelines
- User research applied to uncover how successful is the implementation
- Develop feedback loops
- Review often with data scientists
- Add objectives and key results (OKRs) around RAI

LAUNCH / EVOLVE

- Monitor ML-powered feature/system based on
  - Responsible release criteria
  - Design criteria
  - User reaction
- Keep evolving and improving ML model and H-AI experience

Figure 7. Examples of activities at each stage of the Responsible AI Lifecycle (RAIL)

Envisioning an AI system

Today, we understand that it is critically important for our employees to think holistically about the AI systems we choose to build. As part of this, we all need to think deeply about and account for sociotechnical impacts of these AI systems. That's why we've developed trainings designed to help our teams develop the muscle of asking ground-zero questions, such as, "*Why are we building this AI system?*" and, "*Is the AI technology at the core of this system mature enough?*".

And these are exactly the types of questions you would ask during the first stage of the RAIL with **impact assessment**. Conducting such an assessment is a required first step in the development of any AI system at Microsoft. It includes identifying high priority areas within your AI development, building a way to track and review the process, and securing approvals.

For that purpose, teams must complete an extensive questionnaire, which considers the intended use cases of a product and its potential impacts on stakeholders, and a self-assessment of the potential risks. This process is facilitated and required by ORA and serves as an important tool to help ensure the responsible design, development, deployment, and monitoring of AI system.

This assessment is made easy with the use of the following resources:

- **The** Harms Modeling. A framework for product teams, grounded from our core pillars of responsible innovation. It examines how people's lives can be negatively impacted by technology: injuries, denial of consequential services, infringement on human rights, and erosion of democratic & societal structures. Similar to the Security Threat Modeling, a foundational activity as part of Microsoft SDL, Harms Modeling enables product teams to anticipate potential real-world impacts of technology, which is a cornerstone of responsible development.

  Hereafter is an example of the outcome of a qualitative assessment. Such an outcome is used to inform prioritization of responsible innovation mitigations depending on the encountered types of harm.

Figure 8. Harms Modeling qualitative assessment

- **The** Judgment Call Game. An award-winning responsible innovation game and team-based activity that puts Microsoft's AI principles into action. The game provides an easy-to-use method for cultivating stakeholder empathy by making participants write product reviews from the perspective of a particular stakeholder, describing what kind of impact and harms the technology could produce from their point of view. To learn more about this game, you can download the printable Judgment Call game kit.

- **The Community Jury**. A technique that brings together diverse stakeholders impacted by a technology. It is an adaptation of the citizen jury. The stakeholders are provided with an opportunity to learn from experts about a project, deliberate together, and give feedback on use cases and product design. This responsible innovation technique allows project teams to collaborate with researchers to identify stakeholder values and to understand the perceptions and concerns of impacted stakeholders. Wherever the topic to discuss is related to personal data privacy, the composition of the community jury should be balanced to include individuals with different privacy indices.

See Responsible Innovation: A Best Practices Toolkit.

Defining, prototyping, building an AI system

This is the second stage of building Responsible AI systems, i.e., human-centered systems, with all the activities highlighted in Figure 7 in turquoise blue for the ML model AND in purple for the system or the feature itself, respectively.

While, as far as the former is concerned, the related activities range from data collection and handling to ensuring fairness in performance and transparency of the model(s), building a Responsible AI system is also about building it in a human-centered way.

So, besides and beyond the ML model(s), the latter implies to:

- Tie all technical decisions back to user needs,
- Involve diverse perspectives early in the design of the AI system (see above section) and throughout,
- Plan for failures so users can recover (or take control) when things go wrong or an unexpected situation is encountered,
- Etc.

With that said, it is worth mentioning that this second stage that covers the design and the development of AI systems can be conducted either with today's agile practices - with several iterations and sprints - or not.

Launching, evolving an AI system

Then comes the third stage which is about the responsible deployment of AI systems. This stage includes reinforcing practices and empowering people to use AI responsibly through documentation, gating, scenario attestation, and more.

As we have been rolling out our Responsible AI program across the company, the existence of engineering systems and tools to help deliver on our Responsible AI commitments has been a priority for our teams.

Privacy, and the General Data Protection Regulation (GDPR) experience in particular, has taught us the importance of engineered systems and tools for enacting a new initiative at scale and ensuring that key considerations are baked in by design.

This leads us to the next section.

Leveraging engineering systems and tools, patterns & practices

Although tooling – particularly in its most technical sense – is not capable of the deep, human-centered thinking work that needs to be undertaken when conceiving AI systems, we think it is important to develop repeatable tools, patterns and practices where possible so the creative thought of our engineering teams can be directed toward the most novel and unique challenges, not reinventing the wheel. Integrated systems and tools also help drive consistency and ensure that Responsible AI is part of our engineering teams everyday work.

In recognition of this need, we are embarking on an initiative to build out a "paved road" for Responsible AI at Microsoft with a set of tools, and patterns & practices that help teams easily integrate responsible AI requirements into their everyday development practices. As outlined, Azure ML serves as the foundation for this paved road, leveraging the early integrations of our open-source tools.

We started our journey "paving that road" by the need to assess current processes and figure out what we wanted to change.

As you see in the figure hereafter, tooling is needed for the different stages of the RAIL to sustain the implementation of the requirements for each stage. Consequently, we have and continue to define:

1. *What needs to happen at each stage in terms of implementation (directions)?*
2. *Which tools are available, needed?* Etc.

Figure 9. Examples of activities at each stage of the Responsible AI Lifecycle (RAIL)

These resources can broadly be put into two categories:

1. Guidelines, patterns & practices, which also include trainings, workshops and assessment tools and games among others.
2. Technical tools, which are vehicles to (help) understand, assess and mitigate the AI risks inherent of ML models and the datasets to train them and/or used for inference, and other AI issues which might arise when implementing AI systems along with their ML model(s).

Regarding the former, and beyond the abovementioned resources to help in the initial (impact) assessment, some resources already provide a great, awaited, and welcome contribution for creating (more) responsible Human-AI partnership (with more to come). This is notably the case for the already mentioned Human-AI eXperience (HAX) Toolkit, which is a set of practical tools aiming to help teams strategically create and responsibly implement best practices when implementing AI systems that interact with people.

This toolkit currently consists of four components designed to assist teams throughout the user design process from planning to testing:

1. **The Guidelines for Human-AI Interaction**. A set of 18 generally applicable best practices for designing human-interaction with AI-based products and features initially published in a 2019 CHI paper.

Figure 10. Guidelines for Human-AI Interaction

2. **The** HAX Workbook. A tool to guide teams through planning the time and resources needed to address high-priority items and implementing human-AI interaction best practices.
3. **The** HAX Design Patterns. A set of flexible and reusable solutions to recurring and common Human-AI interaction problems that come up when designing Human-AI systems. The HAX Design Library is a searchable database of the design patterns and implementation examples.
4. **The** HAX Playbook. A tool for generating scenarios to test based on likely Human-AI interaction failures and thus for helping teams identifying and planning for unforeseen errors, such as a transcription error or false positive.

As far as the latter is concerned, and as previously outlined, these tools can be categorized into three broad categories each relating to our six principles:

1. Tools to **understand** the behavior of AI systems. These are used to make AI systems more fair, transparent, and inclusive.
2. Tools to **protect** AI systems data. These are used to make AI systems more secure and privacy-preserving.
3. Tools to establish **control** and governance throughout AI systems development cycle. These are used to make AI systems more reliable and allows people who design and deploy AI systems to be held accountable for how their systems operate.

Module 2 will further explore the first category of tools used to understand the behavior of AI systems while **Module 3** will tackle the tools to protect AI systems data.

Although not examined here for the sake of the length of this guide, you can learn more about governance and control tools by referring to the **Control** section in Microsoft Responsible AI Resources center.

Going forward

As you've seen, we endeavor to put our Responsible AI principles into practice with governance systems that shape the design, the development, the deployment, and the monitoring of AI systems throughout our so-called Responsible AI Development Lifecycle (RAIL).

And as part of that, there's an important point we'd like to stress: This is just the beginning. AI is still a relatively new field, so it should come as no surprise that the processes around it are evolving rapidly as well. Going forward, we plan on refining our governance policies as we invest further in AI, and we advise other businesses do the same.

As we look ahead, first, we'll focus on consistently and systematically enacting our principles through the continued rollout of our (still evolving) Responsible AI Standard and the related governance framework now in place. We are acutely aware that, as the adoption of AI technologies accelerates, new and complex ethical challenges will arise.

While we recognize that we don't have all the answers, the above building blocks of our approach to responsible AI at Microsoft are designed to help us stay ahead of these challenges and enact a deliberate and principled approach. We will continue to share what we learn, and we welcome opportunities to learn with others.

Second, new research and policies, as well as regulation like the freshly announced European AI regulation, will help inform business decisions about what not to do – the scenarios we decline to support because of sensitive use cases and potential for unintended consequences – a framework to guide Yes/No decisions is of no use if the answer is always 'Yes' - you have to say 'No' - Most of the time it will not be Yes/No, but instead, a compromise and an outline of how we reframe the technology to ensure we are responsible and in line with our principles.

Third, there will be increasingly mature toolsets for data scientists, AI engineers, and other AI practitioners – beyond the ones we will discuss in the next modules to help you mitigate these AI risks. These tools will be made available to customers, not only as far as we are concerned, but also as an industry effort as notably outcomes from the Partnership on AI we co-founded.

Ultimately, we will continue to build a culture of Responsible AI across the company. We cannot stress enough that we are just at the beginning of this journey. *What about you?*

Initiating your own Responsible AI journey

"I would argue that perhaps the most productive debate we can have isn't one of good versus evil: The debate should be about the values instilled in the people and institutions creating this technology."

- Satya Nadella

While we recognize that every individual, organization, and region will have their own beliefs and standards that should be reflected in their own AI journey, we hope the previous section provides you with a set of ideas to initiate your own journey and create a holistic approach to Responsible AI if you haven't already done so.

To help you proactively establish our principles, standards, practices, and guardrails for your own AI systems, and thus both anticipate and mitigate their AI risks, and maximize their benefits, you can:

→ Visit our AI Business School page to help you establish and/or adapt a strategy which is suitable for you.

→ Take the Microsoft Learn training freely available online:

- Identify principles and practices for responsible AI (for private organizations).

-or-

- Identify guiding principles for responsible AI in government (for public ones).

This concludes this introduction of Responsible AI, the Microsoft's Responsible AI principles and how they translate into practices company-wide through a related governance framework and a forthcoming standard, and how you can build your own journey.

The rest of this guide, and more particularly the next two modules are dedicated to an investigation of each of these tools, but we deemed it useful in this first module to put these efforts in context because tooling is merely here to enable the practices and sustain the requirements provided in the standards that emerge from a sound governance framework and relies on the other building blocks building blocks as previously depicted to achieve its full potential.

Now, it is high time to get our hands a little bit dirty with Responsible AI tools starting with tools to understand the behavior of ML algorithms.

Module 2: Better understanding your data and the behavior of ML algorithms

The goal of this module is to explore some of the Responsible AI tools we provide for a better understanding of the behavior of ML models, and more particularly Fairlearn, InterpretML and Error Analysis.

But before we investigate each individual tool separately, it is important to put them in context with respect to the RAIL lifecycle depicted in Figure 6 above and to comprehend how these tools can work together to achieve a better understanding of ML algorithms, as opposed to each being used independently.

All three tools we explore in this module fall under the Define/Prototype/Build phase of the RAIL lifecycle (see section **What about a Responsible AI Lifecycle?** above) and act directly on ML models and data they are fed to achieve two main goals:

1. **Model fairness** by ensuring ML algorithms avoid treating similarly situated groups of people in different ways because of sensitive attributes such as race, gender, age, or disability status. Fairness issues are generally due to bias already existing in training data, which is then enhanced by the models, but it might be the case that models induce their own bias into the system.

   For instance, a system used in the hiring process of a company which is fed hiring decisions made by a manager as labels tends to replicate or enhance the bias existing in the manager's decisions when selecting applicants rather than taking full consideration of the capability of a job applicant (most of time this capability is unobserved for people who are rejected). This can lead to a candidate being unrightfully discarded, which is obviously a big fairness problem and shows the extent to which a tool for assessing and mitigating this unfairness is needed.

2. **Model transparency** by leveraging interpretability techniques to understand models' global trends, predictions for selected cohorts of the data as well as explaining individual predictions. Model transparency is indeed a subject of utmost importance because executives and stakeholders need to be able to grasp the value and accuracy of data scientists' results, hence data scientists must be able to explain their models to them. Moreover, some of these predictions can be life changing like credit risk modeling or hiring applicants for a job, and decision makers owe it for the subjects to explain their decision process, which is something a blackbox model can't do.

   This is the reason why there are multiple instances of sensitive settings where people call for black-box models to be barred from making predictions because of their lack of interpretability. One of the most recent such developments involves an ML model used for diagnosing COVID-19 from chest X-rays. Researchers at the University of Washington in Seattle found that the model used relies on confounding factors rather than medical pathology to diagnose COVID-19, creating a delicate situation that could have been avoided had the model been explainable.

But beyond being used independently, the three tools explored in this section are part of an overall scheme illustrated in Figure 11 below, which is comprised of three steps : i) identifying issues for example in terms of model performance, fairness metrics etc., ii) diagnosing the problem by pinpointing what exactly is wrong and finally iii) mitigating the problem with appropriate mitigation techniques.

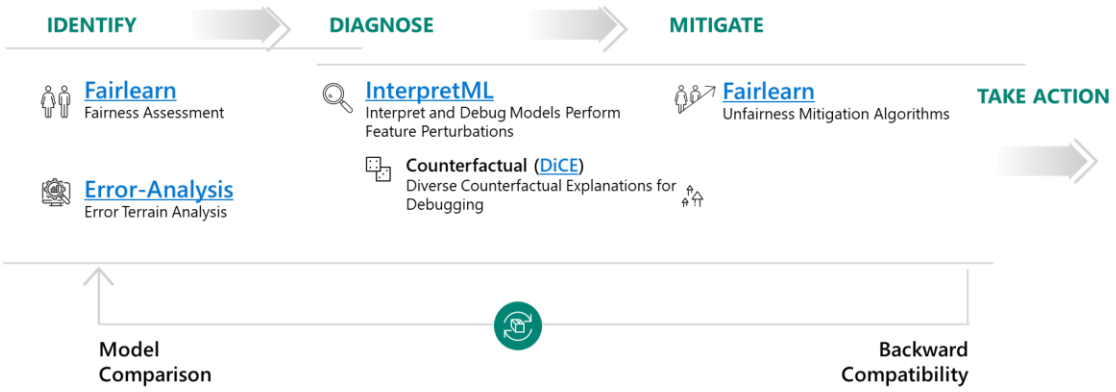


Figure 11. The 4 phased scheme for better understanding of ML models and the associated tools for each phase

Please note that the scheme presented above is by no means perfect and we are engaged in an iterative process of refining the tools. For example, dashboards previously made accessible through each tool independently (e.g., Fairness dashboards, interpretability dashboards and Error Analysis dashboards) are now being grouped together under one roof: the Responsible-AI-Widgets library which will be introduced at the end of this module as part of the Error Analysis section. This novelty is driven by the fact that these three tools are generally used together as we explained in the previous paragraph, so having all the dashboards in the same place comes in very handy.

Now that we have a good overview of the tooling for better understanding of ML models behavior and how these tools can be combined, we are ready to start investigating each of the tools we chose to focus on individually.

Fairlearn

As suggested above, the first challenge encountered in understanding AI systems today is the inability to assess and mitigate unfairness in the ML models. Because, yes, models can be unfair, and this unfairness can come either from the training data which includes some bias or models inducing their own bias into the system.

To address this challenge, we've recently open-sourced a toolkit called Fairlearn. Fairlearn is an open-source Python package that empowers AI practitioners of AI systems to **assess** their algorithms fairness and **mitigate** any observed unfairness issues. It provides state-of-the-art fairness metrics to evaluate your model's fairness along with algorithms for mitigating those fairness issues.

Using Fairlearn, developers and data scientists can leverage specialized algorithms to ensure fairer outcomes for everyone. While Fairlearn can be accessed through the built-in visualizations in Azure ML, the Fairlearn toolkit contains fairness metrics, mitigation algorithms as well as a Jupyter widget for model assessment.

Fairlearn focuses on models' negative impacts on groups of people, such as those defined in terms of race, gender, age, or disability status. For example, a voice recognition system might fail to work as well for women as it does for men, or a system for screening loan or job applications might be much better at picking good candidates among white men than among other groups. The goal of Fairlearn is to detect and help mitigate such biases.

Fairlearn components

There are two components to Fairlearn. The first is metrics and an assessment dashboard for assessing which groups are negatively impacted. The second is a set of algorithms for mitigating fairness issues in a variety of AI tasks and along a variety of fairness definitions!

1. **Assessing unfairness**. For assessing unfairness, the Fairlearn package provides an **interactive dashboard** for evaluating the overall performance of an existing model, and any disparities in the model evaluation metrics such as a disparity in model performance (e.g., accuracy rate, error rate, precision, recall, etc.) or a disparity in selection rate (e.g., loan approval rate) across different groups (e.g., different genders). This enables users to easily detect if there is unfairness against any groups in the existing model, regardless of whether the sensitive attributes have been included during the model training or not.
2. **Mitigating unfairness**. It's perhaps worth calling out that simply removing known sensitive attributes from the model training dataset usually cannot effectively eliminate unfairness in the resulting model, as there are often other features correlated with the removed attribute in the training dataset that would result in unfairness in the model anyway.

   So simply removing sensitive attributes like gender and race does not work as these can usually be induced from other correlated features. Thus, for mitigating fairness-related harms, the Fairlearn package provides more clever implementations of mitigation methods such as Threshold Optimization and the "Reductions" approach as depicted hereafter:

   - **The Threshold Optimization** is a post-processing technique that takes as input an existing classifier and the sensitive feature and derives a transformation of the classifier's prediction (e.g., adjusting the threshold for predicted probabilities) to enforce the specified parity constraints. In short, modifying the decision boundary achieving better results in terms of Fairness metrics (
   - **The "Reductions" approach** takes a standard ML estimator (e.g., a LightGBM model) as a black box and generates a set of retrained models using a sequence of reweighted training datasets. For example, applicants of a certain gender might be upweighted or down weighted to retrain models to reduce disparities across different gender groups. Users can then pick a model that provides the best trade-off between accuracy (or other performance metrics) and disparity, which is quantified by fairness metrics defined in the previous point.

   One key advantage of the "Reductions" approach is that it only requires access to the sensitive features during model training, not when the model is deployed for inferencing. This is because the sensitive features are only used for training data reweighting, and not need to be part of the input features of the actual model itself. This is extremely useful for many applications that do not have access to sensitive attributes for model prediction in production.

For additional information on Fairlearn, you can consider the following resources:

- Project landing page at https://FairLearn.org.
- GitHub repo at https://github.com/fairlearn/fairlearn.
- AI Show video Building fairer AI Systems with Fairlearn.
- White paper Fairlearn: A toolkit for assessing and improving fairness in AI*.
- And more specifically for the Azure ML integration, these two articles:
  - Concept: Machine learning fairness (preview).

- How-to: Use Azure Machine Learning with the Fairlearn open-source package to assess the fairness of ML models (preview)".

With that, let's now illustrate the use of this package with a first hands-on walkthrough.

Hands-on walkthrough: Fair binary classification of credit card default use-case

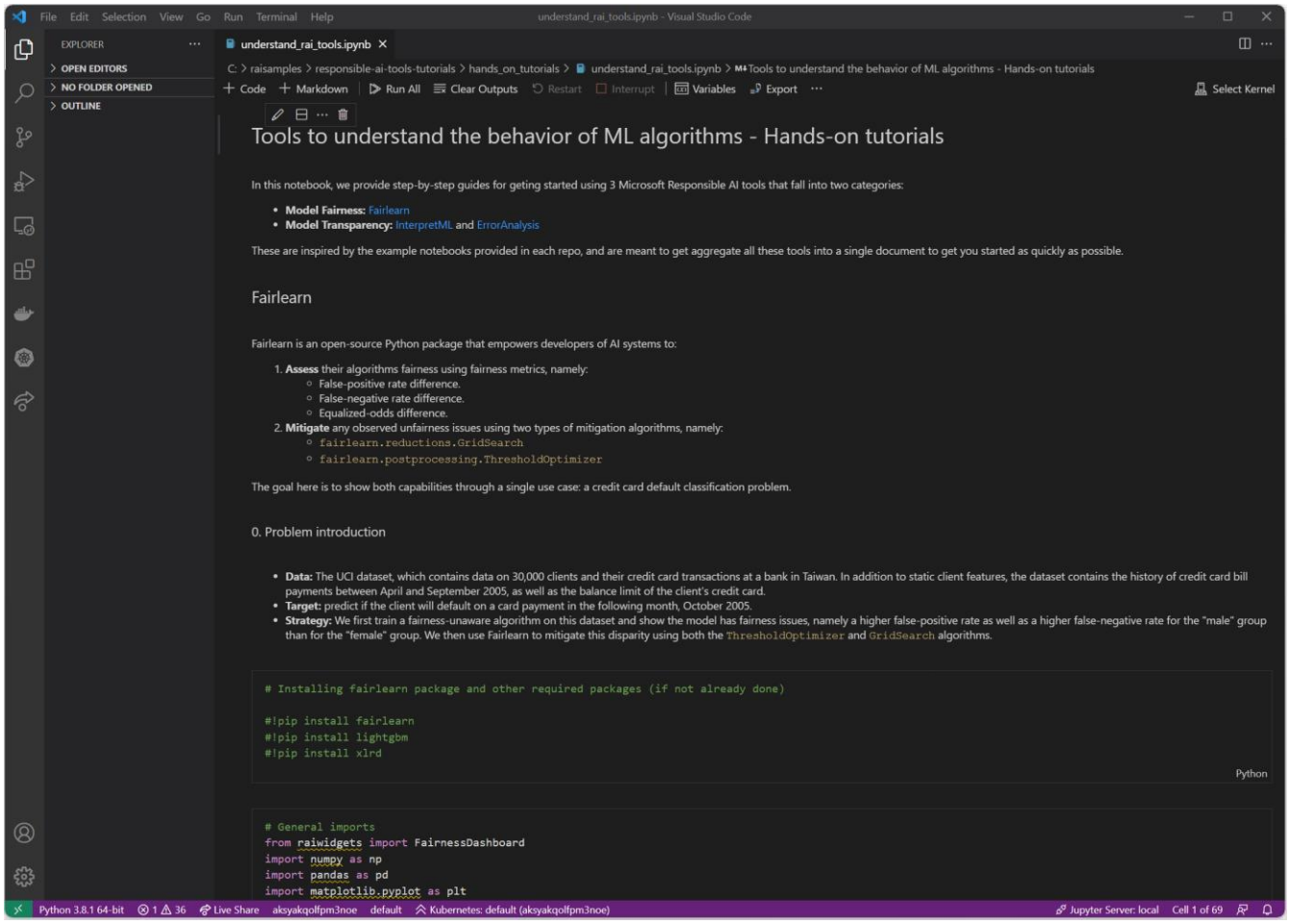
The Jupyter notebook for this hands-on walkthrough is `understand_rai-tools.ipynb` located in the `hands_on_tutorials` directory underneath the folder where you have cloned the samples' repo, for example the folder `raisamples` in our illustration, please refer to the section **Cloning the samples' Jupyter notebooks** above.

To follow this hands-on walkthrough, and the same goes for the other hands-on in this guide, you need to open up this file in the Jupyter environment of your choice, please refer to the section **Guide prerequisites** above.

The goal here is to train a fairness-unaware binary classification algorithm on a credit card default dataset and show the model has a higher false-positive rate as well as a higher false-negative rate for the "male" group than for the "female" group. We then use Fairlearn to mitigate this disparity using both the Threshold Optimization approach (ThresholdOptimizer) and the Reductions approach (GridSearch).

The notebook emulates the problem presented in a white paper that we developed in collaboration with EY. Due to data privacy, we obviously do not use the dataset from the white paper itself, but instead, we use the UCI Credit-card default dataset, a toy dataset reflecting credit-card defaults in Taiwan, as a substitute dataset to replicate the desired workflow.

Let's open up the notebook with the environment of your choice, for example VS Code as an illustration.



Installing the required libraries and data loading

The first step is of course to install the required packages, Fairlearn obviously but also the lightgbm package which is the model we use in this tutorial.

```
!pip install fairlearn
!pip install lightgbm
```

Then we can load the data and have a first look at it

```
# Load the data
data_url = "http://archive.ics.uci.edu/ml/machine-learning-
databases/00350/default%20of%20credit%20card%20clients.xls"
dataset = pd.read_excel(io=data_url, header=1).drop(columns=['ID']).rename(columns={'PAY_0':'PAY_1'})
dataset.head()
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
0	20000	2	2	1	24	2	2	-1	-1	-2	...	0	0	0	0	689	0	0	0	0	1
1	120000	2	2	2	26	-1	2	0	0	0	...	3272	3455	3261	0	1000	1000	1000	0	2000	1
2	90000	2	2	2	34	0	0	0	0	0	...	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
3	50000	2	2	1	37	0	0	0	0	0	...	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
4	50000	1	2	1	57	-1	0	-1	0	0	...	20940	19146	19131	2000	36681	10000	9000	689	679	0

The data consists of static client features, the dataset contains the history of credit card bill payments between April and September 2005, as well as the balance limit of the client's credit card. The target is to predict if a client will default on a card payment in the following month, October 2005.

We then perform some preprocessing where we introduce a bias synthetic feature to create some artificial unfairness for our purposes, this is well detailed in the notebook, but we chose to omit that here.

It is also worth noting that we also completely get rid of the 'sex' feature, but this doesn't help us in any way solve our unfairness issues as you'll see below.

Assessing unfairness

We first fit a fairness unaware LightGBM model to our data:

```
lgb_params = {
    'objective' : 'binary',
    'metric' : 'auc',
    'learning_rate': 0.03,
    'num_leaves' : 10,
    'max_depth' : 3
}
model = lgb.LGBMClassifier(**lgb_params)
model.fit(df_train, Y_train)
# Scores on test set
test_scores = model.predict_proba(df_test)[:, 1]
# Predictions (0 or 1) on test set
test_preds = (test_scores >= np.mean(Y_train)) * 1
```

Now we can use Fairlearn dashboard from ResponsibleAI widgets to show some fairness metrics:

```
mf = MetricFrame({
    'FPR': false_positive_rate,
    'FNR': false_negative_rate},
    Y_test, test_preds, sensitive_features=A_str_test)

mf.by_group
```

This prints the following metrics:

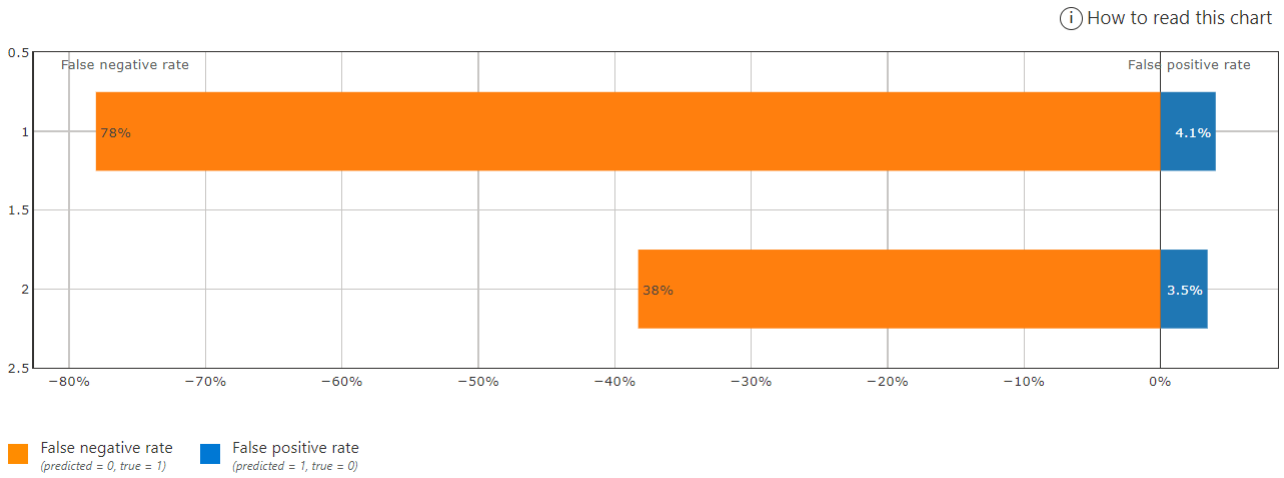


Figure 12. Fairness dashboard showing both False positive rate (FPR) and False negative rate (FNR) for the "male" group (1) and the "female" group (2)

We clearly see that the "male" group suffers from much higher FNR but also FPR than the female group, so even though the overall model performance is quite good at 85% balanced accuracy, males still suffer from much lower accuracy than males, and there is clearly an unfairness problem detected by Fairlearn.

Threshold Optimization approach: Mitigating equalized odds difference with postprocessing

We attempt to mitigate the disparities in the LightGBM predictions using the Fairlearn postprocessing algorithm ThresholdOptimizer. This algorithm finds a suitable threshold for the scores (class probabilities) produced by the Lightgbm model by optimizing the accuracy rate under the constraint that the equalized odds difference (on training data) is zero.

```
postprocess_est = ThresholdOptimizer(
    estimator=model,
    constraints="equalized_odds",
    prefit=True)
```

Since our goal is to optimize balanced accuracy, we resample the training data to have the same number of positive and negative examples.

```
# Balanced data set is obtained by sampling the same number of points from the majority class (Y=0)
# as there are points in the minority class (Y=1)
balanced_idx1 = df_train[Y_train==1].index
pp_train_idx = balanced_idx1.union(Y_train[Y_train==0].sample(n=balanced_idx1.size,
random_state=1234).index)
df_train_balanced = df_train.loc[pp_train_idx, :]
Y_train_balanced = Y_train.loc[pp_train_idx]
A_train_balanced = A_train.loc[pp_train_idx]
```

Now we can fit our ThresholdOptimizer and get the predictions. We only show the results at the end of next section.

```
postprocess_est.fit(df_train_balanced, Y_train_balanced, sensitive_features=A_train_balanced)
postprocess_preds = postprocess_est.predict(df_test, sensitive_features=A_test)
```

"Reductions" approach: Mitigating equalized odds difference with GridSearch

We now attempt to mitigate disparities using the GridSearch algorithm.

```
# Train GridSearch
sweep = GridSearch(model,
                   constraints=EqualizedOdds(),
                   grid_size=50,
                   grid_limit=3)

sweep.fit(df_train_balanced, Y_train_balanced, sensitive_features=A_train_balanced)
sweep_scores = [predictor.predict_proba(df_test)[:, 1] for predictor in sweep.predictors_]
sweep_preds = [predictor.predict(df_test) for predictor in sweep.predictors_]
```

The Figure 13 below shows the results of the mitigation techniques we used.

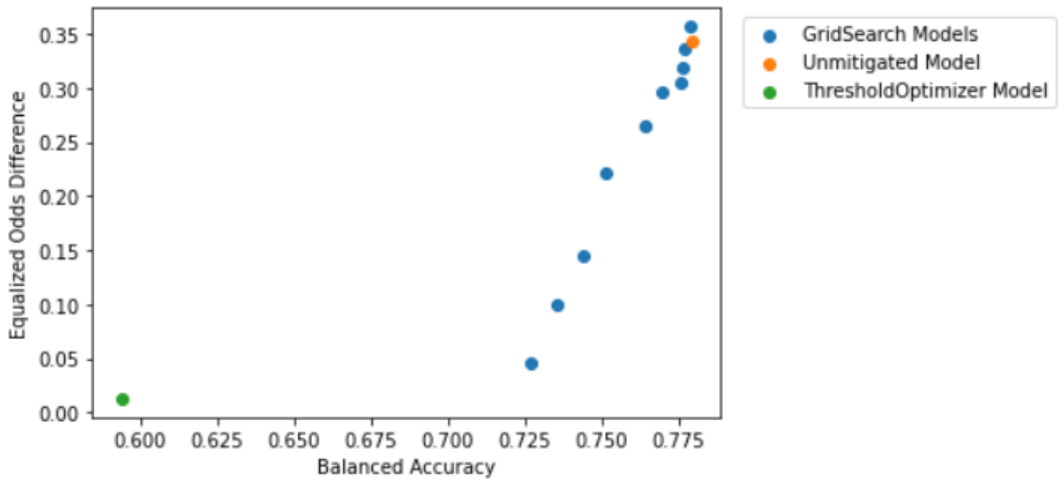


Figure 13. Accuracy vs fairness for Unmitigated and mitigated models

The overall performance measure we consider is the area under ROC curve (AUC), which is suited to classification problems with a large imbalance between positive and negative examples. For binary classifiers, this is the same as balanced accuracy.

As the fairness metric we use equalized odds difference, which quantifies the disparity in accuracy experienced by different demographics. Our goal is to assure that neither of the two groups ("male" vs. "female") has substantially larger false-positive rates or false-negative rates than the other group. The equalized odds difference is equal to the larger of the following two numbers:

1. The difference between false-positive rates of the two groups
2. The difference between false-negative rates of the two groups.

The closer to zero the Equalized odds difference is, the better.

We see that ThresholdOptimizer greatly reduced the disparity in performance across multiple fairness metrics. However, the overall accuracy for the ThresholdOptimizer model were much worse than the fairness-unaware model. With the GridSearch algorithm, we have a better trade-off by mitigating unfairness with very low Equalized odds difference while maintaining the same order of performance as the unmitigated initial model.

In conclusion, the data scientist should choose and deploy the model that balances the performance-fairness trade-off in a way that meets the needs of the business.

InterpretML

Another important aspect of understanding a ML model is the ability to interpret, or explain, its results. Interpretability is essential for:

- Helping detect fairness issues: *Does my model discriminate?*
- ML model debugging: *Why did my model make this mistake?*
- Feature engineering: *How can I improve my model?*
- Cooperation in Human-AI experience: *How can I understand and trust the model's decisions?*
- Regulatory compliance: *Does my model satisfy legal requirements?*
- High-risks AI systems in regulated industries and elsewhere.

So, to make it short, interpretability is needed to ensure there is optimal transparency ML within models to assess, and reason through the predictions it generates or the recommendations it creates. This is where InterpretML, an open-source Python package (the *interpret* package), comes into play.

As its name suggests, it helps AI practitioners interpret models and predictions made by these models, and thus understand which features contribute to their ML model's predictions. As such, it helps them understand their model's global behavior and/or the reasons behind individual predictions on local basis.

For that purpose, InterpretML incorporates state-of-the-art **ML interpretability techniques** under one roof, with a unified API and a built-in visualization platform.

Supported types of interpretabilities

InterpretML exposes two types of interpretabilities:

1. **Blackbox explainability**, which consists of techniques for explaining existing ML models that are well known among data scientists and AI practitioners more broadly, be them global explainability techniques like Partial Dependency Plots or local techniques like LIME or SHAP.
2. **Glassbox interpretability**, which are ML models designed for interpretability (for example linear models, rule lists, generalized additive models).

As far as the former is concerned, we won't get into explaining each of these techniques as this is thoroughly done elsewhere, but we will rather focus on the innovation of the InterpretML package, which is aggregating the results of all these techniques under a unified dashboard as we will show below.

Regarding the latter, InterpretML includes the first implementation of the Explainable Boosting Machine (EBM), a powerful, interpretable, **glassbox model**, designed to have accuracy comparable to state-of-the-art machine learning methods like Random Forest and Boosted Trees, while being highly intelligible and explainable.

This makes EBMs as accurate as state-of-the-art techniques like random forests and gradient boosted trees while still producing exact explanations and being editable by domain experts.

EBM is a generalized additive model (GAM) of the form:

$$g(E[y]) = \beta_0 + \sum f_j(x_j)$$

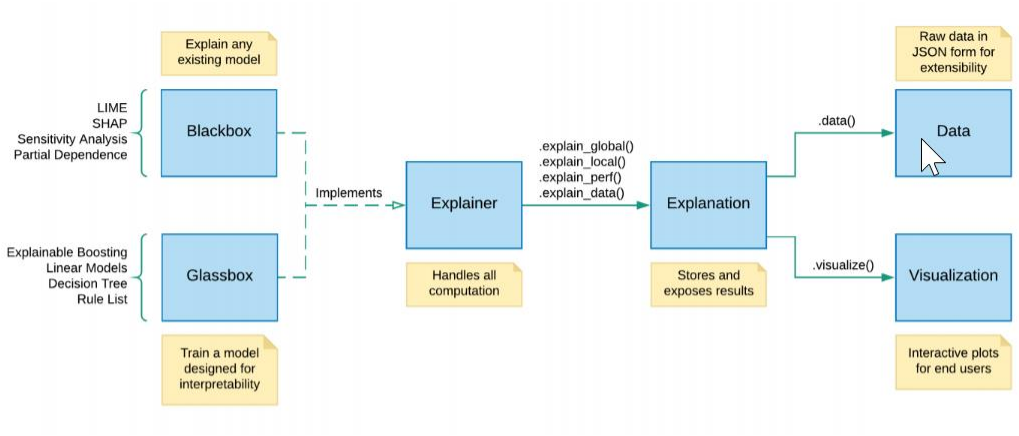
Here g is the link function that adapts the GAM to different settings such as regression or classification. Without going any deep into mathematical explanations, the goal behind showing you this formula is for you to understand that the GAM **learns a separate function f_j for each feature *j***, which means the contribution of each feature to the final prediction can be visualized and understood by plotting f*j* making GAMs highly intelligible.

The particularity of EBMs compared to traditional GAMs is that EBM **learns each feature function f_j using modern ML techniques** such as bagging and gradient boosting and round-robin cycles through features to mitigate the effects of co-linearity.

These two types will be further investigated below through a dedicated hands-on for each, see sections below:

1. **Hands-on walkthrough: Blackbox explainability**.
2. **Hands-on walkthrough: training a Glassbox model with Explainable Boosting Machine**.

The Figure 14 below provides a good overview of the interpretML package API for Blackbox models explainability and Glassbox models interpretability.



Glassbox

```
from interpret import show
from interpret.glassbox import LogisticRegression


clf = LogisticRegression()
clf.fit(X, y)

global_exp = clf.explain_global()
local_exp = clf.explain_local(X, y)

show([global_exp, local_exp])
```

Blackbox

```
from interpret import show
from interpret.blackbox import PartialDependence
from sklearn.neural_network import MLPClassifier

blackbox = MLPClassifier()
blackbox.fit(X, y)

pdp = PartialDependence(blackbox.predict_proba, X)
global_exp = pdp.explain_global()

show(global_exp)
```

Figure 14. InterpretML package API architecture and code examples

Like Fairlearn, InterpretML is also integrated into Azure ML[1].

With InterpretML and Fairlearn, you can gain a very good understanding of your models. Both are available integrated directly in Azure ML.

For additional information on InterpretML, you can consider the following resources:

- Project landing page at https://interpret.ml/.
- GitHub repo at https://github.com/interpretml.
- AI Show (video) How to Explain Models with IntepretML Deep Dive
- And more specifically for the Azure ML integration, these two articles:
    - Concept: Model interpretability in Azure Machine Learning (preview)
    - How-to: Use the interpretability package to explain ML models & predictions in Python (preview)

So, it' time to see InterpretML in actions with a hands-on walkthrough.

Hands-on walkthrough: Blackbox explainability

In this hands-on walkthrough, we will start with the section **InterpretML** of the notebook `understand_rai_tools.ipynb` used so far to illustrate Fairlearn. You can scroll down below the section **Explaining Blackbox Classifiers**.

The purpose here is to show how blackbox model explainability works using a simple binary classification example which consists of predicting income (more or less than 50k) using a Random Forest (RF) classifier.

First, make sure you install the *interpret* Python package using pip:

```
pip install interpret
```

Here is a peak at the dataset we use:

	Age	WorkClass	fnlwgt	Education	EducationNum	MaritalStatus	Occupation	Relationship	Race	Gender	CapitalGain	CapitalLoss	HoursPerWeek	NativeCountry	Income
9646	62	Self-emp-not-inc	26911	7th-8th	4	Widowed	Other-service	Not-in-family	White	Female	0	0	66	United-States	<=50K
709	18	Private	208103	11th	7	Never-married	Other-service	Other-relative	White	Male	0	0	25	United-States	<=50K
7385	25	Private	102476	Bachelors	13	Never-married	Farming-fishing	Own-child	White	Male	27828	0	50	United-States	>50K
16671	33	Private	511517	HS-grad	9	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	40	United-States	<=50K
21932	36	Private	292570	11th	7	Never-married	Machine-op-inspct	Unmarried	White	Female	0	0	40	United-States	<=50K

Figure 15. Subset of the dataset used for income prediction, the target column is "Income"

[1] We've recently updated Azure ML Studio's visualization dashboard with the revamped version 2.0 dashboard of InterpretML currently available in open source. Additionally, we modularized the dashboard so that users can call model performance, dataset explorer, and aggregate/individual feature importance and what-if tabs through separate API calls.

We first train a RF classifier after some dimensionality reduction with PCA.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline

#Blackbox system can include preprocessing, not just a classifier!
pca = PCA()
rf = RandomForestClassifier(n_estimators=100, n_jobs=-1)

blackbox_model = Pipeline([('pca', pca), ('rf', rf)])
blackbox_model.fit(X_train, y_train)
```

Then we use InterpretML to look at:

1. The Model performance.

```
from interpret import show
from interpret.perf import ROC

blackbox_perf = ROC(blackbox_model.predict_proba).explain_perf(X_test, y_test, name='Blackbox')
```

2. The local explanations with LIME and SHAP.

```
from interpret.blackbox import LimeTabular
from interpret import show

#Blackbox explainers need a predict function, and optionally a dataset
lime = LimeTabular(predict_fn=blackbox_model.predict_proba, data=X_train, random_state=1)

#Pick the instances to explain, optionally pass in labels if you have them
lime_local = lime.explain_local(X_test[:5], y_test[:5], name='LIME')


from interpret.blackbox import ShapKernel
import numpy as np

background_val = np.median(X_train, axis=0).reshape(1, -1)
shap = ShapKernel(predict_fn=blackbox_model.predict_proba, data=background_val,
feature_names=feature_names)
shap_local = shap.explain_local(X_test[:5], y_test[:5], name='SHAP')
```

3. The global explanations with Morris Sensitivity and Partial Dependence.

```
from interpret.blackbox import MorrisSensitivity

sensitivity = MorrisSensitivity(predict_fn=blackbox_model.predict_proba, data=X_train)
sensitivity_global = sensitivity.explain_global(name="Global Sensitivity")

from interpret.blackbox import PartialDependence

pdp = PartialDependence(predict_fn=blackbox_model.predict_proba, data=X_train)
pdp_global = pdp.explain_global(name='Partial Dependence')
```

And then we show them all in a single dashboard.

```
# Show them all in one dashboard
show([blackbox_perf, lime_local, shap_local, sensitivity_global, pdp_global])
```

This all results in a multi-tab interactive dashboard where we can select to visualize any of the explanations we defined above. We show for example the performance and Morris sensitivity tabs in figure 9 below, please refer to the associated notebook to explore the other tabs.

Figure 16. Performance and Morris sensitivity tabs from the interpret dashboard

Let's now consider the 'glassbox side' of InterpretML.

Hands-on walkthrough: training a Glassbox model with Explainable Boosting Machine

You can continue with the notebook `understand_rai_tools.ipynb`. You can scroll down below the section **Interpretable glassbox classification using EBM**.

For the sake of simplicity, we use the same dataset used in the previous blackbox explainability hands-on.

We fit an Explainable Boosting Machine (EBM) to the data:

```
from interpret.glassbox import ExplainableBoostingClassifier

ebm = ExplainableBoostingClassifier() ebm.fit(X_train, y_train)
# or substitute with LogisticRegression, DecisionTreeClassifier or any other glassbox model
```

Now we can show the global behavior of the model (how the score varies with the age here):

```
from interpret import show
ebm_global = ebm.explain_global()
show(ebm_global)
```

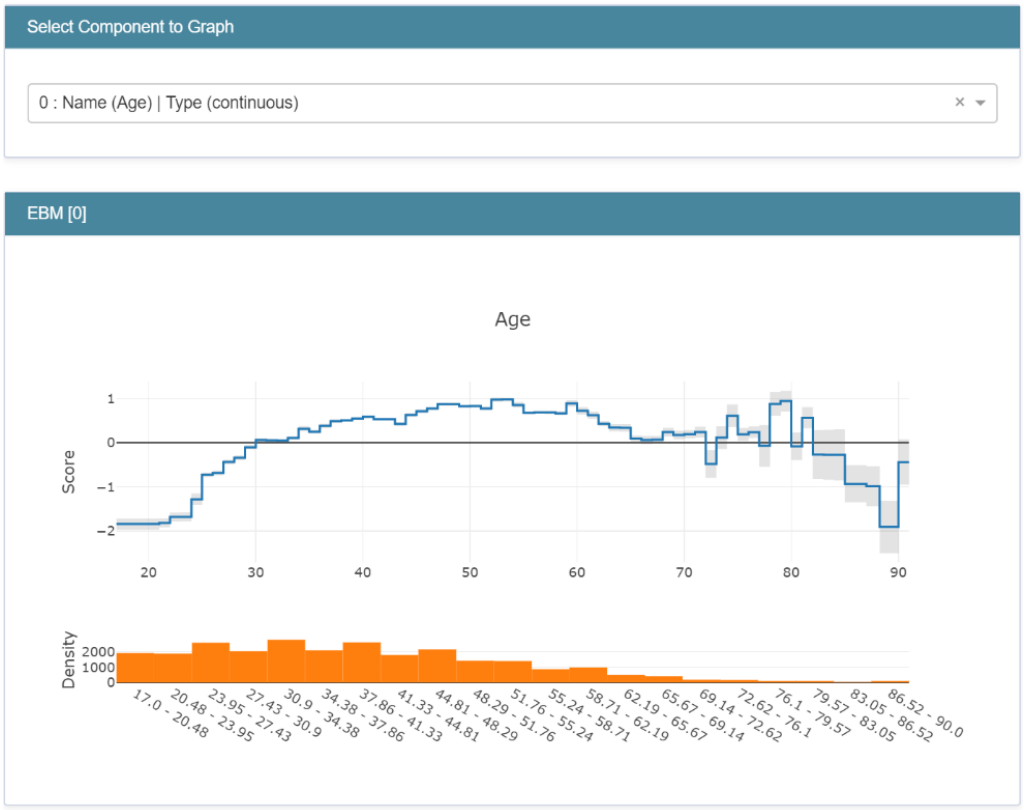


Figure 17. EBMs are interpretable by design, here we show how the score achieved by the model varies with respect to the age feature

And we can also explain individual predictions:

```
ebm_local = ebm.explain_local(X_test, y_test)
show(ebm_local)
```

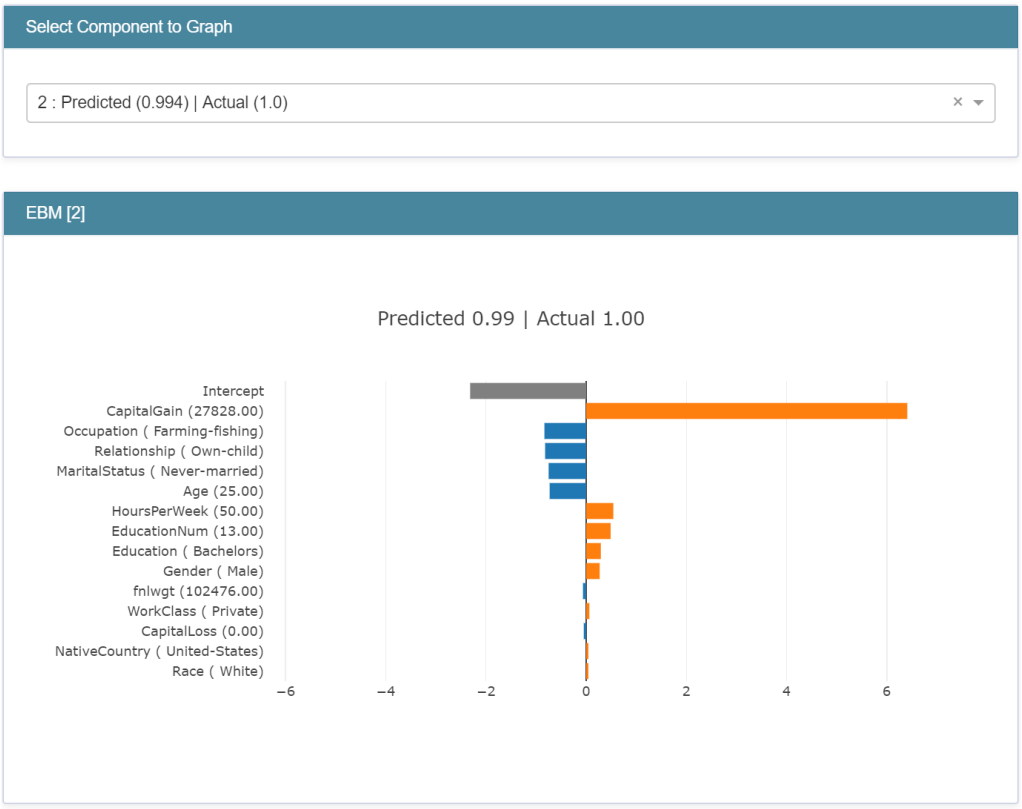


Figure 18. Built-in local explainability for EBMs

It is also worth mentioning that EBM overall performance is much better than other glassbox models like logistic regression or decision trees, as shown in the associated notebook, while still being interpretable by design.

Error-Analysis

Error-Analysis was initially created for our Analysis Platform, and its usage there has shown and validated this tool to be an effective and valuable approach for debugging models.

This internal toolkit has then been released in the open (in partnership with MSR). Error-Analysis is now part of the open-source Responsible-AI-Widgets package that enables you to get a deeper understanding of ML model errors. When evaluating a ML model, a single score like the aggregate accuracy is not sufficient to understand where the model went wrong and may hide important conditions of inaccuracies between cohorts of your data.

This is where Error Analysis comes into play, it allows you to:

- **Identify** cohorts with higher error rates versus the overall benchmark and visualize how the error rate is distributed.
- **Diagnose** the root causes behind these errors.

Let's cover these in order.

Identification of errors

Error Analysis identifies cohorts of data with higher error rate than the overall benchmark. These discrepancies might occur when the system or model underperforms for specific demographic groups or infrequently observed input conditions in the training data. Two different methods are proposed by the Error Analysis toolkit:

- **Decision Tree**, which allows you to discover cohorts with higher error rates across multiple features using an intuitive binary tree visualization. Investigating indicators such as error rate, error coverage, and data representation for each discovered cohort in the decision tree allows you to form hypothesis on what features induce the most failure of your model.

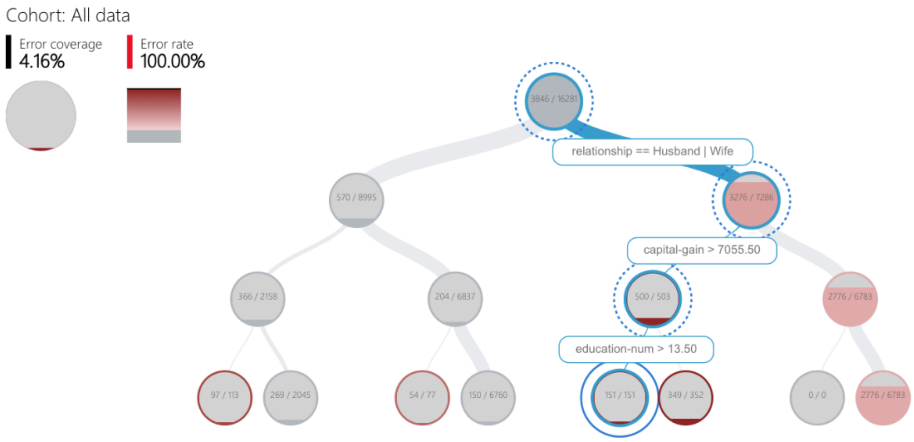


Figure 19. Decision tree showing error rates for each cohort

- **Error Heatmap**: once you form hypotheses of the most impactful features for failure, use the Error Heatmap to further investigate how one or two input features impact the error rate across cohorts.

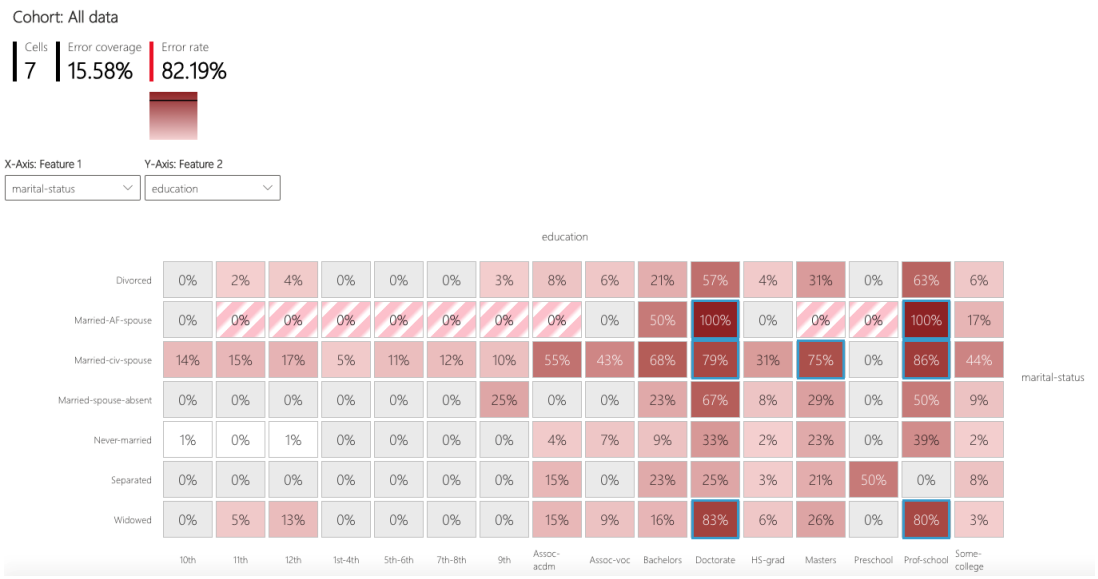


Figure 20. Error heatmap according to two features: education and marital status

Diagnosis of errors

Identifying cohorts with higher error rates is only the first step of the error analysis process, the second step is to debug the errors in these cohorts further through exploratory data analysis and model explainability. The Error Analysis toolkit allows you to perform:

- **Data Exploration**, which compares cohort data statistics and feature distributions with other cohorts or to benchmark data. This allows us you to investigate whether certain cohorts are underrepresented or if their feature distribution is significantly different from the overall data.
- **Global Explanation**, which explores the top K important features that impact the overall model global explanation for a selected cohort of data and compares these features with those from other cohorts or benchmark.
- **Local Explanation**, which enables you to understand individual data points from the cohort have correct or incorrect prediction and compare this local explanation with other cohorts. This allows visual identification of any missing features or label noise that could lead to issues.
- **What-if analysis** (Perturbation Exploration), which applies changes to feature values of selected data point and observe resulting changes to the prediction.

While InterpretML allows for global and local interpretability, Error Analysis complements the picture by allowing you to detect inaccuracies between cohorts of your data, that is to say cohorts with higher error rate than the overall benchmark this is error identification, and then use model explanation for that cohort to understand what drives the error rates up so you can take corrective actions.

For additional information on Error Analysis, you can consider the following resources:

- Project landing page at https://https://erroranalysis.ai
- GitHub repo at https://github.com/microsoft/responsible-ai-widgets/#getting-started

Hands-on walkthrough

In this hands-on walkthrough, we follow the section **Error Analysis** of the same notebook `understand_rai_tools.ipynb` used so far.

The goal is to show how to visualize model errors as well as global and local explanations using the Responsible AI Widget's Error Analysis visualization dashboard. To get there we build a model that classifies types of wine using scikit-learn, and then we analyze model errors and explanations using the Error Analysis dashboard.

Step 1: Import required packages and load the wine data from scikit-learn.

```
from sklearn.datasets import load_wine
from sklearn import svm
from interpret.ext.blackbox import MimicExplainer
from interpret.ext.glassbox import LGBMExplainableModel

wine = load_wine()
X = wine['data']
y = wine['target']
classes = wine['target_names']
feature_names = wine['feature_names']
```

Step 2: Train a SVM classification model.

```
from sklearn.linear_model import LogisticRegression
clf = svm.SVC(gamma=0.001, C=100., probability=True)
model = clf.fit(x_train, y_train)

print("number of errors on test dataset: " + str(sum(model.predict(x_test) != y_test)))
```

This prints the following: "number of errors on test dataset: 25"

We notice that the model makes a fair number of errors, but we have no idea why. This is where the Error Analysis Dashboard is useful.

Step 3: Identification of errors using decision trees and error heatmaps from the ErrorAnalysis dashboard (without explanations – see next step)

```
from raiwidgets import ErrorAnalysisDashboard
predictions = model.predict(x_test)
ErrorAnalysisDashboard(dataset=x_test, true_y=y_test, features=feature_names, pred_y=predictions)
```

The decision Tree tab of the dashboard generated by the previous command shows the following:

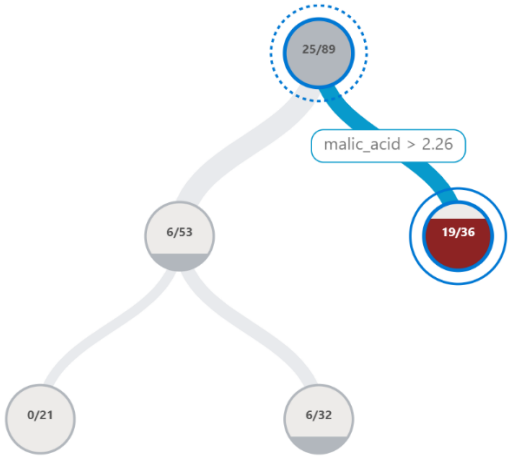


Figure 21. Error heatmap for wine classification

We clearly see from this decision tree that the biggest error rate (19 errors out of 36 predictions – more than half) occurs when the feature malic_acid is higher than 2.26. This is clearly a problem, and we need to investigate this cohort more punctually, which we will do in the next step.

Step 4: Running the Interpret-Community's 'explain_model' globally and locally to generate model explanations.

```
from raiwidgets import ErrorAnalysisDashboard
predictions = model.predict(x_test)
ErrorAnalysisDashboard(dataset=x_test, true_y=y_test, features=feature_names, pred_y=predictions)
```

Step 5: Analyze model errors and explanations using Error Analysis dashboard by feeding model explanations generated in previous Step 4.

```
from raiwidgets import ErrorAnalysisDashboard
ErrorAnalysisDashboard(global_explanation, model, dataset=X_test, true_y=y_test)
```

This way we are able to investigate the specific cohort where we identified a larger error rate in the previous step against the entire dataset, for example by comparing feature importance like in the figure below.

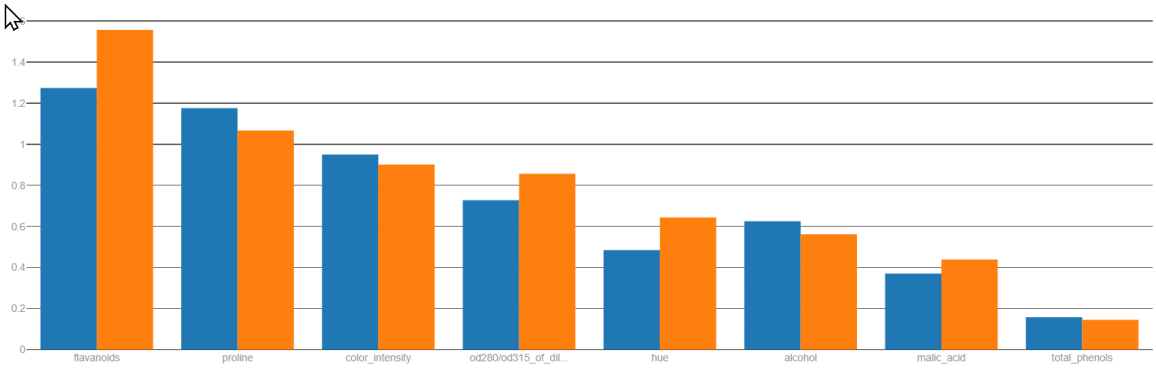


Figure 22. Feature importance for the problematic cohort (in orange) against the entire data (in blue).

This concludes this module about Responsible AI tools to understand the behavior of AI systems, off to the next module which explores tools to protect these AI systems and/or their data(set(s)).

Module 3: Protecting your AI systems and your data assets

Let's now look discuss how to protect both your AI systems and your data assets, i.e., against any adversarial attack, any potential misuse used to train the models and/or during inference.

Similarly, this comprises a set of practices and techniques to articulate as well. With adversarial attacks on both the ML algorithms and data that keep increasing, and as these ML-powered features and/or systems become more pervasive, the need to understand how they fail, whether by the hand of an adversary or due to the inherent design of a system, will only become more pressing to leverage the suitable techniques as part of the design, the development, the deployment, along with the monitoring of these features and/or systems.

Regarding the failure modes, they range:

- From *intentional failures* wherein the failure is caused by an active adversary attempting to subvert the system to attain her goals – either to misclassify the result, infer private training data, or to steal the underlying algorithm.
- To *unintentional failures* wherein the failure is because an ML feature or system produces a formally correct but completely unsafe outcome.

Discussing all the related implications would lead us to a number of considerations to articulate and is definitely outsides the scope of this starter guide.

To better understand how to build a secure AI starting from the threat modeling, we recommend considering the following articles:

- Securing the future of AI and machine learning at Microsoft
- Failure Modes in Machine Learning
- Threat Modeling AI/ML Systems and Dependencies
- AI/ML Pivots to the Security Development Lifecycle Bug Bar

As well as the video AI Security Engineering—Modeling/Detecting/Mitigating New Vulnerabilities.

This also supposes beyond such a state-of-the-art understanding, modeling, and assessment, to consider a number of techniques, would it be in terms of privacy-preserving machine learning (PPML) techniques that applies for the development, to name a few Homomorphic Encryption, Secure Multiparty Computing, and Differential Privacy, or other techniques for the deployment of the considered AI systems, the confidential inference being an illustration with the capabilities we provide with Azure Confidential Computing, See for example the "Data in Use Protection Compass" Workshop.

With that, for the rest of this module, we will focus on the former, and specifically consider two techniques : **anonymization and differential privacy**.

Anonymization is the process of obscuring Personally Identifiable Information (PII) in a manner that prevents it from uniquely identifying an individual. Anonymization reduces the risk of accidental disclosure of PII data, and if a data breach does occur, the stolen information will be of no use to attackers in trying to identify individuals.

Let's shortly share an example to set the context and illustrate the limitations.

Back in 2006, Netflix announced a $1 million prize for improving their movie recommendation service, releasing a movie database which was "carefully" anonymized by deleting personal details and substituting names with random numbers. Less than 18 months later, Arvind Narayanan et al. published the paper entitled Robust "How To Break Anonymity of the Netflix Prize Dataset" where they "successfully identified the Netflix records of known users, uncovering their apparent political preferences and other potentially sensitive information".

If we had to take a single takeaway from the Netflix data re-identification story, it is that "**anonymized data isn't**", as has rightfully said Cynthia Dwork from Microsoft Research (MSR) and one of the pioneers of research in model security and privacy, See Privacy and accuracy: How Cynthia Dwork is making data analysis better.

The above will lead us to explore two tools for the sake of this starter guide.

The first tool called **Presidio** is a personally identifiable information (PII) identifier and anonymizer, which cannot be used standalone to give any privacy guarantees taking into account the previous but can be combined with the second one, the **SmartNoise** system, which provides a concrete implementation of Cynthia's revolutionary Differential Privacy (DP) concept, See Differential Privacy: A Primer for the Perplexed.

So, let's introduce Presidio.

Presidio

Presidio is an open-source data protection and anonymization SDK for text and images providing fast **identification and anonymization of private entities** in text such as credit card numbers, names, locations, social security numbers, bitcoin wallets, US phone numbers, financial data and more.

Presidio's modules include:

1. **The Presidio analyzer** for custom or predefined PII detection in text, leveraging Named Entity Recognition, regular expressions, rule-based logic, and checksum with relevant context in multiple languages.
2. **The Presidio anonymizer** for allowing anonymization of the detected PII entities using different operators.
3. **The Presidio image redactor** for redacting PII text in images.

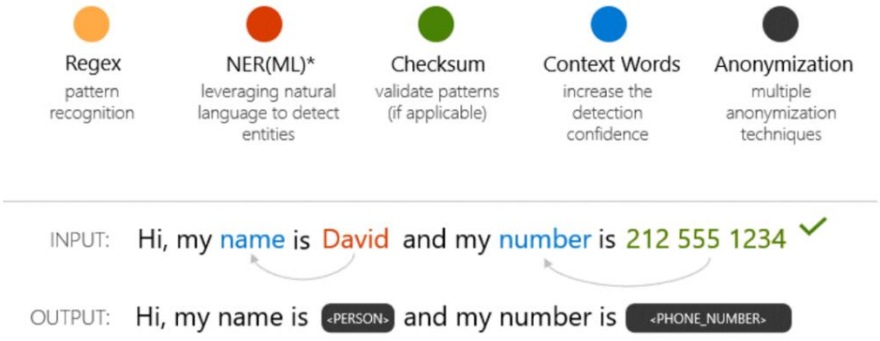


Figure 23. Inside Presidio identification and anonymization modules

*NER: Named entity recognition.

With that, let's see how to use Presidio for simple PII analysis and anonymization on text and how to customize the above Presidio PII analyzer to detect new types of PII entities.

Hands-on walkthrough

The Jupyter notebook for this hands-on walkthrough is `protect_rai_tools.ipynb` located in the `hands_on_tutorials` directory underneath the folder where you have cloned the samples' repo, for example the folder `raisamples` in our illustration, please refer to the section **Cloning the samples' Jupyter notebooks** above.

You can scroll down the section **Presidio**.

Step 1: Installing the presidio_analyzer and presidio_anonymizer libraries using pip along with the spaCy English language model needed by the analyzer.

```
pip install presidio_analyzer
pip install presidio_anonymizer
# Presidio analyzer requires a spaCy language model.
Python -m spacy download en_core_web_lg
```

Step 2: Once the presidio-analyzer package is installed, run this simple analysis script.

```
From presidio_analyzer import AnalyzerEngine

text_to_anonymize = "His name is Mr. Jones and his phone number is 212-555-5555"
analyzer = AnalyzerEngine()
analyzer_results = analyzer.analyze(text=text_to_anonymize, entities=["PHONE_NUMBER"], language='en')

print(analyzer_results)
```

This will print the result of the PII analysis, in this case the detected phone numbers in the provided text.

Step 3: Creating Custom PII Entity Recognizers.

```
From presidio_analyzer import PatternRecognizer

text_to_anonymize = "His name is Mr. Jones and his phone number is 212-555-5555"

titles_recognizer = PatternRecognizer(supported_entity="TITLE",
                                      deny_list=["Mr.","Mrs.","Miss"])

pronoun_recognizer = PatternRecognizer(supported_entity="PRONOUN",
                                      deny_list=["he", "his", "she", "hers"])

analyzer.registry.add_recognizer(titles_recognizer)
analyzer.registry.add_recognizer(pronoun_recognizer)

analyzer_results = analyzer.analyze(text=text_to_anonymize,
                            entities=["TITLE", "PRONOUN"],
                            language="en")
print(analyzer_results)
```

The previous code snippet:

1. Creates custom titles and pronouns recognizers.
2. Adds the new custom recognizers to the analyzer.

3. Calls analyzer to get results from the new recognizers.

It prints the titles and pronouns detected in the provided text.

Step 4: Anonymizing the identified PII entities.

```
From presidio_anonymizer import AnonymizerEngine
from presidio_anonymizer.entities.engine import OperatorConfig

anonymizer = AnonymizerEngine()

anonymized_results = anonymizer.anonymize(
    text=text_to_anonymize,
    analyzer_results=analyzer_results,
    operators={"DEFAULT": OperatorConfig("replace", {"new_value": "<ANONYMIZED>"}),
               "PHONE_NUMBER”: OperatorConfig("mask", {"type": "mask", "masking_char" : "*",
"chars_to_mask" : 12, "from_endv : True}),
                "TITLE": OperatorConfig(“redact”, {})}
)

anonymized_results.to_json()
```

The previous code snippet:

1. Sets up the anonymizer engine.
2. Creates an anonymizer request – text to anonymize, list of anonymizers to apply and the results from the analyzer request.
3. Anonymizes the text.

It prints the anonymized text along with a list of the detected PII entities like this:

```
Text: "His name is <ANONYMIZED> and <ANONYMIZED> phone number is ************".

- Items:

       - {"start": 59, "end": 71, "entity_type": "PHONE_NUMBER", "text": "************", "operator":
"mask"},

       - {"start": 30, "end": 42, "entity_type": "PRONOUN", "text": "<ANONYMIZED>", "operator":
"replace"},

       - {"start": 13, "end": 25, "entity_type": "PERSON", "text": "<ANONYMIZED>", "operator":
"replace"},

       - {"start": 12, "end": 12, "entity_type": "TITLE", "text": "", "operator": "redact"}]}
```

This ends our investigation of Presidio, let’s now jump into the realm of Differential Privacy with the SmartNoise system.

SmartNoise

You said Differential Privacy, *what do you mean?*

Considering membership inference attacks

Sensitive and confidential information about individuals is extensively used and shared between companies, government entities, research organizations, and other parties to train ML models. Using only black-box access to such an ML model, an adversary can determine if a sample was a member of the training set used to build this model, this is called a **membership inference attack**. Inadequate usage of this kind of information can result in significant consequences, such as harm to an individual's reputation, employability, creditworthiness, and insurability.

Invented by Microsoft Research (MSR) and associates, Differential Privacy (DP) is considered the gold standard for protecting individuals' data against membership inference attacks. It provides a mathematically measurable privacy guarantee to individual data subjects and offers significantly higher privacy levels than commonly used disclosure limitation practices like data anonymization. The latter increasingly shows vulnerability to re-identification attacks – especially as more data about individuals become publicly available.

The issue with traditional data anonymization approaches

The crucial problem with anonymized data is that the released records often include **unique combinations of variables** (digital fingerprints) that someone might link to other publicly available information to re-identify specific people. For instance, research has shown that 87% of Americans can be uniquely identified with only three pieces of data: Gender, birthday, and ZIP code.

A typical goal for today's data disclosure practices is to achieve a standard known as *k-anonymity*. This is achieved by minimizing the number of attributes that are particularly vulnerable to re-identification or reducing details by grouping values into brackets (e.g., age brackets). For example, a released dataset satisfies 5-anonymity if at least five records exist for each combination of gender, age, and ZIP code. While this approach likely reduces the hit rate that an attacker can achieve, it is far from solving the problem and fails to provide any reliable privacy guarantee to individuals.

The Differential Privacy (DP) concept

Differential Privacy (DP) requires that any analytical results on a dataset A including an individual's record are identical (or at least very close) to the analytical results on a dataset B where the individual's record has been removed. Differential Privacy aims to **mask the contribution of the individuals record** by adding a precisely tuned amount of random **noise** to the data.

The amount of noise that is introduced to the computation must be chosen carefully. On the one hand, higher quantities of noise increase the level of privacy. On the other side, it is more difficult to derive reliable statistical results when the noise level is too high. There is a tunable parameter available to adjust the amount of noise in the trade-off between privacy and utility. This is known as the **privacy parameter epsilon**. It is also called the privacy budget.

The SmartNoise system

SmartNoise is a joint project by Microsoft and Harvard's Institute for Quantitative Social Science (IQSS) and the School of Engineering and Applied Sciences (SEAS) as part of the OpenDP initiative. It aims to make Differential Privacy broadly accessible.

The SmartNoise tools primarily focus on the "global model" of Differential Privacy where a trusted data collector is presumed to have access to unprotected data and wishes to protect public releases of aggregate information. For example, a hospital having access to patients' information and wishing to release aggregated statistics about these patients without affecting their privacy.

SmartNoise is an open-source project that contains different components for building global differentially private systems. SmartNoise is made up of the following top-level components (only the core library is explored here):

- SmartNoise Core library
- SmartNoise SDK library

The SmartNoise core library includes the following privacy mechanisms for implementing a differentially private system:

- **Analysis**: A graph description of arbitrary computations to perform on the data. These can include statistics like count and mean or utilities like filtering and imputation.
- **Validator**: A Rust library that contains a set of tools for checking and deriving the necessary conditions for an analysis to be differentially private. Support for Python is also available.
- **Runtime:** The medium to execute the analysis. The reference runtime is written in Rust but runtimes can be written using any computation framework such as SQL and Spark depending on your data needs.
- **Bindings:** Language bindings and helper libraries to build analyses. SmartNoise currently provides Python bindings.

Hands-on walkthrough: Protecting statistics against reconstruction attacks

In this hands-on walkthrough, we will explore how data can be protected against reidentification attacks using Differential Privacy and the SmartNoise system by following this notebook from the SmartNoise samples repository. Please be aware that only the most important parts of the code are shown here, so please follow the notebook for completeness.

The goal is to show how an attacker can leverage basic demographic information like age and zip codes to reidentify individuals even when the sensitive data is published in an anonymized format. Then we show how Differential Privacy can help prevent such an attack.

Step 1: Importing the data

We will use anonymized medical dataset and some sample demographic information dataset for performing the identification attack:

```
import reident_tools as reident
from opendp.smartnoise.synthesizers.mwem import MWEMSynthesizer

df_medical = pd.read_csv('data/data_medical.csv', sep=",", encoding="utf-8").infer_objects()
print('Anonymized dataset including sensitive medical information:')
display(df_medical.iloc[:,1:].sample(8))
```

```
df_demographic = pd.read_csv('data/data_demographic.csv', sep=",", encoding="utf-8").infer_objects()
print('Attacker`s data collection with basic demographic information:')
```

This prints the following two samples:

Anonymized dataset including sensitive medical information:

	Gender	Age	Zip	Diagnosis	Treatment	Outcome
22290	F	80-89	329**	Arthritis	31	recovered
19225	M	60-69	797**	Stroke	31	unchanged
29342	F	30-39	952**	Arthritis	41	intensive care
1530	M	40-49	196**	Heart Disease	40	unchanged
21141	F	60-69	972**	Depression	50	recovered
1823	F	50-59	664**	Osteoporosis	33	intensive care
21291	M	10-19	279**	High Blood Pressure	50	recovered
14541	M	40-49	490**	Osteoporosis	42	recovered

Attacker`s data collection with basic demographic information:

	Name	Gender	Age	Zip
17309	Joel Lewis	M	12	65399
2483	Chad Gonzales	M	16	16730
15184	Barry Cannon	M	20	84551
29598	Daniel Baker	M	40	30396
13358	Melissa Johnson	F	16	36679
29264	Christina Phillips	F	12	61586
24430	Carol Fisher	F	12	1956
9520	Donna Jimenez	F	79	10376

Step 2: Reidentification attack

Now, we perform the reidentification attack using the try_reidentification function. As input, we use the data sets generated above (anonymized medical and demographic data).

```
Reident_attack = reident.try_reidentification(df_demographic, df_medical, logger)

print(f'Sample of re-identified patients:')
reident_attack[reident_attack["ID_Match"]==True][['Name', 'Gender', 'Age', 'Zip', 'Diagnosis',
'Treatment', 'Outcome', 'ID_Match']].sample(10)
```

This prints a sample of individuals we were able to identify along with their diagnosis data:

```
Sample of re-identified patients:
```

	Name	Gender	Age	Zip	Diagnosis	Treatment	Outcome	ID_Match
3125	Troy Gonzalez	M	58	69973	High Blood Pressure	36	intensive care	True
5959	Olivia Walker	F	16	76680	Alzheimer	38	intensive care	True
24635	Amber Jones	F	56	39631	COPD	21	intensive care	True
8043	Mrs. Paula Simmons DVM	F	78	14839	Cancer	49	intensive care	True
15418	Oscar Norman	M	47	80099	Depression	30	recovered	True
9360	Sheila Berg	F	89	43955	High Blood Pressure	23	recovered	True
15075	Kristine Lewis	F	25	44018	Alzheimer	36	unchanged	True
1277	Kimberly Bolton	F	12	88284	Alzheimer	25	unchanged	True
23105	Erin Whitaker	F	78	40613	Alzheimer	36	recovered	True
10656	Karen Compton	F	11	25730	Arthritis	30	recovered	True

A total of 9124 individuals were re-identified!

Step 3: Protecting the medical dataset with differential privacy using the MWEM synthesizer from SmartNoise.

Let's look at the data before encoding and synthesizing:

```
df_reident_synth = df_medical[['Gender', 'Age', 'Zip', 'Diagnosis', 'Treatment', 'Outcome']].copy()
df_reident_synth['Zip'] = df_demographic['Zip'].copy()
df_reident_synth['Age'] = df_demographic['Age'].copy()

# Have a quick glance at the data
df_reident_synth.head()
```

	Gender	Age	Zip	Diagnosis	Treatment	Outcome
0	F	10	65418	High Blood Pressure	25	intensive care
1	F	14	65475	COPD	48	unchanged
2	F	10	65484	High Blood Pressure	38	intensive care
3	F	30	27727	Heart Disease	31	unchanged
4	F	36	27772	Diabetes	34	unchanged

Now we encode the data using the do_encode-function to make it compatible with the MWEM synthesizer:

```
# Encode the data set and display it
df_reident_encoded = reident.do_encode(df_reident_synth, ['Gender', 'Age', 'Zip', 'Diagnosis'],
reident.diseases)
df_reident_encoded.head()
```

	Gender_encoded	Age_encoded	Zip_encoded	Diagnosis_encoded
0	0	10	65418	9
1	0	14	65475	2
2	0	10	65484	9
3	0	30	27727	7
4	0	36	27772	3

Finally, we synthesize new data using the SmartNoise MWEM synthetizer (by adding noise to the original data:

```
# Apply the synthesizer to the dataset
synthetic_data = MWEMSynthesizer(Q_count = 400, epsilon = 3.00, iterations = 60,
mult_weights_iterations = 40, splits = [], split_factor = 1)

synthetic_data.fit(df_reident_encoded.to_numpy())
df_synthesized = pd.DataFrame(synthetic_data.sample(int(df_reident_encoded.shape[0])),
columns=df_reident_encoded.columns)

# Compare original and synthetic data
reident.create_histogram(df_reident_encoded, df_synthesized, 'Diagnosis_encoded', reident.diseases)
```

Below, is a histogram comparing the diagnoses distribution of both the original and the synthesized datasets. We see that the distributions are pretty similar, which means that not much information is lost during the synthetization process:

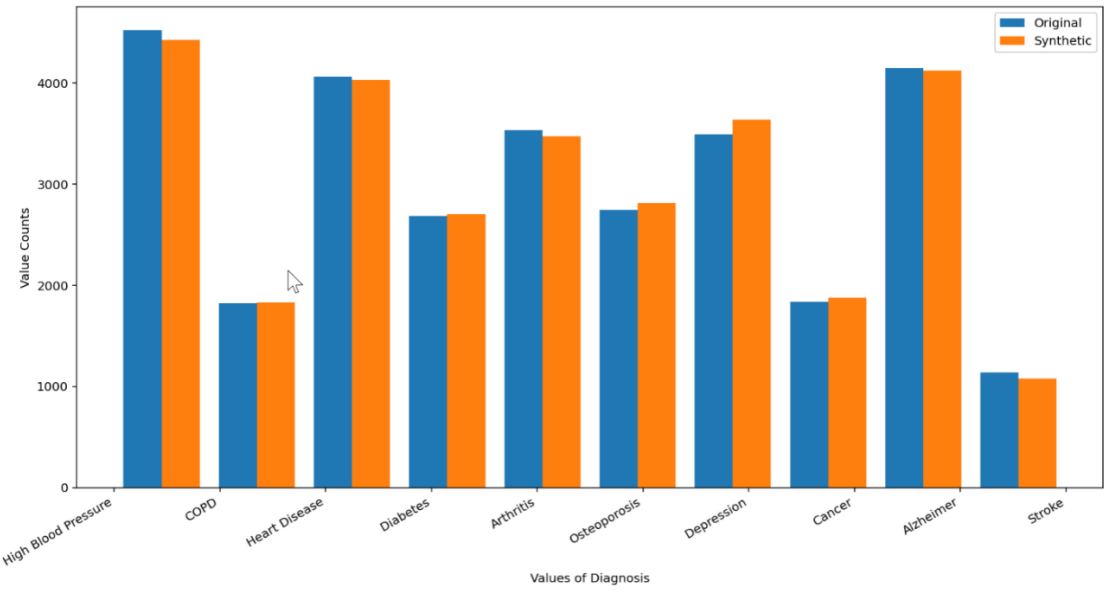


Figure 24. Histogram showing diagnosis distribution for both the original and synthetic datasets

Step 4: Reidentification attack on synthesized data

Finally, we try the re-identification attack on the synthesized data using the try_reidentification_noise-function. As explained, the synthesized data set has new combinations of demographic data, so we do not deal with the raw/real data anymore which drastically reduces the risk of a potential reidentification match.

Here are how the original and synthesized datasets look like:

```
print('Medical Dataset:')
display(df_medical_synth.sample(5))
print('\nSynthesized Demographic Dataset:')
display(df_synthesized.sample(5))
```

```
Medical Dataset:
```

	ID	Gender	Age	Zip	Diagnosis	Treatment	Outcome
2878	30198b29a32c4437ab0f520b4f2ea80c	F	55	98535	Arthritis	50	intensive care
22402	3e9ad6193c4d458b9e593df24b4fb92f	M	86	82318	Diabetes	24	recovered
10670	77454ac7aa13446a8dc738a8eb024e81	F	46	41732	Arthritis	24	recovered
549	62ec56fb72a14e0e9f9c684bf5dd4c00	F	70	56172	High Blood Pressure	41	recovered
2787	20952448800040e5ad716e21c1a962d5	F	73	86049	Cancer	32	unchanged

```
Synthesized Demographic Dataset:
```

	Gender_encoded	Age_encoded	Zip_encoded	Diagnosis_encoded
2992	0	62	79304	8
6094	1	63	28764	8
7410	0	34	45279	0
17414	1	39	95023	9
27195	0	77	50254	6

Now we try to perform the reidentification attack:

```
reident_attack_2 = reident.try_reidentification_noise(df_synthesized, df_medical_synth, logger)

print(f'Found {len(reident_attack_2)} potential matches!')
```

This prints "Found 0 potential matches!" as expected.

All-in-all, this hands-on walkthrough demonstrates the magic behind Differential Privacy, it allows you to protect data against reidentification attacks by masking individual contributions and providing mathematical guarantees of privacy, while still preserving the distribution and thus summary statistics of the data.

To continue your exploration on how you can protect personal data against privacy attacks for your ML models, you can read the white paper Microsoft SmartNoise Differential Privacy Machine Learning Case Studies.

This white paper provides practical guidance on how personal data can be rigorously protected for applications like statistics, machine learning, and deep learning using Differential Privacy. Interestingly enough, it also provides you with a number for interactive demo scenarios:

- Protecting statistics against reconstruction attacks, *sounds familiar*.
- Protecting sensitive data against re-identification attacks.
- Privacy-preserving statistical analysis.
- Machine learning using a differentially private classifier.
- Generating a synthetic dataset for privacy-preserving machine learning.

- Detect pneumonia in X-Ray images while protecting patients' privacy.

The related Jupyter notebooks are located here: https://github.com/opendp/smartnoise-samples/tree/master/whitepaper-demos.

You are all set to go through them.

This concludes this module about tools to protect AI systems data.

As a conclusion

This concludes this starter guide. We hope you have enjoyed this guided tour.

From holistically transforming industries to addressing critical issues facing humanity, AI is already solving some of our most complex challenges and redefining how humans and technology interact.

As part of this guided tour, we have outlined some of the steps we are taking to prioritize Responsible AI along with some of the tooling we use within our company and make available outside in hopes that our experience can help other people and organizations like yours. But we only scratch the surface (and we are only at the beginning).

Going beyond

To continue learning about the passionate subject of Responsible AI, you can visit our Responsible AI Resources page where you can access the entirety of already available tools, guidelines, and other additional resources that will help you create your next AI solution in a (more) responsible manner.

Microsoft France
39 Quai du Président Roosevelt
92130 Issy-Les-Moulineaux