DZone

DZONE TREND REPORT

DECEMBER 2022

Enterprise AI

Machine Learning, Design Paradigms, and Operational Impact

BROUGHT TO YOU IN PARTNERSHIP WITH

KATANA GRAPH

NOW FEATURING THE SOLUTIONS DIRECTORY AND MORE!

Table of Contents

Welcome Letter

By Tyler Sedlar, Software Engineer at DZone

Artificial Intelligence (AI) is part of a software space that is incredibly complex, with the intention of mimicking the human mind by *learning* from a given training dataset. To think of the training process simply is to place items from the dataset in two categories, drawing a line down the middle to separate them. In the case of binary classification, this line delineates whether something *is* or *is not* the topic at hand.

The goal is to place items as far left in category one and as far right in category two. The closer we are to the line, the less room for error we have during classification. Outliers *will* exist, but as far as we're concerned, the bulk of the data is what is important to classify.

AI programs can be taught to learn in two ways: supervised or unsupervised. Supervised learning classifies data based on input that has been pre-labeled by a human, typically, in pairs of data — for example, text content and "spam" or "not spam." Unsupervised learning, on the other hand, learns patterns from unlabeled data. The machine will analyze and cluster the data without the need for human labeling.

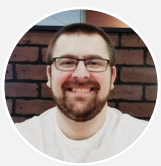
There are many subsets of AI and machine learning (ML), including natural language processing (NLP), computer vision, and speech recognition. NLP is used to aid in identifying parts of speech when given a string of text. Computer vision is used to assign images to categories or check their similarities, which is most helpful to identify unwanted imagery on websites — specifically, on public forums. This software is also applied in technology like self-driving cars. Lastly, speech recognition is integrated into many technologies we use today, such as automated phone calls and smart home integrations.

Though AI is extremely helpful, it can also be detrimental in cases such as "deepfakes," which are videos, images, or voices that are generated based on the actions of real people (e.g., voice clips that mimic a person with political power). With the widespread use of social media, the combination of these two technologies can be quite scary if put to malicious use, making ethics a crucial consideration to prioritize in this space.

At DZone, we've recently started using NLP to help us identify spammy content submissions, with the support of our moderators to confirm content validity — letting our text classifier learn continuously. In the spirit of continuous learning, join us in exploring the latest industry advances in our 2022 "Enterprise AI" Trend Report, featuring DZone research and contributor insights into AI tools, event-driven platforms, MLOps, and more.

Sincerely,

Tyler Sedlar



Tyler Sedlar, Software Engineer at DZone

@tsedlar on DZone | @tsedlar on LinkedIn

Tyler was introduced to software development by creating automation scripts for MMORPGs with Java and has made software development his career since. He also enjoys programming as a hobby as well — noting his favorite languages as Kotlin and Node.js. He otherwise enjoys being around family, playing board games, and playing video games such as LoL, WoW, and OSRS.

ABOUT

DZone Publications

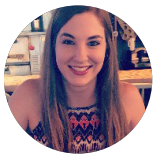
Meet the DZone Publications team! Publishing Refcards and Trend Reports year-round, this team can often be found reviewing and editing contributor pieces, working with authors and sponsors, and coordinating with designers. Part of their everyday includes collaborating across DZone's Production team to deliver high-quality content to the DZone community.

DZone Mission Statement

At DZone, we foster a collaborative environment that empowers developers and tech professionals to share knowledge, build skills, and solve problems through content, code, and community.

We thoughtfully — and with intention — challenge the status quo and value diverse perspectives so that, as one, we can inspire positive change through technology.

Meet the Team



Caitlin Candelmo
Director, Content Products at DZone
@CCandelmo on DZone
@caitlincandelmo on LinkedIn

Caitlin works with her team to develop and execute a vision for DZone's content strategy as it pertains to DZone publications, content, and community. For publications, Caitlin oversees the creation and publication of all DZone Trend Reports and Refcards. She helps with topic selection and outline creation to ensure that the publications released are highly curated and appeal to our developer audience. Outside of DZone, Caitlin enjoys running, DIYing, living near the beach, and exploring new restaurants near her home.

Lindsay Smith
Senior Publications Manager at DZone
@DZone_LindsayS on DZone
@lindsaynicolesmith on LinkedIn

Lindsay oversees the Publication lifecycles end to end, delivering impactful content to DZone's global developer audience. Assessing Publications strategies across Trend Report and Refcard topics, contributor content, and sponsored materials — she works with both DZone authors and Sponsors. In her free time, Lindsay enjoys reading, biking, and walking her dog, Scout.

Lucy Marcum
Publications Coordinator at DZone
@LucyMarcum on DZone
@lucy-marcum on LinkedIn

As a Publications Coordinator, Lucy spends much of her time working with authors, from sourcing new contributors to setting them up to write for DZone. She also edits publications and creates different components of Trend Reports. Outside of work, Lucy spends her time reading, writing, running, and trying to keep her cat, Olive, out of trouble.

Melissa Habit
Senior Publications Manager at DZone
@dzone_melissah on DZone
@melissahabit on LinkedIn

Melissa leads the publication lifecycles of Trend Reports and Refcards — from overseeing workflows, research, and design to collaborating with authors on content creation and reviews. Focused on overall Publications operations and branding, she works cross-functionally to help foster an engaging learning experience for DZone readers. At home, Melissa passes the days reading, knitting, and adoring her cats, Bean and Whitney.

Lauren Forbes
Content Strategy Manager at DZone
@laurenf on DZone
@laurenforbes26 on LinkedIn

Lauren identifies and implements areas of improvement when it comes to authorship, article quality, content coverage, and sponsored content. She also oversees our team of contract editors, which includes recruiting, training, managing, and fostering an efficient and collaborative work environment. When not working, Lauren enjoys playing with her cats, Stella and Louie, reading, and playing video games.

Jason Cockerham
Community Engagement Manager at DZone
@Jason Cockerham on DZone
@jason-cockerham on LinkedIn

Jason heads the DZone community, driving growth and engagement through new initiatives and building and nurturing relationships with existing members and industry subject matter experts. He also works closely with the content team to help identify new trends and hot topics in software development. When not at work, he's usually playing video games, spending time with his family, or tinkering in his garage.

ORIGINAL RESEARCH

Key Research Findings

An Analysis of Results from DZone's 2022 Enterprise AI Survey

John Esposito, PhD, Technical Architect at 6st Technologies

In January 2022, DZone surveyed software developers, architects, and other IT professionals in order to understand how software professionals think about artificial intelligence (AI) and apply AI especially to enterprise applications.

Major research targets were:

1. Spectra of applied techniques that might be considered AI
2. Applications of AI in different domains related to software development
3. Attitudes toward current and future AI

Methods:
We created a survey and distributed it to a global audience of software professionals. Question formats included multiple choice, free response, and ranking. Survey links were distributed via email to an opt-in subscriber list, LinkedIn, DZone Core Slack Workspace, and popups on DZone.com. The survey opened on December 24, 2021 and closed on January 12, 2022, recording 600 responses.

In this report, we review some of our key research findings. Many secondary findings of interest are not included here. Additional findings will be published piecemeal on DZone.com.

Research Target One: Spectra of Techniques That Might Be Considered AI

Motivations:

1. Surely, we can't define "artificial intelligence" if we can't define "intelligence" precisely. But perhaps we might measure both without enumerating necessary and sufficient attributes. We wanted to understand how software professionals recognize something as "AI" or not in order to know what we all denote when talking about AI.

2. Further, the same systems may be considered AI by some but not by others. These systems do what they do irrespective of labels attached by observers, of course. But a label helps both communicate about our work to others and project our own work in new directions.

   For example, if I relabel systems as "AI" that I have already built but not previously labeled "AI," then I may consider new learning directions, new career paths, or new development ecosystems. So it seems important to know what "counts" as AI among software professionals.

3. Consider the following:

   - Thermostats optimize via feedback from their environment.
   - Linear regressions encourage qualitative conclusions from quantitative data.
   - Genetic algorithms optimize more quickly than linear and are inspired by biological processes.
   - Cellular automata generate unexpected complexity from extremely simple procedural definitions.

   Whether or not the elements in this list are commonly called "AI," they all exert discriminatory leverage toward some goal — they all act in steps that their programmers do not need to know. We wanted to know how often these kinds of intelligence-like techniques are applied, "AI"-advertised or not.

USE OF GENERALLY RECOGNIZED AI TECHNIQUES

"Do you even AI?" is a question more for marketers than programmers, but programmers are paid to write software for markets. The name "AI" is vague but inspiring; the name "support vector machine" is precise but flat. So we wanted to know both what specific techniques sometimes called "AI" software professionals are applying and whether software professionals consider themselves to have built an intelligent system in applying these techniques.

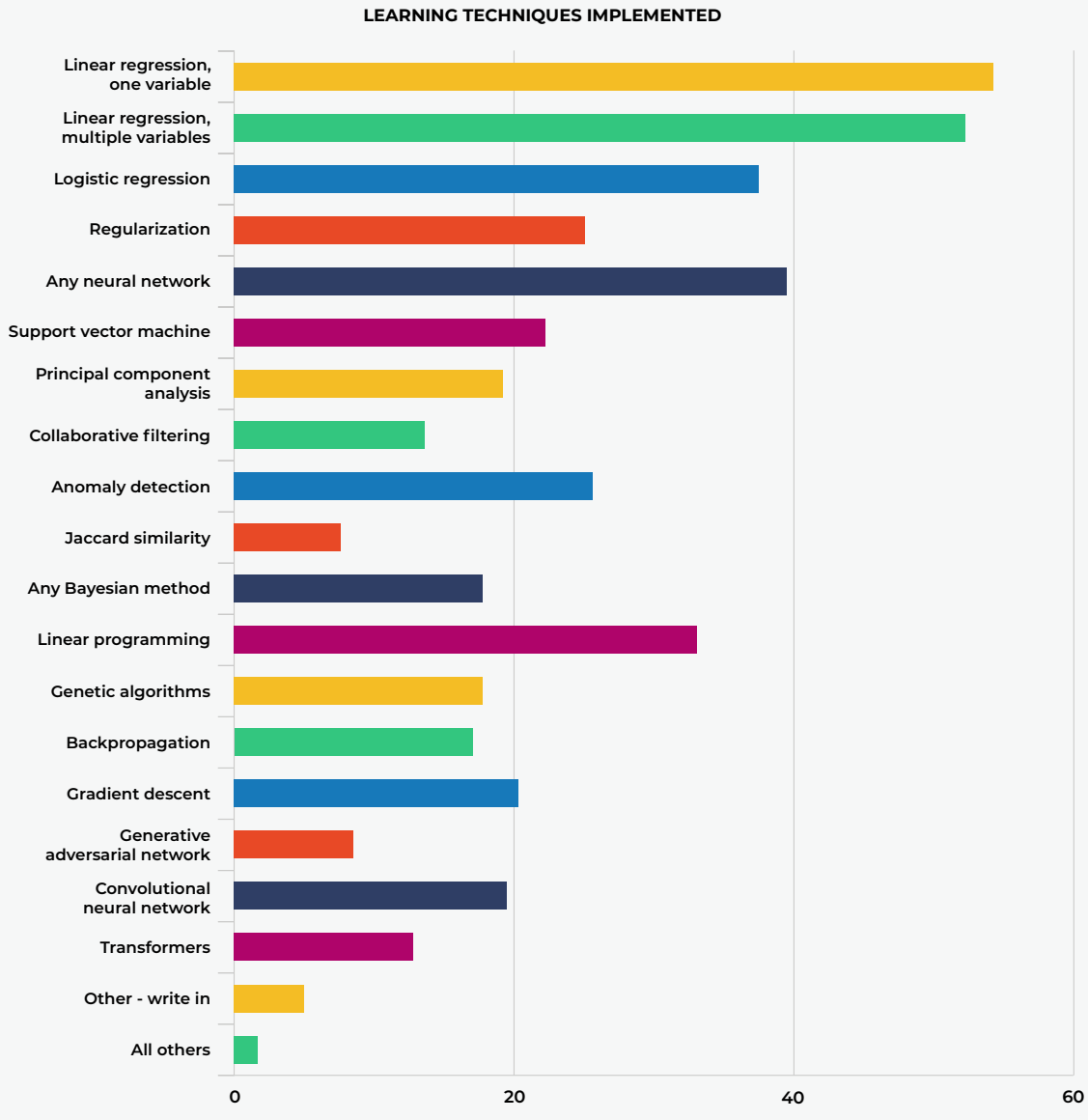
USE OF SPECIFIC TECHNIQUES AND THEIR RELATION TO THE CONCEPT OF "AI"

We asked both questions mentioned above:

Which of the following learning-related techniques have you used and/or implemented? and Have you ever built software with features that you consider "artificial intelligence" in any sense?

Results by technique (n=514):

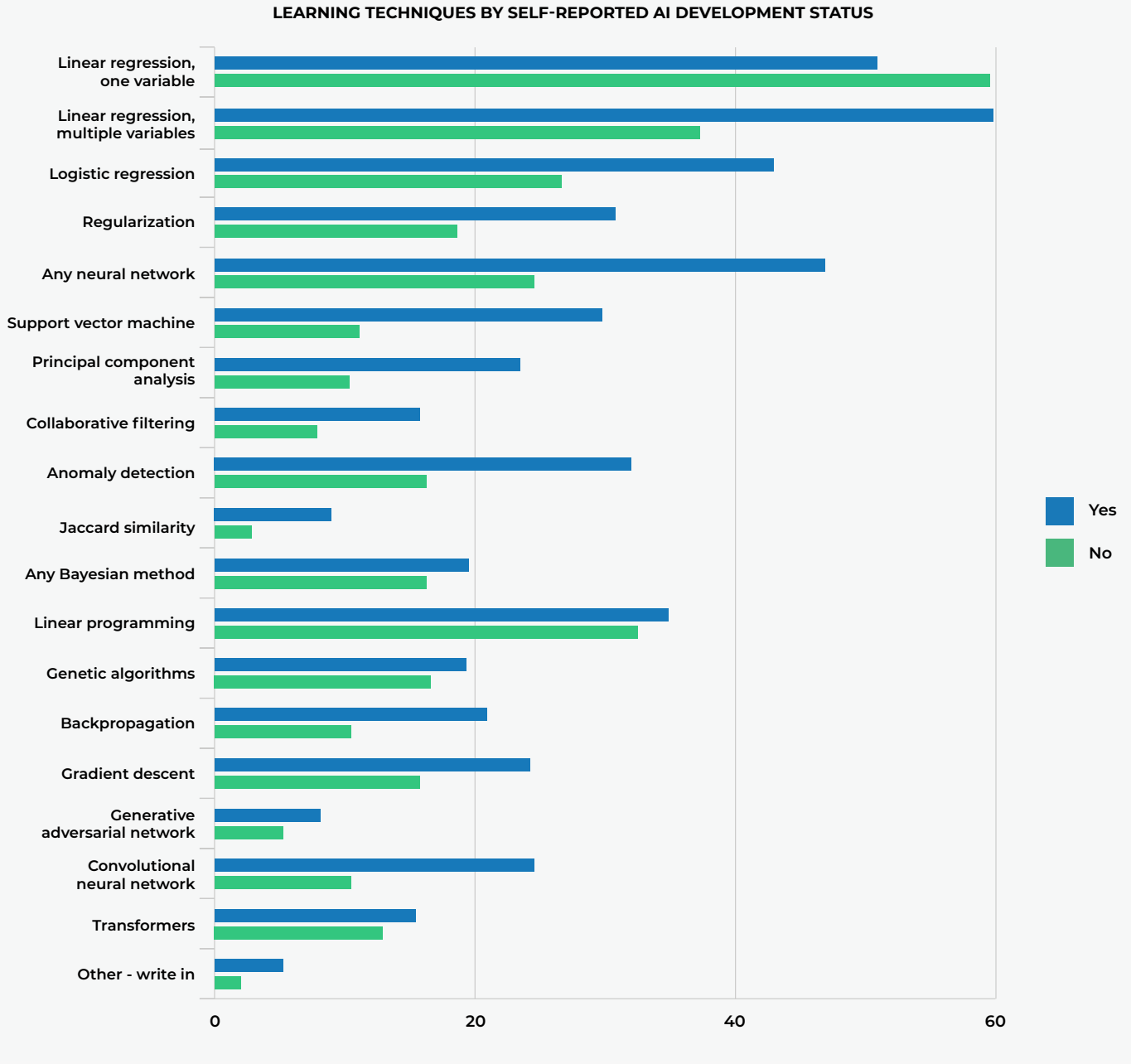
Figure 1



The same results, segmented by whether respondents also reported that they have "built software with features that [they] consider to be 'artificial intelligence' in any sense":

See Figure 2 on next page

Figure 2



Observations:

1. Linear and logistic regressions long predate AGI ambitions; however:

    - Feedforward neural networks are just stacks of linear regressions with "synaptic" nonlinearities (in practice, often logistic sigmoid functions rather than stepwise Heaviside functions) intervening between the layers.
    - The addition of feedback, to form recurrent networks, does not obviate linearities within each layer.
    - Linear regressions solved in purely statistical ways — like ordinary least squares — are conceptually cleaner than more general techniques that accomplish the same line-fitting (e.g., various forms of gradient descent).

    With this, we are therefore not surprised that linear regression is the most common learning-related technique applied by software professionals, with logistic regression coming in a close third behind neural networks.

    More interesting is that a (small) majority has used and/or implemented a statistical technique — not something developers need do every day. Both the rank and (small) majority of respondents having used linear regression hold across multiple segments: years' experience, app types developed, and self-judgment of having built AI systems.

We imagine that the survey title might have introduced some response bias. But we do not think the response bias is large because the overall respondent demographics do not substantially differ from the respondent demographics of other surveys on different topics that were distributed to the same recipient list.

2. Respondents who don't judge themselves to have built any AI are more likely to have used single-variable linear regressions, while respondents who do judge themselves to have built any AI are more likely to have used multivariable regressions.

   We conjecture two possible explanations. First, software professionals may be using a "general complexity" factor in calculating whether linear regression counts as AI. This is consistent with the "magicality" of the vague term, "AI." Second, because multiple and multivariate regression calculations involve covariance matrixes, and matrix algebra techniques are among the first "higher math" techniques used by programmers, the "matrix threshold" may count as higher math enough to push a regression into the "counts-as-AI" bucket.

3. Neural networks are the most common not-merely-statistical-learning-related method used by software professionals. This is so obvious as not to deserve additional comment.

   More interesting, however, is how much the use of neural networks functions as a self-conscious "is-this-AI" threshold: The gap between respondents who consider themselves to have built an AI system and those who do not is greater with respect to the use of neural networks (47.8% vs. 25.1%, a 22.7% difference) than this gap with respect to the use of any other learning-related technique.

4. Principal component analysis (PCA) and anomaly detection are also significant difference makers between responses, "have built AI" and "have not built AI," in software professionals' self-perception, with "have built AI" respondents more than twice as likely to have used PCA (23.4% vs. 11%) and to have performed anomaly detection (31.3% vs. 15.2%).

5. Convolutional neural networks are still used significantly more than transformers (19.1% vs. 14%). But since transformers are being used increasingly in research literature in many domains — especially domains that require consideration of larger input fields (which is greatly aided by transformers' "attention" mechanism) — we are interested to see if this ratio changes. We plan to ask this question in another survey later this year or early next.

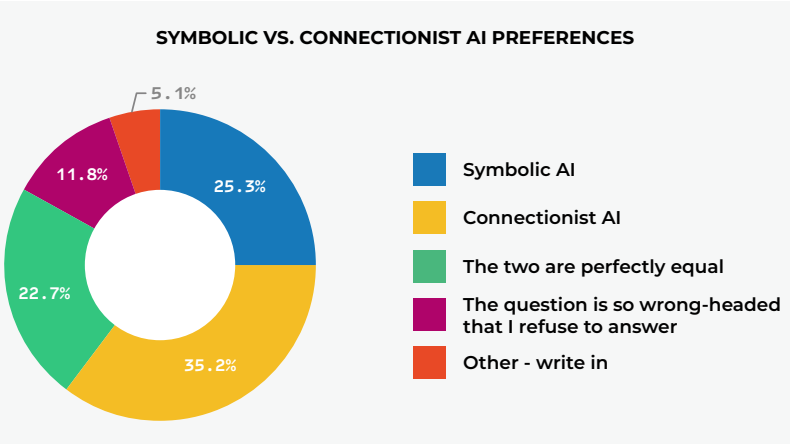
RELATION OF SELF-JUDGED USE OF AI TO EVALUATION OF HIGH-LEVEL AI PARADIGMS

The specific techniques listed in the question discussed directly above might be taught in three types of academic courses: statistics, machine learning, and dynamic programming. These courses apply different mixes of mathematical techniques — algebraic, combinatoric, or analytical. Such techniques carve reality along very different joints, and those differences in world-carvings project into the divergent *symbolic* (from algebra and operations research) vs. *connectionist* (from topology and neurobiology research) approaches to AI. In addition to the more focused techniques treated above, we wanted to understand software respondents' attitudes toward this higher-level symbolic vs. connectionist distinction. So we asked:

On the whole, which is better (by your own definition of "better"): "symbolic" AI (e.g., expert systems) or "connectionist" AI (e.g., multi-layered perceptrons)? Note: obviously both may be good. We're interested in where your preferences lie.

Results (n=586):

Figure 3



Observations:

1. Since neural networks fall under the *connectionist* paradigm and are the most popular single learning-related technique, we are not surprised that more respondents considered connectionist AI better than symbolic AI (35.2% vs. 25.3%).

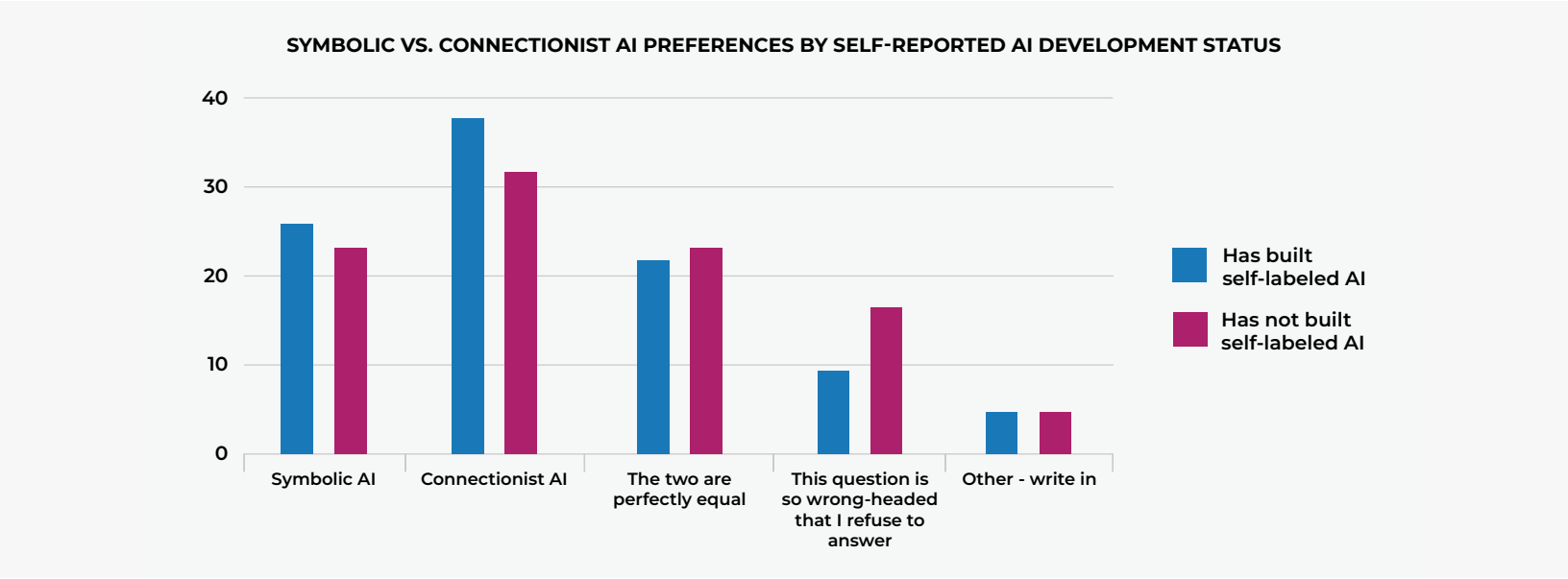
   We are more interested, however, in the persistent partisanship: Fewer respondents considered both paradigms "perfectly equal" (22.7%) than considered one better than the other. This may perhaps reflect:

   - A resurgence of expert systems, a subtype of symbolic AI, in domains like biomedicine (e.g., RDF ontologies).
   - (Indirectly) increasing availability of pre-trained connectionist models like GPT-3 that, in practice, feel more like simple functions than like neural networks (because weights are precalculated).
   - Growing complexity of purely symbolic systems — including enterprise software — that, because of their increasingly hard-to-grasp complexity, feel increasingly opaque and "black box."

2. The percent of respondents who rejected the question entirely is significant (11.8%). We interpret this through the pragmatism of software professionals: What in academic discourse remains somewhat dichotomized seems to implementers simply another set of different tools to be selected appropriately to a specific implementation task.

3. Experience building systems self-described as "AI" significantly impacts answers to this question:

Figure 4

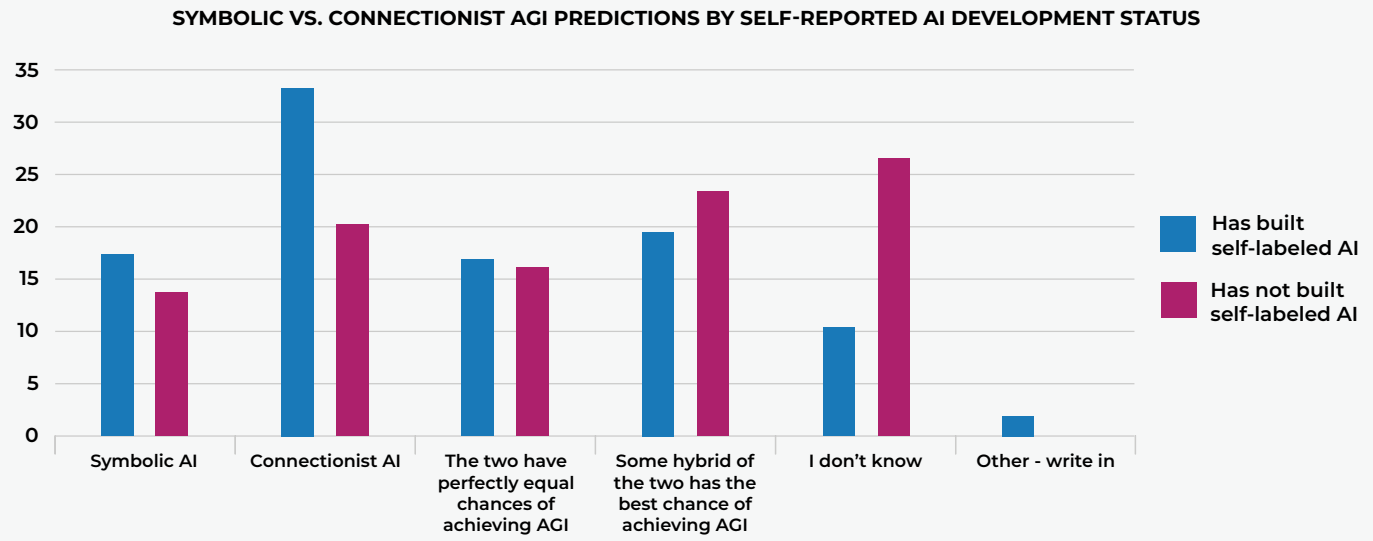


   The gap between those who have built and those who have not built self-described AI is larger for those who consider connectionist AI better than for those who consider symbolic AI better. We may weakly infer that connectionist AI is intrinsically superior, easier to use, or some combination as compared to symbolic AI.

4. In harmony with the last suggestion, we note that the corresponding difference between those who have and have not built self-described AI with respect to the plausibility of achieving AGI by connectionist vs. symbolic AI is also greater — in this case, with an even larger gap:

See Figure 5 on next page

Figure 5



We take this to reinforce our (and the industry's general) impression that connectionist approaches hold significantly more promise for AI in the judgment of relevantly experienced software professionals — at least in the present state of the art, even at the most fantastic levels.

USE OF COMMON AI-ADJACENT TECHNIQUES

Many functions converge without being labeled "AI"; many systems "learn" without being named "machine learning." In addition to AI-explicit techniques, we wanted to learn about the use of AI-adjacent techniques — other ways of training a system, inferring qualitatively from a system, or applying mathematical techniques to software development. So we asked three related questions:

*Have you ever built a non-trivial system that uses **only basic arithmetic** to make qualitative decisions? For example, a "scoring system" that generates scores via simple summation uses only basic arithmetic, but the system that does the summing is considered non-trivial if implementation involves significant engineering challenges (e.g., "big data").*

*Have you ever built a non-trivial system that uses **any mathematics more complex than basic arithmetic** to make qualitative decisions?*

*Have you ever built software that "**learns**" in any sense? For example, a keyword search UI that returns a list of suggested search terms presented in an order determined by any formula that involves previous search inputs counts as "learning" in some sense — even if the formula is no more complex than score(i) = sum(resultCount(searchInput_i...n)).*

Results (n=590, 588, and 589, respectively):

Table 1

USE OF COMMON AI-ADJACENT TECHNIQUES			
Respondent has built a non-trivial system that...	**Yes**	**No**	**Other**
Uses basic arithmetic	70.8% (n=418)	28.1% (n=166)	1.0% (n=6)
Uses mathematics more complex than basic arithmetic	59.4% (n=349)	40.0% (n=235)	0.7% (n=4)
Learns	63.5% (n=374)	35.5% (n=209)	1.0% (n=6)

Observations:

1. A significant majority of respondents have built systems that an external observer might consider "smart." Nearly three quarters (70.8%) have built software that uses some kind of basic arithmetic (e.g., summing) to make qualitative decisions, which is not surprising, but almost 60% (59.4%) have built software that uses some kind of higher-than-counting mathematics to qualitative effect.

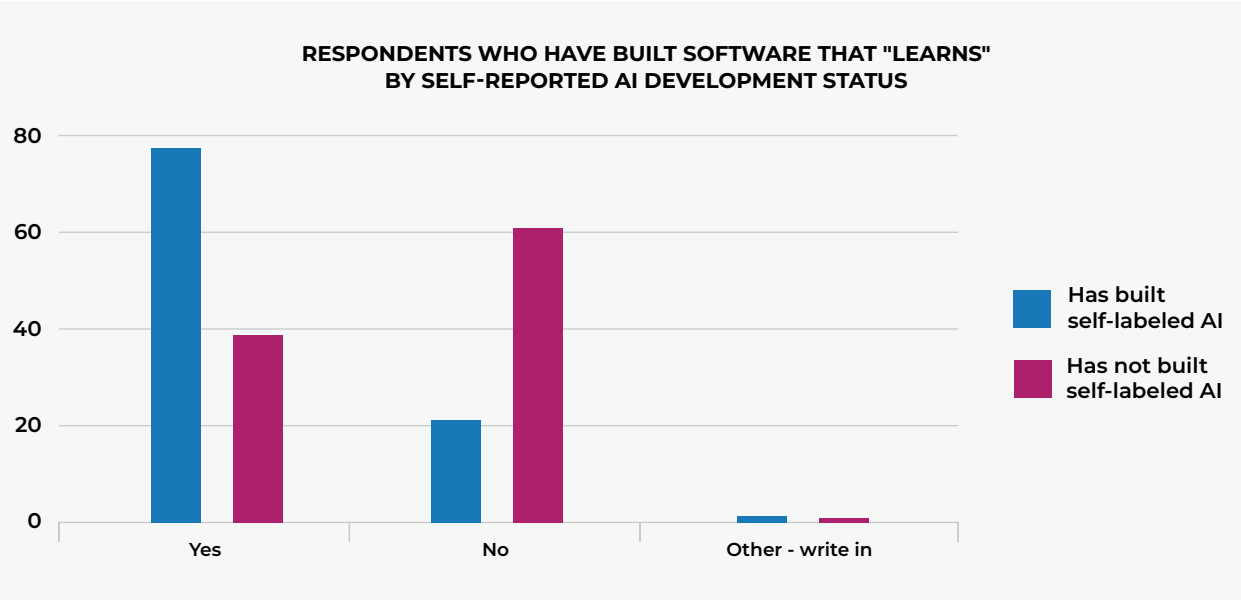
   In previous surveys, our response rates to higher-level mathematical questions have been low, but because of the present subject matter, in future iterations of this survey, we intend to ask which mathematical methods respondents have applied to qualitative decision-making.

   Based on responses to our question about machine learning techniques, we suppose a significant percent of these supra-arithmetical methods have been statistical, but we do not have a breakdown by mathematical method at present.

2. An even larger majority than "supra-arithmetical" responded that they have built software that learns in any sense (63.5% vs. 59.4%). From this we infer two claims:
   - Among software professionals, "learning" systems do not comprise an abnormal or highly segregated subdomain.
   - No higher mathematics than arithmetic are required to build a software system that "learns," as practitioners conceive learning.

3. The latter conclusion (b) further hints at the comparative vacuity of the concepts, machine learning and AI: Almost the same percent of respondents reported that they have built software that "learns" and have "built AI" (62.2%), but the overlap between "learns" and "built AI" cohorts was far from total. Only 78.4% of those who reported having built software that learns also reported having built a system they consider AI:

Figure 6



   We take this to infer that software professionals consider learning to be a much weaker concept than AI.

4. Again, the gap between those who have built AI and those who have not with respect to basic arithmetical vs. higher-mathematical software systems is smaller in the case of the former than the latter — but not colossally so.

   75.9% of respondents who have built non-trivial systems using only basic arithmetic to make qualitative decisions reported having built an AI system, while 66.8% of respondents who reported using higher-than-arithmetic mathematics to make qualitative decisions reported having built an AI system.

Research Target Two: AI Applications in Different Domains Related to Software Development

Motivations:

1. "Is AI for you?" *Yes*, says the CEO at the shareholder meeting; *yes*, says the researcher writing a grant application; and *yes*, says the pundit prognosticating; *maybe*, says the jaded software developer. But also — *maybe*, says anyone interested in solving problems they already have, as opposed to imagining and taking on new problems.

   Perhaps AI will not help a filesystem — or will it? Surely, we need neural networks for search result ranking, or does TF/IDF do exactly what we want? We wanted to find out in which domains software professionals are using AI and finding it useful.

2. Ever since Ludd, in practice, and Marx, in theory — and perhaps since Prometheus, in myth — rational agents have feared, or neutrally anticipated, or thrilled at machines' putative powers to take our jobs away. *Okay, but software developers are Turing machines' puppet-masters — and of course the puppets can do nothing without us puppet-masters, so our jobs are secure from AI.*

   Cue internal dialogue:

   "...Wait a minute. I debug my code... all the time. Therefore, my code does stuff I don't want it to do...all the time. Strings tangled or Pinocchio escaped? Both accounts equally explain the system's failure to do what I told it to. Then autocomplete and...hooray, I don't need to know method names anymore. Then IntelliSense-level autocomplete and...sweet, untyped languages start to feel typed because some computer is telling me I can't use a Double here.

   Wait, the compiler is rewriting my code for me? Wait, the speculative processor is executing commands my code doesn't actually compile to? Whoosh, GitHub is writing whole classes for me while I do little but tab through? ...Is my job actually safe...?"

   This is the sort of developer's mental spiral we also wanted to unpack in our survey.

3. In particular, because the notion of an "enterprise" encodes complexity that cannot easily be managed, enterprises have long combined hierarchical structures of information-hiding with mathematical methods of intellectual augmentation, whether the latter calculations are performed on silicon or on paper.

   Further, operations research is applied more naturally in the enterprise than in, say, pure IT companies, thanks to the former's unfettered complexity and diffuse focus. Operations research is where many optimization (convergence) techniques that undergird machine learning were first developed, and where unsupervised learning systems were already able to infer Newtonian dynamics from raw data in the 1970s.

PRESSURE TO USE AI IN THE ENTERPRISE

To begin, we considered the desire for AI in the enterprise. We did this by comparing the amount of pressure to use AI from within their organization reported by self-identified enterprise developers vs. self-identified web developers, where "type of developer" answers are accepted exclusively, for two reasons. First, widespread use of the web in modern enterprise development means that a developer who identifies as an enterprise developer (and NOT a web developer) is more likely to be involved in the tangle of business complexity proper to the business, rather than the specific technical complexity of web development that is not necessarily proper to the business. Second, these were the two largest cohorts of respondents by development type.

Based on our experience in enterprise consulting, where we have anecdotally noticed vague desire to "use AI" growing rapidly over the past two or three years, we phrased "desire for AI" in terms of "pressure to use AI" where "AI" is not further defined. We asked two questions about this pressure:

How often have you experienced pressure to "use AI" from within your organization?

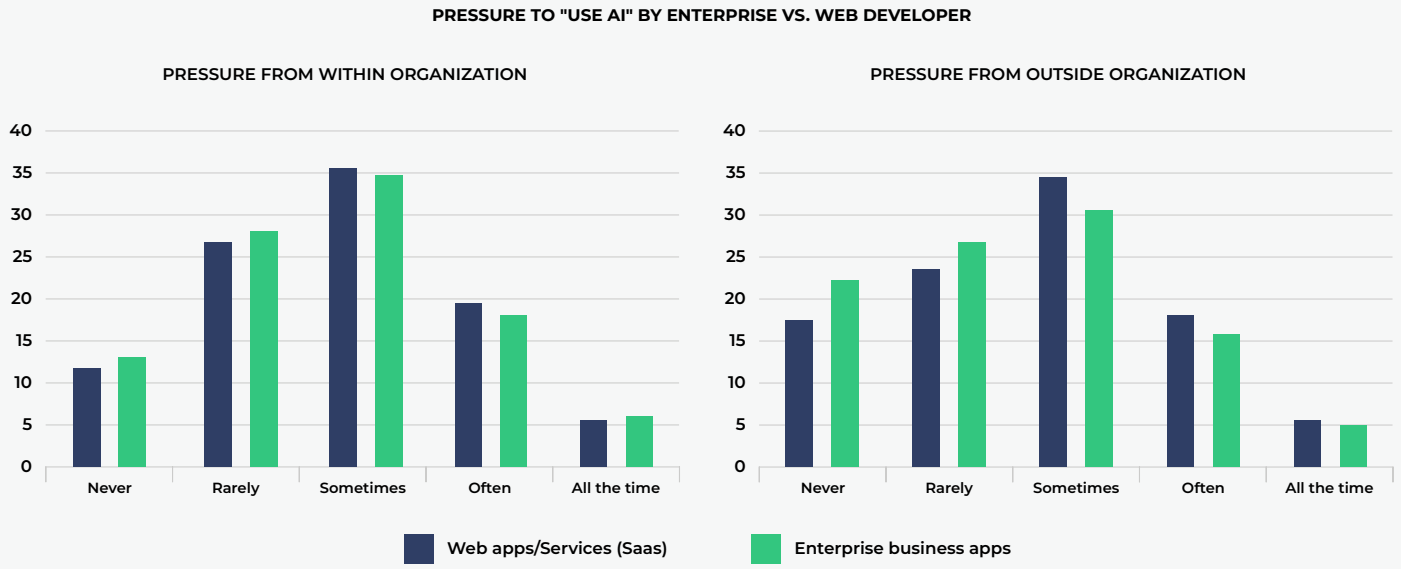
and

How often have you experienced pressure to "use AI" from outside your organization?

We bucketed respondents by "enterprise developer" and "web developer" as reported elsewhere in the survey.

Results (n=583 and 587):

Figure 7



Observations:

1. The enterprise is still behind the world at large in catching "AI fever." We infer this from the near identity of distribution of responses to intra-organizational AI pressure between the enterprise and web developer cohorts, contrasted with the significantly higher degree of extra-organizational AI pressure reported by web developers.

    That is, we take intra-organizational pressure to be mediated by specificity of job description (where neither web developer nor enterprise developer is intrinsically coupled with AI — unlike a data scientist, for example) in a way that extra-organizational pressure is not. We also assume that respondents feel extra-organizational pressure from the point of view of someone with a specific software career.

    Since web development skills are no more AI-adjacent than enterprise development skills, we take the difference in reported extra-organizational pressure to reflect the external career environment rather than self-driven career planning.

2. Nevertheless, enterprises do seem to be exerting noticeable internal pressure toward using AI. We infer this from the greater rate of extra-organizational AI pressures reported by enterprise developers vs. intra-organizational pressure: Only 12.8% of enterprise developers reported never experiencing intra-organizational AI pressure vs. 22.4% reporting never experiencing extra-organizational AI pressure.

    Similar differences hold across "sometimes" and "often" responses to intra- vs. extra-organizational pressures among enterprise developer respondents. Further, while intra-organizational AI pressure is significantly higher vs. extra-organizational AI pressure among enterprise developers, no significant difference between intra- and extra-organizational pressures were reported by web developers.

    We imagine this greater difference between intra- vs. extra-organizational AI pressures among enterprise software professionals reflects the force of enterprise planners who attempt to stay ahead of the industry "hype cycle" (which we take extra-organizational pressure to reflect).

USE OF AI AND LEARNING-RELATED TECHNIQUES IN THE ENTERPRISE

Second, we considered the use of AI and learning-related techniques by enterprise developers from most general to most specific levels, again in comparison with web developers. Overall, enterprise developers and web developers were equally likely to report having built any system that might be considered AI at all — with remarkable precision, 64.3% vs. 64%, respectively.

With respect to specific techniques, however, enterprise developers were slightly more likely to use more sophisticated learning techniques:

Table 2

AI AND LEARNING-RELATED TECHNIQUES USED BY DEVELOPER TYPE		
Method	**Enterprise**	**Web**
Linear regression, one variable	55.9%	60.1%
Linear regression, multiple variables	58.5%	57.7%
Logistic regression	42.4%	37.9%
Regularization	27.9%	24.6%
Any neural network	45.0%	41.1%
Support vector machines	28.8%	24.9%
Principal component analysis	24.0%	18.9%
Collaborative filtering	15.3%	14.5%
Anomaly detection	31.9%	31.1%
Jaccard similarity	9.2%	6.5%
Any Bayesian method	19.2%	19.5%
Linear programming	34.1%	39.9%
Genetic algorithms	18.8%	21.0%
Backpropagation	21.8%	18.3%
Gradient descent	23.1%	22.5%
Generative adversarial network	7.9%	9.2%
Convolutional neural network	19.7%	22.5%
Transformers	16.6%	17.2%
Other	0.8%	3.8%

Most of the techniques that web developers are (slightly) more likely to have used than enterprise developers are less narrowly AI and more generally optimizing, such as single-variable linear regression and linear programming, although web developers are (again, slightly) more likely to use genetic algorithms and convolutional neural networks.

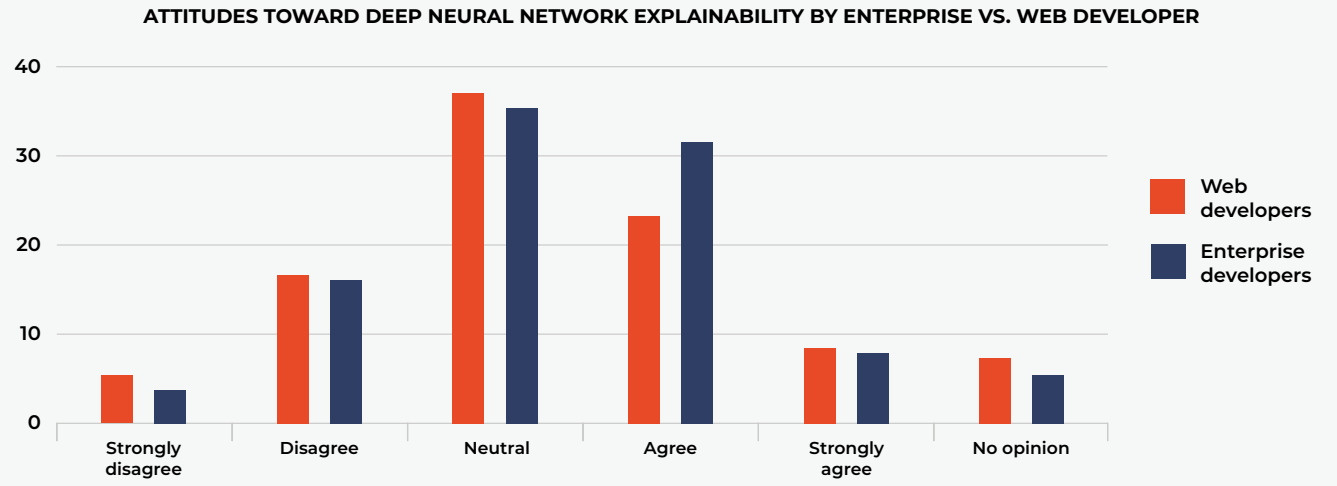
These latter differences are enough to be noise; however, we infer nothing from them. Our general conclusion about the somewhat-more-AI-ish work done by enterprise developers depends on the overwhelming preponderance of slightly more common use of learning-related techniques across a wide range of techniques.

ATTITUDES TOWARD AI IN THE ENTERPRISE

Enterprise developers differ from web developers more with respect to explainability (note the "agree" response difference, especially) than with respect to the use of specific techniques:

See Figure 8 on next page

Figure 8

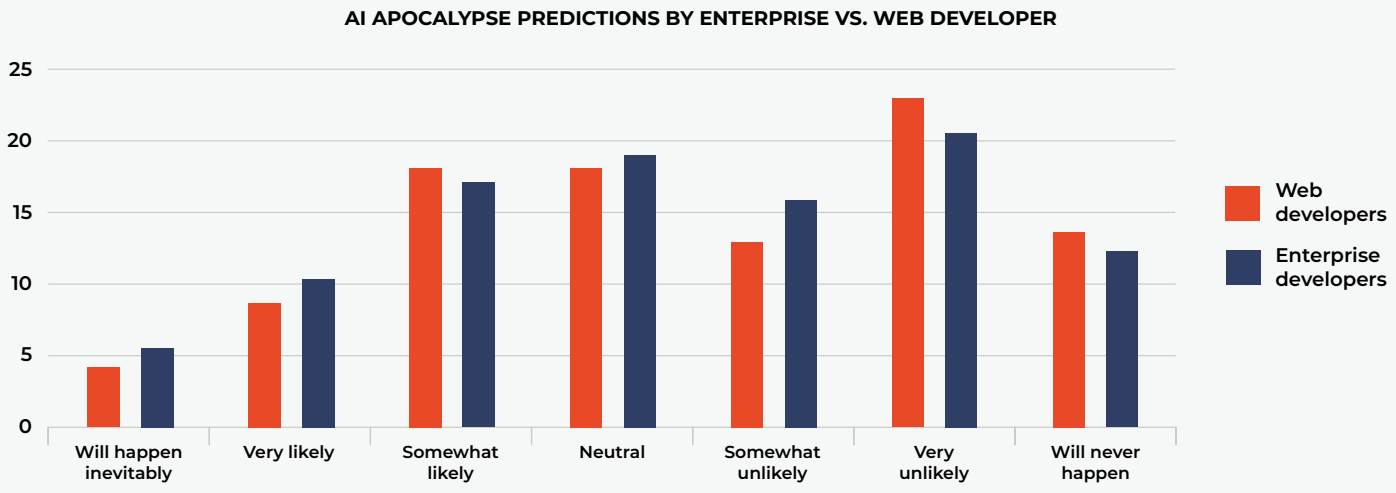


We interpret this as a reflection of the huge degree of tacitness and implicitness of knowledge within the enterprise. That is, in our experience, a great deal of enterprise development involves excavating and formalizing systems that were, in practice, operating in a precise way that was, however, never made explicit or formal enough to turn into code without considerable digging. Contrast this among web developers, for instance: the detailed specifications available from the W3C and the open-source code of the major JavaScript and browser rendering engines.

So we take general lack of explicitness to be a particular pain point among specifically enterprise developers, with concomitantly higher suspicion of black boxes.

Enterprise developers are also more pessimistic about AI *in extremis*:

Figure 9



Although 15.7% of enterprise developers consider the AI apocalypse very likely or inevitable (vs. 13.3% of web developers), with corresponding reverse differences in "very unlikely" and "never happen" responses, we are somewhat surprised that both the difference and absolute numbers are not higher.

And our impressions are echoed by enterprise developers interviewed separately. Perhaps the difference in duct-tapedness between the enterprise and the web at large is not as great as harrowed enterprise developers take it to be. (For the "AI apocalypse" question in general, see discussion below.)

Research Target Three: Attitudes Toward Current and Future AI

Motivations:

1. AI practice has fallen far behind AI theory and prediction several times, but AI practice has resuscitated concepts of intelligence previously abandoned, apparently prematurely, many times.

   Perhaps in 1952, we were just a few years from an Ashby brain; perhaps GPT-3 is "merely" the working out of an insight already encoded in Rosenblatt's 1958 perceptron and Marcus' hybrid "algebraic mind" is enjoying its own panels at major AI conferences two decades after publication. The reality seems to be out of step with the dream, so we wanted to understand how actual creators' dreams fit with the reality now being made.

2. Public conversation about AI is dominated by futurists who rarely write code (anymore) and often fantasize that Moore's Law is an ironclad rule of techno-nature. Actual implementation of AI is done by people who see (and suffer) every day what a house of cards all software systems really are.

   Nick Bostrom fears the AI apocalypse from a machine that is too smart to let us live; we developers fear an AI apocalypse from a machine that somebody wrote `free(nuclearAttackWarningPointer)` a little prematurely while tipsy on New Year's Eve — or so we imagine from our own experience. But we wanted to see whether the broader software professional community agrees.

3. AI has recently been democratized in the form of frameworks with high-level APIs, open-source models pre-trained on massive datasets, cheap GPUs with mature parallelization platforms, dedicated tensor-processing hardware, cheap FPGAs, etc. Many more people can now do "AI" things.

   The attitudes of the developer population at large toward AI will shape the future of AI far more than did the attitude of the same population 20 years ago, and we wanted to know what these attitudes are.

ATTITUDES TOWARD MAJOR AI PARADIGMS

For general attitudes toward connectionist vs. symbolic AI, see discussion under "Research Target One" above, with segmentation by self-reported role in building AI systems. In sum: Connectionism is more likely to be considered better by software professionals and is still more likely to be considered better by software professionals who reported having built software they consider to be AI in any sense. Here, we add some additional findings by less technical segment.

Significant differences in attitudes toward symbolic vs. connectionist AI obtained between senior (>5 years' experience as a software professional) vs. junior (≤5 years' experience) respondents are shown in Table 3.

Observations:

1. Senior respondents are both more sanguine toward connectionist AI and more skeptical of the question. Like the impact of AI-building experience on these preferences discussed above, the stronger senior preference of connectionist AI, again, suggests more connectionist AI promise.

   We imagine this for two reasons:

   - Senior respondents overall have a better sense of software development in practice.
   - Connectionist approaches are the "modern" approach to AI in practice, which might suggest that junior respondents would prefer it more strongly if preference were unaffected by learning experience.

   On the other hand, neural networks have been dominating for more than five years, so the second reason is probably irrelevant.

Table 3

ATTITUDES TOWARD AI PARADIGMS BY EXPERIENCE LEVEL

Response	Senior	Junior
Symbolic AI	23.7%	30.7%
Connectionist AI	36.2%	31.4%
The two are perfectly equal	21.1%	27.7%
This question is so wrong-headed that I refuse to answer	13.1%	7.3%
Other	5.9%	2.9%

2. The comparatively greater preference for symbolic AI among junior respondents is more interesting. A quick survey of AI undergraduate curricula uncovers far less about, for example, expert systems than neural networks, especially if one excludes computer vision as specifically applied to self-driving cars. And yet junior respondents are significantly more likely than senior respondents to consider symbolic approaches better.

   This may reflect the resurgence of symbolic AI and its integration with connectionism (mentioned above), but it may also reflect comparative ignorance of the field in practice: Junior forays into AI (e.g., simple chatbots) are more likely to involve explicit decision trees.

ATTITUDES TOWARD ARTIFICIAL GENERAL INTELLIGENCE

The differences between senior and junior attitudes toward symbolic vs. connectionist AI discussed above obtain, though more weakly, in judgments of the likelihood that one approach or another will produce artificial general intelligence (AGI). More interesting is the significant difference between indifference and attitude toward hybrid approaches:

Table 4

PREDICTIONS FOR SYMBOLIC VS. CONNECTIONIST ACHIEVEMENT OF AGI BY EXPERIENCE LEVEL

Response	Senior	Junior
Symbolic AI	15.2%	18.2%
Connectionist AI	28.3%	27.0%
The two have perfectly equal chances of achieving AGI	13.4%	25.5%
Some hybrid of the two has the best chance of achieving AGI	24.9%	16.1%
I don't know	16.7%	11.7%
Other	1.5%	1.5%

Junior respondents are twice as likely as senior respondents to have effectively no judgment about which paradigm is better suited to AGI (25.5% junior vs. 13.4% senior). But senior respondents are, again, half as likely (24.9% senior vs. 16.1% junior) to suppose that a connectionist-symbolic hybrid is more likely to achieve AGI. Further, among senior respondents, the hybrid AGI predictors are almost as many as the connectionist AGI predictors (24.9% hybrid vs. 28.3% connectionist). However, this similarity of hybrid vs. connectionist AGI prediction does not obtain among those who have built AI systems. Among AI-experienced respondents, specifically, the connectionist AGI prediction is far stronger than any other contender, including hybrid (see Figure 5). Perhaps this simply reflects a craftsperson's pride in their (likely connectionist, as discussed above) work.

More surprising: Symbolic AGI is seen as a more likely AGI contender among AI-experienced respondents than among AI-inexperienced respondents, and AI-inexperienced respondents are more likely to guess that a hybrid approach is more likely to result in AGI. The positive impact of AI-building experience on the bifurcation between the two paradigms is surprising, and at the moment, our only explanation is that "some hybrid approach" is seen by AI-experienced respondents as effectively (and therefore uselessly) agnostic.

ATTITUDES TOWARD THE AI APOCALYPSE

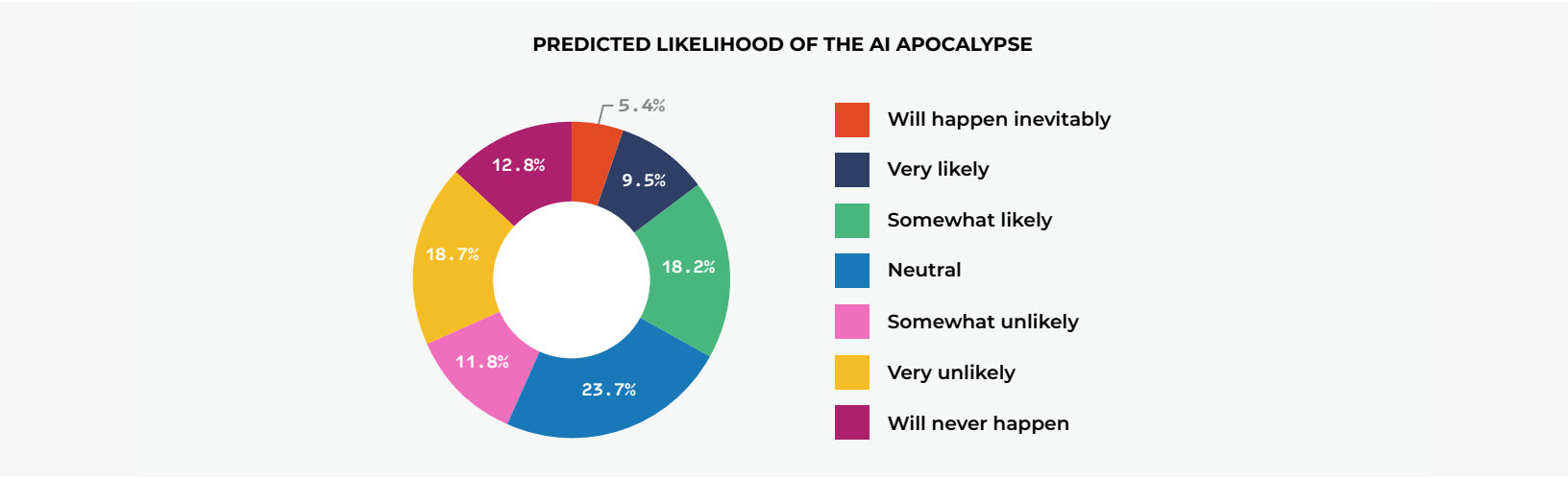
The question, "how likely is it that AI will take over the world and destroy humanity," may seem *prima facie* silly. But we think it is worth considering, especially by developers, for two reasons. First, major academic and industry researchers, with significant funding and broad public audiences, continue to express fear of the "AI apocalypse." But none of those people are actually writing the Allied Mastercomputer's code (using "Allied Mastercomputer" [AM] as shorthand for any AI agent of global catastrophe).

Second, developers already have a precise and well-articulated sense of how software goes wrong — the pain of figuring out what your code is doing that you didn't intend it to is no Hollywood imagining but rather the everyday debugging slog. Developers are, therefore, more immediately responsible for real-world AI systems and more sensitive to the off-the-rails-ness of software than major public interlocutors on the AI apocalypse.

So we asked: *How likely is it that AI will take over the world and destroy humanity?*

Results (n=578):

Figure 10

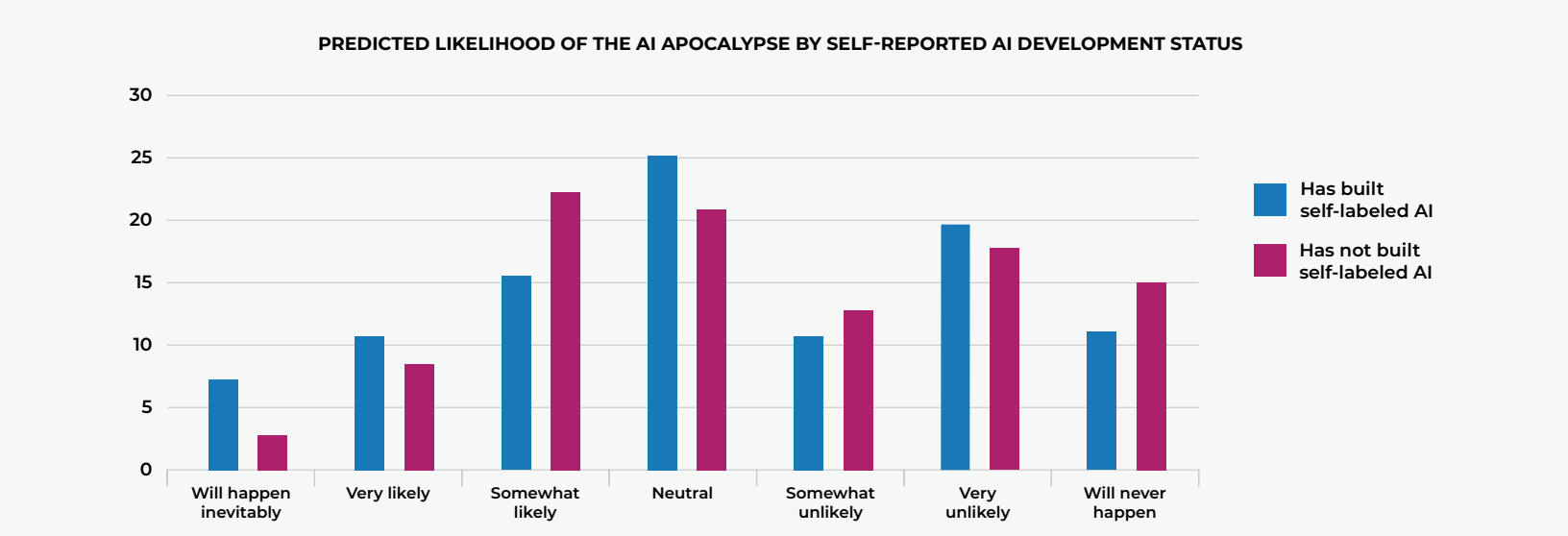


Observations:

1. The picture is overall a little rosy — only a third of respondents predicted the AI apocalypse at any confidence level, while 43.3% predicted no AI apocalypse (at any confidence level) at all. Of course, the fact that our survey respondents are those who will be writing AM's code complicates our interpretation of the response: The people making the catastrophic system presumably do not intend the system's operation to be catastrophic, and yet they are in the best position to foresee how the system's design might result in catastrophe.

2. Respondents who also reported having built AI systems, however, are a little more pessimistic than those who self-reportedly have not built AI:

Figure 11

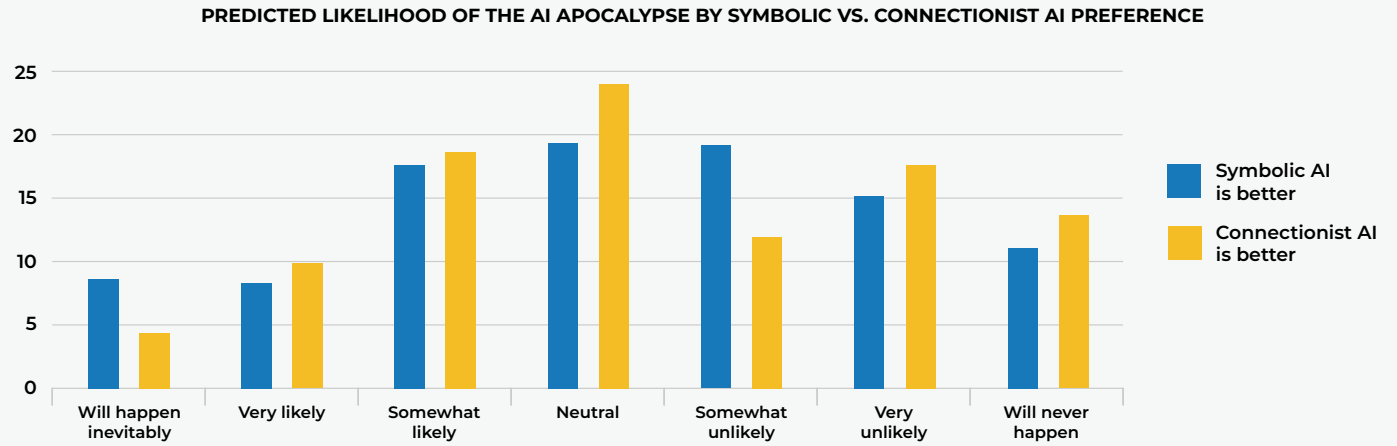


Whereas only 2.9% of those who have not built any AI systems think the AI apocalypse is inevitable, 7% of those who have built AI systems think it is inevitable. And whereas 15.1% of those who have not built AI systems think that the AI apocalypse will never happen, only 11.2% of those who have built AI systems think that it will never happen.

3. Respondents who prefer symbolic AI are also more pessimistic (and less neutral) about the AI apocalypse:

See Figure 12 on next page

Figure 12



This difference is not a reflection of greater experience with connectionist AI: The more-AI-experienced both generally prefer connectionist AI and also think the AI apocalypse is more likely. We find this result intriguing because one might imagine — as pessimistic futurists do — that the black box character of deep neural nets might increase fear of the AI apocalypse by way of general fear of the unknown and more specific difficulty of corrective tuning overrides (how can anyone possibly understand 175 billion GPT-3 parameters?).

We conjecture that respondents who prefer symbolic AI may be more afraid of the AI apocalypse since they may tend to fear underspecified optimization functions more than those who prefer connectionist AI, because those who prefer symbolic AI may be more suspicious of algebraic systems' high sensitivity to small errors. Every developer knows that an optimization aimed ever-so-slightly imprecisely results in a vast distance between intended and actual result, but a mere black box is neutral to desired or undesired effects.

Further Research

This was the first survey on an AI-related topic in our newest, more technical research series. In future surveys, we intend to ask some similar questions in order to measure trends over time in software professionals' use of and attitudes toward AI, especially in the enterprise. In this survey, we gathered data on many topics not discussed in this write-up, including:

- Differences in judgments of what "counts" as AI among developers, sales, marketing, and the general public
- Use of various levels of computer-assisted software development (from simple by-token autocomplete to GitHub Copilot)
- Use of natural language processing (NLP) in and out of the enterprise
- Software professionals' opinions about Turing's "imitation game" as a test for AGI
- Developer involvement in specific ML tasks (e.g., hyperparameter tuning) and collaboration with data science specialists
- Importance of discrete vs. continuous mathematical methods for AI
- Specific tools and frameworks used for AI or machine learning; opinions about the best use cases for AI in the enterprise

We intend to analyze this data in future publications. If you are interested in this data for research purposes, please contact publications@dzone.com and we may be able to share, depending on your research proposal.

John Esposito, PhD, Technical Architect at 6st Technologies

@subwayprophet on GitHub | @johnesposito on DZone

John Esposito works as technical architect at 6st Technologies, teaches undergrads whenever they will listen, and moonlights as research analyst at DZone.com. He wrote his first C in junior high and is finally starting to understand JavaScript NaN%. When he isn't annoyed at code written by his past self, John hangs out with his wife and cats Gilgamesh and Behemoth, who look and act like their names.

CONTRIBUTOR INSIGHTS

AI and Explainability

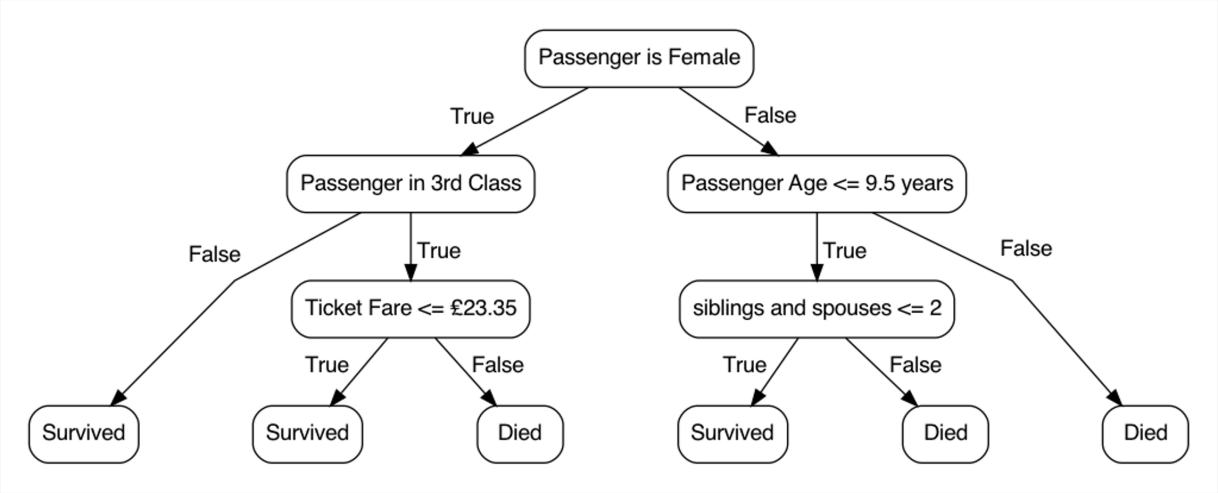
Discover Why Your Models Make Their Decisions

By Hannah Morgan, Lead Data Scientist at Finastra

Explainable artificial intelligence, sometimes referred to as XAI, is exactly what it sounds like — explaining how and why a machine learning model makes a prediction. While models are usually classified as either "black box" or "glass box," it isn't quite as simple as that; there are some that fall somewhere in between. Some models are more naturally transparent than others, and their uses depend on the application.

Naturally transparent models — also called "white box," "clear box," or "glass box" models — are those that can be easily interpreted and often even diagrammed out. Their transparency is fundamental to the algorithm's math, so no additional techniques are needed to interpret the results. As an example, below is a diagram representing a decision tree algorithm predicting if a passenger survived the Titanic. By looking at the figure, we see that the model is picking up on the "women and children first" policy that was enforced during the tragedy.

Figure 1



Black-box models, on the other hand, make predictions that can't be so easily interpreted. It is often claimed that there's no way of knowing how and why the model reached its decision. Of course, how a model reached its decision is just fancy arithmetic — features are input, matrices are multiplied, and an output is returned. However, the models on their own don't tell you why they made their decision. For that, we need additional tools.

Situation Assessment

The transparency level of a model must match its application, a necessary consideration when designing your experiment. Some applications require complete transparency of a model, usually for legal reasons. The most obvious example of this is models for actuarial purposes. In these circumstances, all machine learning must be completely transparent in order to justify any decisions. A glass-box model is usually a requirement for any model based on data with sensitive personal attributes.

While having this complete transparency sounds like a wonderful thing, it's not always necessary or even desirable. Often, black-box models, such as neural networks, perform significantly better at their tasks. It isn't important how a language translation was performed, or Google search results were returned, as long as they are the most accurate and pertinent.

There are cases, too, where only some transparency is needed, such as self-driving cars. It is very important to know what a car was thinking when it made a mistake and what might be some of the possible causes. But it isn't necessary to consider every little detail — the complexity of the input data makes this almost impossible in any case. However, if problems arise, it is important to know that the car didn't see a stop sign or thought a plastic bag was a pedestrian. This helps target mitigation strategies for future iterations of the model.

Let's take a look at examples of both types of models.

Naturally Transparent Models

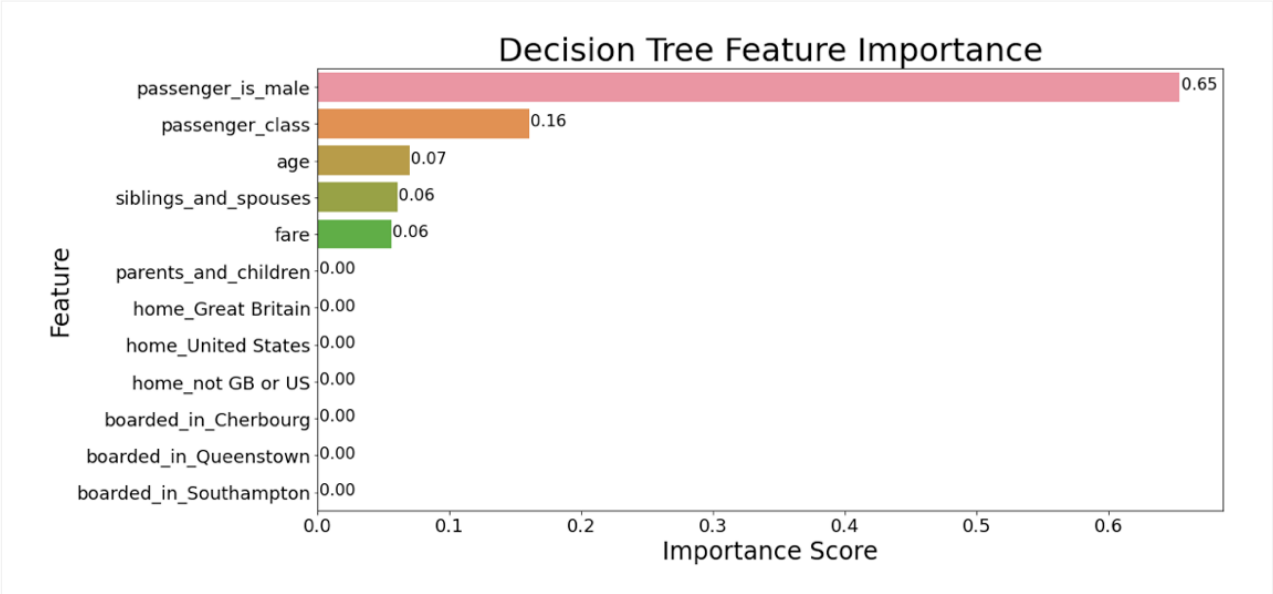
The models often taught to students studying statistics or machine learning are those that are most naturally transparent. In some sense, the underlying transparency of the model is what makes it the easiest concept to grasp.

Linear models, such as linear regression, are perhaps the most transparent models that exist. The model spits out a simple equation, often of the form y = **M x + b**, where M, x, and b are matrices. The coefficients — the matrix **M** — are simply the importances of each feature.

Another extremely simple and transparent model is the decision tree. You can think of a decision tree as a game of 20 questions, only you get to pick the number of yes/no questions. This algorithm has the advantage of being mapped out to make the interpretation extremely clear. Additionally, the model will return the "feature importance" for each column so you can see what is impacting the decision the most.

As an example, we will return to the Titanic decision tree model we used in the introduction. That figure diagrams the algorithms decision paths, while the bar plot below shows the feature importances. Since the model had a maximum depth of only 3, most of the data features weren't used and thus didn't have an importance score.

Figure 2

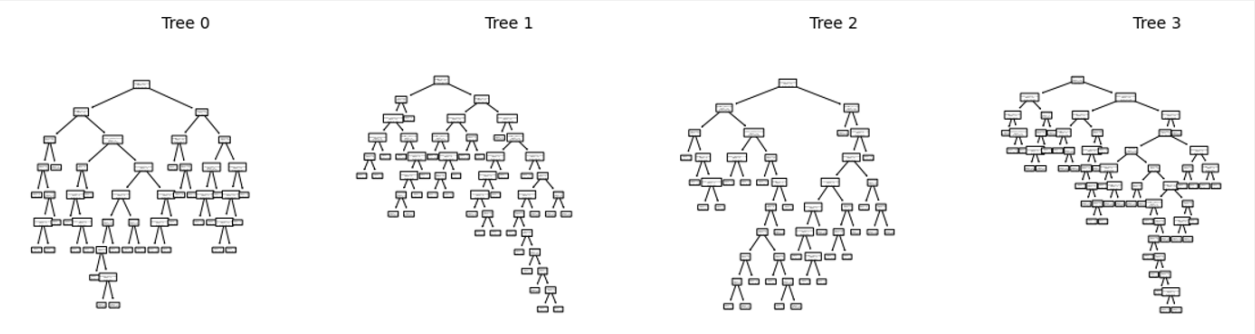


Decision trees by themselves generally don't perform very well, which is why there are ensemble methods such as random forests. These algorithms use many trees, each of which makes a prediction. They then all "vote" to get a final prediction. These algorithms have the advantage of being more accurate than random forests while still having an easily interpretable method. However, the transparency of the model is a little clouded since these models often have hundreds of trees.

Continuing the Titanic example, Figure 3 shows a random forest fitted with 4 trees. Note how even with this small number, it becomes a bit harder to interpret the results.

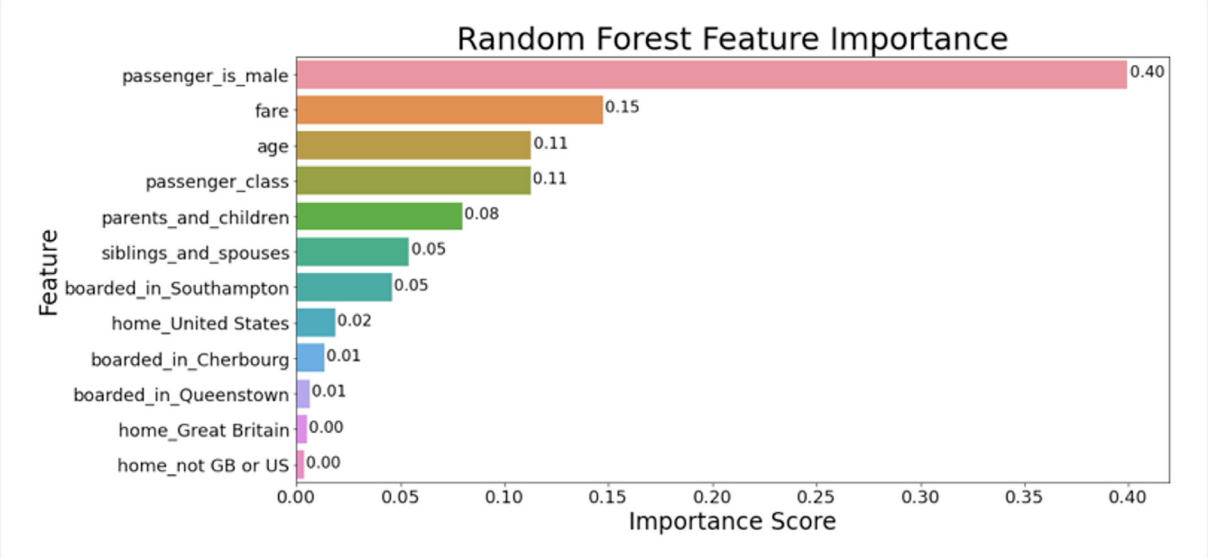
See Figure 3 on next page

Figure 3



Additionally, random forests have the benefit of feature importance scores as well.

Figure 4



Explainability Methods

Interpretability is a fast-growing field in machine learning, leading to many methods, some of which are soon outdated. Explainability methods can be local — they explain a single or global prediction. They explain the overall model. We will go through some of the most popular methods currently available, but keep in mind, this isn't an exhaustive list.

LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS

First published in 2016 by Ribeiro, et al., Local Interpretable Model-Agnostic Explanations (LIME) is a method used to explain the predictions of any classification model. It creates explainability around single predictions by perturbing the data and computing a simpler interpretable model. It is pretty computationally efficient but has the pitfall of only explaining individual predictions instead of the model as a whole. In 2019, Zafar and Khan proposed an improvement on the LIME algorithm, DLIME (where the D stands for deterministic). Instead of using random perturbations, this method uses clustering methods to select new instances.

SHAPLEY ADDITIVE EXPLANATIONS

The Shapley value is a concept based in game theory, first developed by Lloyd Shapley in 1950s, but wasn't applied to machine learning until 2017 by Lundberg and Lee. Since the original model was based in game theory, it had to be adapted to artificial intelligence. Thus, they used an outcome of the model as "the game" and the features of the data as "the players."

The SHAPley Additive exPlanations (SHAP) algorithm computes the contribution to "the game" (i.e., the prediction) for all possible permutations of features, which creates feature importances for that single prediction. By examining so many permutations, it is significantly more costly than the LIME algorithm. In addition to examining the distributions of individual features for observations, SHAP also provides global feature importances.

The left-hand plot below is a bar chart showing the mean SHAP values for a model, thus giving a global feature importance representation. The right-hand plot shows the distribution of SHAP values for all data points.

Figure 5

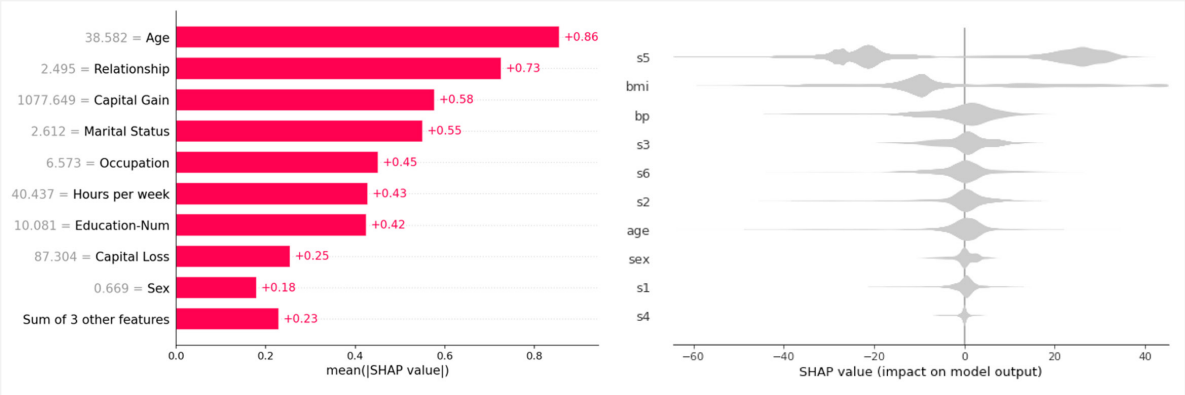
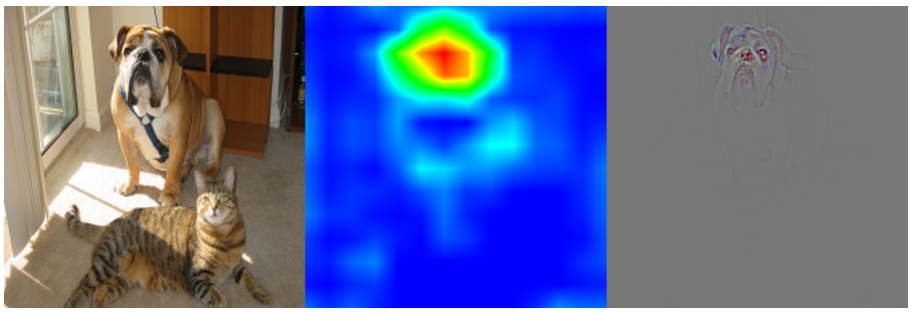


Image source: "Global bar plot," SHAP

GRADIENT-WEIGHTED CLASS ACTIVATION MAPPING

Proposed in 2019 by Selvaraju, et al. , Gradient-weighted Class Activation Mapping (Grad-CAM) is a technique for producing visual explanations for Convolutional Neural Network (CNN)-based models by highlighting important regions of an image. For instance, if a model classifies an image by identifying the animal, Grad-CAM will produce a heatmap on top of the image, showing what is most important in its decision. Since Grad-CAM is a generalization of CAM, it can be used on most kinds of CNN-based models, including image classification, image captioning, or visual question answering. Here is an example of the output for Grad-CAM identifying the word "boxer" in the left-most image.

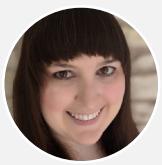
Figure 6



For more details, check out the example on GitHub.

Conclusion

Explainable AI has been given a lot of attention lately over concern around black-box models. Black-box models needn't always be concerning, but when they are, there are many mitigation strategies; some involve avoiding them altogether, while others utilize other algorithms to help explain predictions. It is important to match both the complexity and the interpretability requirements to your use case. Ask colleagues and potential end users for any requirements or preferences. There's nothing worse than having to build a new model from scratch! Hopefully, you now have a few more methods to add to your toolkit.



Hannah Morgan, Lead Data Scientist at Finastra

@hmorgan on DZone | @hannahmorgandatascientist on LinkedIn

Hannah Morgan is the Lead Data Scientist for Finastra's Innovation Team. After receiving a BS in Mechanical Engineering from Caltech and an MS Aerospace Engineering from UT Austin, she switched gears (pun intended) to a career in data science and feels she has truly found her calling. She lives in Austin with her husband, Madame Moose the dalmatian, and Mabel the kitty cat.

KATANA GRAPH

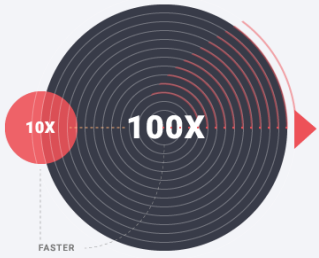
KATANA GRAPH'S

Graph Intelligence Platform

A Revolutionary Approach to Augment Fraud Detection and Prevention.

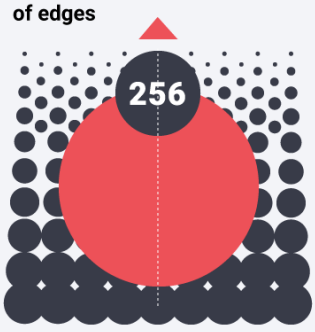
SPEED

Reduces time to insight by **10x - 100x**



SCALE

Scales to **100s of machines & graphs** with **100s of billions of edges**



CLOUD NATIVE

Kubernetes cluster: **GCP, AWS,** and **Azure**

DATA SCIENCE FRIENDLY

Python Jupyter interface & **open ecosystem**

Visit **katanagraph.com** for more information

PARTNER OPINION

Augmenting Fraud Detection and Prevention With Graph Intelligence

By the Katana Graph Team

Traditional fraud detection and prevention solutions lack accuracy and robustness, especially when dealing with large data volumes and sophisticated fraud scenarios. As the world becomes more digital, attackers constantly learn new ways to commit fraudulent activities. These activities, more often than not, cost organizations millions in losses.

Protecting against financial fraud has shown to be an extremely difficult challenge due to:

- A large and ever-growing volume of transaction data from a massively imbalanced, highly skewed nature
- The diverse nature of fraudulent activities and constantly changing fraud patterns
- The need for (near) real-time analytics of financial activities and thoroughly identifying fraudulent ones while minimizing false positives

Rule-based solutions against fraud require significant resources for manual detection and review. They are time consuming, complex, and limited in scope. Additionally, the rule-based algorithms fail to recognize the hidden patterns and are unable to adapt to fraudsters' changing tactics.

Machine learning (ML) tools are well suited for automated fraud detection and can evolve as they analyze additional data. Traditional ML algorithms, however, rely solely on transaction-level features and cannot capture multi-dimensional data. They suffer from laborious feature generation processes and high false positives, especially against complex and evolving fraud scenarios.

Leveraging Graph-Based Analytics and AI to Augment an Existing ML Pipeline

Graph intelligence presents a new frontier of advanced analytics to augment the existing fraud detection applications: It introduces a new approach to extracting and applying features to training and real-time inference to enhance accuracy.

Graph intelligence improves AI/ML models for several reasons:

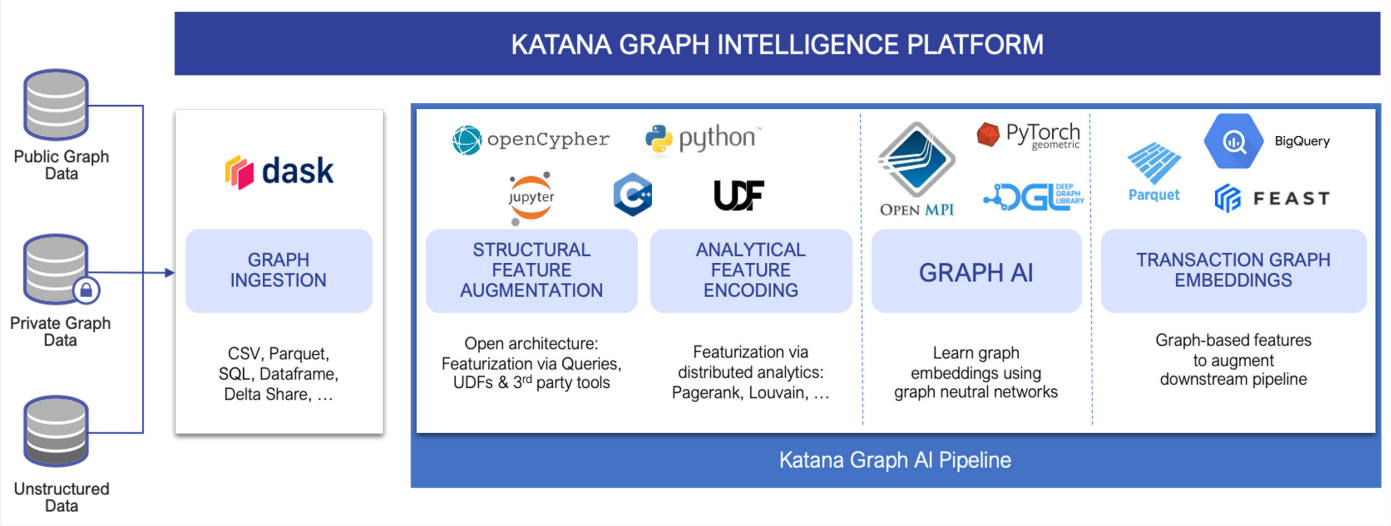
- It enables representation learning and eases the process of feature engineering.
- It captures the context of data by linking together multiple silos of data.
- It learns from relationships between important entities and encodes discriminating features that otherwise would not be captured.

In particular, graph neural networks (GNNs) learn the distribution of benign transaction activities. Fraud manifests are represented as deviations from the normal transaction patterns and distributions.

Katana Graph Intelligence Platform is a revolutionary step to democratizing the use of native graph analytics and AI in a highly distributed manner. It offers an easy-to-use, end-to-end pipeline experience for training and inference of GNNs at scale.

Using the distributed Katana Graph engine, we ingest financial and translation data from multiple sources and convert them to a transaction graph with node entities such as accounts, devices, and more. We then use the knowledge of past transactions to train our GNN model at scale and obtain embeddings for the graph node entities via inference. The graph embeddings are stored in a feature store and updated periodically as the transaction graph evolves. The graph embeddings are leveraged as initial features to train our AI/ML application and apply it in real time.

Figure 1



Conclusion

The Katana Graph Intelligence Platform handles customer identity and financial transaction graphs at terabytes of scale while executing complex graph algorithms with native support for graph analytics and GNNs. Using graph embedding features to augment a customer's productional AI-based fraud detection model has helped achieve about 11 percent higher PR-AUC accuracy and roughly 2 percent higher ROC-AUC accuracy.

CONTRIBUTOR INSIGHTS

Guide to Enterprise AI Platform Selection

Assessing Business Needs and Build vs. Buy

By Thomas Jardinet, IT Architect at Manpower

The use of artificial intelligence (AI) in companies is becoming more and more widespread, possibly even trending toward industrialization. Whether this is done on the basis of an existing data science platform or not, using all-in-one tools or not, or hosting data in the cloud or on-premise, there is a large number of software solutions available and, therefore, choices. This profusion of choices should not make us forget the raison d'être of any IT solution! Between those who promise you the moon and those who want the ultimate (and unfeasible) solution, you must stick to your needs more strongly than ever.

Part I: Assessing AI and Business Needs

DON'T BUY INTO BUZZWORDS!

The first thing to know about assessing AI is not to buy into buzzwords. You know, terms like big data, IoT, blockchain, and so many others where "the revolution expected" was not the reality. I've seen several instances where IT teams were told that they had to implement a specific new technology without thinking about the business need, defined here as the needs of the team and organization. So more than ever, focusing on the need is the very first step to success. Sometimes, AI is used for cases that can be solved by a simple "if-then-else" statement, when AI is actually most useful for problems that are difficult to solve by simple algorithms.

WHAT IS THE NEED?

Of course, this means asking why, and more importantly, what objective you are trying to achieve. Often, when needs are formulated by management, they are not necessarily complete. For example, if you are asked to set up an AI platform for a company, and the company's shareholders are asking for profits to double next year, you need to take that into account. You must be aware of the company's objectives, not only the objectives from your management, but also the organization's needs and its consequences.

But let's get back to the business needs. Of course, it is necessary to make the needs explicit, but it is always a good idea to master the use cases that have already been identified. This requires not just competitive intelligence (has my competitor implemented a relevant use case?) but also meeting vendors, visiting trade shows, and, of course, knowing your company's processes.

WHICH AI USE CASES?

AI use cases are endless, but some are relatively recurrent. Here are some that come up often across multiple industries:

- Marketing automation and definition
- Sales forecasting, lead generation, and analytics-based training
- AI in fraud detection (but can be achieved, at least partly, by CEP platforms)
- Customization of services
- Inventory management
- Administrative tasks such as automated mail, file processing, and decision support
- Decision automation (especially in legal and insurance)
- Predictive maintenance

WHAT ARE SOME ANTI-AI USE CASES?

When wondering whether a use case should or should not use AI, it is worth asking whether a use case should be computerized. The big questions to ask are:

- What are the consequences if an AI solution is wrong?
- What are the implications if an AI solution suffers from bias?
- Can a decision made by an AI project have legal ramifications?
- Does it risk dehumanizing the customer relationship?
- Will it bring real help in a use case where a human remains indispensable?

Part II: Build vs. Buy

When wondering whether to build a platform in house or buy one externally, you need to answer a few more questions, starting with, "Is your need very specific or small?" If your answer to that is "no," then you should be ready to buy! Here is a more extended checklist:

- ☐ Compare the business plan between build and buy
- ☐ If your need is a little specific, does the market contain AI solutions for it?
- ☐ Are there already solutions proposed by vendors for your use case?
- ☐ Is this vendor at significant risk of failing within four years?
- ☐ Could you obliterate the competition with your ideas and own way to use AI?
- ☐ Do you have a critical need that requires you to be fully independent of a vendor?

Part III: AI Enterprise Platforms

AI CAPACITIES CHECKLIST

Here is a list of capacities you must look into and the needs that an AI platform should fulfill:

- ☐ Data integration
- ☐ Data governance
- ☐ Experimentation and development
- ☐ Deployment and monitoring
- ☐ Intelligence engine (ML programs, libraries, etc.)
- ☐ Optimization capacities
- ☐ Collaboration capacities
- ☐ Visualization

LIST OF VENDOR TYPES

There are many vendors in the market, so it's up to you to determine your needs. Here are two general categories of vendors you will encounter and some key differences between them:

	Generic, pure AI platform	Cloud providers' solutions
Completion of offer	Often complete and generally derived from data science; very knowledgeable about the subject.	They have comprehensive, high-quality offerings.
Data integration	They have greater facilities to manage the integration of data from the outside.	Integrate well with their other cloud services. But integrating data that does not come from their own cloud offering is more complex, making multi- and hybrid cloud patterns a bit more difficult to implement.

Table continues on next page

	Generic, pure AI platform	Cloud providers' solutions
Type of target user	This often covers less experienced developers and citizen users.	They are perhaps more oriented toward experienced developers and not citizen users.
Future	Because this "specialized" activity is in a phase of concentration, it is necessary to be vigilant on the health and roadmap of the vendor.	Roadmaps can differ from different cloud providers.

ALTERNATIVES TO ENTERPRISE AI PLATFORMS

Enterprise AI platforms are not the only solutions for the use cases discussed. Two types of platforms may be relevant, depending on whether your use cases are simple or redundant in your industry — "business-oriented solutions" and robotic process automation (RPA).

"BUSINESS-DEDICATED" PLATFORMS

In some fields, you can have "old" vendors that sell solutions focused on one kind of subject. Especially in manufacturing, you have some historic vendors who embraced AI and offer ready-to-use AI solutions to help manage a factory, enable predictive maintenance, etc. These solutions can sometimes be straightforward to use and cover some of your use cases.

ROBOTIC PROCESS AUTOMATION

RPA is a bundled solution that tries to "robotize" human gestures. These solutions are complemented with OCR solutions, but they also can write and send responses via email to cover a number of AI use cases. The ROI of this kind of solution can be exceptional. Nevertheless, managing dependencies between RPA and manipulated applications can be very difficult. Ideally, RPA should be considered if your business software rarely evolves.

Conclusion

Hopefully, these insights will help prepare you for when your boss says, "We need AI for these operations!" There is a big gap in understanding what AI can do and what we would like it to do. From assessing business needs to focusing on the right direction, building it yourself or buying from a vendor, and managing on-premise or in the cloud, you now have the tools necessary to make the right choice for your business use case.

Thomas Jardinet, IT Architect at Manpower

@thomas-jardinet on DZone | @ThomasJardinet on LinkedIn | @ThomasJardinet on Twitter

As an IT architect with 17 years of experience, I oversee business projects by defining their architectures, whether functional, applicational, or technical, studying the best path forward. As a supporter of flattened organizations, I also accompany them on the organizational side, and above all, I seek both an intellectual and human exchange.

CONTRIBUTOR INSIGHTS

MLOps for Enterprise AI

An Assessment of Current Trends, Tools, and Maturity Models for MLOps

By Sibanjan Das, Data Science Manager at ServiceNow

There was a time when building machine learning (ML) models and taking them to production was a challenge. There were challenges with sourcing and storing quality data, unstable models, dependency on IT systems, finding the right talent with a mix of Artificial Intelligence Markup Language (AIML) and IT skills, and much more. However, times have changed. Though some of these issues still exist, there has been an increase in the use of ML models amongst enterprises.

Organizations have started to see the benefits of ML models, and they continue their investments to bridge the gap and grow the use of AIML. Nevertheless, the growth of ML models in production leads to new challenges like how to manage and maintain the ML assets and monitor the models. Since 2019, there has been a surge in incorporating machine learning models into organizations, and MLOps has started to emerge as a new trending keyword. Although, it's not just a trend; it's a necessary element in the complete AIML ecosystem.

Advantages of MLOps

A simple Google search on MLOps will give you a variety of definitions, which is due to the fact that it's an emerging field and everyone has their own thoughts on what it encompasses. Similarly, MLOps is all about supporting the AIML ecosystem to manage the model lifecycle through automation, producing reliable and quality results consistently without performance degradation, and ensuring scalability of the AIML products.

The primary goal of MLOps is to maximize our ML model performance, increase agility in model development, and improve ROI. However, it is not easy to achieve. A complete automated MLOps solution has various components that provide power to this engine. These components include:

- **Automated ML model building pipelines** – Similar to the concept of continuous integration and continuous delivery (CI/CD) pipelines in DevOps, we try to set up ML pipelines to continuously build, update, and make the models ready to go into production accurately and seamlessly.
- **Model serving** – This is one of the most critical components, which provides a way to deploy the models in a scalable and efficient way so that the model users can continue using the ML model results without loss of service.
- **Model version control** – This is an important step in an AIML workflow, which enables tracking the code changes, maintaining data history, and enabling collaboration between teams. This brings scalability in providing agility to experiments and reproducing experimentation results whenever required.
- **Model/data monitoring** – Another critical component that helps measure the key performance indicators (KPIs) related to ML model health and data quality.
- **Security and governance** – This is a very important step to ensure access controls to ML model results and to track activities to minimize the risk associated with the consumption of results by an unintended audience and bad actors. Nowadays, data is the real asset, and ML provides guidance based on this data. Therefore, it is essential to protect and manage the results, so they stay only with the intended audience.

MLOps Maturity Assessments

It is highly unlikely for someone to have all these components automated together in one go, and it is also unrealistic to think that this can be achieved within a short period. To track the progress and measure the level of automation achieved using MLOps, Google and Microsoft came up with maturity models.

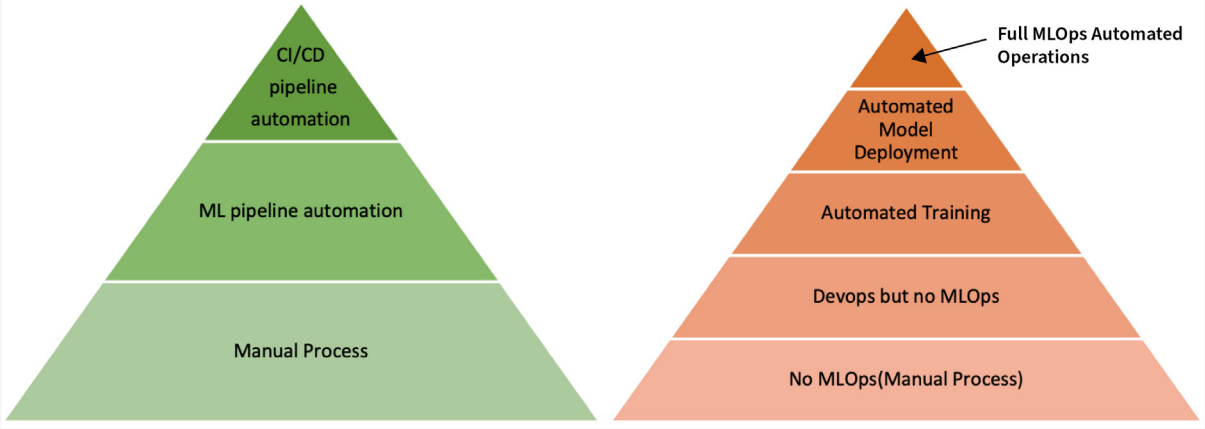
Google's maturity model consists of three levels:

- Level 0 – Manual process
- Level 1 – ML pipeline automation
- Level 2 – CI/CD pipeline automation

Similarly, Microsoft's maturity model consists of five levels:

- Level 0 – No MLOps
- Level 1 – DevOps but no MLOps
- Level 2 – Automated Training
- Level 3 – Automated Model Deployment
- Level 4 – Full MLOps Automated Operations

Figure 1: Comparing Google's and Microsoft's maturity models



Personally, I feel Microsoft's maturity model provides a better way to track the MLOps in an enterprise, as it is more detailed and the stages go beyond just CI/CD pipeline automation, which is just one of the components of a fully automated MLOps function. A fully automated MLOps system includes components related to automated model monitoring, prescribed improvement areas, and a goal for zero-downtime systems with reliable results each and every time.

Choosing the Right Tool for Your Use Case

Until now, we have discussed the various components in an MLOps system and the maturity levels, which can provide us with a benchmark on where we are in the MLOps transformation journey. The next question is: Are there any tools available in the market to help in this MLOps transformation journey? My answer would be: "Many." But there is no one-size-fits-all solution.

The following are some of the open-source tools that are considered to be a part of the MLOps ecosystem:

1. **MLFlow** – This is an open-source MLOps platform that can be used for managing an end-to-end ML lifecycle. It allows you to track the model experiments, manage and deploy ML models, package your code to reusable components, manage the ML registry, and host results as REST endpoints. You can get more information from their GitHub page.
2. **Evidently AI** – This is an open-source tool used to analyze and monitor ML models. They have features that allow us to monitor model health, data quality, and analyze data using various interactive visualizations.
3. **DVC** – This is an open-source version control system for ML projects. It helps with ML experiment management, project version control, collaboration, and makes the models shareable and reproducible. Built on top of the widely famous version control tool GitHub, it is designed to overcome some of the challenges very specific to ML model development, such as handling large data files and tracking ML model codes and metrics.

Apart from these, if you are ready to invest money, you can go for some packaged MLOps tools provided by Fiddler, DataRobot, Arize.ai, Comet, and many more.

Bottom Line

Which tool to use depends only on you! We are in the early phases of the MLOps evolution, and the process is only going to mature in the coming years. Everyone's problem is unique. Some want to deploy models quickly in production, some want an efficient way to secure and govern ML models, and others might require help with model serving and monitoring. We need to decide our top priority, and then start the transformation process, which will provide a quick path to realize value and ROI.

Sibanjan Das, Data Science Manager at ServiceNow

@sibanjandas on DZone | @sibanjandas on Twitter | @sibanjan on LinkedIn

Sibanjan Das is the team manager for Enterprise AI/ML initiatives at ServiceNow. In his current role, he works with a team of highly skilled data scientists, ML data engineers, and MLOps engineers, working together to make ServiceNow a data-driven and AI-driven organization. He believes in sharing knowledge with the community and has been actively writing books and articles as well as mentoring professionals and students interested in transitioning to AI/ML and IT.

CONTRIBUTOR INSIGHTS

Deploying AI With an Event-Driven Platform

Modern Reference Architectures Used to Deploy AI in the Enterprise

By Timothy Spann, Developer Advocate at StreamNative

Today, many large organizations are deploying artificial intelligence (AI) models with an event-driven platform in order to solve two common challenges of leveraging enterprise AI. First, to meet their data needs, enterprises often require a variety of model types that are built on different machine learning (ML), deep learning, and AI languages, frameworks, tools, and systems. These models are tied to various ways of deployment, using tools such as PyTorch, scikit-learn, XGBoost, DJL.AI, spaCy, TensorFlow, ONNX, PMML, Apache MXNet, and H2O. As a result, developers and data engineers need to deploy their models in diverse deployment environments with varying characteristics and restrictions, which makes accessing and managing the models complicated.

Deploying AI with an event-driven platform is a great method for integrating model access within streaming applications. Organizations can deploy and utilize all available libraries, frameworks, and models as part of an event-driven AI platform with open-source model servers and streaming platforms. This simplifies operations, increases flexibility, enables data durability, adds unlimited storage, improves resiliency, and enables near-limitless scalability.

Second, an event-driven AI platform democratizes access to enterprise AI, as the barrier to adopting a single AI platform is lower than adopting multiple platforms. Both technical and non-technical users now have real-time access to model classification results that are enhanced by streaming analytics. This offers numerous benefits to organizations — many use cases not only require AI applications but also need them to be accessible to various types of users within the organization and beyond. Below are the most common use cases that I have seen within different types of organizations all over the world in recent years:

- Visual question and answer
- NLP (natural language processing)
- Sentiment analysis
- Text classification
- Named entity recognition
- Content-based recommendations
- Predictive maintenance
- Fault and fraud detection
- Time-series predictions
- Naive bayes

Democratizing Access to Model Classification Results in Real Time

In this section, we look at what you need in an event-driven AI platform, along with the details for how to build one. We will also cover advanced features, demonstrate their benefits, and finally, uncover why democratizing access to your AI classifications is important, who those can serve, and how.

DESIGNING YOUR EVENT-DRIVEN AI PLATFORM

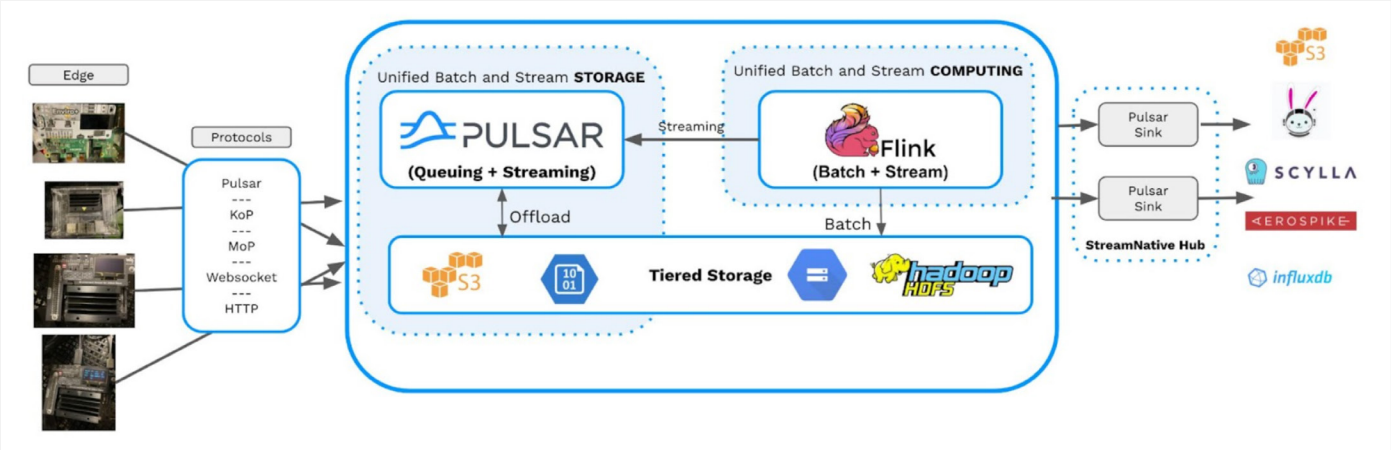
There are a number of crucial features required in an event-driven AI platform to provide real-time access to models for all users. The platform needs to offer self-service analytics to non-developers and citizen data scientists. These users must be able

to access all models and any data required for training, context, or lookup. The platform also needs to support as many different tools, technologies, notebooks, and systems as possible because users need to access everything by as many channels and options as possible.

Further, almost all end users will require access to other types of data (e.g., customer addresses, sales data, email addresses) to augment the results of these AI model executions. Therefore, the platform should be able to join our model classification data with live streams from different event-oriented data sources, such as Twitter and Weather Feeds. This is why my first choice for building an AI platform is to utilize a streaming platform such as Apache Pulsar.

Pulsar is an open-source distributed streaming and publish/subscribe messaging platform that allows your machine learning applications to interact with a multitude of application types, which enables countless users to consume your ML application results. Apache Pulsar is also a general platform capable of handling all possible messaging applications and use cases, as it supports the major messaging protocols and clients in most languages and systems. You will be able to exchange messages with Spark streaming applications, producing and/or consuming messages.

Figure 1: Example architecture for an event-driven AI platform



USE CASE: APACHE PULSAR FOR ML APPLICATIONS

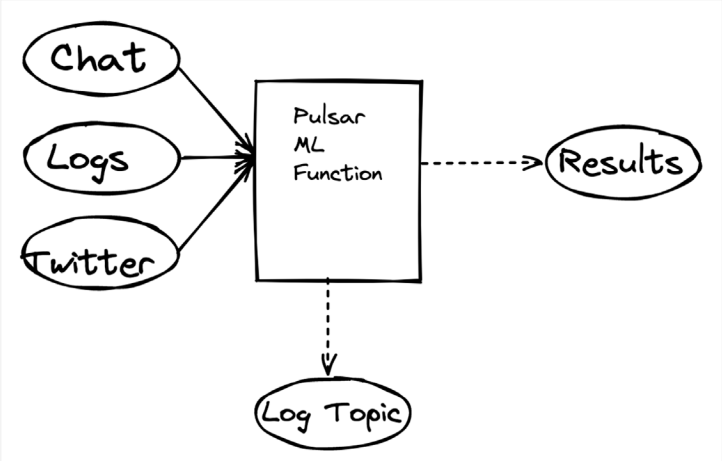
For example, with Pulsar SQL, you can run queries against all outputs of your ML applications via SQL, and with Pulsar's tiered storage capability, the output can amount to petabytes worth of data. Citizen data scientists can run these queries from any JDBC-compliant tool, which includes most spreadsheets, query tools, notebooks, web tools, and more. Supporting standard ANSI SQL opens the door to a huge set of tools, making the query results much more accessible to the non-developer world.

Everyone in your organization will also be able to access your AI data or send events to your AI models through the existing protocols. Apache Pulsar supports a large number of protocols, including MQTT, AMQP, WebSockets, REST, and Kafka. So your legacy applications can now add ML functionality without additional development, recompile, rebuild, redeployment, upgrade, rework, or new libraries.

Since Pulsar supports multiple consumers and producers, you are not tied to any one mechanism for triggering your ML models. By deploying your models as stateless functions in Pulsar, you open up a whole new world of possibilities.

In Figure 2, you will see how our ML function can accept messages from three topics or streams of data. Logging, which is a concern of late, is handled as asynchronous topics without concern for external libraries. Machine learning results are then published to log topics that all users can subscribe to.

Figure 2: ML function routing



SUBSCRIBING TO REAL-TIME CLASSIFICATION RESULTS

Another important requirement when deploying ML models is providing access to the results of classifications and the ability to make calls to your model from anywhere, by anyone you choose, and at any time. Democratizing access to model classifications results must be done in real time. We need many applications, developers, and systems to receive the results of classifications as they happen. Some users will need the ability to use SQL queries to get these results continuously, while other users need this data as aggregates on an ad hoc basis. There will be applications that run continuous processes that need event-by-event access to these results or may need to join them with other live data sources.

A platform built on Apache Pulsar will also enable any citizen data scientist, nontechnical user, data engineer, front-end developer, SQL developer, data scientist, and analyst to call your models and return results in the format that they need. They can do this in real time with no coding. The results of model classifications are accessible in universally available messaging topics, which can be read by any number of clients, including WebSockets, REST, and many more tools.

These topics can be accessed via all major messaging protocols for many existing legacy applications. An event-driven AI platform enables as many simultaneous accesses to these results as you need without worrying about scaling, latency, performance, or contention. All access is decoupled and scaled anywhere via geo-replication when needed. Results can be queried in real time with applications such Apache Spark and Apache Flink. For low-code development, you can easily connect to Pulsar topics from Apache NiFi and quickly create visual pipelines of your data.

Conclusion

Any enterprise can design and deploy an event-driven AI platform utilizing available open-source projects, frameworks, and tools. The time to start deploying these is now. A platform that developers, engineers, data scientists, data engineers, citizen data scientists, and non-technical users can collaborate on and share without complexity or difficulty is necessary.

When you are ready to build your own event-driven AI platform, make sure your system has all the features you need. The table below summarizes a few of the required capabilities and the nice-to-haves.

Table 1

System Requirements	Bonus Features
Open source	Geo-replication
Support for necessary ML and deep learning libraries	Support for multiple clusters
Support for existing pre-built models and your own models	Unit testing
Serverless code execution	Integrated logging
Choice of hosting environment — from on-prem to multi-cloud	Local run mode
Support for asynchronous communication, multi-tenancy, and multiple programming languages	Support for multiple protocols (e.g., MQTT, AMQP, Kafka, WebSockets)
Decoupling of models from users and model execution from model results access; concurrent access to results	
Accessibility from web apps and for citizen data scientists and nontechnical users	
Classification subscriptions	

Once you have the needed features, it is easy to trial an initial proof of concept with a simple Docker container running Pulsar, a simple Python machine learning model, and command-line tools to produce input data and consume the final model results.

We have reviewed many advantages of an event-driven platform for model deployment — ranging from the ease of model deployment and the ability to rapidly develop applications to the solutions for enterprise needs like scalability and low latency. With the flexibility to run thousands of models — whether on-prem, in Kubernetes, any cloud environment, or geo-replicated globally — you are ready to deploy your event-driven AI platform and create greater accessibility to your model classifications results. The time is now.

Timothy Spann, Developer Advocate at StreamNative

@bunkertor on DZone | @PaaSDev on Twitter | streamnative.io

Tim Spann is a Developer Advocate for StreamNative. He works with StreamNative Cloud, Apache Pulsar, Apache Flink, Apache NiFi, Edge AI, TensorFlow, Apache Spark, InfluxDB, Aerospike, ElasticSearch, Lakehouses, and deep learning. Tim has over a decade of experience with the IoT, big data, distributed computing, messaging, streaming technologies, and Java programming.

CONTRIBUTOR INSIGHTS

Where Spring Boot Meets Machine Learning Services: A Study

The Deep Java Library Brings Java Developers Into the ML Game

By John Vester, Technical Architect at CleanSlate Technology Group

In 1978, Bell Labs computer scientist Brian Kernighan introduced the most commonly known programming task in history:

```
main( ) {
        printf("hello, world\n");
}
```

This very simple C program was designed to communicate a system-generated message. The approach continues to be an initial step when learning programming languages, frameworks, and even system integrations. By establishing this basic premise, application code subsequently produces applications that manage intellectual property for the organizations which own them.

Now, almost 45 years later, the concept of machine learning (ML) has matured in a manner offering a competitive approach to understanding and solving daily business needs. As a subset of artificial intelligence, ML introduces models that are trained so that algorithms can be used to make decisions without being explicitly programmed to do so.

While Python maintains an edge in the ML space, Java developers have the option of using the Deep Java Library (DJL) to employ ML patterns while leveraging their existing Java knowledge. DJL is an open-source, high-level, engine-agnostic Java framework for deep learning that allows established models to be integrated with existing applications.

This article will demonstrate how Spring Boot and Java 11+ can be used with DJL to auto-classify images uploaded for a fictional photo competition.

Example Use Case

Consider an animal photo competition held over social media. Contestants simply need to upload a photo and include a specified tag to enter the competition. The most impressive photos for a given set of categories will receive some form of prize. Assume the following animal categories have been established:

- Cat
- Dog
- Elephant
- Lion
- Zebra
- Unknown (for all other categories and will be discarded)

As photos are tagged and uploaded, integration services will retrieve the image and user's information and then submit the image for auto-classification. This process will be handled by a brand-new service called the `machine-learning-service`.

The service will return the following information upon completing the classification request:

- Original file name provided (`Tilly-the-Cat.jpg`)
- Best match from classification model (`n0221567 cat, tiger cat`)
- Probability of the best match found (`8.677305101727`)
- Animal category found (`CAT`)

Judges will then be able to review the submissions at the category level, allowing them to focus on selecting the best photos.

Ideal Design

In this example, the service team tasked with creation of the `machine-learning-service` has a strong background in Java and the Spring Boot framework. The team can build upon existing standards for introducing a RESTful API that can accept binary images that need to be classified. Upon classification of the image, the results will be returned to the API consumer using the following JSON schema:

```
{
    fileName: string,
    value: string,
    probability: number,
    type: string
}
```

The only challenge for the service team is to understand how the auto-classification will work.

DEEP JAVA LIBRARY (DJL) IN ACTION

The DJL includes an existing classification model, which should provide the necessary criteria modeling required for the `machine-learning-service` to utilize to classify contestant submissions. The following logic can be configured as a Java `Bean` and automatically load when the service starts:

```
@Slf4j
@Component
public class CriteriaConfig {
    private static final int RESIZE_WIDTH = 224;
    private static final int RESIZE_HEIGHT = 224;

    private static final String DEFAULT_LAYERS_KEY = "layers";
    private static final String DEFAULT_LAYERS_VALUE_STRING = "50";

    @Bean
    public Criteria<Image, Classifications> criteria() {
    log.info("Establishing Criteria<Image, Classifications>");

        ImageClassificationTranslator translator = ImageClassificationTranslator.builder()
                .addTransform(new Resize(RESIZE_WIDTH, RESIZE_HEIGHT))
                .addTransform(new ToTensor())
                .build();

        Criteria<Image, Classifications> criteria = Criteria.builder()
                .setTypes(Image.class, Classifications.class)
                .optApplication(Application.CV.IMAGE_CLASSIFICATION)
                .optFilter(DEFAULT_LAYERS_KEY, DEFAULT_LAYERS_VALUE_STRING)
                .optTranslator(translator)
                .optProgress(new ProgressBar())
                .build();
```

Code continues on next page

```
        log.info("Service will utilize criteria={}", criteria);
        return criteria;
    }
}
```

With the criteria established, auto-classification of a given `InputStream` (image) can leverage the `ZooModel` provided by DJL using a very small amount of code:

```
@RequiredArgsConstructor
@Slf4j
@Service
public class ClassificationService {
    private final Criteria<Image, Classifications> criteria;

    public ClassificationDTO classifyImage(InputStream, String fileName) throws Exception {
        Image image = ImageFactory.getInstance().fromInputStream(inputStream);

        try (ZooModel<Image, Classifications> model = ModelZoo.loadModel(criteria);
             Predictor<Image, Classifications> predictor = model.newPredictor()) {
            Classifications classifications = predictor.predict(image);

            ClassificationResults classificationResults = ClassificationUtils.
convertClassifications(classifications);

            return new ClassificationDTO(classificationResults.getBest(), fileName);
        }
    }
}
```

The results of the `classifyImage()` method are returned as a newly created `ClassificationDTO` object:

```
@Data
public class ClassificationDTO {
    private String fileName;
    private String value;
    private double probability;
    private Type;
}
```

Introducing the Spring Boot Application

Aside from using Spring Boot 2.6.2 and Java 11, the following dependences are required for the `machine-learning-service`:

```
   <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jersey</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>ai.djl.spring</groupId>
            <artifactId>djl-spring-boot-starter-mxnet-auto</artifactId>
```

Code continues on next page

```
            <version>0.11</version>
        </dependency>
        <dependency>
            <groupId>ai.djl.spring</groupId>
            <artifactId>djl-spring-boot-starter-pytorch-auto</artifactId>
            <version>0.11</version>
        </dependency>
        <dependency>
            <groupId>ai.djl</groupId>
            <artifactId>basicdataset</artifactId>
            <version>0.14.0</version>
        </dependency>
    </dependencies>
```

To process the request, the `ClassificationController` is added, as shown below:

```
@RequiredArgsConstructor
@Slf4j
@CrossOrigin
@RequestMapping(produces = MediaType.APPLICATION_JSON_VALUE)
@RestController
public class ClassificationController {
    private final ClassificationService classificationService;

    @PostMapping(value = "/classify")
    public ResponseEntity<ClassificationDTO> classifyImage(@RequestParam("file") MultipartFile file)
{
        try {
            return new ResponseEntity<>(classificationService.classifyImage(file.getInputStream(),
file.getOriginalFilename()), HttpStatus.ACCEPTED);
        } catch (Exception e) {
            log.error("There is an issue with file={}, message={}", file.getName(), e.getMessage(), e);
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
    }
}
```

The controller will accept an HTTP `POST` request that includes the binary image and return the `ClassificationDTO` noted above.

Spring Boot in Action

Starting the `machine-learning-service` will present the following information to the console:

Figure 1

```
:: machine-learning-service :: Running Spring Boot 2.6.2 :: Port #8585 ::

2022-01-08 11:33:36.029  INFO 7446 --- [           main] .g.j.m.MachineLearningServiceApplication : Starting MachineLearningServiceApplication using Java 11.0.11 on johnvester02.local
2022-01-08 11:33:36.030  INFO 7446 --- [           main] .g.j.m.MachineLearningServiceApplication : No active profile set, falling back to default profiles: default
2022-01-08 11:33:36.720  INFO 7446 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8585 (http)
2022-01-08 11:33:36.726  INFO 7446 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-01-08 11:33:36.726  INFO 7446 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.56]
2022-01-08 11:33:36.786  INFO 7446 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-01-08 11:33:36.786  INFO 7446 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 721 ms
2022-01-08 11:33:36.822  INFO 7446 --- [           main] c.g.j.mls.configs.CriteriaConfig         : Establishing Criteria<Image, Classifications>
2022-01-08 11:33:36.830  INFO 7446 --- [           main] c.g.j.mls.configs.CriteriaConfig         : Service will utilize criteria=Criteria:
    Application: CV.IMAGE_CLASSIFICATION
    Input: interface ai.djl.modality.cv.Image
    Output: class ai.djl.modality.Classifications
    Filter: {"layers":"50"}

2022-01-08 11:33:37.099  INFO 7446 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8585 (http) with context path ''
2022-01-08 11:33:37.111  INFO 7446 --- [           main] .g.j.m.MachineLearningServiceApplication : Started MachineLearningServiceApplication in 1.432 seconds (JVM running for 2.057)
```

The `CriteriaConfig` class successfully loaded the classification criteria, allowing the service to quickly auto-classify images. As an example, the following image was downloaded from Pixabay and renamed to be `cat.jpg` for simplicity:

Figure 2

Using a simple `cURL` command, the image can be auto-classified using the `machine-learning-service`:

```
curl --location --request POST 'http://localhost:8585/classify' \
--form 'file=@"./cat.jpg"'
```

The Spring Boot service and DJL logs the following messages during processing:

```
Loading:     100% |████████████████████████████████████████|
[12:04:16] ../src/nnvm/legacy_json_util.cc:209: Loading symbol saved by previous version v1.6.0.
Attempting to upgrade...
[12:04:16] ../src/nnvm/legacy_json_util.cc:217: Symbol successfully upgraded!
2022-01-08 12:04:17.995  INFO 7446 --- [nio-8585-exec-1] c.g.j.m.services.ClassificationService
: classificationDTO=ClassificationDTO(fileName=cat.jpg, value=n02123045 tabby, tabby cat,
probability=8.329780578613281, type=CAT)
```

This results in the following JSON response being returned:

```
{
    "fileName": "cat.jpg",
    "value": "n02123045 tabby, tabby cat",
    "probability": 8.376285552978516,
    "type": "CAT"
}
```

The tabby cat classification has the highest probability of a match, which is what the `machine-learning-service` returns to the requestor. If the `DEBUG` log level is enabled in the `application.yml`, all the classification results are logged to the console:

```
[
  {
    "className": "n02123045 tabby, tabby cat",
    "probability": 8.329780578613281
  },
```

Code continues on next page

```
  {
    "className": "n02123159 tiger cat",
    "probability": 6.066715240478516
  },
  {
    "className": "n02124075 Egyptian cat",
    "probability": 5.044580459594727
  },
  {
    "className": "n02123394 Persian cat",
    "probability": 2.0124764442443848
  },
  {
    "className": "n02127052 lynx, catamount",
    "probability": 1.495771884918213
  }
]
```

For more information on probability values, please check out the documentation.

Conclusion

Starting in 2021, I have been trying to live by the following mission statement, which I feel can apply to any IT professional:

> *"Focus your time on delivering features/functionality which extends the value of your intellectual property. Leverage frameworks, products, and services for everything else."*
>
> *– J. Vester*

The Spring Boot framework and Deep Java Library fall into line with my personal mission statement. In a very short amount of time, a service was created to completely meet the requirements to auto-classify images — without requiring any knowledge of a new programming language.

If desired, the DJL could be further employed to train and utilize custom models for use cases where one of the included models is not an ideal match. While additional time is required to understand, introduce, and train custom models, there is still no need to learn a new program language or service-tier framework to meet the needs of the request.

The full source code for this example is available on GitLab.

John Vester, Technical Architect at CleanSlate Technology Group

@johnjvester on DZone | LinkedIn, Twitter, GitLab, GitHub, dockerhub | Personal Blog

Information Technology professional with 30+ years of expertise in application design and architecture, feature development, project management, system administration, and team supervision. Currently focusing on enterprise architecture/application design utilizing object-oriented programming languages and frameworks. Prior expertise building (Spring Boot) Java-based APIs against React and Angular client frameworks. CRM design, customization, and integration with Salesforce. Additional experience using both C# (.NET Framework) and J2EE (including Spring MVC, JBoss Seam, Struts Tiles, JBoss Hibernate, Spring JDBC).

CONTRIBUTOR INSIGHTS

AI-Powered Knowledge Graphs

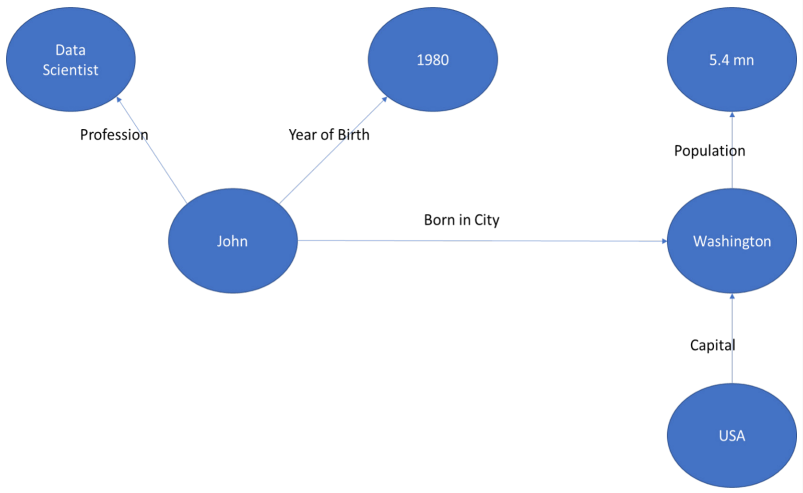
The Holy Grail of Omniscience

By Dr. Tuhin Chattopadhyay, Professor at Jagdish Sheth School of Management

The Knowledge Graph: What It Is, The Rise, and The Purpose

A knowledge graph (KG) is a semantic network of an organization, a topic where the nodes are known as the entities and the edges are the relationships. It is a framework comprising a set of related yet heterogeneous data — like image, sound, text, video, numbers, etc. — that gives a semantic interpretation and lets researchers run complex algorithms on graph data to generate insight. The RDF (Resource Description Framework) triplestore, a graph database, stores the data as a network of objects or RDF triples that segregate the information into subject-predicate-object expressions. A simple example of relations among entities is shared in Figure 1 for ease of understanding.

Figure 1: Relational knowledge graph



Mathematician Leonhard Euler, the father of graph theory, used graphs to calculate the minimum distance the emperor of Prussia had to travel to visit Königsberg. With the revolution of big data, organizations started looking beyond traditional relational databases like RDBMS. The NoSQL movement lets organizations store both structured and unstructured data in data lakes. Different types of databases, like MongoDB for documents and Neo4j for graph databases, came into existence with capabilities of graph storage and processing. However, they were not free from problems as there was a lack of formal data schemas and consistencies to run the complex analytics models. KGs bridged the gap and instantly became the cynosure of all large organizations.

There exists a three-fold goal of KGs. First and foremost, a KG helps users by searching to discover information more quickly and easily. Secondly, a KG provides side and contextual information in developing an intelligent recommendation engine. Finally, it can help answer queries and make predictions through Knowledge Graph Question Answering (KGQA). The key basis for generating the answers from the questions are shared below in Table 1.

Table 1: Basis of KGQA

	Description
Semantic parsing	• Parses the natural language question • SPARQL is used to search the KG
Information retrieval	• Natural language questions are transformed into structured queries to find possible answers • Feature and topic graphs are used to retrieve the best answer
Embedding	• Calculates proximity scores between questions and plausible answers • Uses the vector modeling approach
Deep learning (DL)	• DL on NLP is applied, like multi-column convolutional neural networks (MCCNNs), for image analysis • Bidirectional long short-term memory (BiLSTM) is used to understand the questions better

Developing the Knowledge Graph

Automated knowledge acquisition and semantic mapping are the pillars of developing a KG. The process of ontology engineering for knowledge acquisition starts with ontology learning that aims to automatically learn relevant concepts and establish relations among them. To achieve this, first the corpus is parsed to identify the collocations and, subsequently, it retrieves the semantic graph. Entity enrichment takes place by crawling semantic data and merging new concepts from relevant ontologies.

Figure 2: Process of entity enrichment

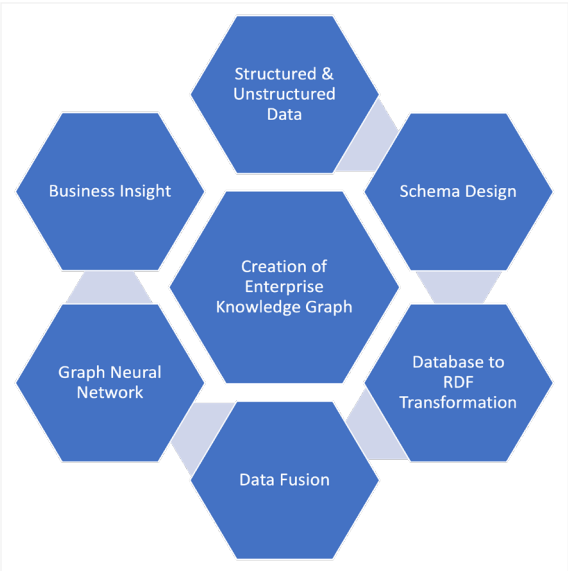
Integrating heterogeneous data from structured sources demands mapping the local schemas to the global schema. Global-as-view (GAV), a mediation-based data integration strategy, is implemented where the global schema acts as a view over source schema to convert the global query into a source-specific query. Detecting the semantic type is the first step for automated semantic mapping, which is followed by inferring the semantic relation.

Data are initially modeled using RDF and subsequently RDF Schema (RDFS), and Web Ontology Language (OWL) adds semantics to the schema. Semantic information can also be mapped in a hierarchical way through relational vectors. Graph neural networks (GNNs) — like graph convolutional networks (GCNs) or gated graph neural networks — are used for object detection and image classification of graph data.

Enterprise Knowledge Graph

Organizations of today's era are in pursuit of discovering the hidden nuggets of information, so they are interlocking all their siloed data by consolidating, standardizing, and reconciling. Thus, an enterprise knowledge graph provides an explicit representation of knowledge from business data in the graph. An integrated data enterprise possesses the power of the web of knowledge that uncovers critical hidden patterns to monetize their data.

Figure 3: Steps to develop an enterprise knowledge graph



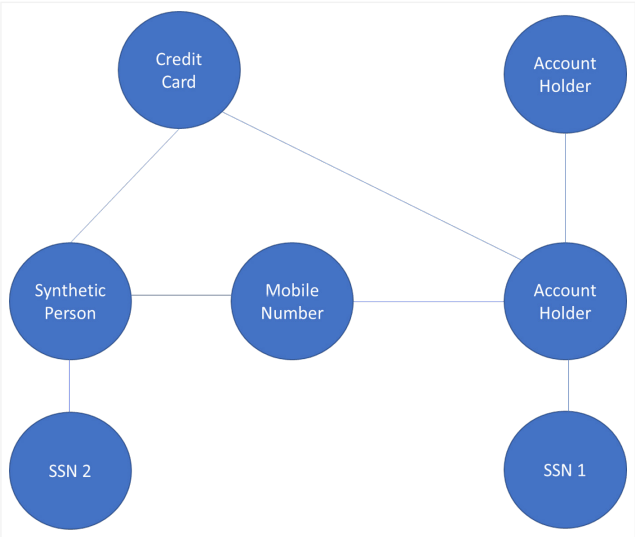
Real-World Knowledge Graphs

We are inundated with data in the present world. KGs give meaning and purpose to connected data with several applications, of which a few are shared below.

FINANCIAL SERVICES KNOWLEDGE GRAPH

KGs have wide applications in financial services, ranging from fraud detection and tax calculations to financial reporting and stock price prediction. Fraud rings comprising a few people collectively committing a fraud can easily be identified by examining the topology of the subgraphs.

Figure 4: Fraud detection through a knowledge graph



Stock price can be predicted by linking the sentiments associated with the news of the respective company. Hedge funds and banks use KGs for better predictions by mapping their existing models with the alternate data provided by KGs.

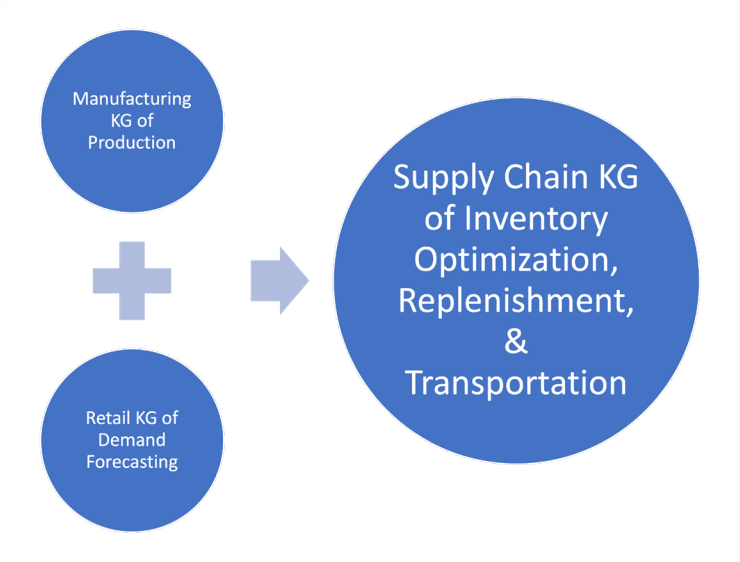
MEDICAL SCIENCE

Biomedical concepts and relationships are represented in the form of nodes and edges. By applying KGs, medical imaging analysis can be used for disease classification, disease medication and segmentation, report generation, and image retrieval. Textual medical knowledge (TMK) from the Unified Medical Language System (UMLS) is analyzed to generate key medical insights and personalized patient reports.

REAL-TIME SUPPLY CHAIN MANAGEMENT

Supply chain organizations use KGs to optimize the stock of inventories, replenishment, network and distribution management, and transportation management. The connected supply chain KG takes the inputs from the manufacturing KG of production, including personnel, plus the retail KG, which comprises the real-time and forecasted demand for better prediction and management (Figure 5).

Figure 5: Constituent components to develop a supply chain knowledge graph



Conclusion

A knowledge graph has the power to create a virtual world where all entities are connected with a proven relationship. Sophisticated machine learning algorithms are applied to prune those connections where the probability of a relationship is slim. Thus, proven relationships among all objects in the world can be established through a KG.

With all the past and present data, a KG produces deep insights by recognizing the patterns. A KG also helps us predict the future with all the relevant data leading to a phenomenon. The future KGs could be even more powerful with the road ahead shared below:

- **Graph of Things** (GoT) – GoT is an innovative project that aims to merge both the high-volume streaming data of Internet of Things (IoT) and the static data of the past.
- **Quantum AI for KG** – Quantum AI can leverage the power of quantum computing for running the GNNs on the KG and can achieve the results beyond the capabilities of traditional computing.

A world with all the information connected through a KG would indeed be magnificent if the benefits are harnessed for the welfare of society. AI on top of KG, when used with the right intent, will make the world a better place to live.

Dr. Tuhin Chattopadhyay, Professor at Jagdish Sheth School of Management

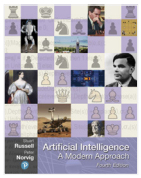
@tuhinc on DZone | @tuhinai on LinkedIn | tuhin.ai

Dr. Tuhin Chattopadhyay is a celebrated Industry 4.0 thought leader among both the academic and corporate fraternity. Recipient of numerous prestigious awards, Tuhin is hailed as India's Top 10 Data Scientists by Analytics India Magazine. Dr. Tuhin serves as Professor of Analytics at JAGSoM in Bengaluru, India alongside driving his AI consultancy organization.

ADDITIONAL RESOURCES

Diving Deeper Into Artificial Intelligence

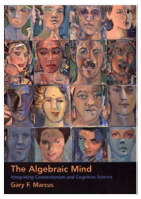
BOOKS



Artificial Intelligence: A Modern Approach

By Stuart Russell and Peter Norvig

As one of the most authoritative AI resources available, this academic textbook covers AI and related subjects like machine learning, neural networks, and NLP at great depths. It's a perfect companion for anyone exprienced with AI who is looking to comprehensively expand their technical and mathematical knowledge and skill sets.



The Algebraic Mind: Integrating Connectionism and Cognitive Science

By Gary F. Marcus

Engage further with the symbolic vs. connectionist discussion covered in our 2022 Enterprise AI Key Research Findings with this go-to cognitive science book. The author seeks to integrate these two opposing theories, examining the nature of cognitive architecture and presenting an insightful analysis that takes the conversation to an even higher level.

TREND REPORTS

Data Persistence

While data management tools and strategies have matured quickly as of late, the complexity of architectural and implementation choices has intensified too, creating unique challenges and opportunities for those designing data-intensive systems. This report examines the industry's current state, presents analyses of our research findings, and features contributor insights into microservice polyglot persistence, data storage solutions, and more.

Data Warehousing

As the demand for informed business decisions and analytics continues to skyrocket, data warehouses are becoming more popular. This Trend Report explores adoption across industries, including common data tools like data lakes, data virtualization, and ETL/ELT. Readers will find original research, an interview with "the father of data warehousing," and articles covering helpful tips and best practices.

REFCARDS

Data Pipeline Essentials: Strategies for Successful Deployment and Collecting Analytical Insights

Data pipelines allow organizations to automate information extraction from distributed sources while consolidating data into high-performance storage for centralized access. In this Refcard, readers will explore the fundamentals of data pipelines — from common pipeline processes and core components to deployment considerations, implementation challenges, and advanced strategies.

Getting Started With Robotic Process Automation

Technologies like AI, ML, and NLP have led to the innovation of software robots that can interact with computer-centric processes to help reduce the manual, time-consuming, and repetitive actions performed by humans. This Refcard dives into robotic process automation (RPA), exploring key components and techniques, process standardization, and configuration of an example RPA tool.

PODCASTS

Practical AI: Machine Learning, Data Science

If you are looking to keep up with the newest advances in AI — and still learn about the practical, real-world applications of AI — this is the show for you. The hosts, along with featured guests of all experience levels, cover topics that include AI, machine and deep learning, neural networks, AIOps, MLOps, and more.

Eye On AI

Catch this bi-weekly podcast, hosted by Craig S. Smith of the New York Times, about what is on the horizon for AI. Listeners will discover insights from those making changes in the industry and learn how machine intelligence applies in broader contexts as well as the global implications of AI that we should consider.

ADDITIONAL RESOURCES

Solutions Directory

This directory contains AIOps, MLOps, and various AI/ML tools to help you do analyze data, automate processes, improve customer experience, and more. It provides pricing data and product category information gathered from vendor websites and project pages. Solutions are selected for inclusion based on several impartial criteria, including solution maturity, technical innovativeness, relevance, and data availability.

DZONE'S 2022 ENTERPRISE AI SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
2022 PARTNER Katana Graph	Katana Graph	Graph intelligence platform	By request	katanagraph.com
Company	**Product**	**Purpose**	**Availability**	**Website**
[24]7.ai	Conversations	Conversational AI	By request	247.ai/products/conversations
	Engagement Cloud	CCaS		247.ai/products/engagement-cloud
Ada	Ada	Conversational AI, ML-driven analytics	By request	ada.cx
Aisera	AI Customer Intelligence	Conversational AI	By request	aisera.com/products/ai-customer-intelligence
	AI Service Desk	Conversational AI, automation, ticket AI		aisera.com/products/ai-service-desk
	AIOps	AIOps, incident management		aisera.com/products/aiops
Alluxio	Alluxio	Data orchestration for analytics and ML	Free tier	alluxio.io/products
Altair	DesignAI	AI- and simulation-driven design	By request	altair.com/designai
	Knowledge Studio	ML and predictive analytics		altair.com/knowledge-studio
	SmartWorks	Build analytics applications and scalable automation		altair.com/smartworks-analytics
Alteryx	Intelligence Suite	Computer vision, text mining, and automated ML	Trial period	alteryx.com/products/intelligence-suite
	Machine Learning Platform	ML automation	By request	alteryx.com/products/alteryx-machine-learning
	Promote	Model deployment and management, MLOps	Trial period	alteryx.com/products/alteryx-promote
Amazon Web Services	Amazon Augmented AI	Human review of ML predictions	Free tier	aws.amazon.com/augmented-ai
	Amazon DevOps Guru	ML-powered DevOps		aws.amazon.com/devops-guru
	Amazon SageMaker	Build, train, and deploy ML models		aws.amazon.com/sagemaker
Amelia	Amelia Integrated Platform	Conversational AI, AIOps	By request	amelia.ai
Anaconda	Anaconda Distribution	Python distribution	Open source	anaconda.com/products/distribution
	Enterprise DS Platform	End-to-end data science and ML	By request	anaconda.com/products/enterprise
Anodot	Anodot	Autonomous business monitoring and anomaly detection	By request	anodot.com

DZONE'S 2022 ENTERPRISE AI SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
Apache Software Foundation	Apache Marvin-AI	MLOps platform	Open source	marvin.apache.org
	MADlib	Big data ML in SQL		madlib.apache.org
	Mahout	Create scalable performant ML applications		mahout.apache.org
	MXNet	Deep learning library		mxnet.apache.org
	OpenNLP	ML-based toolkit		opennlp.apache.org
	SINGA	Distributed training of deep learning and ML models		singa.apache.org
	Spark MLlib	Scalable machine learning library		spark.apache.org/mllib
	UIMA	Unstructured content analysis		uima.apache.org
Artificial Solutions	Teneo Platform	Conversational AI	By request	artificial-solutions.com/teneo
Baidu	DuerOS	Conversational AI	By request	dueros.baidu.com/en/html/dueros
BigML	BigML	Consumable, programmable, and scalable ML	Free tier	bigml.com
Cloudera	CDP Machine Learning	ML workflow optimization	By request	cloudera.com/products/machine-learning.html
Databricks	Databricks	Data lakehouse platform	Trial period	databricks.com
Dataiku	Dataiku	End-to-end AI platform	Trial period	dataiku.com
DataRobot	AI Cloud Platform	End-to-end AI platform	By request	datarobot.com
Domino	Enterprise MLOps Platform	MLOps platform	Trial period	dominodatalab.com
DQLabs	DQLabs	Data quality and observability	By request	dqlabs.ai
Dynatrace	Dynatrace	End-to-end observability	Trial period	dynatrace.com
EdgeVerge	XtractEdge	Document AI	By request	edgeverve.com/extractedge-platform
Google Cloud	Dialogflow	Conversational AI	Trial period	cloud.google.com/dialogflow
	Vertex AI	Managed ML platform		cloud.google.com/vertex-ai
Grok	Grok AIOps	AIOps for intelligent operations	Trial period	grokstream.com
H2O.ai	H2O AI Cloud	End-to-end AI platform	By request	h2o.ai
IBM	Cloud Paks	Hybrid cloud AI platform	By request	ibm.com/cloud-paks
	SPSS Modeler	Data visualization for AI and ML	Trial period	ibm.com/products/spss-modeler
	Watson	Data analytics processor	By request	ibm.com/watson
Inbenta	Inbenta	Conversational AI	Trial period	inbenta.com
Intel	OpenVINO	Deep learning runtime compiler	Free	intel.com/content/www/us/en/developer/tools/openvino-toolkit
Kaldi	Kaldi	Speech recognition toolkit	Open source	kaldi-asr.org
Kare	Kare	AI automation for customer experience	By request	karehq.com
Kasisto	KAI	Conversational AI	By request	kasisto.com
Keras	Keras	Deep learning API	Open source	keras.io
KNIME	KNIME Analytics Platform	Data analytics platform	Free	knime.com/knime-analytics-platform

DZONE'S 2022 ENTERPRISE AI SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
Kore.ai	Kore.ai	Conversational AI	Trial period	kore.ai
MatConvNet	MatConvNet	MATLAB toolbox for computer vision apps	Open source	github.com/vlfeat/matconvnet
MathWorks	MATLAB	Programming and numeric computing platform	Trial period	mathworks.com/products/matlab.html
	Simulink	Model-based design and simulation		mathworks.com/products/simulink.html
Meya	Meya	Chatbot development platform	Trial period	meya.ai
Micro Focus	IDOL Unstructured Data Analytics	Unified AI-driven analytics platform	By request	microfocus.com/en-us/products/information-data-analytics-ido
Microsoft	Microsoft Cognitive Toolkit	Deep learning toolkit	Open source	github.com/microsoft/cntk
Microsoft Azure	Azure Applied AI Services	AI services for common business processes	Trial period	azure.microsoft.com/en-us/products/applied-ai-services
	Azure Cognitive Services	AI servicecs for cognitive capabilities		azure.microsoft.com/en-us/products/cognitive-services
	Azure Kinect DK	Spatial computing developer kit with AI	By request	azure.microsoft.com/en-us/products/kinect-dk
	Azure Machine Learning	End-to-end ML platform	Trial period	azure.microsoft.com/en-us/products/machine-learning
	Azure OpenAI Service	Advanced, large-scale generative AI models		azure.microsoft.com/en-us/products/cognitive-services/openai-service
mlpack	mlpack	Header-only C++ ML library	Open source	mlpack.org
MoreSteam	EngineRoom	Data analyzation tool	Trial period	moresteam.com/engineroom
Natural Language Toolkit	NLTK	Python program builder for NLP	Open source	nltk.org
Neuroph	Neuroph	Java neural network framework	Open source	neuroph.sourceforge.net
NVIDIA	NVIDIA AI Enterprise	End-to-end AI development and deployment	Trial period	nvidia.com/en-us/data-center/products/ai-enterprise
	NVIDIA CUDA-X AI	GPU-accelerated library for AI	Free	nvidia.com/en-us/technologies/cuda-x
	NVIDIA Riva	GPU-accelerated speech AI SDK	Trial period	nvidia.com/en-us/ai-data-science/products/riva
OpenNN	OpenNN	C++ neural networks library for ML	Open source	opennn.net
OpenText	OpenText Magellan	AI and analytics platform	By request	opentext.com/products/magellan-platform
Oryx	Oryx 2	Lambda architecture for real-time ML	Free	github.com/oryxproject/oryx
OutSystems	OutSystems.AI	AI and ML for OutSystems	Free tier	outsystems.com/ai
Posit	tidymodels	Modeling and ML framework	Open source	tidymodels.org
Progress	NativeChat	Chatbot development platform	By request	progress.com/nativechat
PyTorch	PyTorch	ML framework	Open source	pytorch.org
Qlik	AutoML	No-code, automated ML	By request	qlik.com/us/products/qlik-automl
	Sense	Cloud analytics	Trial period	qlik.com/us/products/qlik-sense
Rainbird	Rainbird	No-code intelligent automation platform	By request	rainbird.ai/platform

DZONE'S 2022 ENTERPRISE AI SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
RapidMiner	RapidMiner	Data science platform with MLOps and AI	By request	rapidminer.com/platform
Salesforce	Einstein Intelligence	AI-powered application for predictive analytics	By request	salesforce.com/products/einstein/overview
SAP	SAP Predictive Analytics	Predictive analytics platform	Open source	help.sap.com/docs/sap_predictive_analytics
SAS	SAS Viya	AI-based automation	Trial period	sas.com/en_us/software/viya.html
Scikit Learn	Scikit Learn	ML in Python	Open source	scikit-learn.org/stable
Shogun	Shogun	Toolkit of algorithms and data structures for ML problems	Open source	github.com/shogun-toolbox/shogun
Skyjed	Skyjed	AI-powered product lifecycle management and governance	Free tier	skyjed.com
Skymind	Skymind	AI ecosystem builder and investment company	Free	skymind.global
SmarTek21	IntelliTek.ai	Conversational AI	By request	intellitek.ai
Snorkel	Snorkel Flow	Platform for data-centric AI development	By request	snorkel.ai/snorkel-flow-platform
spaCy	spaCy	NLP in Python	Open source	spacy.io
Splunk	Splunk IT Service Intelligence	AIOps for monitoring and observability	Trial period	splunk.com/en_us/products/it-service-intelligence.html
Stanford University	Stanford CoreNLP	NLP toolkit	Open source	stanfordnlp.github.io/corenlp
Superwise	Superwise	ML monitoring and observability	Free tier	superwise.ai
TensorFlow	TensorFlow	Software library for AI and ML	Open source	tensorflow.org
TIBCO	TIBCO Spotfire	AI-based analytics platform	Trial period	tibco.com/products/tibco-spotfire
UMass Amherst	MALLET	Java library for NLP and ML	Open source	mimno.github.io/Mallet/index
Uniphore	Uniphore X Platform	Conversational AI	By request	uniphore.com/x-platform
University of Waikato	Massive Online Analysis	ML for data stream mining	Open source	moa.cms.waikato.ac.nz
	Weka	Java-based AI and ML data mining		cs.waikato.ac.nz/ml/weka
Unravel	Unravel	AI-powered observability	Trial period	unraveldata.com/platform/optimization
Vaticle	TypeDB	Strongly-typed database	Free	vaticle.com
WhyLabs	WhyLabs	AI-powered observability	By request	whylabs.ai/observability
Wipro	Holmes	AI development platform	By request	wipro.com/holmes
Wit.ai	Wit.ai	NLP interface for applications	Open source	wit.ai

INTRODUCING THE

AI Zone

What do developers need to know about artificial intelligence? Whether it's understanding real-world AI applications or tutorials on using algorithms in data science, this Zone covers a wide range of topics.

Keep a pulse on the industry with topics such as:

- Neural networks
- Machine learning in development
- Deep learning
- AI applications

VISIT THE ZONE

TUTORIALS

CASE STUDIES

BEST PRACTICES

CODE SNIPPETS