A critical overview of AutoML solutions

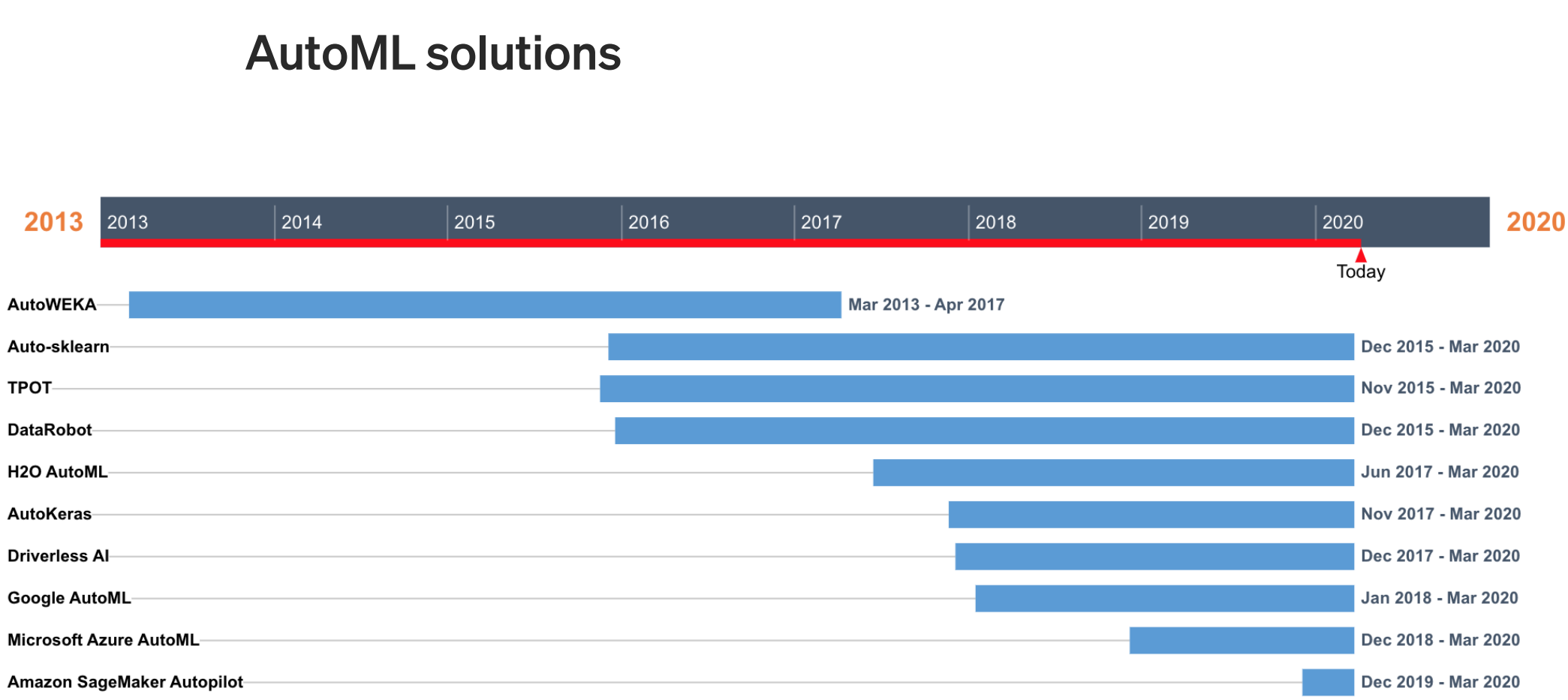
Bahador Khaleghi Follow
Apr 2, 2020 · 22 min read

In an earlier article, I discussed how enterprise machine learning (ML) faces several challenges when it comes to successful implementation and adoption of ML solutions and why AutoML is the next natural evolutionary step to help alleviate some of those challenges. In this article, I present an overview of the existing AutoML solutions, highlight their key capabilities, and discuss some of their major remaining gaps.

Methodology

I will explore capabilities of AutoML solutions by answering these six main category of questions:

- **Functionality**: this is perhaps the most important category and refers to the specific steps of the end-to-end ML workflow that are being partially or fully automated.
- **Versatility**: what types of ML problems, e.g., classification, regression, forecasting, or ranking can be handled? What ML methods are included? Does it have data-type specific provisions such as handling text data for NLP tasks?
- **Flexibility**: how much customization power and control users have over the automated workflow steps?
- **Scalability**: how large of a problem in terms of dataset size can be handled? Are multiple CPUs, GPUs, or TPUs supported? Does it scale up or out? Can it be run on the cloud?
- **Transparency/Trust**: what is offered to help users better understand and trust results of automated workflow steps? Possible offerings include model explanations, model fairness assessment, model documentation, model blueprint visualization and so on.
- **Ease of use**: is solution available for free? How easy is the installation? Is there a GUI, if so, how user-friendly is it? What APIs are offered to programatically interact with the solution?

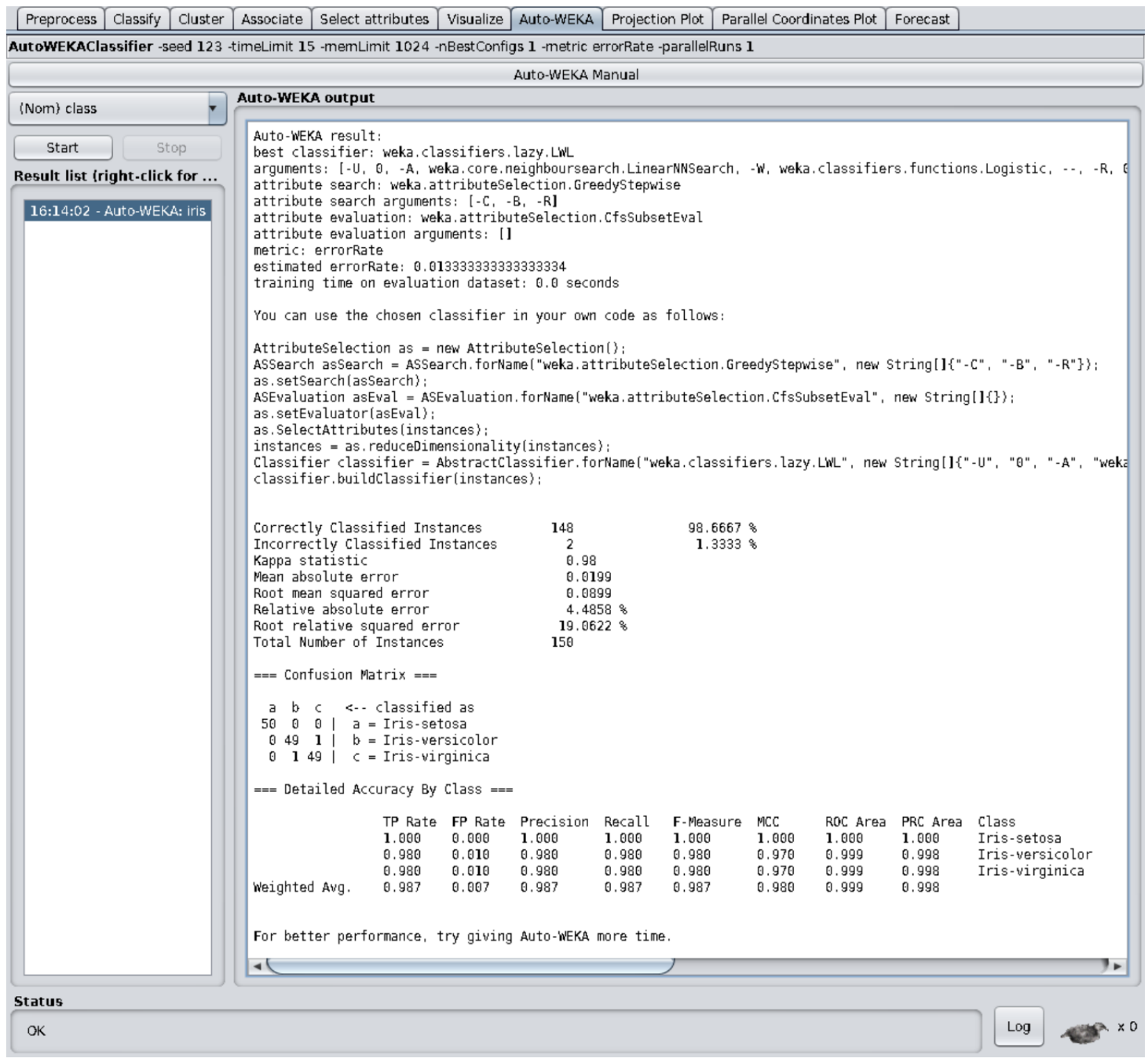


Historical timeline of AutoML solutions landscape including early open source solutions first introduced in 2013, subsequent enterprise solutions, and the more recent cloud based solutions

AutoML has only recently become a trendy topic within the ML community. Nonetheless, early AutoML solutions that originated in academia such as

AutoWEKA date back to a few years ago. Later on, some startups, including H2O and DataRobot, started developing their own AutoML solutions. The most recent wave of AutoML solutions are developed by major cloud providers including Amazon, Microsoft, and Google. As such, there are a relatively large number of solutions in the AutoML space and newer ones are emerging all the time. Our discussion is not meant to be comprehensive. Instead, I aim at presenting an overview of some of the most popular and capable solutions.

AutoWEKA



Example AutoWEKA run on the Iris dataset

AutoWEKA was first introduced by a research team at the University of British Columbia in 2013. It is one of the earliest open source AutoML solutions and has evolved over the years although there has been no new releases since 2017. As suggested by its name, AutoWEKA relies on the WEKA library as its ML backend.

Functionality: simultaneously tackles model selection and model learning steps of the ML workflow using a Bayesian optimization approach called

SMAC.

Versatility: can tackle classification and regression type of ML problems. AutoWEKA offers a rather extensive suite of ML methods though it excludes modern deep neural networks. No support for learning from unstructured data such as text or image is offered. In addition, no time series forecast modelling is available.

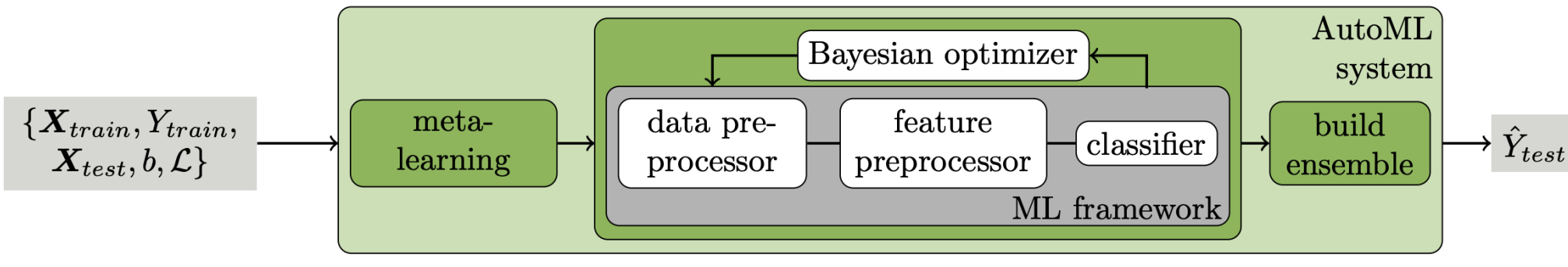
Flexibility: being perhaps the oldest AutoML solution, very little customization is possible, users can choose the examined ML methods, and time and memory limits.

Scalability: offers basic parallelism through multi-threading. However, no support for GPUs, distributed computing, or cloud is provided.

Transparency/Trust: documentation, in form of a Javadoc and a user guide, is provided.

Easy of use: is open source and quite easy to install (requires only Java 8 as a prerequisite). A basic GUI is offered. Has a Java API and a command-line interface. A Python wrapper API is also available.

Auto-sklearn



The AutoML approach of Auto-sklearn

Similar to AutoWEKA, Auto-sklearn was developed in academia at the University of Freiburg in 2015. It is based on the highly popular scikit-learn ML library. The project has been continuously developed over the years and remains active as of today. Check out this video for a quick demo of Auto-sklearn in action.

Functionality: builds on the Bayesian optimization based approach proposed by AutoWEKA for solving model selection and model learning as a single problem. It uses a meta-learning step to warm-start a Bayesian optimization that results in a considerable efficiency improvement. Moreover, it can automatically construct the final model as an ensemble of models evaluated during Bayesian optimization. Finally, Auto-sklearn offers partial automated feature engineering by including a set of feature preprocessing methods such as PCA in its optimization search space.

Versatility: can tackle classification and regression type of ML problems. Auto-sklearn offers a rather extensive suite of ML classification and regression methods. Similar to AutoWEKA, it does not support deep neural networks. Both numerical and categorical feature types are supported. However, neither text or image data are supported.

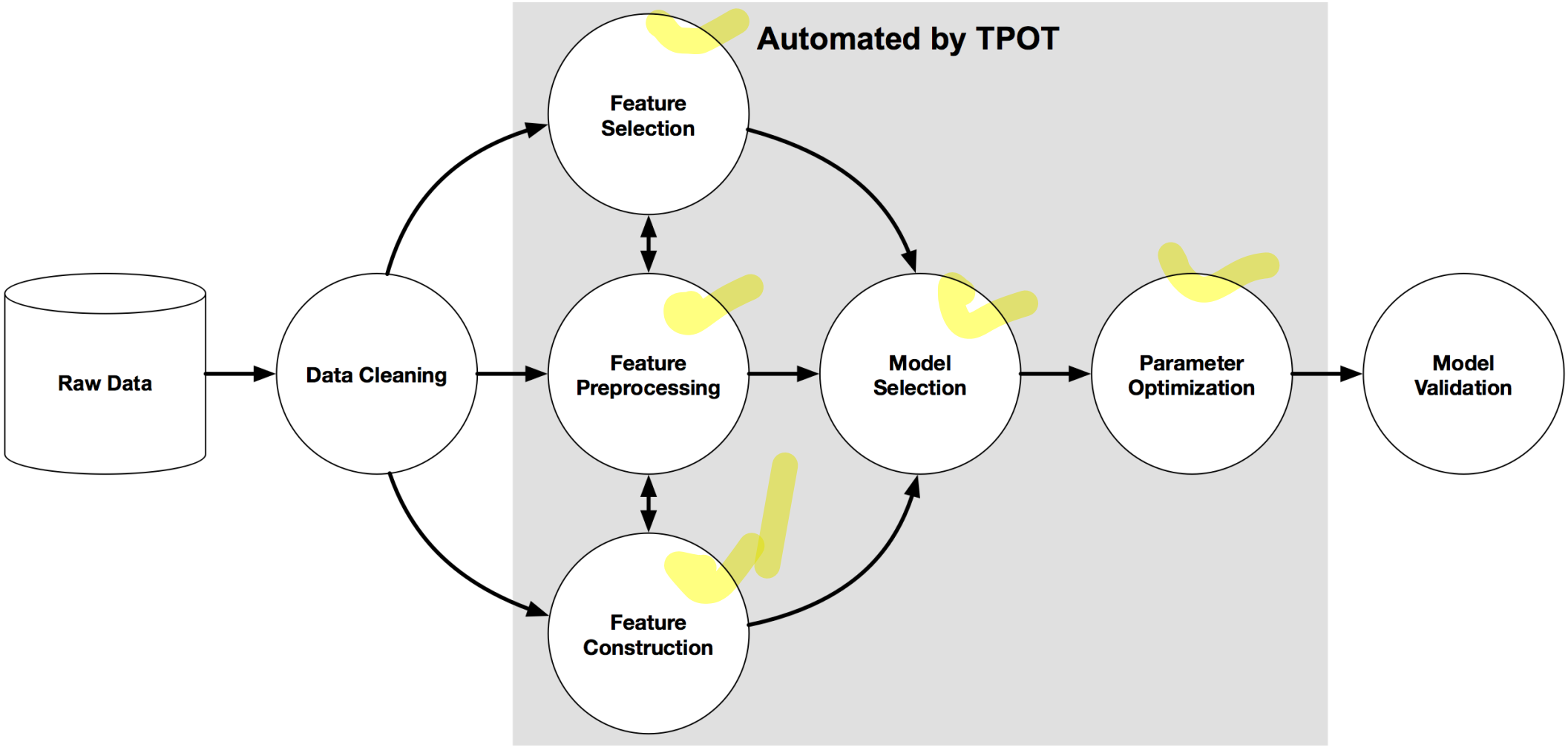
Flexibility: can be extended with new classification, regression and feature preprocessing methods by implementing a wrapper class and registering it to auto-sklearn. Users can also restrict the optimization search space to improve efficiency. Finally, users can define their own custom metric to fit ML models.

Scalability: basic parallel computation is provided. However, no GPU, distributed computing, or cloud support is offered.

Transparency/Trust: an API documentation and a manual, constituting basic functionality descriptions and some references examples, are provided.

Easy of use: is open source and easy to install. However, no GUI is provided and only a Python API is available.

TPOT



An example of ML workflow automation with TPOT

TPOT, or Tree-based Pipeline Optimization Tool, was developed at the University of Pennsylvania in 2015. The project has been actively developed ever since. It relies on the scikit-learn ML library as its backend. Having being developed by researchers at the Computational Genetics Lab, TPOT performs AutoML using an algorithm based on genetic programming, a well-known evolutionary computation technique for automatically constructing computer programs. You can check out a quick demo of TPOT here.

Functionality: is able automatically find an ML pipeline of feature preprocessing, model selection and model learning. While most AutoML solutions focus on model selection and model learning, TPOT also pays attention to feature selection and feature engineering by optimizing a more comprehensive pipeline. The details of this algorithm are not provided but its makers admit that it can be quite time-consuming to run.

Versatility: offers the same suite of ML methods as Auto-sklearn to tackle classification, and regression problems. No support for problems involving unstructured text and image data or time series forecasting is offered. Moreover, TPOT cannot handle categorical features automatically.

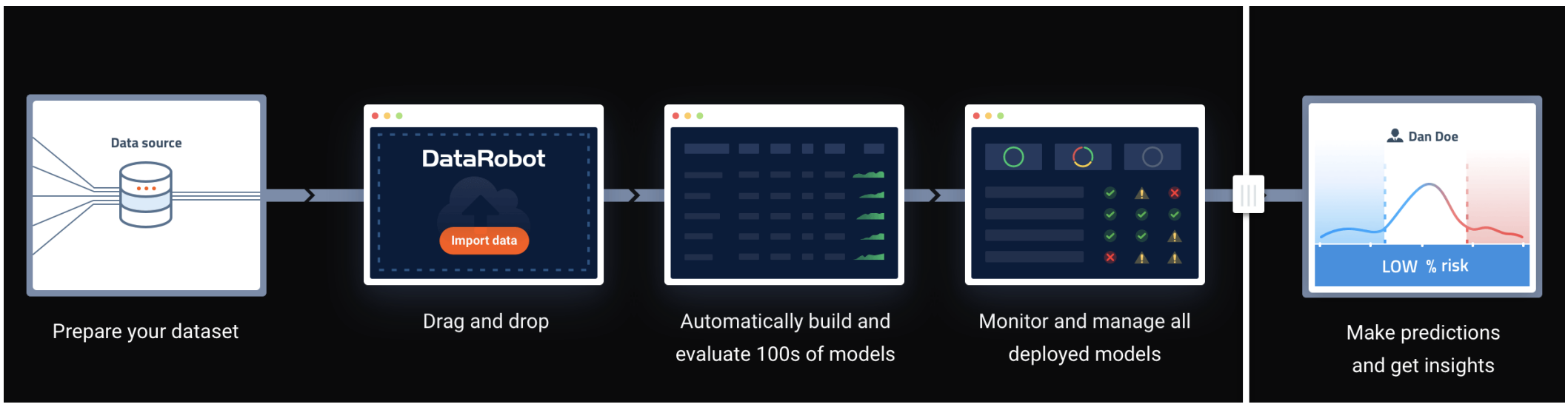
Flexibility: allows users to restrict the optimization search space in terms of feature transformers, ML methods, and their hyper-parameter choices. Moreover, users can define their own optimization score.

Scalability: supports distributed computation through dask. However, no GPU or cloud support is provided. In addition, to enhance scalability TPOT allows users to specify a subset of features, e.g., based on their domain knowledge, to be included in the genetic optimization process.

Transparency/Trust: an API documentation and some references examples are provided.

Easy of use: is open source and easy to install. However, no GUI is provided and only a Python API is available. Users can also use TPOT through the command line.

DataRobot



How DataRobot Delivers Enterprise AI

DataRobot launched their AutoML solution in late 2015, which make it one of earliest enterprise AutoML solutions. The solution is primarily focused on speedy model development and deployment, as well as ease of usage. DataRobot aims at creating a new class of "citizen data scientists" capable of creating ML solutions without having a deep ML expertise. Check out this video for a quick overview of DataRobot AutoML solution.

Functionality: provides automated feature engineering, model selection and learning functionalities. Provides a leaderboard of candidate solutions where users can rely on a speed vs accuracy plot to pick the desired solution while maintaining a tradeoff between model efficiency and complexity. ML model ensembling in form of stacking is available although it is only applicable to relatively small datasets due to computational cost. In addition, a suite of exploratory data analysis capabilities are provided. Users can perform model deployment as a REST server or export a scoring code that can be run in any Java environment. Once deployed, a model governance service offers several features such as drift detection, prediction accuracy monitoring, and service health stats.

Versatility: can tackle classification, regression and forecasting type of ML problems. Moreover, problems involving structured (aka tabular), as well as unstructured text data are supported. Uses top open-source ML algorithms such as deep neural networks based on TensorFlow, H2O, XGBoost, and DMTK. In addition, offers built-in support for automatically dealing with categorical and numerical features.

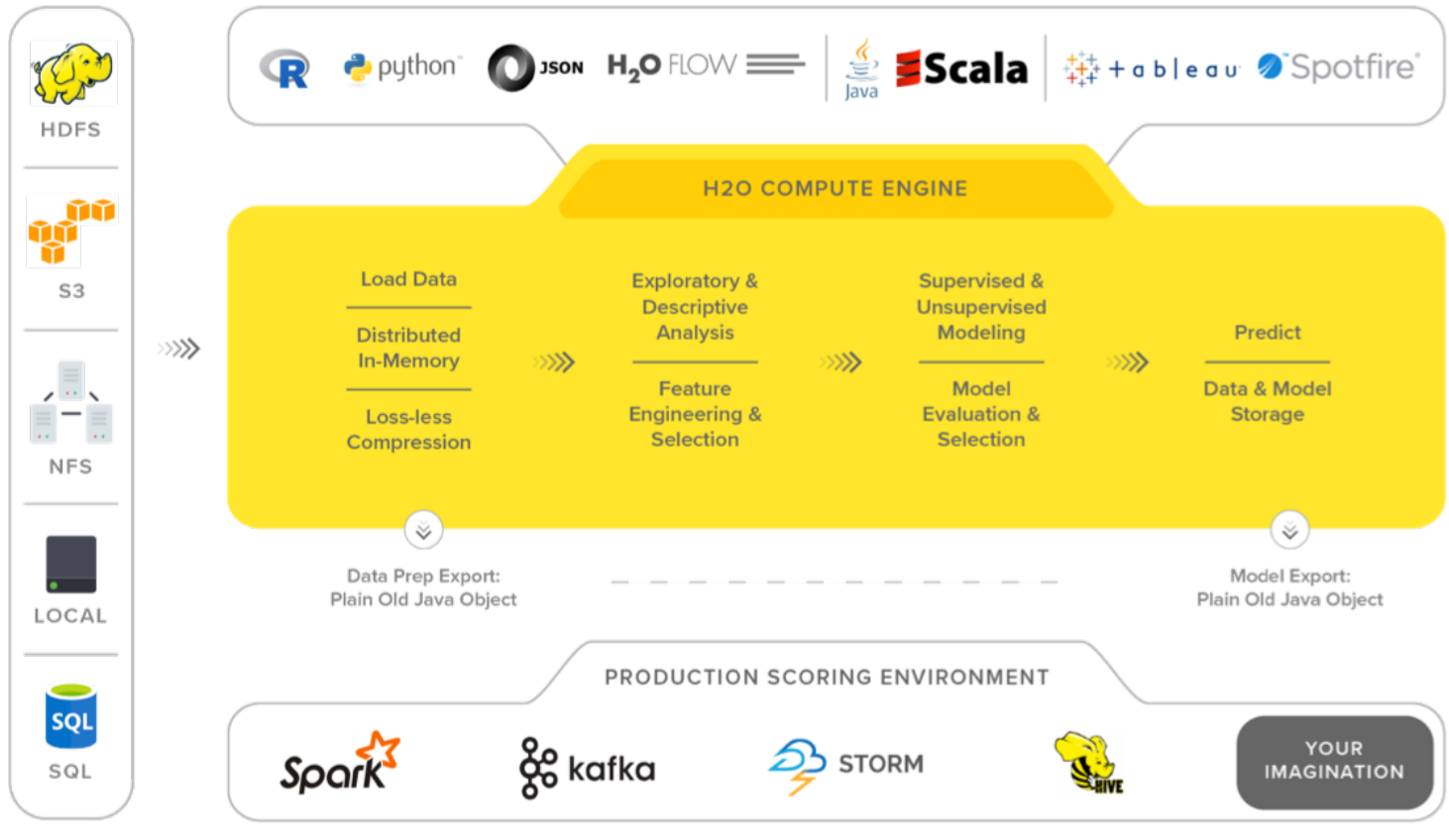
Flexibility: users can control various parameters of modelling process, e.g., adjust the cross validation folds through a GUI. Defining custom models or metrics does not seem to be possible at this time.

Scalability: uses parallel processing to efficiently evaluate multiple ML solutions. Moreover, it can be run on a Hadoop or Spark cluster and is available as a cloud solution.

Transparency/Trust: several types of model explanations are provided. First, feature impact obtained through random shuffling of given feature values and observing its impact on prediction accuracy. Second, feature effect, which is similar in nature to partial dependency plots. Third, feature fit that highlights parts of dataset where model has a systematically low performance. Finally, variable effects (aka feature contribution) type of explanation per model prediction are provided. Model blueprints, an automatic visualization of combinations of data preprocessing steps and machine learning algorithms comprising each candidate ML solution, are also offered to improve transparency. The provided documentation is in form of API reference and Wiki. While the API reference is comprehensive, the Wiki pages seem to lack some details. Finally, an automatically generated compliance document is provided to facilitate model deployment in regulated industries.

Easy of use: is an enterprise solution and requires a commercial license. Offers a clean GUI for users to interact with the solution. A Python client API is also provided. Can be installed on premises, as well as most major cloud platforms.

H2O AutoML



The H2O architecture

H2O is a popular platform for developing ML solutions at scale using in-memory and distributed computations. It enjoys widespread adoption and is estimated to be used by over 18,000 enterprises worldwide. H2O AutoML, introduced in 2017, is based on the H2O as its ML backend and can be used for automating ML workflow at scale. A recent survey shows H2O AutoML to also enjoy an impressive adoption rate among other AutoML solutions. Check out a tutorial on H2O AutoML here.

Functionality: performs model learning and hyper-parameter tuning at scale for a random grid of base models. The final model is constructed as an stacked ensemble of these base models. As such, H2O AutoML automates model selection, learning, and finalization steps of the ML workflow. The final model can either be downloaded as a binary or be packaged as a MOJO (Model Optimized Java Object) that can be easily embedded into any Java environment. The MOJO method makes model deployment significantly easier and very flexible. In fact, a diverse set of design patterns are outlined for deployment of MOJOs. Please note the MOJO does not include any data pre-processing or feature transformation steps. Finally, before deployment users can perform model validation using several performance metrics and graphs.

Versatility: can handle classification and regression type of ML problems. The suite of base learners offered includes common ML methods such as GBM, XGBoost, GLM, and RF. Both numerical and categorical features are directly supported. Unstructured data such as text or image are not supported yet. In addition, time series forecasting problems cannot be tackled.

Flexibility: users can assert some control over the AutoML optimization process through a set of parameters. These parameters dictate choices such as allotted time to the AutoML process, max number of models to be considered, which model types to be included and/or excluded, optimization stopping criteria, and more. However, users do not have control over the hyper-parameter values considered by the grid search yet.

Scalability: supports distributed computation natively on Hadoop clusters, as well as on Spark clusters through the Sparkling Water. In addition, its XGBoost type of models can leverage GPUs for improved efficiency. H2O is available on all major cloud platforms.

Transparency/Trust: supports model explanations in form of feature

contribution per prediction using the TreeSHAP method. Relative importance of features for all predictions are also provided for tree ensemble type of ML models. Moreover, users can enforce monotonicity constraints to developed base models to enhance their explainability. H2O comes with a comprehensive documentation and large community of active users. Finally, users have access to a set of logs to enable debugging and auditing tasks.

Easy of use: is open source and easy to install. A notebook style web UI called Flow is provided. Moreover, it has both Python and R APIs.

AutoKeras

AutoKeras System Overview

AutoKeras was first introduced in 2017 by researchers at the Texas A&M University. It has been actively developed ever since with the version 1.0 being recently released. Is relies on a popular deep learning library called Keras, as well as TensorFlow, as its ML backend. You can check out this quick tutorial on how to develop a simple binary classifier using AutoKeras here.

Functionality: enables efficient architecture search and hyper-parameter tuning of deep neural network models using a novel Bayesian optimization approach. As such, automatic model selection and model learning is

provided. Moreover, automatic feature engineering, at least to some extent, is implied. This is because with deep neural networks under the hood, feature engineering is (arguably) considered to be reduced to architecture search. The model architecture discovered by the AutoKeras can be easily exported as a Keras model for deployment.

Versatility: can tackle classification and regression type of ML problems involving tabular data, as well as unstructured image or text data. Time series forecasting is not offered yet. Automatic handling of both categorical and numerical features is provided. Finally, AutoKeras supports complex problems involving multi-modal data, e.g., image data along with its meta-data, and multi-task learning, i.e., predicting multiple targets from the same input features.

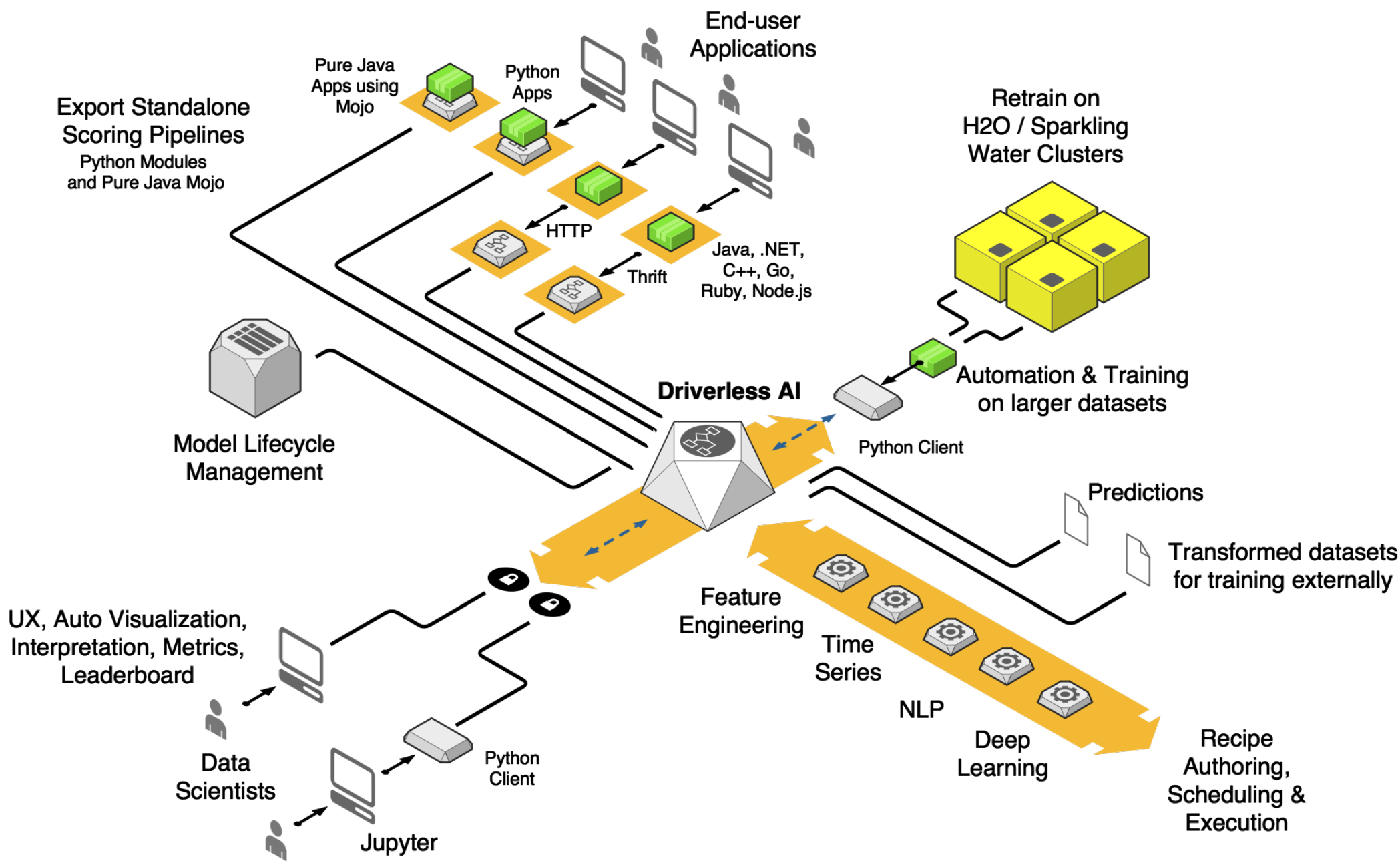
Flexibility: allows users to specify their own neural network building blocks and high-level architecture in order to customize the model optimization process. The recurrent type of neural network models are not supported yet.

Scalability: runs in parallel on multiple CPUs and GPUs for improved efficiency. No cloud or distributed computation is offered.

Transparency/Trust: an API documentation, a set of tutorials, and a few basic reference examples are provided.

Easy of use: is open source and easy to install. No GUI is available and only a Python 3 API is provided.

Driverless AI



The Driverless AI architecture

Driverless AI is an enterprise solution for automating the end-to-end ML workflow. It was introduced in 2017 by H2O.ai, the same company offering the H2O platform discussed earlier. While the open source H2O AutoML is mainly focused on scalability, Driverless AI is focused on providing a wider suite of capabilities aimed at making expert data scientists much more efficient. You can check out an end-to-end demo of Driverless AI here.

Functionality: provides a suite of capabilities to fully or partially automate several steps of the ML workflow. Driverless AI provides automatic detection of basic data types and visualization of data to facilitate data preparation, e.g. detecting outliers. In addition, it uses a sophisticated genetic algorithm to perform simultaneous feature engineering, model selection, model learning, and model finalization. Several performance evaluation metrics and diagrams are provided to enable model validation and diagnostics. The final scoring pipeline, made of feature transformations and an ensemble model, is automatically packaged for deployment as either a MOJO or a Python whl file. In particular, the MOJO scoring pipeline is suitable for applications requiring real-time scoring and can be easily

ported and deployed on any Java supporting environment including Amazon Lambda and a REST server.

Versatility: can tackle classification, regression, and forecasting type of problems involving structured data, as well as unstructured text data. Both numerical and categorical types of feature are handled automatically. Driverless AI offers a wide suite of popular ML algorithms including XGBoost, LightGBM, GBM, Decision Trees, Isolation Forest, RuleFit, and deep neural networks based on TensorFlow.

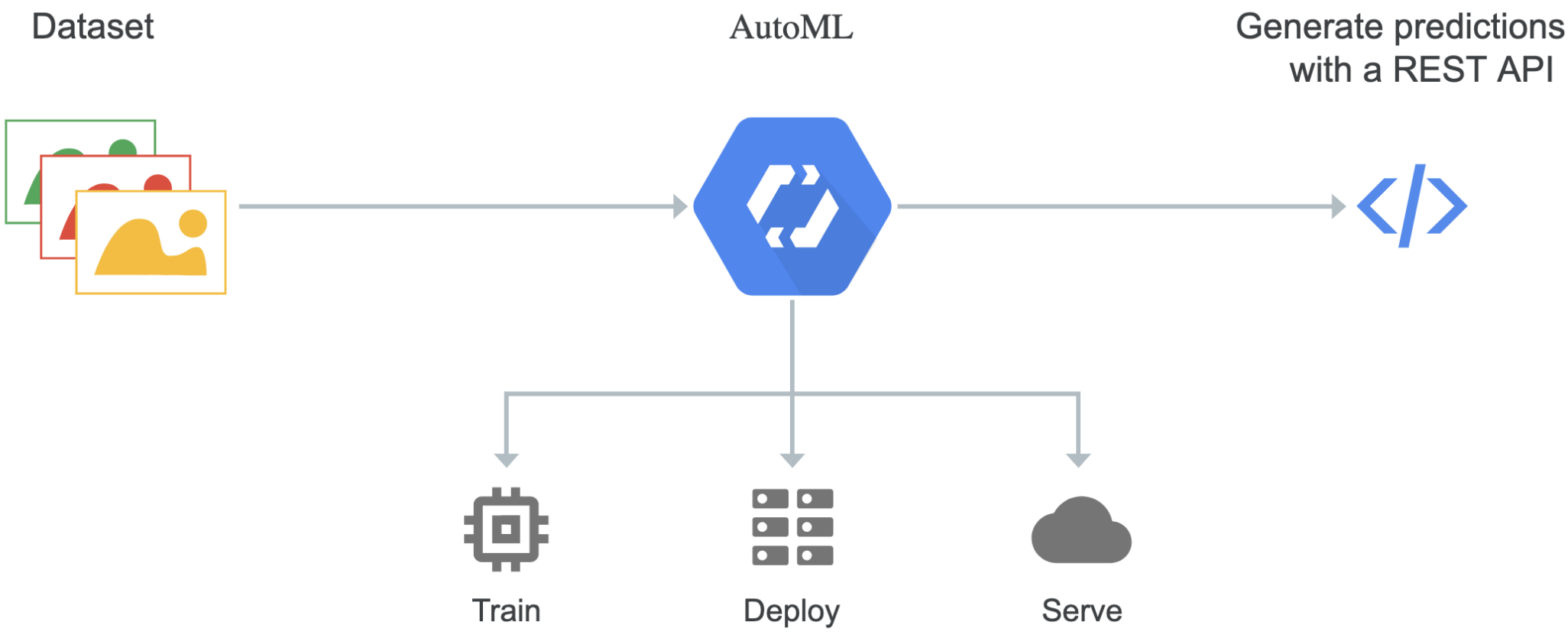
Flexibility: highly customizable through an extensive set of expert settings, as well as the BYOR (Bring Your Own Recipe) feature that enables users to define custom data sources, feature transformations, ML models, and optimization scores. Moreover, an extensive set of open source custom recipes are offered by H2O.ai. Finally, users can easily dictate a tradeoff between their desired model accuracy, experimentation time, and model explainability using three knobs.

Scalability: can leverage multiple CPUs and GPUs to speed up experimentation and is available on all major cloud platforms. No distributed computation is offered.

Transparency/Trust: provides an extensive suite of capabilities including several types of model explanations such as Shapley plots, partial dependency plots, and decision tree surrogate, as well as model fairness assessment through disparate impact analysis, and model sensitivity analysis. Moreover, an auto-generated report is provided that details all steps taken by Driverless AI to develop the final scoring pipeline. This is complemented by the ability to visualize the scoring pipeline to further enhance transparency. Finally, Driverless AI provides an extensive suite of logs to enable auditing and advanced debugging tasks.

Easy of use: is an enterprise solution and requires a commercial license to use. However, Driverless AI is also offered through a special academic licensing program. Moreover, users can easily try Driverless AI for free for a time period of three weeks. Driverless AI installation is relatively easy and can be performed as native install on several operating systems or through Docker or cloud-specific images. Users can interact with Driverless AI through both its elaborate GUI or Python and R client APIs.

Google AutoML



How Google AutoML works

Google cloud provides a suite of products to facilitate implementation of ML workflow using a machine learning as a service business model. Google Cloud AutoML was introduced in early 2018 to offer automated machine learning capabilities to primarily non ML experts. The initial release was limited to vision applications only. Users can leverage Google's proprietary pre-built models, as is, using simple APIs. Alternatively, if necessary, users can customize the pre-built models to meet their business needs using transfer learning and neural architecture search technologies. Check out a quick demo of Google AutoML for tabular data here.

Functionality: provides a data labeling service to help with data preparation step of the ML workflow. For unstructured data, Google AutoML Vision, NLP, and Video services rely primarily on deep neural networks where feature engineering is believed to be equivalent to architecture search. For tabular data, Google AutoML Tables automatically applies a set of common feature engineering functionalities. Moreover, automatic model selection and learning are provided. Several metrics of model performance and graphs are offered to help users perform model validation and pick a final model. The final model can be exported into a REST server wrapped as a Docker image that is highly portable and can be easily deployed. Users can also export their model in a format optimized for deployment on edge devices. Once deployed, a number of Google cloud services can be used to perform tasks such as continuous evaluation of model performance and model versioning.

Versatility: can solve classification and regression problems involving unstructured text, image, and even video data, as well as structured data. The ML models used for unstructured data are primarily based on deep neural networks. For structured data, a number of popular ML models including linear, gradient boosting trees, and AdaNet are used. In addition, users can develop custom language translation models. Time series forecasting problems are not supported at the moment. Finally, both numerical and categorical feature types are handled automatically.

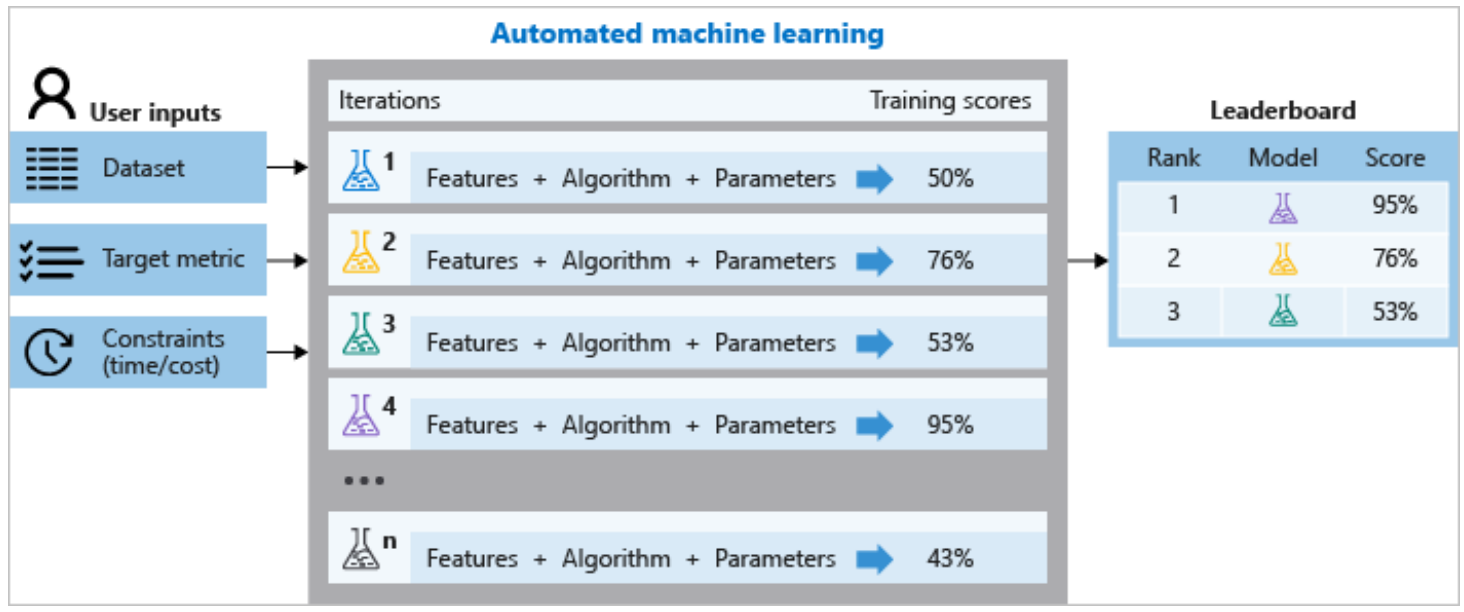
Flexibility: users can use some of client API parameters to specify AutoML optimization time budget and objective. Google's AutoML is targeted at non ML experts and, as such, offers fairly limited customization capabilities in terms of model development and optimization process.

Scalability: is cloud based and scalable although it still has limitations in terms of dataset set size and dimensions. Efficiency is enhanced through parallelizing of model learning process. Furthermore, users can use cloud instances with GPU or TPU for higher efficiency.

Transparency/Trust: Two per prediction (local) explanations are offered both of which are of feature attribution type. The first is based on the integrated gradient method applicable to ML models with differentiable objective function, e.g. deep neural networks. The second is based on the Shapley values method applicable to ML models with non-differentiable objective function, e.g. tree ensembles. In addition, global feature importance explanations are provided. Moreover, the What-If tool allows performing sensitivity analysis to gain a better sense of model robustness. A set of logging services including model architecture logs and audit logs are provided to enhance transparency. The provided documentation is extensive and includes user guides, APIs reference, and how-to tutorials .

Easy of use: is an enterprise cloud solution where users pay for the services they use over time, e.g. pay by hour for time spend training models. It has a web UI and is relatively easy to manage and use. Client API libraries are provided in several languages including Python, Java, PHP, and more. Although, the language options vary depending on the specifically chosen AutoML service.

Microsoft Azure AutoML



How Azure AutoML works

Similar to Google, Microsoft has been offering a suite of ML products as part of its Azure cloud computing platform for some time. Azure AutoML was introduced in late 2018 as part of Azure ML studio, Microsoft's main ML as a service offering. It is mainly targeted at non-ML experts and is meant to provide them with tools for rapid ML model development and deployment. Check out an introduction and demo of Azure AutoML here.

Functionality: offers a data preview and statistics feature to allow users validate their data before developing models. Moreover, a data labeling service facilitates manual labeling of large volumes of data. Automated feature engineering, model selection, and model learning services are also provided, which output a leaderboard of candidate ML pipelines. Various performance metrics and charts are provided to allow users select the best ML pipeline for a given application. The chosen pipeline can be easily deployed on either ACI or AKS compute engines with a REST endpoint. Alternatively, users can download the pipeline as a python pickle file whose dependencies are specified as JOSN and Conda *yml* files. Once deployed, a model management service offers various services such as data drift detection.

Versatility: tackles classification, regression, and time series forecasting ML problem types. Can automatically handle numerical and categorical feature types, as well as unstructured text data through feature engineering. A rich set of models are available for each of the classification, regression, and forecasting problem types. Azure AutoML supports deep learning for classification problems only at this time. Model ensembling through voting and stacking is also supported.

Flexibility: offers customization through advanced settings such as exit criteria specification, model validation approach, number of concurrent

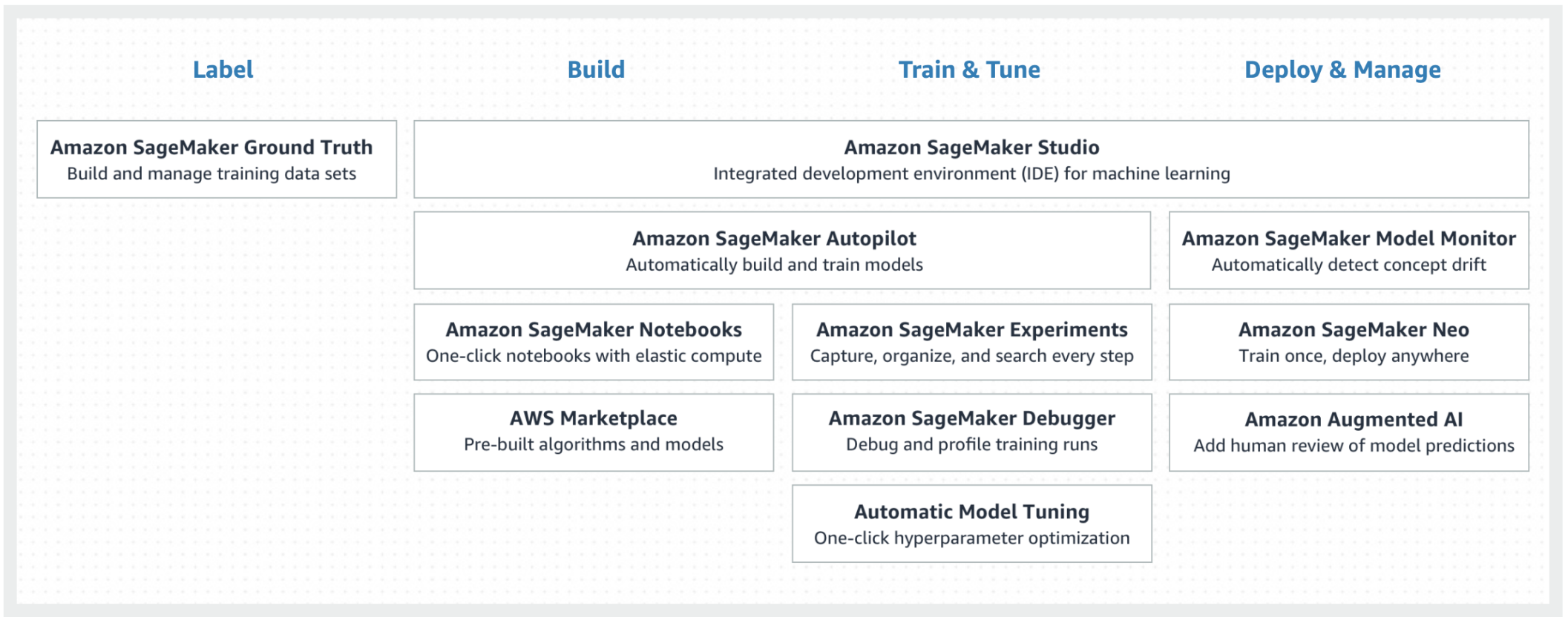
model trials, ML methods to include/exclude, and more. The Python API provides a much finer level of control through a configuration object.

Scalability: offers distributed computation and auto-scaling across a cluster of CPU or GPU nodes in the cloud for high scalability.

Transparency/Trust: provides both local (per prediction) and global feature importance type of explanations. The local explanations are obtained using a different variants of the Shapley values and the global explanations are derived using random shuffling or surrogate model (mimic) techniques. An ML pipeline graph, provided as JSON file, as well as a suite of logs are provided to improve transparency. An extensive documentation is provided including how-to guides, API references, and tutorials.

Easy of use: is an enterprise cloud solution and users are charged only for the computational and storage resources they use. APython SDK and the Azure Studio UI are the available interfaces.

Amazon SageMaker Autopilot



Amazon SageMaker suite of tools

SageMaker is Amazon's suite of tools to ease implementation of different steps of the ML workflow. Autopilot is the AutoML component of the SageMaker ecosystem introduced in late 2019. It is promoted as an AutoML solution targeted at non ML experts but also offering enough visibility and control to also be useful to ML experts. Check out this video series for a

walkthrough demo of the SageMaker Autopilot.

Functionality: users are provided with an automatically generated data exploration notebook to gain a better understanding of their dataset and eliminate potential issues before starting with modelling task. Autopilot performs automatic feature engineering, model selection and model learning and packages everything as an Inference Pipeline. The SageMaker Inference Pipeline allows users to combine data pre-processing, prediction, and post-processing tasks into a single deployable artifact. The pipeline can be easily deployed onto an EC2 instance. Autopilot yields a leaderboard of all evaluated and tuned pipelines. To select the final pipeline, users are provided with performance metrics and charts in terms of both model accuracy, latency, and size. In addition, users can use SageMaker Ground Truth in tandem with the Autopilot to facilitate data preparation tasks. SageMaker Neo aims to automatically compile an inference pipeline to run in the most efficient manner on a target platform, e.g. ARM, Intel, or NVIDIA processors. Lastly, SageMaker Model Monitor service continuously observes deployed models in production and detects issues with model quality such as data drift.

Versatility: can tackle both classification and regression type of ML problems . It is limited to tabular data for now. Moreover, regression is limited to linear regression only. On the other hand, users can use the DeepAR method along with SageMaker's hyper-parameter tuning service to tackle time series forecasting problems at scale with little manual intervention. Both numerical and categorical feature types are handled automatically.

Flexibility: users can directly "hand tune" the process of developing each candidate model by directly modifying its original Python code, presented as a commented notebook, by the Autopilot. Moreover, users can define tuning metrics and specify hyper-parameter range. Finally, users can define custom alerting schemes for the model monitor service.

Scalability: is a cloud based solution and easily scalable. Supports GPUs to further speed up the training process specially for deep neural network based models. The hyper-parameter optimization (HPO) of Autopilot can run multiple parallel jobs and uses a Bayesian optimization approach to enhance efficiency. Supports distributed model learning on multiple EC2 instances as well. For deployed models, SageMaker provides an automatic scaling service.

Transparency/Trust: to provide full transparency, users can see the original code, as a commented Python notebook, used to develop each candidate model. However, no model explanation capabilities are provided a this time. Extensive logging is offered through the Amazon CloudWatch service. The Documentation includes elaborate developer guides and an API reference.

Easy of use: is an enterprise cloud solution, is fairly easy to manage and use within the AWS ecosystem. Its pricing model is similar to that of Google and Azure AutoML discussed earlier. Provides a web UI and a Python client API.

Other Noteworthy Solutions

There are some other, perhaps lesser know but still interesting, solutions that are worth mentioning:

- **Uber's Ludwig**: is an open source toolbox based on TensorFlow for developing deep neural network based models with minimal coding. Moreover, Ludwig enables distributed training of models on multiple GPUs or even multiple nodes using Uber's Horovod platform. Check out an introductory presentation on Ludwig here.
- **Saleforces's TransmogrifAI**: is an open source library developed in Scala and based on Spark to enable automated feature engineering, model selection and learning at scale. Check out an overview of TransmogrifAI here.
- **Darwin**: is an enterprise solution by SparkCognition for automatic feature engineering, model selection and learning. It also provides some functionality through an interactive UI to enable data preparation. Check out a quick overview of Darwin here.
- **DARTS**: is an algorithm to automate architecture design for deep neural network based models. It has been shown to yield high-performance convolutional and recurrent network architectures using a continuous relaxation, hence differentiable, architecture representation. Importantly, DARTS promises to be orders of magnitude more efficient than conventional neural architecture search based on evolutionary or reinforcement learning methods. A nice overview of neural architecture search methodologies is presented here.
- **Featuretools:** is a open source Python implementation of the Deep Feature Synthesis algorithm, for automatic feature engineering using relational data, e.g., a multi-table dataset.

- **Seldon:** is an open source platform that makes it easier to deploy ML models at scale on a Kubernetes cluster. The models can be developed in Python, R, Java, NodeJS, and even Go, and are wrapped as docker container before deployment.

Final thoughts

As interest in AutoML continues to surge, its solutions landscape will become increasingly more competitive and evolve faster. That being said, there is still plenty of room for more widespread adoption of AutoML. Early AutoML solutions were primarily developed by academic researchers. They were primarily focused on automating model selection and model learning steps of the ML workflow and were simple to use. However, early AutoML solutions often offered primitive interfaces and little or no support for workflow steps beyond modelling such as feature engineering, model validation, deployment, and monitoring.

Modern AutoML solutions' treatment of the ML workflow automation is much more comprehensive and includes several steps. This is particularly the case for enterprise and cloud based solutions. Beyond automating ML workflow functionalities, they also offer improved scalability, flexibility, versatility, and transparency/trust capabilities. In particular, explainability and transparency are already important requirements for successful adoption of ML solutions in enterprise specially in highly regulated industries. When using AutoML to develop ML solutions, they become even more important.

Non open source AutoML solutions have the obvious drawback of licensing or usage cost. In addition, they are often more sophisticated and come with an initial learning curve to master. However, these initial time and money investments, could be well worth it in the long run due higher overall productivity of ML experts through AutoML. That being said, most cloud based AutoML solutions are tightly integrated with the rest of cloud based offerings of their vendor. This make the threat of cloud vendor lock-in a real possibility.

Comparative studies of AutoML solutions are far and few between. A benchmark to compare AutoML solutions was recently published where all of the open source solutions discussed in this article, except AutoKeras, are evaluated across 39 datasets. Two set of experiments are conducted using default parameter settings of all solutions and time budgets of 1 and 4

hours. A tuned random forest model is used as baseline. None of the solutions consistently outperforms the rest. AutoWEKA tends to overfit when running for a longer time especially on multi-classification problems and yields the poorest overall performance. Some experiments involving high dimensional data or highly multi-class problems are challenging for AutoML solutions as none of them outperform the baseline. In addition, problems involving imbalanced datasets are challenging for some solutions.

Another more recent and comprehensive study of AutoML solutions evaluates a large number of AutoML solutions in terms of their suite of functionalities, as well as empirical performance across nearly 300 datasets. In terms of functionality, open source solutions are reported to often have a more restricted suite of offerings, which is agreeable with our observation. The empirical study does not include some of the enterprise solutions or those lacking a Python wrapper interface. Once again, none of solutions consistently outperforms all of others. However, when accounting for combined computational efficiency, predictive performance, and stability H2O AutoML and AutoKeras yield the best overall results. For binary classification and regression problems H2O AutoML rapidly and consistently converges to best performing solutions. For multi classification problems, AutoKeras performs slightly better than the rest.

Current AutoML solutions have come a long ways since their early days few years ago. Modern solutions offer a lot of functionality while maintaining relatively high flexibility, versatility and ease of use. Some solutions also offer high scalability and emphasis on transparency/trust. However, there are at least two main areas with room for improvements. First, automatically tackling complex ML problems involving high-dimensional data, many or highly imbalanced classes remains a challenge. Second, almost all existing solutions are dedicated to supervised learning problems and little attention has been paid to unsupervised or reinforcement learning type of ML problems.

I hope this this two part blog post series to have provided you with a solid understanding of AutoML fundamentals and its state-of-the-art solutions. Please feel free to reach out to me on LinkedIn or leave a comment below if you have any questions or suggestions.

Sign up for Analytics Vidhya News Bytes

By Analytics Vidhya

Latest news from Analytics Vidhya on our Hackathons and some of our best articles! Take a look.

Your email

Get this newsletter

By signing up, you will create a Medium account if you don't already have one. Review our Privacy Policy for more information about our privacy practices.

AI | Machine Learning | Automl | Automation | Enterprise Ai