Introducing AutoML

—

Paco Nathan
IBM Data Science Community

Part 1:
“You probably have a gradient problem”

Part 1:
“You probably have a gradient problem”

Show a brief example of supervised learning, which illustrates:

- *Learners*
- *Loss Function*
- *Regularization Term*
- *Gradient Descent*

A look inside a machine learning algorithm (1 of 2)

Part 1:
“You probably have a gradient problem”

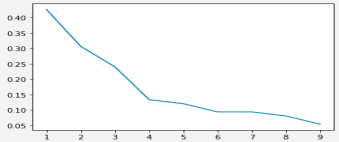
Show a brief example of supervised learning, which illustrates:

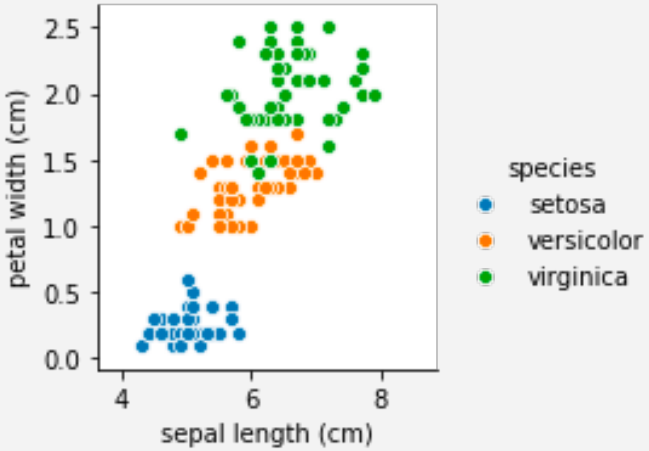
- *Learners*
- *Loss Function*
- *Regularization Term*
- *Gradient Descent*

Decision trees

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$$

$$\min_{f \in H} \left[\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \right]$$





See notebook: [iris.ipynb]

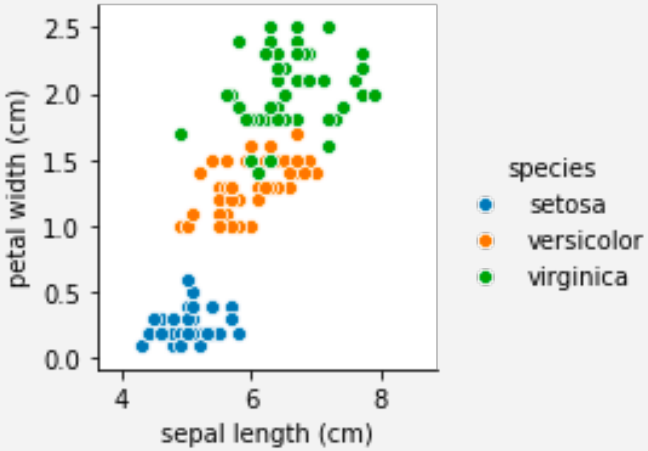
Part 1:
“You probably have a gradient problem”

A look inside a machine learning algorithm (2 of 2)

Let’s distinguish between:

- *Parameters*
  - Elements for how a model has learned to differentiate a gradient
- *Hyperparameters*
  - Used to guide models to search for optimal parameters

Part 1:
“You probably have a gradient problem”



Let’s distinguish between:

- *Parameters* **≤ log(4 dimensions) × 10 trees**
  - Elements for how a model has learned to differentiate a gradient
- *Hyperparameters* **N = 10**
  - Used to guide models to search for optimal parameters

See notebook: [iris.ipynb]

Supervised Learning :

- Differentiate gradients of input features (indep. variables) to predict labels

Unsupervised Learning:

- Mostly the same as supervised learning, except that the algorithm generates proxies for labels

Active Learning:

- Mostly the same as supervised learning, except that people override label predictions when ML model results have high uncertainty; *aka active learning, aka human-in-the-loop*

Self-Supervision:

- Create learning tasks among parts of different views of an encoded object to infer proxies for labels; then mostly the same as Unsupervised Learning above, construct a gradient, then mostly the same as Supervised Learning above

Deep Learning:

- Many layers of neural networks for the Supervised Learning above, connected by autoencoders, etc.

Reinforcement Learning:

- 1+ agents explore/exploit a simulation to construct a gradient; then mostly the same as Supervised Learning above

Weak Supervision:

- Leverage relatively noisy input from subject matter experts, modeled with label functions; then construct a gradient about the human + machine decisions that go into constructing the training set used for Supervised Learning above

Transfer Learning:

- Reuse a model previously trained on other data by fine-tuning high level decisions with your data; then mostly same as Supervised Learning above – especially used in deep learning

Knowledge Graph:

- Graph representation of the context for features used to train Machine Learning models, which can construct a gradient to be differentiated by graph embeddings used for features in the Supervised Learning above

Meta-Learning:

- Leverage relatively noisy input from subject matter experts, modeled with label functions; then construct a gradient about the human + machine decisions that go into constructing the training set used for Supervised Learning above

Meta-Learning

That last definition, *meta-learning*, can be viewed as a combination of...

- Knowledge graph (or database)
- Weak supervision
- Active learning
- Potentially some human-in-the-loop

… all used to build ML workflows

And all of the above pretty much describes the scope of where AutoML picks up, so far!

Part 2:
“But it’s really a search problem”

Formulating ML Problems as Search

Formulating ML problems as Search (1 of 5)

The challenge of building ML models is to find the best hyperparameters for a given use case and set of input data.

That isn’t a simple task. When we develop ML models, we must address several challenging and sometimes conflicting questions...

Formulating ML problems as Search (2 of 5)

- Which ML algorithms would fit well with the use case?
- How much training data is needed or even available?
- Which features in the data are best to use for training models?
- What hyperparameter settings produce the best results?
- How to evaluate different models and compare their relative trade-offs?
- How could the features be transformed prior to training to improve results?
- Have any assumptions introduced potential risks for the use case?
- Will the resulting models satisfy requirements for the production environment?

Formulating ML problems as Search (3 of 5)

Other caveats to keep in mind:

- Many human decisions must go into the mix
- Hiring lots of AI expert talent is difficult and quite expensive
- Data science teams really benefit from diversity: it's a "team sport" where many different perspectives combine to build solutions

Formulating ML problems as Search (4 of 5)

Now imagine if it were possible to look at **all** of the possible variations of input data (*all possible gradients*) and then compare those with **all** of the possible variations of ML models (*optimized parameters*)?

Imagine if many perspectives and learnings from prior use cases could be accumulated and leveraged by others?

That right there is a *search* problem!

Formulating ML problems as Search (5 of 5)

The goals of AutoML could be stated as a search problem, one which helps you to...

- Find the appropriate hyperparameters that converge to optimal after training and you've identified the path toward a good solution for an ML model
- Identify features and data transformations that produce optimal training sets, and you've enhanced the steps along that path
- Learn from history: decisions that go into collecting and preparing data; decisions about which features and model selections appear to optimize for your intended use case; and feedback that comes from running the ML model in production

Search meets other considerations

Search meets other considerations (1 of 3)

One current controversy: the “search problem” for optimizing the hyperparameters in a large ML model may be quite expensive to solve.

See **“Show Your Work”** about effective trade-offs in ML modeling algorithms **https://arxiv.org/abs/1909.03004**

See **“Energy and Policy Considerations for Deep Learning in NLP”** which describes how training the new transformer models for natural language can require up to 5 times the carbon footprint of the total lifetime of operating an automobile. **https://arxiv.org/abs/1906.02243**

Question: *do research papers represent scientific advances if their results become too difficult for others to reproduce? Or if the work required contradicts ethics and policy considerations?*

Search meets other considerations (2 of 3)

Handling the issues about Fairness and Bias in data are provably hard problems.

See "Why it's hard to design fair machine learning models"
See "Task-agnostic Universal Black-box Attack on Computer Vision DNNs"

AI Trust:

- AIF360 http://aif360.mybluemix.net/
  - OSCON 2019: "Removing Unfair Bias" (part 1, part 2)

- AIX360 http://aix360.mybluemix.net/

Search meets other considerations (3 of 3)

For the full, end-to-end ML lifecycle, many different dimensions must be optimized:

- “accuracy” could have different meanings:
  - *precision*: how many positives are relevant
  - *recall* (aka sensitivity): identifying true positives
  - *specificity*: identifying true negatives
  - *perplexity*: variability in the data and patterns
- confidence vs. uncertainty in predicted results
- cost (computing) – how much $$ spent on training
- cost (time) – diminishing returns for $$ spent on optimization
- scalability: throughput per worker
- resource footprint (e.g., memory size, low power) for edge model drift
- fairness and bias
- privacy (leaking PII)
- security risks / attack surface
- compliance, explainability, AI trust

Part 3: AutoML in Practice

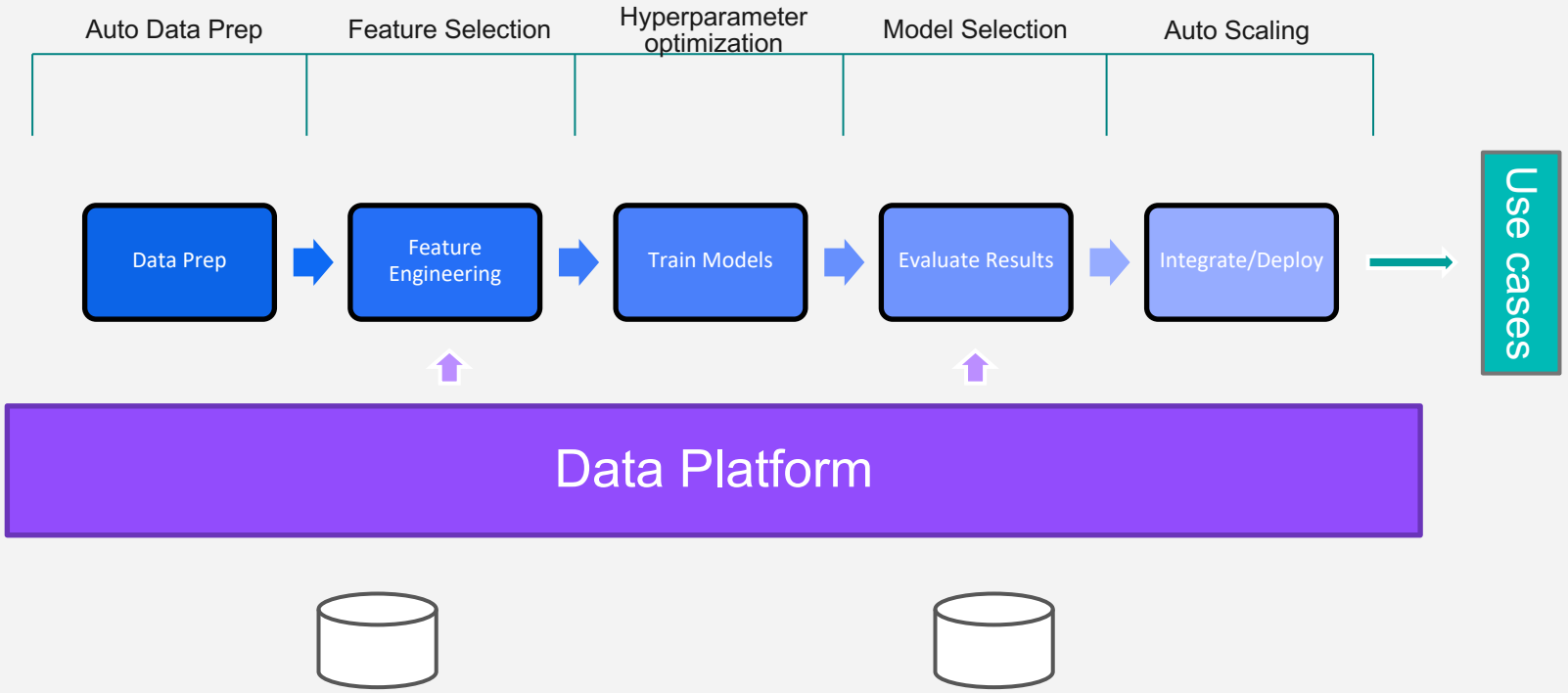
End-to-End ML Lifecycle
(1 of 2)

ML Ops is about managing the *end-to-end lifecycle* for ML models used in production.

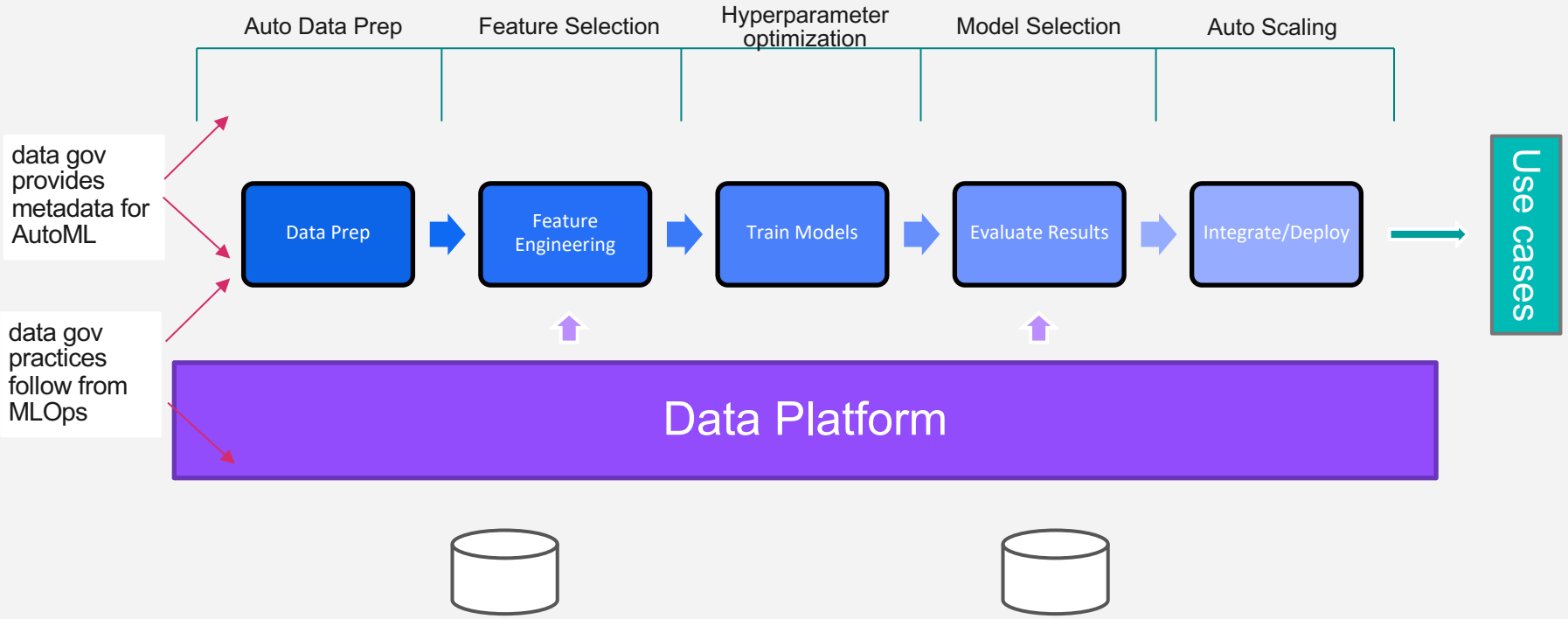
Where can automation and machine learning be applied throughout that end-to-end lifecycle?

See **“**[**ML Ops Day, OSCON 2019**]**”**

End-to-End ML Lifecycle
(2 of 2)



End-to-End ML Lifecycle
(2 of 2)



OSS related to AutoML

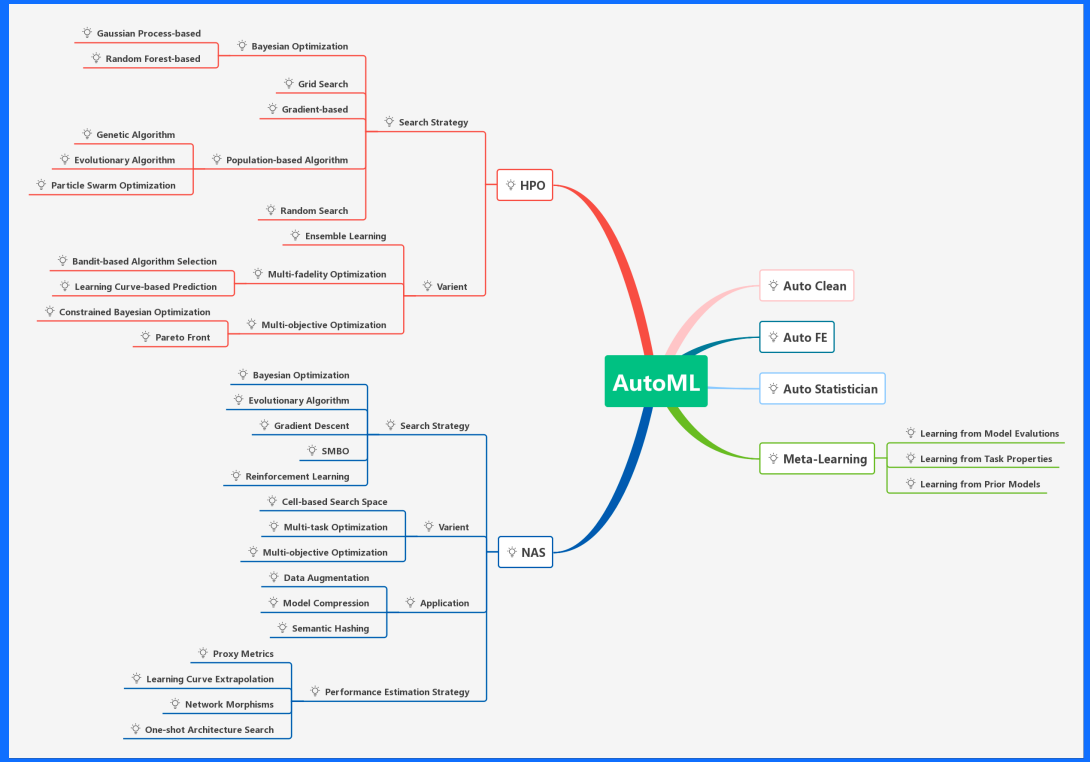
	A	B	C	D	E	F	G
1	project	sponsors	url	category	license	article	purpose
2	Amundsen	Lyft	https://github.com/lyft/amur	knowledge graph	ASL 2	https://eng.lyft.com/amundsen-ly	metadata, data catalog
3	Databook	Uber	#N/A	knowledge graph	#N/A	https://eng.uber.com/databook/	surfaces and manages metadata abou
4	Knowledge Graph	Airbnb	#N/A	knowledge graph	#N/A	https://medium.com/airbnb-engi	helps us categorize our inventory and
5	Marquez	WeWork, Stitch Fix	https://github.com/Marquez	knowledge graph	ASL 2	https://marquezproject.github.io/	collect, aggregate, and visualize a dat
6	WhereHows	LinkedIn	https://github.com/linkedin/	knowledge graph	ASL 2	https://engineering.linkedin.com	data discovery and lineage for Big Dat
7	Ground Context	UC Berkeley RISE Lab	http://www.ground-context.	knowledge graph	ASL 2	http://www.vikrams.io/papers/gr	subsumed into a larger research proje
8	Simon	Genular	https://github.com/genular	KDD, model generation	AGPL 3	https://github.com/genular/simor	AutoML back-end/front-end
9	automl-toolkit	Databricks	https://github.com/databricl	data prep, feature selection, model	Databricks	https://databricks.com/blog/2019	AutoML plug-in for MLflow + Spark, nc
10	MLflow	Databricks	https://github.com/mlflow/m	fits into meta-learning, deployment	ASL 2	https://databricks.com/blog/2019	tracks workflow, collects telemetry nee
11	AIF360	IBM	https://github.com/IBM/AIF	repair imbalanced training data	ASL 2	https://youtu.be/a0bTPMvUJXI	detect and fix bias (e.g., imbalance pr
12	PALEO	Determined AI	https://talwalkarlab.github.i	performance modeling for deep lear	ASL 2	https://talwalkarlab.github.io/pal	estimate the scalability and performan
13	TPOT	U Penn	https://github.com/Epistasis	optimize ML pipelines	LGPL 3	https://link.springer.com/chapter	optimize ML pipelines using genetic pr
14	Recipe	LAIC	https://github.com/laic-ufm	optimize ML pipelines	GPL 3	https://link.springer.com/chapter	evolve ML pipelines using genetic prog
15	TransmogrifAI	Salesforce	https://transmogrif.ai/	optimize ML pipelines	BSD 3	https://engineering.salesforce.co	compiled Scala for Spark, to automate
16	Ludwig	Uber	https://github.com/uber/lud	end-to-end ML pipelines	ASL 2	https://eng.uber.com/introducing	code-free deep learning toolbox for Ho
17	Hyperopt	Jasper Snoek	https://github.com/jaberg/hy	HPO	BSD 3	http://jaberg.github.io/hyperopt/	Python library for optimizing over awk
18	spearmint	SEAS, Harvard	https://github.com/HIPS/Sp	HPO	Harvard	https://arxiv.org/abs/1403.5607	Bayesian optimization for global optim
19	AutoKeras	DATA Lab, TAM	https://github.com/keras-te	NAS an HPO for deep learning	MIT	https://autokeras.com/	Python library for AutoML with Keras
20	Ray Tune	UC Berkeley RISE Lab	https://github.com/ray-proje	scalable HPO	ASL 2	https://arxiv.org/abs/1807.05118	scalable HPO for deep learning and de
21	auto-sklearn	AutoML-Freiburg	https://automl.github.io/aut	HPO, meta-learning, ensembles	MIT -ish	http://papers.nips.cc/paper/5872	leverages recent advantages in Bayes
22	Auto-WEKA	AutoML-Freiburg	https://github.com/automl/a	HPO, model selection	GPL 3	http://www.jmlr.org/papers/volum	automatic selection of models and hyp
23	auto_ml	Preston Parry	https://pypi.org/project/autc	most of the above	MIT	https://auto-ml.readthedocs.io/e	Python library designed for production

vendors | open source | Explore

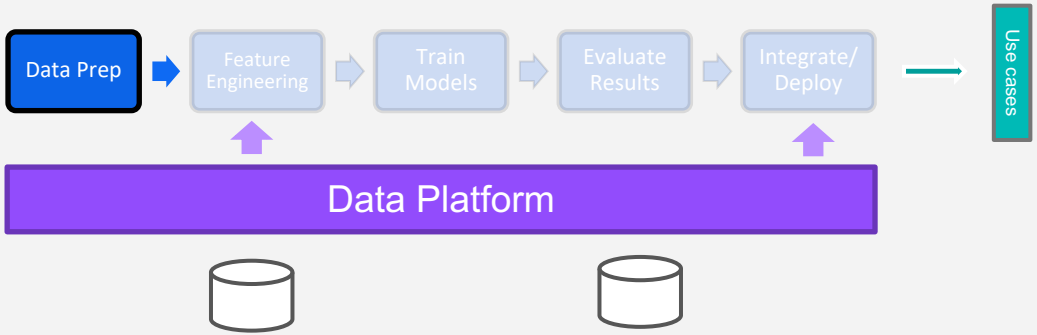
https://derwen.ai/s/fg9p

Taxonomy of AutoML

https://github.com/hibayesian/awesome-automl-papers



Stage 1: Auto Data Preparation

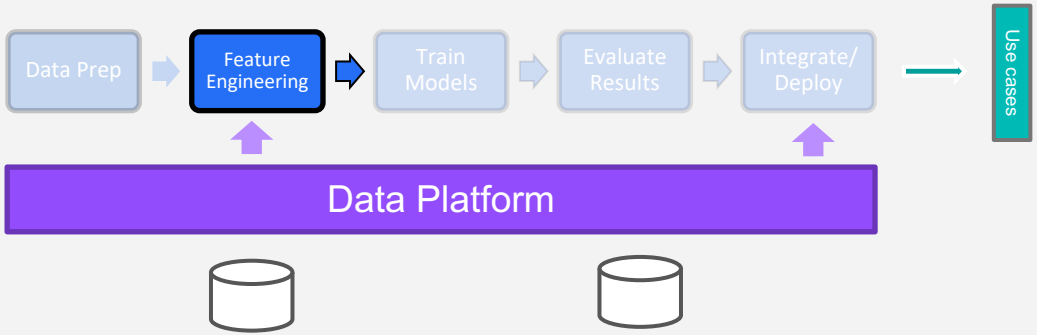


The first stage of the ML lifecycle typically requires data preparation – the proverbial 80% of time spent by data science teams. Weak supervision is indicated here, for what is increasingly called *auto data preparation*.

Examples: HoloClean, Snorkel
See "Data Cleaning is a Machine Learning Problem that needs Data Systems Help!"

See "Labeling, transforming, and structuring training data sets for machine learning"

Stage 2: Feature Engineering

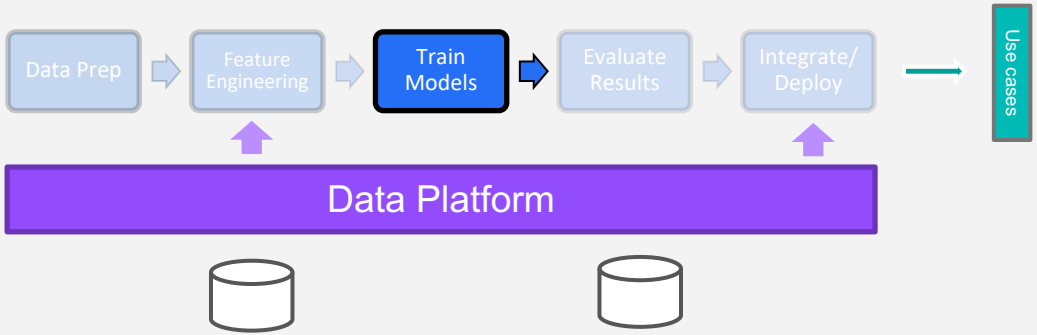


The next stage in the ML lifecycle is *feature engineering*, sometimes called "feature selection" – i.e., transform the data to create useful gradients to use during ML model training.

Examples: MLBox, automl-toolkit
See "Learning Feature Engineering for Classification"
See "On the Stability of Feature Selection Algorithms"

Recommended: "Feature Engineering for Machine Learning" by Alice Zheng, Amanda Casari

Stage 3: Hyperparameter Optimization

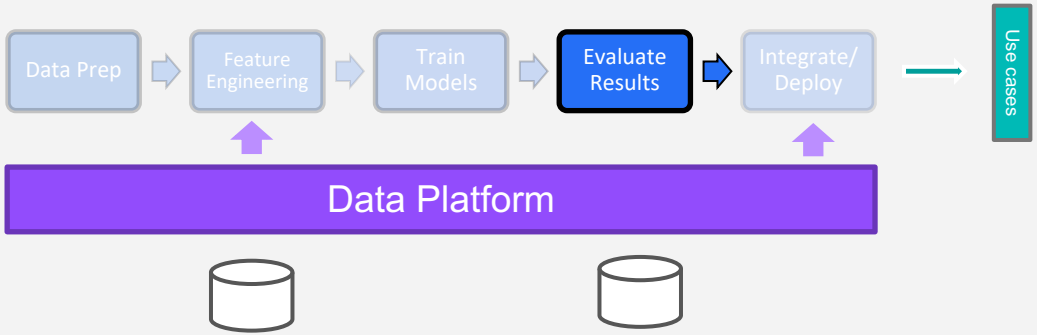


HPO: If we have training data and we have an algorithm defined by a loss function and regularization term, searching for the best hyperparameters to use is an optimization problem.

This provides yet another good use of gradients – at a meta level.

Examples: Ray Tune, HyperOpt, AutoKeras
See "Stop doing iterative model development"

Stage 4: Model Selection and Ensembles

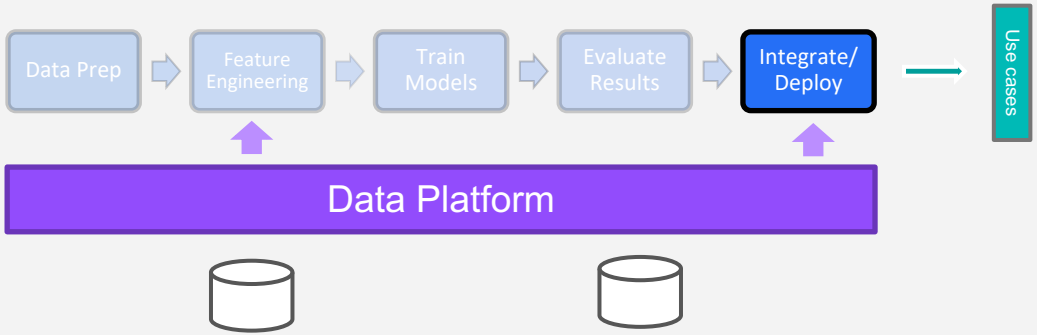


How does one select which algorithms fit best with the use case, and how to compare models that have been trained by different algorithms?

Examples: **auto-sklearn**
See "**Introduction to Ensemble Learning**"

Also: the use of ensembles – combinations of multiple ML models – is a powerful ML technique, and some AutoML tools at this stage help build ensembles.

Stage 4: Deployment



Some models get deployed as services and must consider *auto-scaling*.
Examples: Watson Studio, SageMaker, Knative

Other models get deployed embedded in mobile devices or other edge use cases. These need model compression to fit into the available system resources for memory, power, network, etc.
Examples: TensorFlow JS

See "EIE: Efficient Inference Engine on Compressed Deep Neural Network"
See tinyML Summit

Watson Studio: AutoAI vs AutoML

	AutoAI	Traditional AutoML
Transfer Learning	✓	
Neural Network Search	✓	
Data Preparation	✓	✓
Advanced data refinery	✓	
Feature Engineering	✓	✓
Hyperparameter Optimization	✓	✓
One click deployment	✓	
Explainability and de-biasing	✓	
AI lifecycle management	✓	

Your Watson Studio Trial:
http://ibm.biz/BdzAVJ

Looking Ahead (1 of 3)

Watch for more about *meta-learning*, i.e., leveraging the history of what your team (and other teams) have learned from experience with ML workflows.

In other words, implementing "semi-automated data science".

Examples: **lale**, **TPOT**

- "SmartML: A Meta Learning-Based Framework for Automated Selection and Hyperparameter Tuning for Machine Learning Algorithms" Mohamed Maher, Sherif Sakr (2019-03-26)
- "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks" Chelsea Finn, Pieter Abbeel, Sergey Levine (2017-07-18)
- "Learning to Optimize" Ke Li, Jitendra Malik (2016-06-06)
- "Cross-disciplinary perspectives on meta-learning for algorithm selection" Kate A. Smith-Miles (2009-01)

IBM

Looking Ahead (2 of 3)

Watch for more news about *program synthesis*, i.e., how to use ML to augment how people create software.

For example, tools for generating code based on examples of **dataframes**.

AI-augmented *auto-completion* features in IDEs are closely related.

Examples: **AutoPandas**, **TabNine**

IBM

Looking Ahead (3 of 3)

One controversy: is the subtext of AutoML about getting people to use a **~**100x increase in cloud-based computation, to offset the difficulty of hiring AI experts?

Then again, rapid evolution of hardware may help make data science work much faster, less expensive, and enable AutoML to flourish.

Meanwhile, keep watching this excellent resource for details about how AutoML research is evolving:

Awesome-AutoML-Papers

IBM

Conclusion

Consider the opportunities, although also keep in mind the caveats and ethical/policy considerations.

Meanwhile this field and the adjacent field of hardware accelerators is evolving rapidly ... stay (auto) tuned!

Conclusion

Consider the opportunities, although also keep in mind the caveats and ethical/policy considerations.

Meanwhile this field and the adjacent field of hardware accelerators is evolving rapidly ... stay (auto) tuned!

The point about training ML models in clever ways – in other words, more intelligently – that's where we begin to get AI helping to improve AI. That's the promise of AutoML!

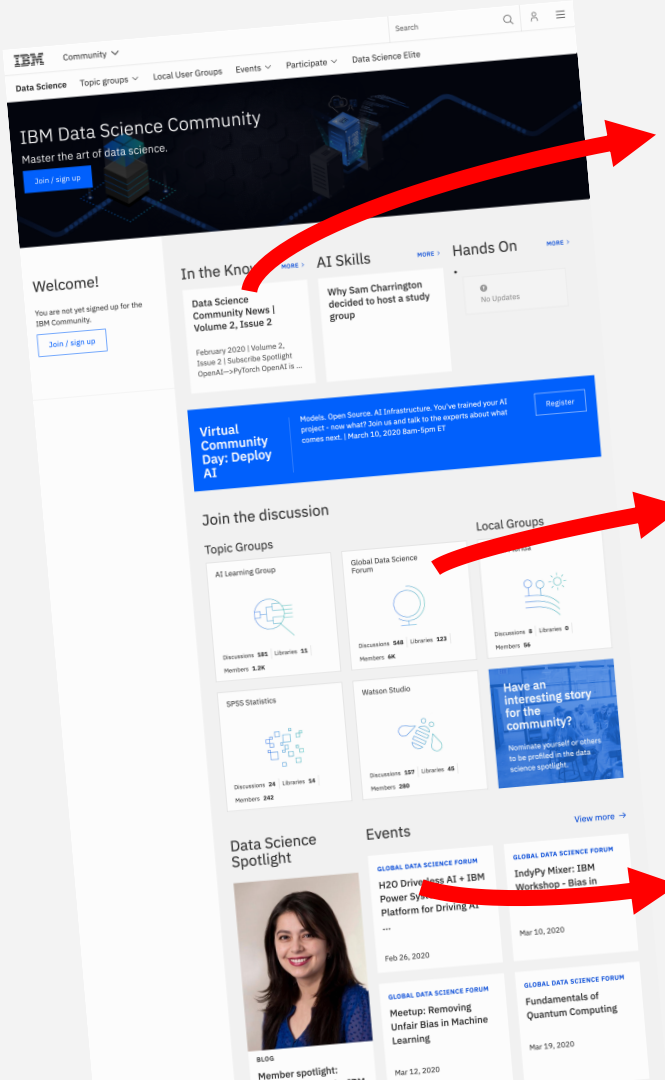
IBM

Demo: Coding Examples

- **Lale** https://github.com/IBM/lale
  - See notebooks: **demo_pipeline_hyperopt.ipynb**, **demo_aif360.ipynb**

- **Ray Tune** https://docs.ray.io/en/latest/tune.html
  - See examples in notebook: ray_tune.ipynb

Also see the article on which this talk was based, plus a related discussion thread where we can talk further about AutoML: https://ibm.biz/paco-automl-1



Learn

Resources galore & experts on tap (just ask)

Share

10,000+ members and counting!

Engage

Post blogs, start forum discussions, join webinars, online hackathons & events

Special Offer: IBM & Coursera

Join the

IBM Data Science Community

and get a complimentary month of select IBM Data Science & AI Specialization Programs on **Coursera**

http://ibm.biz/community-coursera