

DEEP LEARNING IN 2019

FROM EXPERIMENTATION
TO PRODUCTION



Contents

- 1 Introduction
- 2 Staffing the machine learning team
- 6 Level up your ML code from notebook to production
- 10 What can data scientists learn from software engineers?
- 14 How to make reproducible experiments?
- 20 The right tools for machine learning
- 25 Machine learning infrastructure for everyone

INTRODUCTION

Welcome to the Deep Learning guide book for 2019! If 2017 was about data exploration and data lakes, 2018 was definitely about model exploration and moving towards production ready model building. Moving from exploration to production-ready model building requires new ways of thinking to ensure reproducibility and team work.

This eBook starts from the most important part in your deep learning: your team. We'll continue with how to prepare your code for production ready experimentation. Deep learning code requires constant experimentation and changes in both the code as well as the training data before tuning externalized hyper parameters. We'll also touch upon one of the most popular ways of experimenting with deep learning, namely Jupyter Notebooks and why they aren't ideal for production-scale deep learning. Finally we'll round up with what a deep learning experiment consists of and how to build reproducible experiments. Lastly we'll present the most common deep learning platforms and how they differ from each other so that you can make a sound decision on what to build on top of.

We thank all our customers, partners and users from feedback in building this eBook. Deep Learning has changed tremendously in the past year and converged more towards the internal platforms of the technology unicorns that have been built with deep learning first already years before the rest. Uber's Michelangelo, AirBnB's BigHead, Facebook's FB Learner Flow and Netflix' and Google's own platforms have all been an inspiration to us at Valohai as well.

We hope this eBook sheds some lights on the deep learning scene for 2019!

STAFFING THE MACHINE LEARNING TEAM

Today's machine learning teams consist of people with different skill sets. There are a bunch of different roles that are needed when you start building production level machine learning solutions, but first, let's compare the two key roles: data scientist vs. machine learning engineer.

DATA SCIENTIST

Data scientists are people who work with data and build machine learning models. They clean and interpret data and build models using a combination of machine learning algorithms and data.

Data Scientists come often from the academic field and their background is usually in university research projects. It might be surprising how big of a technological gap there is between research and real world production systems. In theoretical or research settings, repeatability, record






3 / STAFFING THE MACHINE LEARNING TEAM

keeping, testing and collaboration play a much smaller role compared to production systems where teams of 10 or more researchers might work on the same data and models to optimize.

Researchers have a fixed dataset that they train a model on and once satisfied with the results, they write a paper and often never go back to the code ever again or actually deploy the model at scale for real world use cases. resources like GPUs.

Real world applications are fundamentally different in a few major aspects:

- Teams are larger
- People leave for other companies
- New people come in
- The world changes, data evolves and models go stale
- Models go into production and therefore must be tested and monitored
- The main driver is long term ROI instead of getting the research result



These requirements present a host of issues that require a lot of software infrastructure support. This is mainly DevOps work that involves building data pipelines, automated testing, autoscaling computational clusters and ensuring high availability for serving models.



MACHINE LEARNING ENGINEER

Machine learning engineers are the support troops of researchers and data scientists. Machine learning engineers rarely touch the models or are interested in the form or contents of the data they work with.

Machine learning engineers do anything from data lake set up and management to building easy to use computational clusters for training and finally ensuring high availability deployment of models.

Machine learning engineers come from software development and DevOps backgrounds and have started to specialize in ML infrastructure. These people are familiar with containers, container orchestration – tools like Docker and Kubernetes –, running clusters in various compute clouds or on-premise and building robust deployment pipelines.

Machine learning in production is very complicated – in many ways even more complicated than normal web scale systems – and you should treat the engineering side with respect, choose the right tools to support the team and staff enough to move faster.

In addition, you should not expect data scientists to build the engineering side of your machine learning stack.

The data and machine learning modelling are vast and complicated areas of expertise and require a lot of study on their own.

If you want the most out of your machine learning team, make sure these two areas work seamlessly together and staff both sides equally.



YOUR MACHINE LEARNING TEAM



PROBLEM DOMAIN EXPERT

Communicates with data scientist about what different values mean and provides domain specific knowledge. Understands what needs to be predicted and why. Some understanding of machine learning.



DATA SCIENCE LEADER

Deep knowledge into data science and is able to make decisions and lead a team.



DATA SCIENTIST

Explores and tries to understand correlations in the gathered data. Understands well how statistics and machine learning works and knows some programming



MACHINE LEARNING ENGINEER

Transforms models created by the data scientist for production. Excellent in writing maintainable code and has a good understanding how machine learning works.



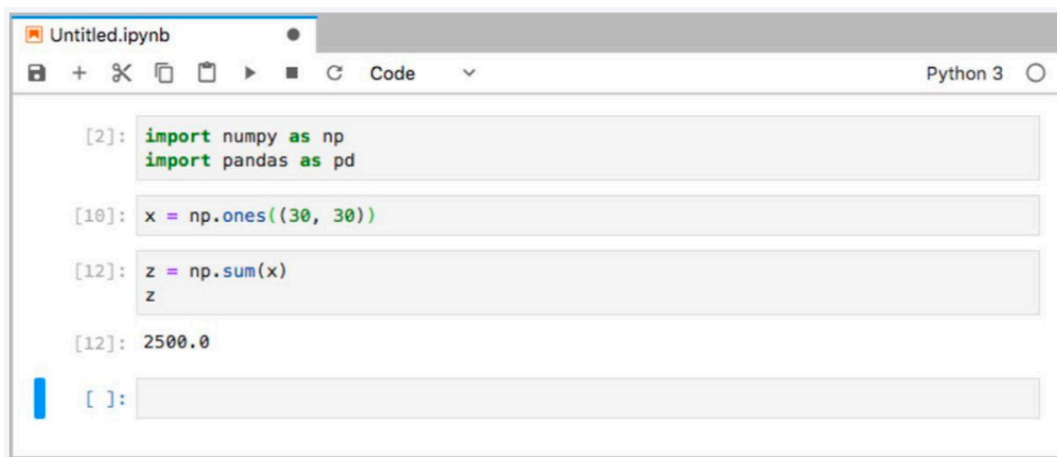
DATA ENGINEER

Takes care of collecting the data and making it available for the rest of the team. Knows how to plan maintainable data pipelines and manages how the data is tested. Good in programming.

LEVEL UP YOUR ML CODE

FROM NOTEBOOK TO PRODUCTION

There are couple main reasons why we advocate for using regular, linear Python scripts instead of Jupyter/IPython notebooks when going from initial exploration work to production.



```
[2]: import numpy as np
import pandas as pd

[10]: x = np.ones((30, 30))

[12]: z = np.sum(x)
z

[12]: 2500.0

[ ]:
```

Developing a machine learning model for a new project starts with groundwork and exploration, to understand your data and figure out the approaches to try. A popular choice for this groundwork is Jupyter, an environment where you write Python code interactively. In Jupyter



notebook's cells you can evaluate and revise and it is an attractive, visual choice (and many times the right choice) – for this step of data science work.

Since Jupyter kernels, the processes backing a notebook's execution, retain their internal state while the code is being edited and revised, they're a highly interactive, fast-feedback environment.

However, while convenient, Jupyter notebooks can be hard to reason about exactly because of this retention of state, since the state of your environment may have changed in a non-linear fashion (or worse yet, left in an inconsistent state) after re-evaluation of an earlier cell. It's entirely possible to have a saved notebook that can't be successfully evaluated after relaunching the kernel.

Since production-grade code is supposed to be easily tested and reviewed, as we've learned as an industry, this isn't desirable at all.

It can also be difficult to keep track of the exact versions of dependencies you've used during development. For instance, a model that worked fine with a certain version of TensorFlow might not run at all with a newer one down the line, and it's tedious to try and figure out what exactly was being run at the time of exploration.

JUPYTER NOTEBOOK IN PRODUCTION. OR NOT..

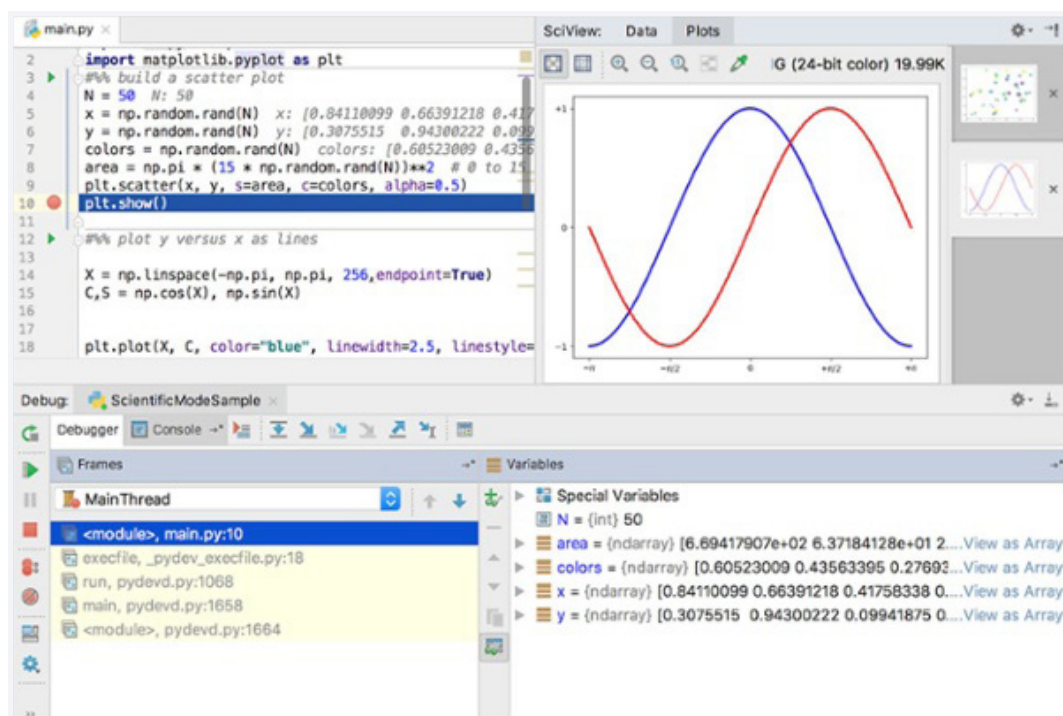
Let's assume you've played nice and been fastidious enough to not run into these problems – all your dependencies are locked down and you've found out



8 / FROM NOTEBOOK TO PRODUCTION

that you can actually run your notebook non-interactively with `jupyter nbconvert --to notebook --execute notebook.ipynb` and maybe pipe the output into a file, for tracking results. You'll inevitably want to run your training code using different parameters (say, learning rates, network structures, etc.); `jupyter nbconvert --execute` isn't really conducive for that, and editing the notebook, or maybe a separate configuration file, to change constants is just silly, too.

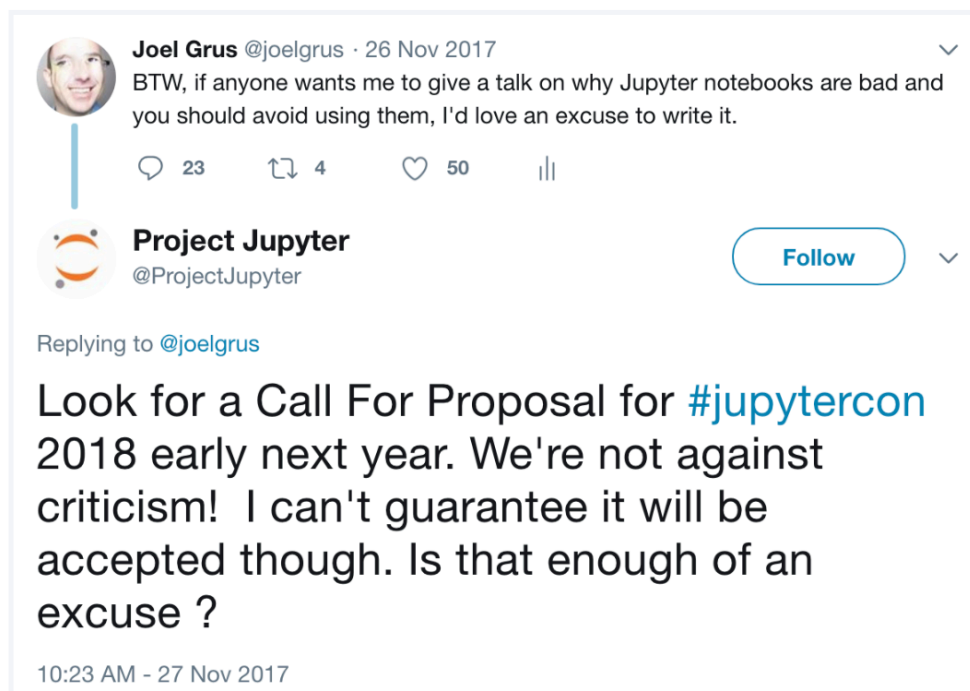
These are some of the reasons why we advocate for using regular, linear Python scripts instead of Jupyter/IPython notebooks when going from initial exploration work to something resembling production. Another thing is that you get to develop using your favorite editor/IDE, be it vim or Emacs or VSCode or PyCharm (which, by the way, has an excellent Scientific Mode), instead of being confined to a browser. Switching from notebooks to regular code also



lets you refactor your solution to a more modular, more easily testable and reviewable package.

Of course there are drawbacks; development is less interactive, and since there is no persistent state, everything needs to be evaluated or loaded from scratch at every invocation of your script. On the other hand this property makes you think about preprocessing your data to a faster-to-load format earlier. When you extract the preprocessing code from other code, it becomes easier to maintain and more reproducible, as well.

@joelgrus talked about this same topic at JupyterCon2018! The slides are hilarious and you can check them [here](#).



WHAT CAN DATA SCIENTISTS LEARN FROM SOFTWARE ENGINEERS?

If developers used to be the rock stars of the dotcom era, Data Scientists are quickly overtaking them as the new Whitesnake cover bands of the 2020s. Although both might be sporting the same hobo beards, Data Scientists are getting their work done with just sticks and stones as their tools while us Software Engineers have every tool in the universe.

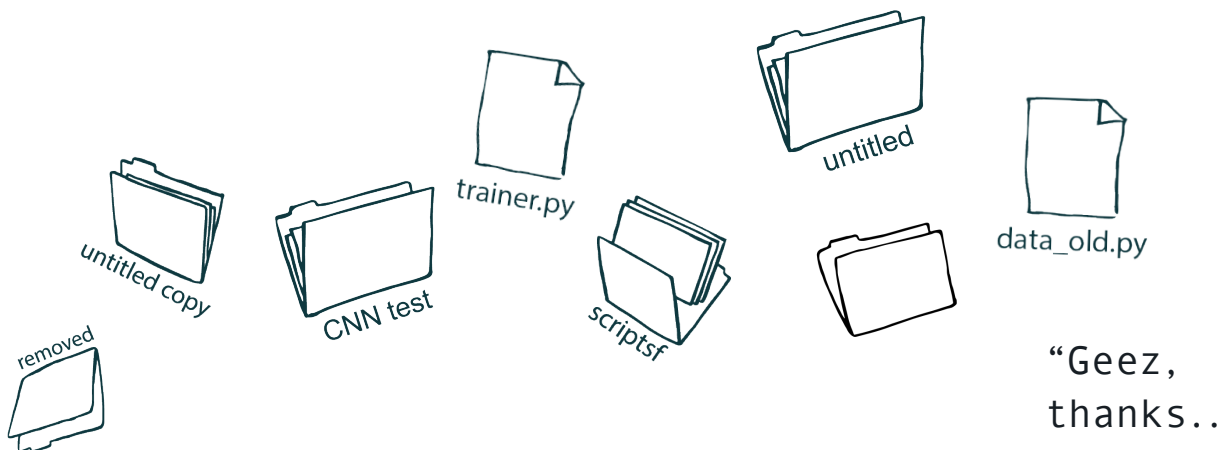
DATA SCIENCE TOOLS

Data Scientists may be building the next Tower of Babel for all we know, but they're stuck with steam engines and pitchforks as their tools.

Where software engineers couldn't live without version control for code, reality for data scientists too often is manual bookkeeping of experiment data, model algorithms, testing environments and training parameters



in an Excel file tossed around over Slack to other team members.



When Software Engineers want to deploy an app to production, they don't need to build their own servers anymore – they just deploy to the cloud. Data Scientists, on the other hand, too often have dance a rain dance before training their models on a server. They need to SSH into a server, install the latest Nvidia drivers, Python dependencies and a clusters of 100 GPUs hosting docker containers over Kubernetes. Software engineers would know how to do it, but the difference is, that's their job.

Guess how relevant this is for getting data scientists' job done and how much more they could get done if they could just double the 10% they spend on actual machine learning algorithms?

And it doesn't stop here. Let's look at team collaboration tools and methodologies. When software engineers use JIRA and go agile, data scientists need to invent their own ways. Imagine if you had to re-learn everything every time you switched projects!



THE ML INFRASTRUCTURE AT FACEBOOK, AMAZON, NETFLIX, GOOGLE ET.AL.

Most big players in machine learning have seen this problem and started solving it for themselves. For example, Über has built their own Michelangelo toolset for doing version control and server management. Likewise Airbnb has Bighead, Netflix has something of their own, as do Google and Facebook. These are all in-house proprietary toolsets to make sure their data scientists' time is not wasted on pipeline orchestration and management tools.



Not every company however has the muscles to invest into ML orchestration and many large corporations still fail to see big impact it will have on business. As a result these early movers have a huge advantage over the rest—not to mention how impossible it will be for startups to get started when most of your time goes into building tools for themselves.

Unlike software engineering, data science is not a simple if-clause somebody wrote in a stored procedure in 1979 running on your DB2, but because the world changes and data changes, you constantly need to re-train the models.



THE KNIGHT ON THE WHITE HORSE

There are however solutions that don't involve inventing everything yourself.

For example here at Valohai we're in the business of building the same tools for the masses that the big players have built in-house for themselves. You could of course build it yourself as well – and at the end of the day you already know what to build: Version control, server orchestration and support for ML frameworks and distributed learning.

The important thing is you'll have something to help you concentrate on your actual work: Building predictive models.



HOW TO MAKE REPRODUCIBLE EXPERIMENTS?

The key to reproducible experiments is proper version controlling. Most companies today are able to systematically store only the version of code. Some more advanced teams have a systematic approach to store and version the models but almost none are able to go beyond that.

Systematic means an approach where you are able to take a model in production and look at the version of code that was used to come up with that model and that this is done for every single experiment, regardless if they are in production or not.

Find a version controlling test on the next page. Cross a box if you save and archive respective information from every experiment.



INPUT

Yes, we store
this information



TRAINING CODE



PARAMETERS



DATASET



EXECUTION



USED HARDWARE



ENVIRONMENT



EXPERIMENT COST



OUTPUT



MODEL



LOGS



RESULTS



STATISTICS



How many boxes did you check?

TRAINING CODE

The exact version of code used for running this experiment. This is usually the easiest, as it can be mostly solved by doing regular software development version control, with Git for instance.

PARAMETERS

One of the important things in running and replicating an experiment is knowing the (hyper)parameters for that particular experiment. Storing parameters and being able to see their values for a particular model over time will also help gain insights and intuition on how the parameter space affects your model. This is exceptionally important when team grows.

DATASETS

Versioning and storing datasets is one of the most difficult problems in the machine learning version control, especially when datasets grow in size and just storing full snapshots is no longer an option.

At the same time it is also one of the most important things to store for reproducibility and compliance purposes, especially in light of new and upcoming laws that will increase the requirement for backtracking the decisions done by machine learning models and model fairness.

USED HARDWARE

Storing hardware environments and statistics serves two



purposes: Hardware optimization and reproducibility. Especially if you are using more powerful cloud hardware it is important to get easy visibility to hardware usage statistics.

It is way too easy and common today to use large GPU instances and not utilize them to their full extent.

It should be easy to see the GPU usage and memory usage of machines without having to manually run checks and monitoring. This will optimize usage of machines and help debug bottlenecks.

The difficulty of reproducibility should not increase one bit when experiments grow in scale and executions are done on multi node clusters. New team members should be able to train the same models on new sets of data with just looking at the configuration and running an execution. Especially deep learning with large data can easily require a setup of several multi-GPU machines to run and this should not bring any kind of overhead to actually running the code. Looking at the history can also help you choose the right cluster size for optimal training time for future runs.

ENVIRONMENT

Package management is difficult but completely mandatory to make code easy to run. We have seen great benefits in a container based approach and tying your code to a Docker image that can actually run it will greatly speed up collaboration around projects.



EXPERIMENT COST

The cost of experiments is important in assessing and budgeting machine learning development. Charging granularity in existing clouds using servers shared within a team does not give you useful insights to actual experiment amounts and cost per experiment (CPE). Worst case, you can have multiple teams working within the same cost structure. This makes it impossible to assess the usefulness of investments per project.

MODEL

Model versioning means a good way to store the outputs of your code. The model itself is usually small and relatively easy to store, but the difficult part is bringing the model together with all the other things listed in this post.

LOGS

Training execution time logs are essential for debugging use cases but they can also provide important information on keeping track of your key metrics like accuracy or training speed and estimated time to completion.

RESULTS

Storing a model without results usually does not make a lot of sense. It should be trivial to see the performance and key results of any training experiment that you have run.

Valohai saves and archives all this information making everything version controlled and 100% reproducible.





VALOHAI

DEEP LEARNING MANAGEMENT PLATFORM

Try Valohai today and
unlock the free features!

[VALOHAI.COM](https://valohai.com)



THE RIGHT TOOLS FOR MACHINE LEARNING

If machine learning is a team sport, machine learning platforms must be the playing fields. And to up your machine learning game, you must have the proper environments to do it.

Machine learning platforms are services that support organizations developing machine learning solutions and there are multiple companies developing such tool. What are the best platforms and how to choose the right one for you? These platforms focus on one or more components of a machine learning system; 1) managing data, 2) building models, and 3) serving predictions.

As the terminology used with various machine learning offerings can be quite convoluted, let's start by untwining the high-level terms first.

Simply put, you can think of **analytics platforms, data science platforms, machine learning platforms, and deep learning platforms** as synonyms.



The main thing that differs is the core focus; deep learning platforms offer GPUs for neural network training while data science platforms focus more on traditional data science like decision trees and linear regressions. The specific terminology is more of a marketing thing.

MACHINE LEARNING SYSTEMS

1) Managing data

→
Data collection
Data exploration
Data storage
Annotation

2) Building models



3) Serving predictions

MONITORING & VERSIONING ALL THE WAY

Machine Learning as a Service (MLaaS) providers offer API-based microservices with pre-trained models and pre-defined algorithms such as Google Cloud Vision API or Amazon's Rekognition services.

Terms together is an enormous, confusing ball of jargon yarn even for the most educated people, especially as most of the terms haven't been generalized yet, making marketing materials patchwork of gobbledygook.



MACHINE LEARNING PLATFORM CATEGORIES

Machine learning platforms can be grouped into seven broad categories based on their core focus

Business Intelligence data science platforms analyze common business information – we are talking about market research, website visitor information, sales numbers, financial records or anything that most companies record already. Point-and-click interfaces and predefined algorithms are the main common feature with all of these platforms. Easy to use, expensive to buy, favor domain expertise over data science and assume deeper partnership with the service provider.

Data Management platforms focus on storing and querying your data. They are your best bet if you are e.g. proficient in writing Spark jobs but don't have the in-house expertise or capacity to maintain big data clusters.

Digitalization data science platforms focus on digitalization of manufacturing or other more traditional companies by data automation, usually involving predictive maintenance, productivity bottleneck detection, and uptime predictions. The type of the analyzed data is domain specific, e.g., machine sensor information or vehicle fuel usage.

Infrastructure data science platforms feel more like IaaS providers than PaaS or SaaS. This category is in many ways the opposite of business intelligence platforms, requiring a lot of additional glue code to get your machine learning system going. They are ideal for organizations requiring highly customized solutions.



Lifecycle Management platforms focus on the projects and workflows to build machine learning solutions. You define the problem scope, acquire/explore/transform the related data, create/validate/optimize solution hypotheses by modeling and finally deploy/version/monitor the prediction-giving model. These are the most full-fledged end-to-end services that require only a modest amount of glue code while not sacrificing too much extensibility.

Notebook hosting platforms focus on offering Jupyter notebooks or RStudio workspaces for exploratory data analysis. These are naturally the first places to start as an individual data scientist, but shared notebooks can cause compounding technical debt to your machine learning system if they remain your primary way of versioning and delivering machine learning code.

Record-keeping platforms focus on visualizing machine learning pipeline steps and keeping history on what each artifact, like a model, consists of. These platforms rarely actually run any code, they mainly work as an add-on that plugs in to get reporting rolling.

Note that these categories aren't exclusive. For example, business intelligence platforms can include handling of big data, and they frequently do, but the categorization helps to find the core focus of the platform compared to the other services.



Service/ Provider	Category	Focus areas
Azure ML Studio	Business Intelligence	point-n-click graphs
Magellan Blocks	Business Intelligence	point-n-click graphs
Pachyderm	Data Management	container-based, data pipelines, collaboration
MapR	Data Management	hadoop-based, performance, customization
MAANA	Digitalization	point-n-click, industry uptime
Uptake	Digitalization	point-n-click, process automation
Spell	Infrastructure	deep learning
Google ML Engine	Infrastructure	deep learning (TensorFlow), DIY
Valohai	Lifecycle Management	deep learning, collaboration, optimization, deployment
cnvrg.io	Lifecycle Management	deep learning, collaboration
Azure Notebooks	Notebook Hosting	exploration
Domino data lab	Notebook Hosting	exploration, collaboration, modeling
Comet	Record-keeping	visualization, record-keeping





MACHINE LEARNING INFRASTRUCTURE FOR EVERYONE

Machine learning infrastructure is one of the biggest things to concentrate on when building production level machine learning models. If you have a business background and want to understand the requirements of machine learning development, continue reading.

WHY SHOULD YOU UNDERSTAND THE CONCEPT?

Currently data scientists, who should concentrate on valuable AI development, need to do lots of DevOps work before they are ready to do the thing they do best: playing with the data and algorithms. Larger companies like Uber and Facebook have built their competitive advantage around machine learning and have proper tools and processes in place, but it has taken years to build those.

Majority of companies are left out as they don't have the knowledge nor the resources to build an efficient and scalable machine learning workflow. The gap between big

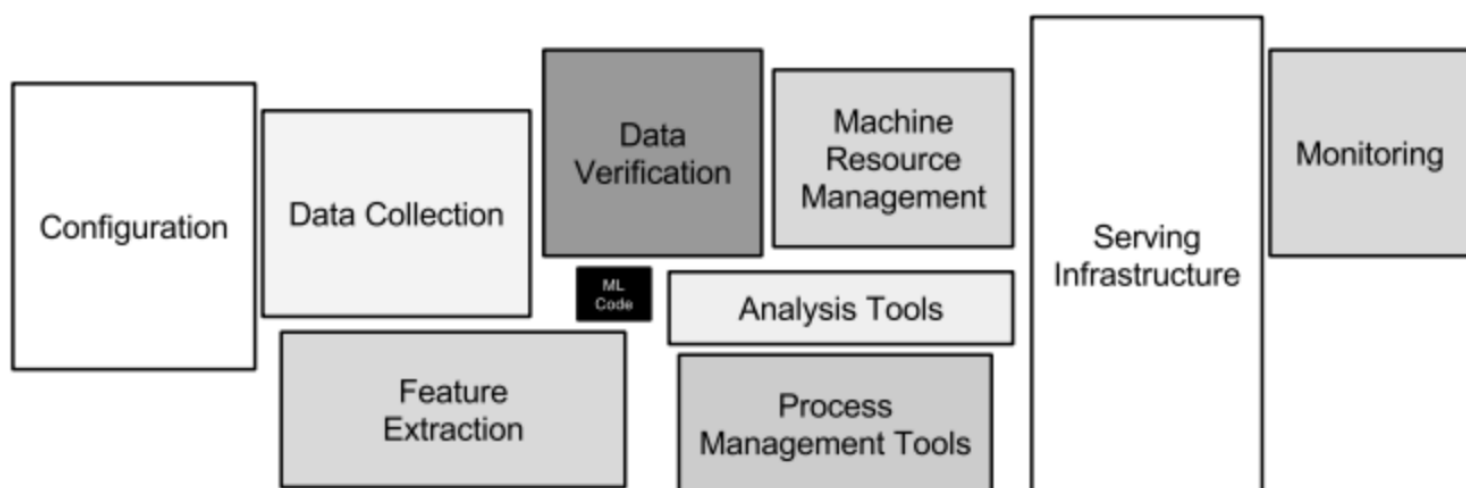


and small players grows.

This is the reason why companies (Yes, yours too.) need to have a mutual understanding between the business development and data scientist teams on what it really takes to build production level machine learning. Building and managing the machine learning infrastructure is one big part of the development work and it is not going to directly bring in any revenue for the company, but the good news is that it can be automated.

WHAT IS THIS MACHINE LEARNING INFRASTRUCTURE THEN?

The image here illustrates the scale and different parts that needs to be taken into consideration in machine learning development. Machine learning code is that small black box in the middle but the required infrastructure is vast and



Source: Google Research Team
 “Hidden Technical Debt in Machine Learning Systems”



very complex. Valohai helps tackle all of this extraneous, but inevitable, infrastructure around the actual revenue generating ML code.

WHEN DO WE NEED INFRASTRUCTURE MANAGEMENT?

Proper tools and processes to manage infrastructure aren't only for saving data scientists' time, but it becomes particularly useful when finalized models don't perform as planned. You probably have heard about the recent unfortunate incident when an Uber self-driving car drove over a pedestrian. Another example of an unwanted end result is a face recognition model that recognized only people with white skin tones as people.

In both of these examples, an explanation for the malfunctioning model is required in order to fix the flaws. There might be some problem with the data or with preprocessing it, or maybe some parameters worked better than others. It is not a foregone conclusion that machine learning teams actually have proper history available.

To drive the point home, here are couple real life examples of data scientist teams' ineffective version control.

One team of hundred data scientists kept track of their models by posting the binary file of the model into a Slack channel. Compare this to a situation where sales people would not have any CRM and they would just jot their client history down to a Slack channel and try to find notes from there later on. Sounds insane, right?



One member of a 50 person machine learning team keeps track of executions in an Excel sheet that is located on his own computer. This can be compared to a situation where a salesman would have a spreadsheet on his own laptop and no other sales team member would know what companies has been contacted and what has been the end result of the meetings. And an even more accurate comparison would be if a salesman would write all different hypotheses of the end results of his every single customer to the spreadsheet. Data scientists can have multiple scenarios regarding one data set and all of these should be tracked somehow.

Now have a discussion with your machine learning team about how they store different versions of their experiments. If their answer is something similar to the examples above, make a plan how to streamline their work.





**Register today and try the
deep learning management
platform for free.**

VALOHAI.COM