# HOW TO BE A(N EVEN BETTER!) DATA SCIENTIST IN 2018?

It's that time of year again: time to make (and hopefully keep this time!) your new year's resolutions. If one of yours is to be a better data scientist, you're in luck. We asked talented data scientists coming from different backgrounds and experiences what their advice would be for becoming a better data scientist.

## THIS IS WHAT THEY SAID

### HARIZO.

While cleaning data or training a model, start simple with something that just works, and improve later. Avoid premature optimization!

### ALEX W.

Don't be afraid to learn or use new languages and methods, and know when to sacrifice model complexity for rapid speed and prototyping. A lot of times you can save yourself much time and have just as good of a model by using a simple algorithm or high-level API.

Trying to optimize and use complex models like deep learning, again and again, can be a big waste of time. Data science for production is about speed, efficiency, and using the right tools instead of being overly theoretical in your algorithm creation. Most of the time, it's about having good data and not the algorithm!

## MATT.

My advice: start simple! When you start a project, you always think about all the interesting - and complicated - stuff you could do (try this new method of auto feature generation, try this new deep learning open source library, etc.).

It's tempting, but to understand if it's really useful, you have to understand what you can achieve with the most obvious features and basic algorithms. So I would advise you to first benchmark the problem with basics to have ideas of what you can score and then what score you can expect with more complex stuff.

This approach will give you a more global understanding, because improving your model is an infinite job. You will always have new ideas for features, data sources you could add, etc. - but is the time spent worth it?

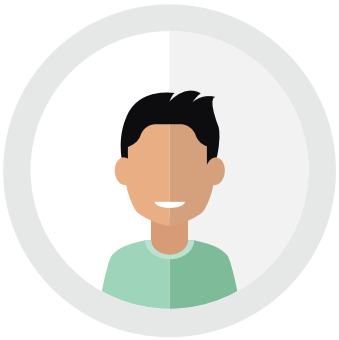
## ALEX C.

To me, a good data scientist balances time across exploration and exploitation (aka multi-arm bandits). While exploring new topics (reading papers, trying out new libraries, etc.), you should exploit your current toolbox to generate business results.

Don't stray away too far from the business, but don't stop learning new stuff.

## JORIE.

My advice is to learn enough architecture so that that you are comfortable setting up and managing your resources -- Hadoop, Spark, different Python and R environments, cloud services (e.g., AWS), remote clusters, read/write access, etc.



## DU.

Not every company needs big data (whatever that means), not everyone needs deep learning, and not every problem needs machine learning.

However, we, as data scientist, all need data ethics. Data science is maturing as a scientific field, and in science, you don't compromise.

Have the courage to say that you did not find any significant results. Have the guts to tell shareholders that the metrics currently implemented are misleading. Be rigorous, be honest, be humble, and keep learning.

## SAMUEL.

My number one tip is to focus on evaluation metrics as early as possible (i.e., to understand the problem in detail - this involves talking to the users of the project!) and start with a simple benchmark.

Other high-ranking tips:

- Don't forget pen and paper (or whiteboard) to write / sketch things when explaining (or thinking carefully about) something!
- Have sanity checks (which can be charts) along the way of your data workflows since many things can go wrong in there.
- Think about the units of features (like in physics) and be extra careful when performing renormalizations / groupings.
- Think about the marginal and joint distributions of features (again, charts help): you want features that have entropy, are «far» from each other and still «close» to your target (in terms of mutual information).
- Think Bayesian especially when dealing with small samples (sometimes adding a thoughtful prior is better than averaging over a sample of size 1).
- Understand and accept that a ML model does not say much about causality of the underlying reality.
- Broaden culture in computational algebra / algorithms / data structures to understand the performance trade-offs of jobs and computations.
- Code for the maintainer (aka «as if the person who ends up maintaining your code is a violent psychopath who knows where you live»).