Robert Chang Follow
Data @Airbnb, previously @Twitter
Jun 20, 2015 · 15 min read

# Doing Data Science at Twitter

A reflection of my two year Journey so far. Sample size N = 1

## Motivation

On June 17, 2015, I celebrated my two year #Twitterversary @Twitter. Looking back, the Data Science (short for DS) landscape at Twitter has shifted quite a bit:

- Machine Learning has played an increasingly prominent role across many core Twitter products that were previously not ML driven (e.g. “While you are away”)
- Tool wise, we’ve moved away from Pig and all new data pipelines are now written in Scalding, a Scala DSL built on top of cascading that makes it easy to specify Hadoop MapReduce jobs

> Organizationally, we switched to an embedded model where DS are now working closer than ever with the product/engineering teams

And these are only a handful of changes among many others! On a personal note, I've recently branched out from Growth to PIE (Product, Instrumentation, and Experimentation) to work on the statistical methodologies of our home grown A/B Testing platform.

Being at Twitter is truly exciting, because it allows me to observe and learn, first hand, how a major technology company leverages data and DS to create competitive edges.

Meanwhile, demands and desires to do data science continued to skyrocket.

> *"Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it"—Dan Ariely*

There are many, and I mean many, discussions around how to become a data scientist. While these discussions are extremely informative (I am one of the beneficiaries), they tend to over-emphasize on techniques, tools, and skill-sets. In my opinion, it is equally important for aspiring Data Scientists to know what it is really like to work as a DS in practice.

As a result, as I hit my two year mark at Twitter, I want to use this reflection as an opportunity to share my personal experience, in the hope that others in the field would do the same!

## Type A Data Scientist v.s. Type B Data Scientist

Before Twitter, I got the impression that all DS need to be unicorns—from Math/Stat, CS/ML/Algorithms, to data viz. In addition to technical skills, writing and communication skills are crucial. Furthermore, being able to prioritize, lead, and manage projects are paramount for execution. Oh yeah, you should also evangelize a data driven culture. Good luck!

A few months in into my job, I learned that while unicorns do exist, for the majority of us who are still trying to get there, it is unrealistic/infeasible to do all these things at once. That said, almost everything data related is tied to the term DS, and it was a bit daunting to find my place as a newbie.

Overtime, I realized that there is a overly simplified but sufficiently accurate dichotomy of the different types of Data Scientists. I wasn't able to articulate this well until I came across a Quora answer from Michael Hochster, who elegantly summarized this point. In his words:

> ***Type A Data Scientist:*** *The A is for Analysis. This type is primarily concerned with making sense of data or working with it in a fairly static way. The Type A Data Scientist is very similar to a statistician (and may be one) but knows all the practical details of working with data that aren't taught in the statistics curriculum: data cleaning, methods for dealing with very large data sets, visualization, deep knowledge of a particular domain, writing well about data, and so on.*

> ***Type B Data Scientist:*** *The B is for Building. Type B Data Scientists share some statistical background with Type A, but they are also very strong coders and may be trained software engineers. The Type B Data Scientist is mainly interested in using data "in production." They build models which interact with users, often serving recommendations (products, people you may know, ads, movies, search results).*

I wish I had known this earlier. In fact, as an aspiring DS, it is very useful to keep this distinction in mind as you make career decisions and choices.

Personally, my background is in Math, Operations Research, and Statistics. I identified myself mainly as a Type A Data Scientist, but I also really enjoy Type B projects that involved more engineering!

## DS at early stage start-ups, growing start-ups, and those who achieved scale

One of the most common decisions to make while looking for tech jobs

is the decision between joining a large v.s. small company. While there are a lot of good general discussions on this topic, there isn't much information specifically for DS—namely, how the role of DS would change depending on the stage and size the company.

Companies at different stages produce data in different velocity, variety, and volume (the infamous 3Vs). A start-up trying to find its product market fit probably don't need Hadoop because there isn't much data. A growing start-up will be more data intensive but might do just fine using PostgreSQL or Vertica. But a company like Twitter cannot efficiently process all its data without using Hadoop and the Map-Reduce framework.

One important lesson I learned at Twitter is that a Data Scientist's capability to extract value from data is largely coupled with the maturity of the data platform of its company. Understand what kind of DS work you want to get involved, and do your research to evaluate if the company's infrastructure can support your goal is not only smart, but paramount to ensure the right mutual fit.

> **At early stage start-ups:** the primary analytic focus is to implement logging, to build ETL processes, to model data and design schemas so data can be tracked and stored. The goal here is focused on building the analytics foundation rather than analysis itself
>
> **At mid-stage growing start-ups:** Since the company is growing, the data is probably growing too. The data platform needs to adapt, but with the foundation laid out already, there will be a natural shift to insight generation. Unless the company leverages Data Science for its strategic differentiation to start with, many analytics work are around defining KPI, attributing growth, and finding the next opportunities to grow
>
> **Companies who achieved scale:** When the company scales up, data also scales up. It needs to leverage data to create or maintain competitive edge. e.g. Search results need to be better, recommendations need to be more relevant, logistics or operations need to be more efficient—this is the time where specialist like ML engineers, Optimization experts, Experimentation designers can play a huge role in stepping up the

game.

By the time I joined Twitter, it already has a very mature data platform and stable infrastructure in place. The warehouse is clean and reliable, and ETL processes are processing hundreds of Map-Reduce jobs easily on a daily basis. Most importantly, we have talented DS working on data platform, product insights, Growth, experimentations, and Search/Relevance, along with way other focus areas.

## My Journey

I was the first dedicated Data Scientist on Growth, and the reality is, it took us a good few months before Product, Engineering, and DS converged on how DS can play a critical role in the process. Based on my experience working closely with the product team, I categorize my responsibilities into four general areas:

**Product Insights**

**Data Pipeline**

**Experimentation (A/B Testing)**

**Modeling**

Let me describe my experience and learning in each of these topics.

## Product Insight

- One of the unique aspects of working for a consumer technology

company is that we can leverage data to understand and infer the voice and preference of our users. Whenever a user interacts with the product, we record useful data and metadata and store them for future analyses.

This process is known as **logging** or **instrumentation,** and is constantly evolving. Frequently, DS might find a particular analysis difficult to perform because the data is either malformed, inappropriate, or missing. Establishing a good relationship with the engineers is very useful here because DS can help engineers to identify bugs or unintended behaviors in the system. In return, engineers can help DS to close "Data Gaps" and to make data richer, more relevant, and more accurate.

Here are a few examples of product related analyses I performed at Twitter:

> **Push Notification Analysis**—How many users are eligible for push notifications? across user segment? across clients? What are the tap rates of different push notification types?
>
> **SMS Delivery Rates**—How do we calculate Twitter's SMS delivery rates across different carriers? Are our delivery rates in emerging countries poorer? How can we make them better?
>
> **Multiple Accounts**—Why do certain countries have a higher ratio of multiple accounts? What drive people to create multiple account?

Analyses come in different forms—sometimes you are asked to provide straightforward answers to simple data pulls (*push analysis*), other times you might need to invent and come up with new ways to calculate a new but important operational metrics (*SMS delivery rates*), and finally you might be tasked to understand deeper about user behaviors (*multiple accounts*).

Generating insights through product analysis is an iterative process. It requires challenging the questions being asked, understanding the business context, and figuring out the right dataset to answer the questions. Over time, you will become an expert in where the data lives and what they mean. You will get better at estimating how much time it

will take to carry out an analysis. More importantly, you will slowly move from a **reactive** state to **proactive** state and start suggesting interesting analyses that product leaders might not think of, because they don't know the data exists or that disparately different data sources can be complementary and combined in a particular way.

> ***Skill used here:***
>
> Logging and Instrumentation. Identifying Data Gaps. Establish good relationships with engineers
>
> Ability to navigate and identify relevant datasets and how to use them
>
> Understand different types of analyses and get better at time estimates on how long or difficult they will take
>
> Know your query language. Typical data munging skills using R or Python

## Data Pipeline

- Even though type A data scientists might not produce codes that are directly user facing, surprisingly often, we still commit codes into the

codebase for the purpose of **data pipeline processing**.

If you heard of the operation | (pipe) from Unix that facilities the execution of a series of commands, a data pipeline is nothing but a series of operations, when streamed together, helped us to automatically capture, munged, aggregated data on a recurring basis.

Before Twitter, most of my analysis are ad-hoc in nature. They are mostly run and executed once or few times on my local machine. The codes were rarely code reviewed, and they are most likely not version controlled. When a data pipeline is created, a new set of concerns start to surface such as dependency management, scheduling, resource allocation, monitoring, error reporting, and alerting.

Here is a typical process of creating a data pipeline:

> You realized that the world would be a better place if a dataset can be produced on a recurring basis
>
> Upon confirming the need, you started off by designing the final product first, such as designing the data schema of the output dataset.
>
> Write your code, either in Pig, Scalding, or SQL, depending on where your data lives.
>
> Submit for **code reviews,** and be prepared to get feedbacks and make additional changes because either your business logic is incorrect or your code was not optimized for speed and efficiency **#shipit**.
>
> There might be a step of testing and dry-running your job to make sure everything works as intended.
>
> **Merge your code into master.** Deploy the code and schedule your job!
>
> Set up monitoring, error reporting, and alerts in case things go awry

Obviously, a pipeline is more complex than an ad-hoc analysis, but the advantage is that this job can now be run automatically, the data it produces can be used to power dashboards so more users can consume

your data/results. More importantly but a subtle point, this is a great learning process to pick up engineering best practices, and it provides the foundation in case you ever need to build specialized pipelines such as a Machine Learning Model (I will talk about this more in the last section) or a A/B testing platform.

> ***Skills Used Here:***
>
> Version control, the most popular by far is Git
>
> Learn how to do code reviews and provide feedbacks effectively
>
> Know how to test, dry-run, and debug when job fails
>
> Dependency management, Scheduling, Resource Allocation, Monitoring, Error Reporting, Alerting

## Experimentation (A/B Testing)

- Right at this moment, it's very possible that the Twitter app you are using is slightly different from mine, and it's entirely possible that you actually have a feature that I do not see. Under the hood, since Twitter has a lot of users, it can direct a small % of its traffic to experience a new feature that is not yet public, so to understand how these specific users react to it compared to those who don't (the control group)—this is known as A/B testing, where we get to test which variant, A or B, is better.

  I personally think A/B testing is one of the unique perks of working for a large consumer tech company. As a Data Scientist, you get to establish

**causality (**something really hard to do with observational data) by running actual randomized, controlled experiments. At Twitter, *"It's rare for a day to go by without running at least one experiment"—Alex Roetter, VP of Engineering*. A/B testing is ingrained in our DNA and our product development cycle.

Here is the typical process of running a A/B test: ***Gather Samples*** -> ***Assign Buckets*** -> ***Apply Treatments*** -> ***Measure Outcomes*** -> ***Make Comparisons***. Surely this sounds pretty easy, no? On the contrary, I think A/B testing is one of the most under-appreciated and tricky analytics work, and it's a skill that's rarely taught in school. To demonstrate my point, let's revisit 5 steps above again and some of the practical problems you might run into:

> **Gather Samples**—How many samples do we need? How many users should go into each bucket? Can we ensure that the experiment will have sufficient power?
>
> **Assign Buckets**—Who are eligible to be in the experiments? and where in the code should we start assigning buckets and showing treatments? Would the placement introduce data dilution (i.e. some users are assigned to treatment but never see it)?
>
> **Apply Treatment—**Are there any other teams in the organization running experiments that are competing for the same real estate in the app? How do we deal with experiment collision and ensure our data is not contaminated?
>
> **Measure Outcome—**What is the hypothesis of the experiment? What are the success and failure metrics of this experiment? Can we track them? and How? What additional logging do we need to add?
>
> **Make Comparisons—**Suppose we see that the # of users who logged-in increase dramatically, is it due to noise? How do we know if the results are statistically significant? Even if it is, is it practically significant?

Addressing each and every question from above requires a good command of Statistics. Even if you are as rigorous as possible when designing an experiment, other people might fall short. A PM would be incentivize to peek the data early, or to cherrypick the results that they

want (because it's human nature). An engineer might have forgotten to log the specific information we need to calculate the success metric, or the experiment codes can be written in the wrong way and unintended bias is introduced.

As a Data Scientist, it's important to play the devil's advocate and help the team to be rigorous, because time wasted on running a ill-designed experiment is not recoverable. Far worse, an ill-informed decision based on bad data is far more damaging than anything else.

> ***Skills Used Here:***
>
> **Hypothesis Testing**: Statistical test, *p*-values, statistical significance, power, effect size, multiple testing
>
> **Pitfalls of Experimentation**: Carryover effect, metrics cherry-picking, data dilution, bucket anomaly

## Predictive Modeling

**&**

## Machine Learning

- My first big project at Twitter was to augment a set of fatigue rules to our existing email notification product so to reduce spams to our users. While this is a noble gesture, we also know that email notification is one of the biggest retention levers (we know this causally because we ran experiments on it), so finding the right balance is the key.

  With this key observation, I quickly decided to focus on trigger based email, email types that tend to burst arrive at users' email inbox when interactions happen. Being an ambitious new DS who is trying to prove his value, I decided to build a fancy ML model to predict email CTR at the individual level. I marched on and aggregated a bunch of user level feature in Pig and built a random forest model to predict email click. The idea is that if a user has a consistent long history of low CTR, we can safely holdback that email from that user.

  There was only one problem—all of my work was done in my local machine in R. People appreciate my efforts but they don't know how to consume my model because it was not "*productionized*" and the infrastructure cannot talk to my local model. **Hard lesson learned!**

  A year later, I found a great opportunity to build a churn prediction model with two other DS from Growth. This time around, because I had accumulated enough experience building data pipeline, I learned that building a ML pipeline was in fact quite similar—There is the training phase, which can be done offline with periodic model updates through Python; There is the prediction part, where we aggregate user features daily, and let the prediction function does its magic (mostly just dot product) to produce a churn probability score for each user.

  We built the pipeline in a few weeks, confirmed that it has good predictive power, and rolled it out by writing the scores to Vertica, HDFS, and our internal key-value store at Twitter called Manhattan. The fact that we make the scores easily query-able by analysts, DS, and engineering services helped us to evangelize and drive use cases of our model. And it was the biggest lesson I learned about building models in production.

  I deliberately ignore the steps one needs to take in building a ML model in this discussion so far—framing the problem, defining labels, collect training data, engineer features, build prototypes, and validate and test

the model objectively. These are obviously all important, but I feel that they are fairly well taught and many good advices have been given on this very subject.

I think most of the brilliant DS, especially type A DS have the opposite problem, they know how to do it right but are not sure how to push these models into the ecosystem. My recommendation is to talk to the Type B Data Scientists who have a lot of experience on this topic, find the set of skills that are needed, and find intersections and hone your skills so you can pivot to these projects when the time is right. Let me close this section by the following quote:

> *"Machine learning is not equivalent to R scripts. ML is founded in math, expressed in code, and assembled into software. You need to be an engineer and learn to write readable, reusable code: your code will be reread more times by other people than by you, so learn to write it so that others can read it."—Ian Wong, from his guest lecture at Columbia Data Science class*

Well said.

> ***Skills Used Here:***
>
> **Pattern Recognition:** Identifying problems that can be solved using modeling technique
>
> **All the basics of modeling and ML** : Exploratory data analysis, building features, feature selection, model selection, training/validation/testing, model evaluation
>
> **Productionization:** all the things mentioned above about data pipeline. Set up your ouput so it can be query by different people and services

## Some After Thoughts ....

Being a Data Scientist is truly exciting, and the thrill of finding a particular insight can be as exciting as an adrenaline rush. Building a data pipeline or ML model from ground can be deeply satisfying, and there are a lot of fun 'playing God' when running an A/B tests. That said, this road ain't easy and pretty, there will be a lot of struggles along

the way, but I think a motivated and smart individual will pick these things up quickly.

Here are some additional information I found very useful along the way, and I hope you find them useful too:

**Data Science and Software Engineering:**

Software Development Skills for Data Scientists

Building Analytics at 500px

How I Became a Data Scientist Despite Having Been a Math Major

The Data Engineering Ecosystem: An interactive map

**A/B Testing:**

So, you need a statistically significant sample?

Experiments at Airbnb

When should A/B testing not to be trusted when making decisions

**Recruiting:**

What it is like to be on the data science job market

On picking where to work

It's a long journey, and we are all still learning. Good luck, and most importantly, have fun!