Why becoming a data scientist is NOT actually easier than you think

I was just doing some late night reading and came across this article:

http://gigaom.com/data/why-becoming-a-data-scientist-might-be-easier-than-you-think/

TL;DR - You can take the ML course on Coursera and you're magically a data scientist, because three really intelligent people did it. I disagree.

I'm not claiming the people referenced in this article are not data scientists who score high in Kaggle competitions. They're probably really intelligent people who picked up a new skill and excelled at it (although one was already an actuary, so he is basically doing machine learning in some form already).

Here is my problem with it - being a data scientist usually requires a much larger skill set than a basic understanding of a few learning algorithms. I'm taking the Coursera ML course right now, and I think it is great! Here is what I didnt learn though:

Programming Languages and Other Technologies:

Most data scientists and the companies that employ them are not using Matlab/Octave. They have backend web services written in Java, Python, Scala, or Ruby. These languages are not covered. Python has libraries like Scipy, Numpy, and Scikit-learn that are great for solving numerical problems. Java has a bunch of libraries too like the Mahout math library [2]. R is used by most statisticians (again not covered in the course). When your boss (or a customer) comes to you and says you need to integrate an algorithm into a pre-existing web service (example -they need a recommendation engine), and you say "I only know Matlab" that is going be a huge problem. You don't just pick up Java/Python/C++/Scala/whatever in a few days on the job. You have to be somewhat familiar with these languages to understand large, pre-existing code bases. It wouldn't hurt to have a decent understanding of existing technologies like Django, ROR, Groovy, Lift, etc. because you're going to have to integrate your amazing algorithm into one of them. If you only know Python but the rest of the company is using Java, you better know about Thrift, Avro, Google ProtoBufs or something smiliar. I digress ...

Big Data Software:

Most data scientists are working on problems that can't be run on single 512MB RAM machine (the data sets on Coursera are tiny). They have large data sets that require

distributed processing. To do this, you need to understand map-reduce, distributed files systems and be able to utilize Hadoop. Even if you don't know Java, you still need to know that Hadoop streaming exists, how do use it, and know a scripting language (Hadoop streaming does not currently support Matlab or Octave). Again - not something you just pick up in a few days. If you're going to be doing distributed machine learning, you will probably want to use Mahout. Some of the algorithms covered in the Coursera course have distributed versions implemented into Mahout (Clustering, PCA, Regression), but most do not exist yet (SVM, Distributed-User-Recommendations, and cross-validation/performance metrics for distributed versions of the algorithms). You're going to have to know Java (and the Hadoop/Mahout APIs) to implement them from scratch, or use a different algorithm that you may or may not be familiar with. Even if you do not need to use a distributed algorithm, it would probably be good to know how to spin up a 64GB instance on ec2, login, install some software, and run your algorithm on the cloud.

Other useful learning algorithms

Coursera skipped over Bayesian learning [1]. A lot of systems use this (or some form of this) in production, but you would know nothing about it (I'm not saying you couldn't learn it, but I am obviously disagreeing with this article).

Feature Extraction:

You can use every algorithm from the ML course and build hundreds of different classifiers to solve a real-world problem (you could even combine them!), but if your features suck, the performance of your classifier is going to suck also. Extracting good features usually requires a deep understanding of the problem, the underlying distributions of the data, and/or an familiarity of how the data is being generated. It might help to also know about Convolution, Wavelets, Time Series Analysis, Digital Signal Processing, Fourier Transforms, etc. Feature extraction could be a whole Coursera course by itself.

Data cleaning:

Data preprocessing - Coursera sets up all the data sets for you. They even write the scripts to load the data. (see week 6 SVM email classification, they wrote all of the regex expressions to clean the emails for you) That doesn't work in the real world. Real-world data is ugly, and unstructured. You need to know regular expressions and UNIX commands like sed, grep, tr, cut, sort, awk, and map/reduce to clean these data sets up and put them into "Coursera" format. Notice I said UNIX commands, which implies you need to be somewhat comfortable on UNIX/LINUX, which may be a steep learning curve if you're currently using Windows.

Probability & Statistics

The ML course touches on some of these topics, but real-world problems usually require a much deeper understanding. Examples:

Are your features dependent or independent (Chi Squared test).

How do you interpret p-values?

How do you set up confidence intervals?

What is the F-test?

What is standard error?

Should I use AROCs to test the performance of this algorithm?

What is a ROC curve?

What is Hypothesis testing and when can you / do you use it.

I could go on - but you get the point . It's important to know this stuff and know how and when to apply it.

Databases:

Where is all this data being stored? It's fine in flat files for the purposes of the ML course, but when you show up to your first day at the job, it's going to be stored in MySQL, Postgres, MongoDB, Casandra, CouchDB, and/or on the HDFS. Your going to have to improvise.

Visualization:

If you think Matlab 2D plots are awesome, check out D3.js. Oh, I forgot to mention, you need to know javascript, functional programming, and the general learning curve for the D3.js APIs.

Debugging:

Debugging is briefly covered in the ML course (Neural Networks - Gradient Checking, which was an optional programming excerise), but when an algorithm isn't working correctly, you're going to have to quit treating it like a black-box and dive in. That's when shit gets real - you actually do need to know about Conjungate Gradients, Partial Differential Equations, Numerical Analysis, Lagrange Multipliers, Numerical Linear Alegbra, Convex Optimization, Vector Calculus, Stochastic Processes, and potentially a lot of other subjects. The ML course is only 8 weeks, and they can't cover these topics,

because they require years of experience, and probably some form of technical training beyond a BS in whatever. You could just ask the guy with a few years more experience that is about to take your job though.

Intuition

Beyond what I have highlighted above, being a good data scientist (usually) takes years of experience. It requires more than just knowing how machine learning algorithms work. It's knowing what questions to ask and how to convey the answers to investors, management, and customers. No online course, even from Stanford professors, is going to be able to teach you that.

UPDATE:
I'm currently available for Django/Machine-Learning/Design consulting. If you are interested, please contact me: info@mathandpencil.com. For more info, please check out my firmMath & Pencil

You can follow me on twitter: @josephmisiti

[1] http://en.wikipedia.org/wiki/Bayesian_inference

[2] https://github.com/apache/mahout/tree/trunk/math