(Source)

Lessons from a Year in the Data Science Trenches

Five takeaways from learning to put machine learning in production at Cortex Building Intelligence

Will Koehrsen Follow
Dec 9, 2019 · 13 min read

Over the past year, I've gone from the simple world of writing Jupyter Notebooks to developing machine learning pipelines that deliver real-time recommendations to building engineers around the clock. While I have room for improvement (I still make plenty of coding and data science mistakes), I've managed to learn a few things about data science that we'll go through in this article. Hopefully, with the lessons below, you'll avoid many of the errors I made learning to operate on the day-to-day data science frontlines.

1. **Production data science is mostly computer science**
2. **Data science is still highly subjective**
3. **People and communication skills are crucial**
4. **Use standard tooling and be slow to adopt new technologies**
5. **Hide the internal complexity of data science with external simplicity**

Author's note: This is written from a single perspective, which does not represent the entire field of data science. Keep in mind this is from an end-to-end (concept to deployed machine learning system) data scientist working in the building energy sector, providing real-time recommendations for operating buildings more efficiently. If you've had different experiences or would like to contradict me, the comment section is waiting for your reply.

. . .

Production data science is mostly computer science

When asked about the hardest part of the job, I firmly reply it's not machine learning, considering all our ML looks like:

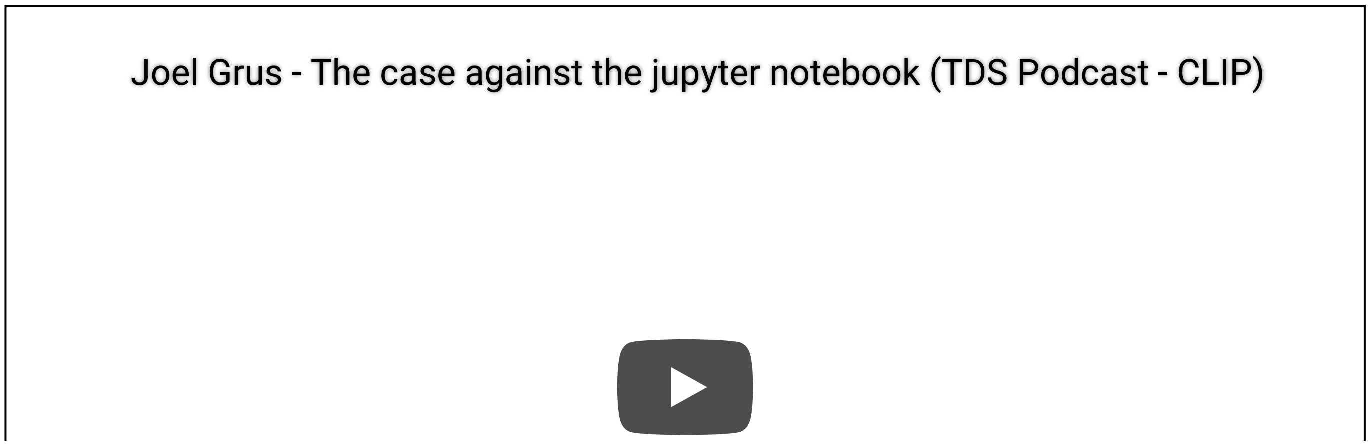
```
from sklearn import Model

Model.fit(features, target)
predictions = model.predict(testing_features)
```

Instead, the hardest parts of data science are developing everything that occurs before and after modeling. Before we have: loading data from a database, feature engineering, data validation, and data processing pipelines (assuming our job starts after data is ingested). After we need to verify the results, set tasks to run automatically on a schedule, write results back to our database and send off webhooks to trigger other services.

These peripheral actions, which comprise most of the work in machine learning, all require solid computer science practices. A few of these practices which pertain to developing code are writing many short functions each of which does one thing well, developing classes to implement related functions, proper naming conventions, writing unit tests on the code and data, writing code that is meant to be read, and not repeating code. Also, there are other computer science practices to apply around the code itself, such as version control, code reviews, continuous integration, code coverage, and deployment, which has now spawned an entirely separate field of machine learning operations (MLOps).

Although I managed to make the mechanical engineering -> data scientist transition, in retrospect, it would have been more productive to do engineering -> computer science -> data science. The second approach would have meant I didn't have to unlearn the poor coding practices I picked up in data science classes. In other words, I think it's easier to add data science on top of a solid computer science background than to learn data science first and then take up computer science (but both routes are possible).

Computer science involves an entirely different way of systematic thinking, methodically planning before coding, writing code slowly, and testing code once it's written. Clean code is in stark contrast to the often free-wheeling nature of data science with dozens of half-written notebooks (we've all had notebooks called `Untitled12.ipynb`) and an emphasis on getting immediate results rather than writing rather error-free code that can be re-used.


Joel Grus - The case against the jupyter notebook (TDS Podcast - CLIP)

See also https://www.youtube.com/watch?v=7jiPeIFXb6U for why notebooks aren't always great

All data scientists can benefit from a course on computer science best coding practices. The ability to structure scripts and packages, to write clean code, to test and to document code makes the transition from exploratory data science to production machine learning more manageable. Moreover, they instill a model of thinking that leads to re-usable code easy for others to understand. Even academic data scientists, who typically write data science scripts to analyze data for papers would be helped by better practices. Perhaps the reproducibility issues in science would improve if scientists wrote cleaner code and included unit tests to verify inputs, outputs, and function behavior.

There are many topics to learn in data science, and it can feel overwhelming at times. However, computer science should not be viewed as an add-on; instead, it should be seen as foundational for data scientists who want to see their code operationalized. Fortunately, there are plenty of resources, such as software carpentry, which anyone can use to learn and apply these practices.

Data science is still highly subjective

Data science promises to make optimal decisions using data instead of human judgment. It's a noble cause, but one that is far from current reality because the data and the methods we use to analyze it are subject to human influence to a significant degree. Even the supposedly objective field of data science rests on human behavior. As Vicki Boykis puts it in her excellent Normcore Tech newsletter, Neural Nets are just people all the way down.

Every step of a typical machine learning system is affected by personal choices. Here are a few of those decisions:

- **Collecting data**: What data do we gather? What sensors do we use? Who do we survey? How do we phrase our questions?
- **Feature engineering**: What features do we make? Do we use domain knowledge or automated feature engineering? How do we fill in missing values? What observations should be removed?
- **Modeling**: What hyperparameters should we use? How complex should we make the model?
- **Validation**: What is the metric for evaluation? What is the validation procedure? What level of performance do we need?
- **Deployment:** Do we trust these numbers enough to show to customers? Do we need a human evaluating the predictions for a sanity check?

Inevitably, through this process, different humans will arrive at different conclusions. One example of this is documented in the paper Many Analysts, One Data Set, which describes how data scientists can arrive at conflicting decisions with the same dataset because they employ varied methods. It's no exaggeration to say that you can use a single dataset to prove an argument and its antithesis by altering the analysis. What this shows is you shouldn't put too much faith in any conclusion drawn from one study, but should instead look (with skepticism) at meta-analyses (and read *How to Lie with Statistics*).

Furthermore, human bias, whether on purpose or by accident, makes its way into the data, thereby influencing machine learning models. As shown in books like *Weapons of Math Destruction*, ceding decisions to machines does not eliminate discrimination, but codifies existing prejudices that appear in real-world data. The objective, ending biases decisions with data science, is noble, but as long as humans are involved, we cannot blindly rely on machine learning predictions.

· · ·

Does the subjectivity in data science mean we should give up all idea of truth? I think we should reframe the question: instead of searching for the one right answer, we use data science, however flawed, to move in the direction of better solutions. After all, data science is a sub-field of science, with the goal of becoming less wrong over time. Also, the more people working on a problem, and comparing their work, the closer we move to better outcomes. Those 20 scientists might have carried out 20 different analyses, but if they had then compared their methods and worked together, the resulting effort would be superior to any of the individual projects.

When practicing data science, we must remember that data science, like any field, is not without its flaws, and should not be trusted without question. Practicing responsible data science means presenting results with uncertainty intervals, looking for reasons to disprove your conclusions, comparing your results to other similar work, and be realistic when presenting findings.

Since data science is dependent on human judgment, we need to realize …

People and communication skills are crucial

While seemingly obvious (is there any field where communication skills are a negative?), I'm reminded daily of the need to effectively communicate machine learning to people across the technical spectrum. It's not enough to know your ML jargon; you need to be able to meet people where they are in their understanding and tell them only the details they need to know.

(As a slightly humorous example, my job is "computer stuff" to some people and a half-hour discussion on the particulars of machine learning to others.)

At least in our case, machine learning decisions do not replace human choices (even when they are more accurate), because it's up to the building engineers to use our recommendations. (Autonomous building operating is probably further away than autonomous vehicles). It's not enough to build the model, show how accurate it is, and give the results to customers expecting them to implement the predictions immediately. Data scientists still have to master the messy art of social interaction. You can produce the best machine learning model possible, but if you can't convince people to use it, then it will have no impact.

The most common people aspects of my job are explaining methods, both through writing and presentations to both internal and external groups, understanding how our customers currently make decisions and talking to domain experts to translate their knowledge into data science systems. None of these were mentioned in college, where I was told data scientists could hide behind perfectively objective numbers.

· · ·

Even after you explain how the computer arrived at a decision, the recommendation may be ignored because people are not entirely rational. When presented with an objectively better choice, people may choose another option for all sorts of reasons: habit, lack of trust, familiarity, misinformation.

Consider the scenic route choice: sometimes, people, for seemingly no logical reason, will take a significantly longer route between two places. Why? Because there is more beautiful scenery along the way. A naive data scientist might show only the shortest route recommended by the model, but, a data scientist who understands her customers will know they want to see more than endless miles of the interstate on their trip.

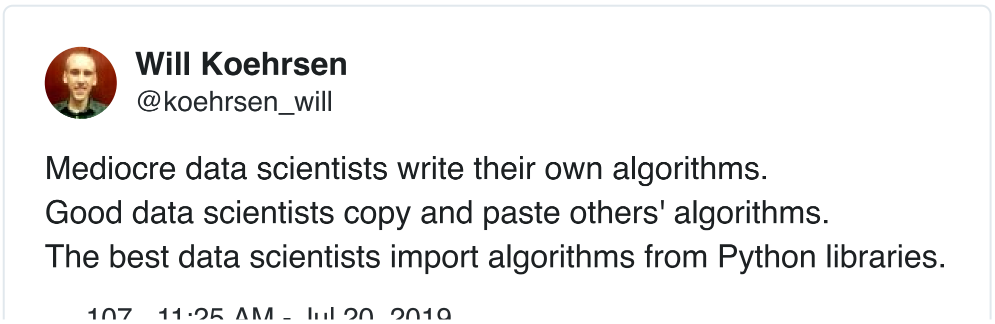
Likewise, optimal machine learning predictions may not be used because accuracy is not the only consideration. For example, we predict the ideal time for a building engineer to start heating their building, but many engineers will still turn on equipment earlier because they do not want tenants to experience discomfort. This isn't rational (we time our recommendations to ensure the building will be at the correct temperature by the time tenants arrive), but, until we remove humans from the decision-making process, we're going to have to adapt our computer systems to humans instead of the other way around.

Perhaps in addition to your computer science classes, take a few courses in sociology to understand your fellow humans (or read behavioral economic books such as *Misbehaving* by Richard Thaler or *Thinking, Fast and Slow* by Daniel Kahnemann, both Nobel winners in economics).

Use standard tooling and be slow to adopt new technologies

What's the best way to ensure your algorithm doesn't contain any errors? Import a model from `sklearn` instead of writing it yourself. Unless you are doing cutting-edge research, there is almost no reason to write your own version of a machine learning model. Instead, use functions from widely used and well-tested libraries, standard tooling, to accomplish tasks.

In a recent tweet, I said the worst data scientists write their own algorithms and the best import them from standard libraries. I was sort of joking, but I stand behind the principle that using well-tested code from open-source libraries is almost always more efficient than developing your own code.



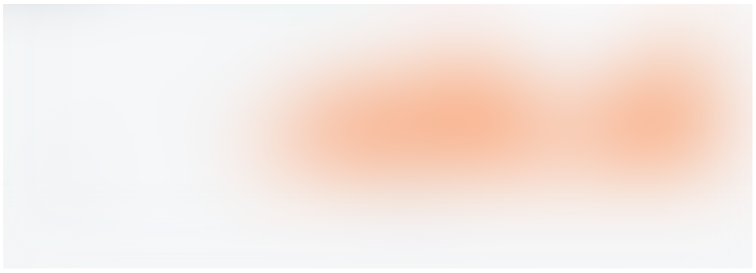
Will Koehrsen
@koehrsen_will

Mediocre data scientists write their own algorithms.
Good data scientists copy and paste others' algorithms.
The best data scientists import algorithms from Python libraries.

107 11:25 AM - Jul 20, 2019

The logic of using standard tooling applies to more than machine learning models. Everything you could want to do to a dataset is already implemented in `pandas` (assuming you use Python) so look for a solution there first. Likewise, there are standard libraries for statistics, plotting, testing, scheduling, deploying tasks, and most parts of the machine learning pipeline.

I took over my position from 2 Ph.D. data scientists who had felt the urge (probably to justify their degrees) of inventing their own data structures, metrics, algorithms, file loading, etc., which resulted in a mass of jumbled code that no one understood. My first six months of the job were mostly replacing hundred line scripts with three import statements, and, to this day, I'm proud to be a net-negative contributor to our machine learning library.



Via Negativa: addition by subtraction

Moreover, don't switch to the new library/technology/framework/database just because it's new. Standard tools like SQL databases, sklearn for machine learning, and pandas for data manipulation work fine. They may be bland because they are (relatively) old, but they are also tested and reliable. Being an early adopter may seem fun at first, but it'll quickly become exhausting as you battle through bugs and limited documentation.

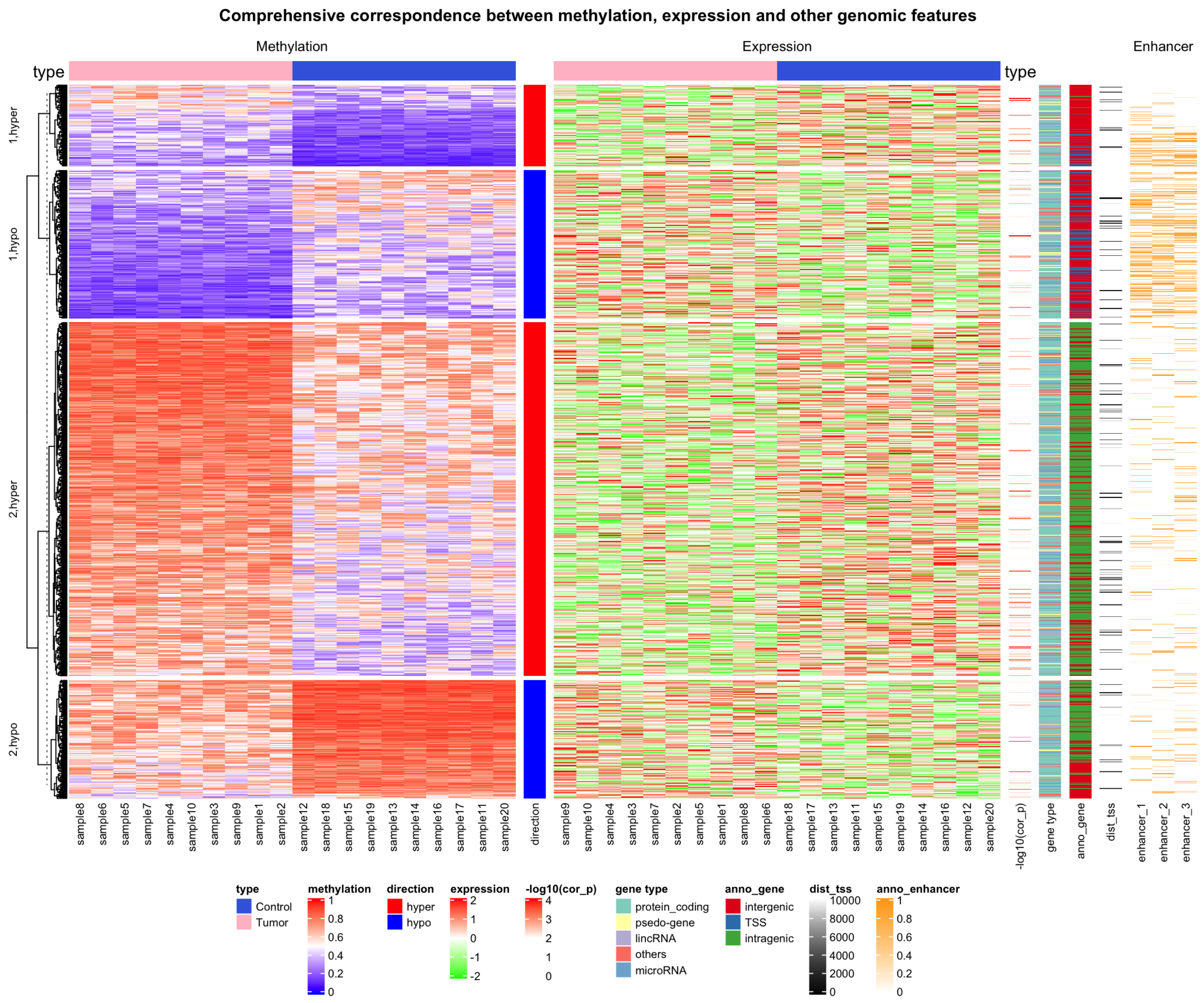
While new technologies drive the media cycle, they often have little to no immediate impact on the people and companies actually doing work (Beware Engineering Media). My younger self cannot believe I'm saying this, but I now prefer boring, proven technologies to exciting, novel ones that have yet to deliver results. Internally, our engineering team has lengthy debates about upgrading library versions, and, if there's no discernible benefit or need, then we don't upgrade because there's a new release. Adding a library to our machine learning codebase requires a demonstrated need as another library means another dependency to manage.

The most long-lived companies are those that do mundane things and move slowly (like Caterpillar), while startups that move fast and do "cool" things tend to dissipate in a few years. The most robust machine learning systems are not going to be ones using cutting-edge technologies, but those that stick with the tried-and-tested data science standard tooling.

Hide the internal complexity of data science with external simplicity

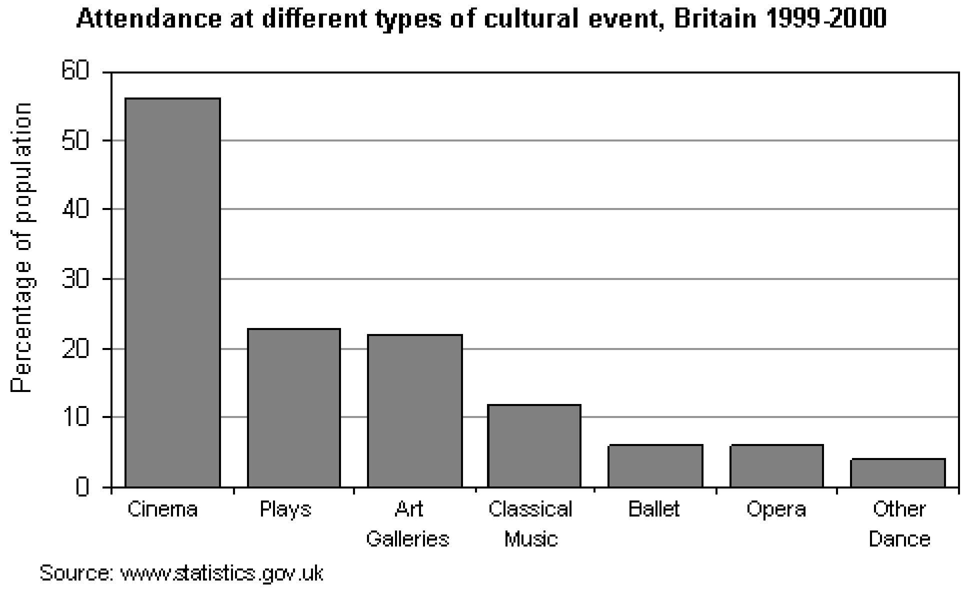
Computers are very good at processing large quantities of numbers. Humans can barely handle comparing a few numbers. To most effectively combine computers' and humans' abilities, we should use computers to analyze large datasets and present only the most critical figures to humans making decisions. Millions of numbers in, as few numbers as possible out. Complex models on the inside, actionable recommendations on the outside.

Over the past year, I've developed a theory that the more data points a chart has, beyond a small number (maybe the number 7?), the less useful it is. Humans just do not have the capacity to analyze complex quantitative diagrams with accuracy. Heatmaps are cool, but has anyone made a crucial decision from a heatmap with 1000s of datapoints in contrast to a bar chart with five numbers?



Cool, but what am I supposed to do with this information? (Source)

As someone who generally enjoys perusing numbers and listening to details of machine learning models, I've had a hard time getting used to the idea that most people do not want more information. Customers and people making decisions desire the takeaway, and that's it. Less ink means a better chart. (If you need help making charts, consult The Visual Display of Quantitative Information or Fundamentals of Data Visualization).



Dull? Probably. Informative? Absolutely.

The argument for external simplicity does not mean using only linear models. Data science can involve complex algorithms and highly technical operations. Only the external-facing part of data science must be simple enough for a non-technical person to understand. Nonetheless, be careful about making your model so complex that even you don't understand it. Is it worth a blended model for a small accuracy gain at the expense of not being able to explain your model?

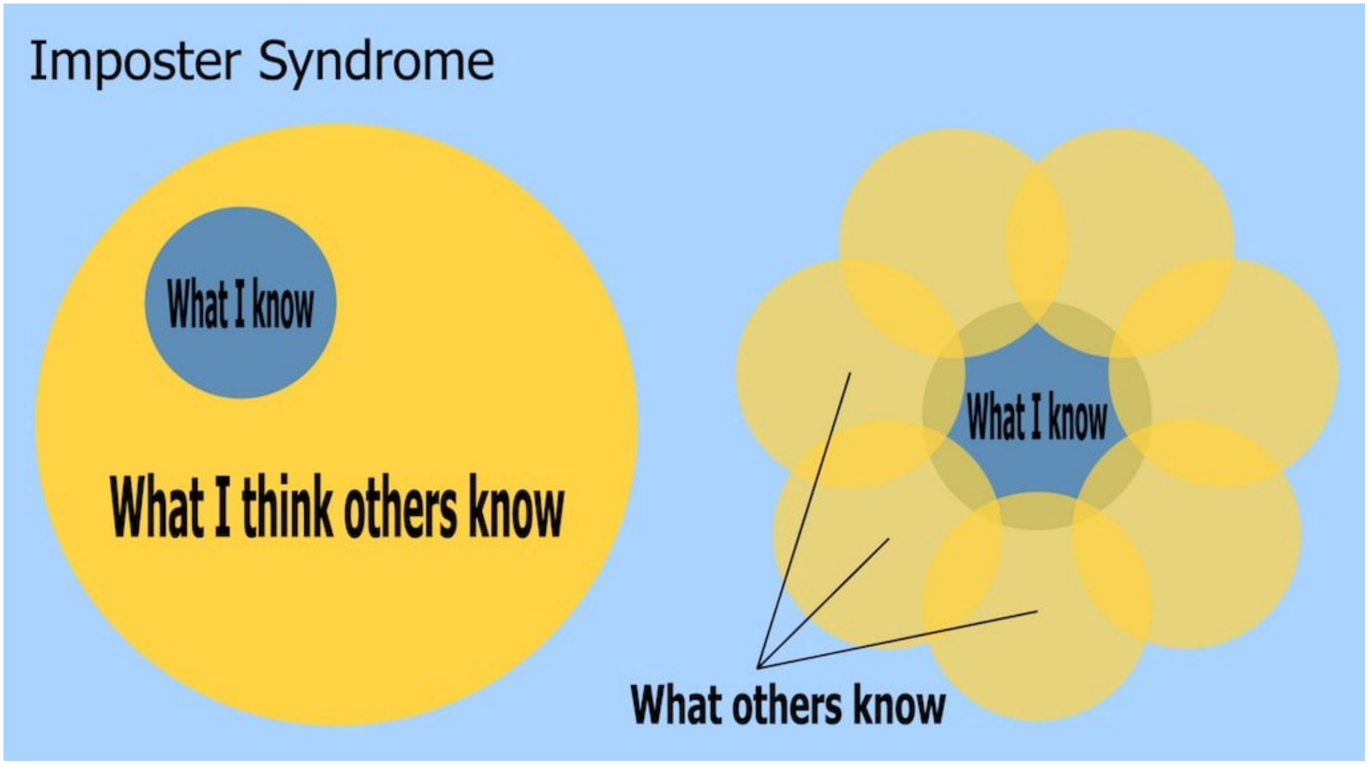
For masking internal complexity with external simplicity, use tools that can help describe a model's decision. SHAP values are a useful technique, and you can layer on additional methods. To explain a building's optimal start time recommendation, we take SHAP values on all features, including engineered features, and combine them into human-understandable feature groups, like weather and internal building conditions. We take a sophisticated machine learning algorithm, simplify it with SHAP values for us to understand, then streamline it further with our knowledge before presenting to customers.

One method for simplifying quantitative information is to prepare a report starting with only one number, and then add other numbers as needed (this also works for graphs). Instead of starting with dozens of figures and removing them, this additive approach ensures that no extraneous stats make it into presentations and reports. Remember that people are not computers, and you shouldn't present results as if they are.

+1 Everyone feels imposter syndrome and makes mistakes; don't let either hold you back

Finally, because this is a significant issue in data science (and other professions), here's a bonus lesson as encouragement: Don't let imposter syndrome or mistakes get you down.

Everyone feels some sense that they don't belong or that they'll eventually be "found out" as incapable of the job, so don't fret if you do. You're not the only one with these thoughts, learning new things is as vital as producing results, and, if you are relatively new, there are benefits to being a novice (like finding new ways to solve problems). Furthermore, it's easy to look around and see people achieving great success, but what you don't see is all the failures they encountered along the way (a form of survivorship bias).



People have different areas of knowledge (Source)

Even the best performer started out as a beginner who made (and continues to make) her share of mistakes. Errors do not mean you shouldn't be a data scientist or computer programmer; they mean you have an opportunity to learn how to do things better. We need more people and more diverse people in data science, and I worry we are excluding skilled candidates by portraying data scientists as elite and having data science positions require years of experience. You can only gain expertise by working in a field, it's not something you have before starting a career. The fact is there's no "typical" path into the data science field. If you think you don't belong because of background or inexperience, the good news is that's a cognitive distortion; data science is not a profession reserved only for a few elite.

. . .

Conclusions

After a year in the field, my initial boundless optimism for data science has shifted to cautious enthusiasm. Machine learning can solve a narrow range of problems well (better than humans) but is not a cure-all for human errors. It's critical to recognize the limitations of the field to avoid overselling data science. Nonetheless, approached with a realistic attitude and keeping these lessons in mind, machine learning can deliver impressive results. The best machine learning systems will help humans, not supplant them, by allowing us to do our jobs more efficiently.

. . .

I appreciate feedback and constructive criticism. The best place to reach me is below in the responses, or on twitter @koehrsen_will.

Data Science | Machine Learning | Education | Towards Data Science | Writing