Data Science Best Practices and Processes

This (draft) document is meant to help onboard new data scientists and serve as a reference for current data scientists. Data science consists of a multidisciplinary field encompassing machine learning & statistics, knowledge representation, visualization, writing, and software engineering. This document covers a variety of subjects and best practices from how to run an Agile data science process, to coding standards, and to project management.

There is no right way to do data science but many ways to do it wrong. This document represents just one approach. Data science is a team sport, and consensus is the best way to define what process works best.

Above all else, we're scientists, and that comes with ethical and process implications. Keep this in mind always.

Code of Conduct

In addition to your organizations ethics and conduct standards, there are a few essential principles we hold dear:

- Act in good faith: we are here to accomplish several goals - deliver value to the company, one another, and ourselves. Whenever you have a choice to make about work or how you treat a colleague, ask whether, in doing so, you further these goals.
- Consider things from other people's perspectives: Working on a team is hard if we expect our colleagues to do all the work of communication. Give people the benefit of the doubt when you disagree, and try to understand what they are really saying and why they might be doing so.
- Create a welcoming environment: This team recognizes that racism, sexism, gender discrimination, and classism can severely impact the way people move through both life and the workplace. It is our policy to proactively address these problems by educating ourselves about them, self-monitoring and letting our colleagues, regardless of their background, get on with work. At the same time, we actively seek to improve the work environment.

Ethics

Ethics is an essential concern in machine learning. Focusing on the ethical impact of a model will potentially reduce bias and improve results. Try to think through the hidden biases and potential misuses of any model you build. Think about this in model choice - try to use transparent methods like traditional statistical models and decision trees when interpretability is important.

8 Principles for ethical machine learning. [https://ethical.institute/]

- Human augmentation: assess the impact of incorrect predictions and, when reasonable, design systems with human-in-the-loop review processes.

- Bias evaluation: continuously develop processes that allow me to understand, document, and monitor bias in development and production.
- Explainability by justification: develop tools and processes to continuously improve transparency and explainability of machine learning systems where reasonable.
- Reproducible operations: develop the infrastructure required to enable for a reasonable level of reproducibility across the operations of ML systems.
- Displacement strategy: identify and document relevant information so that business change processes can be developed to mitigate the impact towards workers being automated.
- Practical accuracy: develop processes to ensure my accuracy and cost metric functions are aligned to the domain-specific applications.
- Trust by privacy: build and communicate processes that protect and handle data with stakeholders that may interact with the system directly and/or indirectly.
- Data risk awareness: develop and improve reasonable processes and infrastructure to ensure data and model security are being taken into consideration during the development of machine learning systems.

Project Philosophy

Projects should, to the best of our ability, be broken up into as many small pieces as possible and is feasible. Dependencies between projects should be managed by tools and proper software configuration management (packages) rather than by including code from one project directly inside another project.

This practice encourages modular design and also encourages independence between projects. Sometimes it is better to implement multiple similar pieces of code in different projects than it is to expect a single library to cover all of these slightly different cases in a simple, coherent way.

When a subset of a project reaches the point of being its own library or utility, don't be afraid to factor it out - git can even preserve the history of the sub-project in a new repository for you.

Many choices we make as software developers are arbitrary, in the sense that they don't significantly impact productivity. Whether a team uses R or Python is probably one such choice, for instance, since both languages provide roughly the same capabilities or their costs and benefits balance out. The value of fixing such decisions isn't in the choices themselves, but on reducing the complexity of the development process for the data science community at large. In other words, while we strive to find the best possible practices to use, don't take this document as bald assertions that these are the optimal choices. The point is to simplify teamwork, collaboration, and add clarity to our process.

Project Planning and Accounting

Data science should be run in an Agile fashion. Agile is a general type of project management process, used mainly for software development, where and solutions evolve through the collaborative effort of self-organizing. This includes Kanban and SCRUM.

- Generally, a data scientist will work on only one or two projects per sprint.
- Project artifacts should be stored in content stores like Jira, Confluence
- Code, documentation and all software artifacts for a project *must* exist in a git repository
- Data and private information should *not* exist in any git repository. Automation in each project should,instead, fetch necessary data to a location outside the git repository (or ignored by it). Instructions about where to get private information required for the project should be in the project documentation.
- Projects must have defined completion criteria before starting

Project planning is hard. The point isn't to get estimates exactly right but to create a manageable history of effort. It is also very good for productivity to resolve questions of priorities and to plan ahead of time so that when we do get to work, we can do so without distraction.

Agile

Negative results are not a failure and in fact, are expected in a healthy data science practice.

The Agile Manifesto

Value: - Individuals and interactions over processes and tools - Working software over comprehensive documentation - Customer collaboration over contract negotiation - Responding to change over following a plan

Agile Data Science Manifesto

Agile Data Science is organized around the following principles:

- Iterate, iterate, iterate: tables, charts, reports, predictions.
- Ship intermediate output. Even failed experiments have output.
- Prototype experiments.
- Integrate the tyrannical opinion of data in product management.
- Climb up and down the data-value pyramid as we work.
- Discover and pursue the critical path to a killer product.
- Get meta. Describe the process, not just the end state.

Sprints

Try to run an agile process with the aim of coordinating downtime throughout the year optimally. No team runs full steam all year - by being honest about the need for creative downtime and team bonding, we'll be more productive.

- Two week sprints
- Mandatory daily stand-ups - 10-15 minutes by the clock
- Stand-ups are not a time for announcements
- Hold lab-style development meetings for the whole team to discuss technical matters and review interim results
- Use chat regularly to communicate- a distributed development culture requires this
- Share code snippets and demo results frequently through the sprint
- retrospectives are important
- scoping of work is important
- code reviews *VERY* important

Planning & Estimation

During this phase, we choose a set of tasks to try to accomplish in the sprint, and we break them down into sub-tasks, which can be completed in one day or less. We generate enough of these sub-tasks that we can reasonably expect to exhaust them in the current sprint. We may also pull such subtasks from the backlog.

Work is, of course, organized in pieces more substantial than one-day tasks. Some teams manage these in terms of Projects, Epics, Stories, and other sorts of tickets.

All projects which take multiple days of work should be well documented. These include but not limited to projects involving DevOps, R&D, presentations, academic collaborations, etc.

Execution

A ticket should be small enough to complete in one day or less. When you work on a ticket, you create a local branch from master for the given project and do your work there. You have some latitude on how you actually work in this branch, but the end result should be a series of small commits which implements the feature.

At this stage, you will rebase your code on the most recent version of master, and either merge your branch into master and push or make a pull request. This is roughly the *git flow* workflow. Read about it here:

http://nvie.com/posts/a-successful-git-branching-model/

Repeat until the end of the sprint.

Demo

Each sprint ends with a sprint demo, which shows either some interesting results of your work or a two-three slide summary of the work you've accomplished in the sprint. Analytics' ultimate purpose is to provide insight into data - if your work does that, be sure to present that result in the most engaging way possible, given the time constraints.

It's very important to include stakeholders in the demo. The content needs to be understandable for non technical managers and customers. It's OK to have technical content but be clear where it is, and always respect the audience. The demo is not a time to show other data scientists your skills at mathematics and statistics - there are other venues for that. Don't derail the sprint demo with side conversations, planning, or peer review. The demo is to communicate what was accomplished during the sprint.

Retrospectives

After the demo comes a retrospective, this is an opportunity to reflect on how the agile process is working. What's working well, what needs to improve, and an action plan for improvements.

Open Time

Two days between each sprint and the next are open times. Use this time to work on long-shot projects or for learning new techniques that aren't necessarily immediately applicable to the current work. Don't use this time for refactoring code: this task can and should be included in sprints since it is a bona fide cost of doing business. Do use sprint time for learning critical path techniques.

The Master Backlog and the Open Business Questions List

Keep a document in the enterprise store that describes at a high level what are the primary business questions and concerns that drive the need for data science. This content should come from the business community and will not cover all that is worked on by data science. The aim should be to give the customer what they want as well as what they need but might not be aware of.

Keep a master backlog of all ideas and use cases - from well formulated to the not so well thought out. Cull this as a team regularly, refining and adding detail to the use cases that make sense to keep. This should be open to all data scientists to collaborate on.

While we may have SME's in certain algorithm or enterprise domains, try not to factor the work in a way that creates knowledge silos. This makes review hard, adds risk to code maintenance, and can stifle innovation.

Statistical Analysis vs. Machine Learning and R vs. Python

There are no rules here. Try to use what's best for the job. R has really good statistics and markdown capabilities. Python might be better for production use when all else is equal. There is a lot of overlap in functionality between R & Python. Generally, R is used for statistical analysis. For Bayesian statistics, the tooling available in Python might be equivalent (JAGS, Stan, TF Probability). Python, Java, R, Spark/Scala are all acceptable for machine learning applications.

[https://github.com/matloff/R-vs.-Python-for-Data-Science]

Version Control**

Version control is one of the most effective tools data scientists have to ensure smooth collaboration and total awareness of the history and meaning of software artifacts. Understanding git is required.

Git Fundamental Ideas

For those unfamiliar with git, this reference may be helpful. https://git-scm.com/book/en/v2

Don't think of your project as a bunch of files. Think of it as a series of commits which are, in actuality, simply *diffs* applied to the previous state of the repository.

A git repository literally is just a set of commit chains and some tools to merge different chains and apply the patches inside each to the current state of the project. But it is all *diffs* at the end of the day.

- Always make small commits that do one thing.
- Always submit pull requests or merge to master a chain of small commits which result in a running, test-passing project.
- Don't document too much code inline. Documentation in source code gets out of date and can make code less readable.
- Do write very informative commit messages. These are perpetually attached to the code they refer to and furnish perpetual, contextual documentation that doesn't drift out of date.
- Do refactor code aggressively. If you follow rules 1-4 it will always be easy to find when things went wrong and to recover the latest, meaningful working version.
- Do get comfortable with git's fancier, patch management capabilities. Learn to revert and cherry pick. Learn to rebase and clean up messes. Know what a detached head state is.

There are different types of git workflows. Get familiar with these.

- fork and pull
- feature branch
- feature branch + merge request

- gitflow / master + develop

For projects with multiple data scientists where the ultimate aim is production, the gitflow workflow is probably best.

Definition of Done for Data Science

bbcrevisit - complete this section

Workflow

If an analysis is to be repeated, it needs to be scripted in a repeatable fashion or exposed via an API. The minimal output of a project always consists of a report. Additional artifacts include R and Python packages, intermediate data for further analysis, blog posts, etc.

Testing

Code should be tested where possible. Any algorithms should have test data passed through. This is not necessary for well-used algorithms in common frameworks (LM in R, for example). This is not only for the benefit of the workflow - but for the benefit of other team members and future interns.

Data Validation

Data should be randomly sampled at various points in the workflow. Any transformations should be manually extracted. This data - manually transformed where needed - should be spot-checked against the original source.

Hardware

Favor disposable hardware and persistent data paradigms. A small community of data scientists should be responsible for maintaining DevOps best practices like connecting containers to git, how to orchestrate multi-container flows, and managing how EMR workflows are developed and deployed.

Languages

- R
- Python
- SQL
- Spark : Scala / Python / R

IDE's vs Notebooks

Notebooks are a great way to start a project but are not appropriate for commercial deployment. Use them, but keep in mind that it's easier to use an IDE from

the start if you know you're going to be developing a package. Use templates for software configuration management.

[https://github.com/uwescience/shablona

[https://usethis.r-lib.org/articles/articles/usethis-setup.html]

Modern devops

All of the modern software engineering paradigms apply to data science.

- Containerization
- API's and REST
- Continuous integration
- Unit testing
- Code coverage
- Logging with levels (DEBUG INFO WARN ERROR) and streams (file stdout etc) : log4r , python logger
- linters for code standards: such as flake8 (which combines the tools pep8 and pyflakes) and lintr
- Auto documentation tools - Doxygen, Roxygen, Sphynx

These are best incorporated in an Agile data science practice by using platforms and templates. GitLab offers a platform for CI. The tidyverse has a library for creating and maintaining R packages - usethis. There are package templates for Python projects as well. [https://github.com/uwescience/shablona] is particularly comprehensive. Look for easy ways to incorporate these into your practice.

A common pitfall in data science is to spend too much time on these concerns. The risk being that all your time will be spent tooling and optimizing and not innovating. Try to strike a good balance here.

Data Science Reading List

This is not a comprehensive list of reading material, but it's an excellent place to start for the basics. The machine learning material generally requires multivariate calculus, probability, statistics, and linear algebra.

Big Data

- Hadoop: The Definitive Guide, 4th Edition by Tom White
- Learning Spark by Matei Zaharia , Patrick Wendell , Andy Konwinski , Holden Karau
- Advanced Analytics with Spark, 2nd Edition by Josh Wills , Sean Owen , Sandy Ryza , Uri Laserson

Statistics

Theory

- Mathematical Statistics and Data Analysis 3rd Edition by John A. Rice
- Mathematical Statistics 2nd Edition by Wiebe R. Pestman
- Mathematical Statistics: Basic Ideas and Selected Topics 1st Edition Peter J. Bickel , Kjell A. Doksum 6

Applied

- Linear Models with R by Julian J. Faraway
- Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models by Julian J. Faraway
- Generalized Linear Models by P. McCullagh and John A. Nelder
- Reliability: Probabilistic Models and Statistical Methods by Lawrence Mark Leemis

Machine Learning

- An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics) by Gareth James,Daniela Witten,Trevor Hastie, Robert Tibshirani
- (Intermediate) The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics) by Trevor Hastie,Robert Tibshirani, Jerome Friedman • Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series) by Kevin P. Murphy
- Probability for Statistics and Machine Learning: Fundamentals and Advanced Topics (Springer Texts in Statistics) Springer Texts in Statistics DasGupta, Anirban
- Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series) 1st Edition by Daphne Koller (Author), Nir Friedman
- Deep Learning (Adaptive Computation and Machine Learning series) by Ian Goodfellow (Author), Yoshua Bengio (Author), Aaron Courville
- Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning) by Bernhard Schlkopf (Author), Alexander J. Smola

Core Competencies

- git [https://www.csc.kth.se/utbildning/kth/kurser/DD2385/material/gitmagic.pdf]
- SQL
- R
- Spark
- Python

General Advice & Thoughts on data science

The highest and most beautiful things in life are not to be heard about, nor read about, nor seen but, if one will, are to be lived. Soren Kierkegaard

Data science is not a new field - statistics, information theory, cybernetics (the science of communications and automatic control systems), and AI have been around a long time. What's new are the engineering breakthroughs that have commoditized previously esoteric software and advances in hardware that have allowed for training much larger models. What else is new is the attention being paid to the field by business leaders.

To the managers of data scientists

Understand the resources and support required to maintain your data scientists. Don't socialize too early. Data science is a craft where many projects have negative results or fail. Do take the time to understand the difference between negative results and failure due to technical reasons. Understand the difference between the different types of data scientists.

[https://hbr.org/2018/11/the-kinds-of-data-scientist]

When is your data big?

- When it cant fit in memory on one computer
- When processing takes more than a few hours

The data science community needs to maintain a few experts responsible for calcualting features on large data sets. Not everyone needs to learn Spark, but be familiar with it and understand how you might be able to write feature calculations in a way that can be deployed in Spark as a custom aggregation.

C++?

Yes, please.

If you want to do anything serious in developing packages and machine learning algorithms, then C++ is a must. Consider Fortran as well. It's not widely discussed, but a vast amount of data science is done running Fortran libraries written long ago. NIST statistics libraries, Arpack, CSuite, and Lapack are a few examples.

On Metrics and Being Data-Driven

Enterprise efforts to be more data-driven can devolve into a culture of false certainty. The influence of results and metrics is dictated not by their reliability but rather by their abundance and the confidence with which they're presented.

This can lead to bad or misguided decision making. The remedy for this is to develop a strong culture of science and a code of ethics in the practice of data science.

Sometimes, more data and more analytics are thrown at a problem when what's needed is a hypothesis-based approach.

Fallacies and Failures in Judgment

Biases not only come from the models we fit. They are an inherent part of the human world the data scientist interacts with. Be familiar with these biases. Try to understand the human biases that arise in the enterprise - utility theory provides a framework to understand how money can make decisions 'irrational' from a modeling point of view. Pricing, commission rates, deals, resource allocations are areas fraught with bias. Be sensitive to these and try to model around them or account for them directly.

- Lucid Fallacy: Mistaking the complex real-world to the well-posed problems of mathematics and laboratory experiments.
- Iatrogenics : Harm done by the healer, like the doctor doing more harm than good.
- Naive Interventionism: intervention with disregard to iatrogenics. The preference or perceived obligation, to "do something" over doing nothing.
- The Agency Problem: A moral hazard and conflict of interest that may arise in any relationship where one party is expected to act in another's best interests.
- Narrative fallacy: Our need to fit a story or pattern to series of connected or disconnected facts. The statistical application is data mining. Fitting a convincing and well-sounding story to the past as opposed to utilizing experimental methodology.

On relevance

Be kind to yourself when evaluating the relevance of your work to the enterprise. The company has hired talented staff, and you want to make an impact. It can sometimes be hard to understand why a model is not adopted, or why one of many competing models developed in isolation is adopted over another. This is part of science, while it may all be correct - it might all be relevant. Focus instead on the correctness of the results and the process in which it's developed. Be kind to your leaders when assessing their reasoning behind making a decision regarding models and results.

On model complexity

Fit a variety of models, especially if using black-box methods. Simple models can provide a form of model validation for more complex models. Most business leaders understand effect size and significance. Models that are interpretable

may sometimes provide more actionable insights than black-box methods with better predictive performance.

Do you want to learn Tensorflow? Many frequentist and Bayesian models can now be fit in Tensorflow. It's not just for CNN,s, BERT, and LSTM's. Learn how to fit a linear model, wide and deep models, or investigate Tensorflow-Probability to learn about powerful samplers for large Bayesian models.

On credit and accountability

We're all responsible for the work that the data science community produces. Any failure of one is a failure for all.

Give credit where credit is due, but keep in mind that work properly captured in agile artifacts speaks for itself.

Data science code should maintain a copyright notice.