Data Science is Software

DRIVENDATA

AI Research
~~Data Science~~ is Software

How doing a *Totally New Thing*
makes you more likely to repeat
the mistakes of the past

AAAI Workshop on Reproducibility in Machine Learning
27 January 2019

DRIVENDATA

About me

Software Engineering
then
Data Science

Co-founder of **DrivenData**

@pjbull | @drivendataorg

DRIVENDATA

DRIVENDATA

Active Competitions

UNTIL EARLY 2018 $100,000

Early detection of abnormal lung tissue saves lives. This is a new type of data challenge where competition-tested algorithms are just the beginning. Help build open software that brings

LET'S GO! →

2 WEEKS LEFT $50,000

Cod, haddock, flounder - these iconic fish have supported New England's fishing fleets for generations. When they're not swimming around the depths of the North Atlantic Ocean,

dmytro CURRENT LEADER

COMPETE →

2 MONTHS, 1 WEEK LEFT

Using environmental data collected by various U.S. Federal Government agencies—from the Centers for Disease Control and Prevention to the National Oceanic and Atmospheric

Divyanshu Suri

COMPETE →

4 MONTHS, 2 WEEKS LEFT

Can you predict whether a donor will return to donate blood given their donation history? Good data-driven systems for predicting donations and supply needs can improve the entire

rcking2 CURRENT LEADER

COMPETE →

4 MONTHS, 3 WEEKS LEFT

The UN's Millennium Development Goals provide the big-picture perspective on international development. We can use these indicators to help understand where to

hristo.buyuklie... CURRENT LEADER

COMPETE →

5 MONTHS, 1 WEEK LEFT

Can you predict which water pumps are faulty? Using data from Taarifa and the Tanzanian Ministry of Water, can you predict which waterpoints are functional, which need some repairs,

nxxx CURRENT LEADER

COMPETE →

1 YEAR, 5 MONTHS LEFT

We're rebooting our first prized competition for fun and education! Budgets for schools and school districts are huge, complex, and unwieldy. It's no easy task to digest

marielgh CURRENT LEADER

COMPETE →

Completed Competitions

COMPETITION HAS ENDED $16,000

Data on penguin populations are limited primarily due to the fact that most monitored colonies are nearby permanent research stations. This means that any remote populations are

ambarishg 1ST PLACE

RESULTS →

COMPETITION HAS ENDED $1,000

US presidential elections come but once every 4 years, and this one's a big one. There are lots of people trying to predict what will happen. Can you top them? In this challenge, you'll predict

tallmeasure 1ST PLACE

RESULTS →

COMPETITION HAS ENDED $10,000

Contribute to open, cutting-edge research on the use of wearables in promoting health and independence for seniors. Passive monitoring can help detect when something is wrong, all

Daniel_FG 1ST PLACE

RESULTS →

COMPETITION HAS ENDED $15,000

Your challenge is to develop a model that will predict the yield of Dar Si Hmad's fog nets for every day during an evaluation period, using historical data about meteorological conditions and

ulery 1ST PLACE

RESULTS →

COMPETITION HAS ENDED $5,000

Metis wants to know: can you identify a bee as a honey bee or a bumble bee? These bees have different behaviors and appearances, but given the variety of backgrounds, positions, and image

ea

RESULTS →

COMPETITION HAS ENDED $5,000

The City of Boston regularly inspects every restaurant to monitor and improve food safety and public health. As in most cities, health inspections are generally random, which can increase

LilianaMedina 1ST PLACE

RESULTS →

COMPETITION HAS ENDED $5,000

Recent literature suggests that the demand for women's health care will grow over 6% by 2020. Given how rapidly the health landscape has been changing over the last 15 years, it's

giba 1ST PLACE

RESULTS →

COMPETITION HAS ENDED $7,500

Budgets for schools and school districts are huge, complex, and unwieldy. It's no easy task to digest where and how schools are using their resources. Education Resource

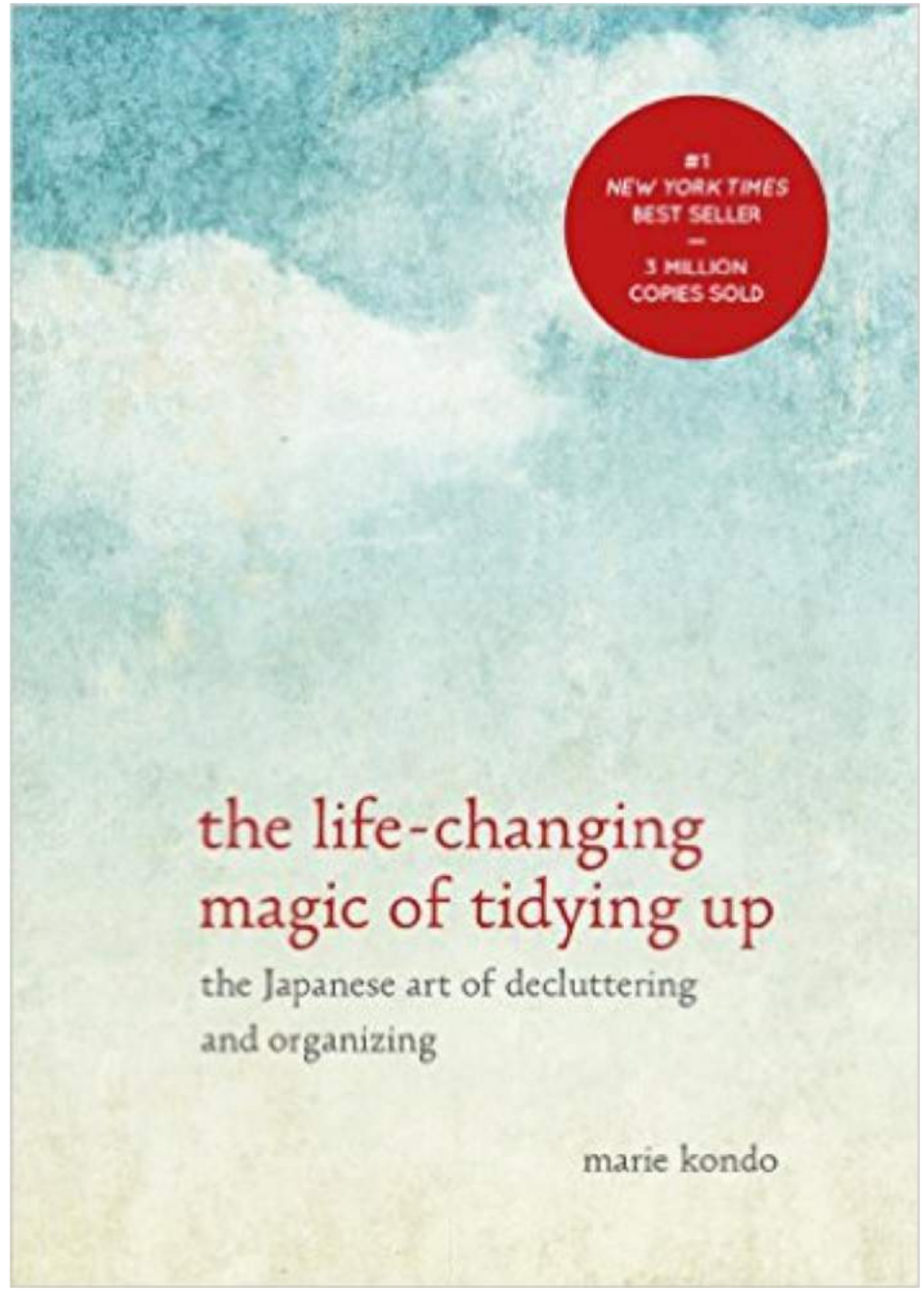
quocnle 1ST PLACE

RESULTS →

```
.
├── Inspection_count_min.jpeg
├── README.Rmd
├── README.html
├── dd_dictionary.csv
├── mallet.rar
├── scripts\ and\ data
|   ├── AllViolations.csv
|   ├── PhaseIISubmissionFormat.csv
|   ├── build_rev_tm.R
|   ├── docsAsTopicsProbs_noStopwords.txt
|   ├── feature_eng.R
|   ├── features_test_phase2.csv
|   ├── features_train_phase2.csv
|   ├── learning_final.R
|   ├── negative-words.txt
|   ├── positive-words.txt
|   ├── rand_neg.txt
|   ├── restaurant_ids_to_yelp_ids.csv
|   ├── rev_tm.txt
|   ├── review_sentiscored.csv
|   ├── run.R
|   ├── sentiment_script.R
|   ├── sub_2_PhaseII_h20.csv
|   ├── yelp.stops
|   └── yelp_academic_dataset_business.json
├── varimp_gbm1.jpeg
├── varimp_gbm2.jpeg
└── varimp_sev.jpeg
```

```
.
├── AllViolations.csv
├── BusinessClass.py
├── GenLearningData.py
├── GenTestingData.py
├── InspectionClass.py
├── LearnTest.py
├── PhaseIISubmissionFormat.csv
├── PhaseIISubmissionFormat_final.csv
├── PhaseIISubmissionFormat_test.csv
├── README.txt
├── ReviewClass.py
├── restaurant_ids_to_yelp_ids.csv
├── yelp_boston_academic_dataset
└── yelp_duplicate_ids.csv
```

```
.
├── Step\ 1\ -\ install\ necessary\ software\ and\ packages.txt
├── Step\ 2\ -\ one-off\ step\ to\ create\ postgresql\ server\ instance\ and\ a\ database.txt
├── Step\ 3\ -\ one-off\ step\ to\ create\ tables\ and\ views\ in\ postgresql.py
└── Step\ 4\ -\ The\ only\ file\ to\ run\ when\ you\ want\ to\ run\ models\ and\ generate\ new\ scores.py
```


#1
NEW YORK TIMES
BEST SELLER
—
3 MILLION
COPIES SOLD
the life-changing
magic of tidying up
the Japanese art of decluttering
and organizing
marie kondo

- Cookiecutter Data Science
- Why use this project structure?
  - Other people will thank you
  - You will thank you
  - Nothing here is binding
- Getting started
  - Requirements
  - Starting a new project
  - Example
- Directory structure
- Opinions
  - Data is immutable
  - Notebooks are for exploration and communication
  - Analysis is a DAG
  - Build from the environment up
  - Keep secrets and configuration out of version control
  - Be conservative in changing the default folder structure
- Contributing
- Links to related projects and references

Cookiecutter Data Science

A logical, reasonably standardized, but flexible project structure for doing and sharing data science work.

Why use this project structure?

> We're not talking about bikeshedding the indentation aesthetics or pedantic formatting standards — ultimately, data science code quality is about correctness and reproducibility.

When we think about data analysis, we often think just about the resulting reports, insights, or visualizations. While these end products are generally the main event, it's easy to focus on making the products *look nice* and ignore the *quality of the code that generates them*. Because these end products are created programmatically, **code quality is still important**! And we're not talking about bikeshedding the indentation aesthetics or pedantic formatting standards — ultimately, data science code quality is about correctness and reproducibility.

It's no secret that good analyses are often the result of very scattershot and serendipitous explorations. Tentative experiments and rapidly testing approaches that might not work out are all part of the process for getting to the good stuff, and there is no magic bullet to turn data exploration into a simple, linear progression.

That being said, once started it is not a process that lends itself to thinking carefully about the structure of your code or project layout, so it's best to start with a clean, logical structure and stick to it throughout. We think it's a pretty big win all around to use a fairly standardized setup like this one. Here's why:

bit.ly/cookieds | drivendata.github.io/cookiecutter-data-science

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.testrun.org
```

Data Innovation for Social Impact

The Machine Learning Reproducibility Checklist (Version 1.0)

For all **algorithms** presented, check if you include:

- ❑ A clear description of the algorithm.
- ❑ An analysis of the complexity (time, space, sample size) of the algorithm.
- ❑ A link to a downloadable source code, including all dependencies.

The Machine Learning Reproducibility Checklist (Version 1.0)

For all **algorithms** presented, check if you include:

- ❑ A clear description of the algorithm.
- ❑ An analysis of the complexity (time, space, sample size) of the algorithm.
- ❑ A link to a downloadable source code, including all dependencies.

“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

— Martin Fowler, "Refactoring: Improving the Design of Existing Code"

Validity

Why does reproducibility matter?

"The fact that an analysis appears in print has no relationship to the likelihood of its being correct."

— Akin's Laws of Spacecraft Design, #17

David Akin
Director of Space Systems Laboratory, University of Maryland
Member, NASA Task Force on Technology Readiness

`execution != correctness`

A corollary: code that executes the first time you try to run it has at least one bug you haven't found yet.

Code-level reproducibility is correlated with code correctness.

Why does reproducibility matter?

Validity

Progress

François Chollet

@fchollet

Follow

Buggy code is bad science. Poorly tuned benchmarks are bad science. Poorly factored code is bad science (hinders reproducibility, increases chances of a mistake). If your field is all about empirical validation, then your code *is* a large part of your scientific output.

12:26 AM - 15 Jul 2018

$$\text{reproducibility} \propto \frac{1}{p(\text{you say “it works on my machine”})}$$

Reproducibility facilitates collaboration, which is essential for scientific progress.

(Also, studies have shown that well-documented articles receive more citations.)

-- Odd Erik Gundersen, et al.
"On Reproducible AI: Towards Reproducible Research,
Open Science, and Digital Scholarship in AI Publications"

Why does reproducibility matter?

Validity

Progress

Applications

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

041c0808-367a-49ff-82dc-c55ec6e75487

Elephant	0.9850
Blank	0.0084
Human	0.0025
Duiker	0.0003
Chimpanzee	0.0002

Created: 2 months, 3 weeks ago
Original location: eleph.mp4

"The concept of technical debt was first introduced by Ward Cunningham in 1992 [. . .] Deferring the work to pay it off results in increasing costs, system brittleness, and reduced rates of innovation [...] *machine learning packages have all the basic code complexity issues as normal code, but also have a larger system-level complexity that can create hidden debt*."

— *Sculley et. al., "Machine Learning: The High Interest Credit Card of Technical Debt"*

The pace at which AI research is incorporated into real applications has been rapidly increasing, but the cost of this code in technical debt is higher than traditional software.

Corollary: Good ideas are left behind when they are locked up in bad code.

Why does reproducibility matter?

Validity

Progress

Applications

How do we do reproducible research?

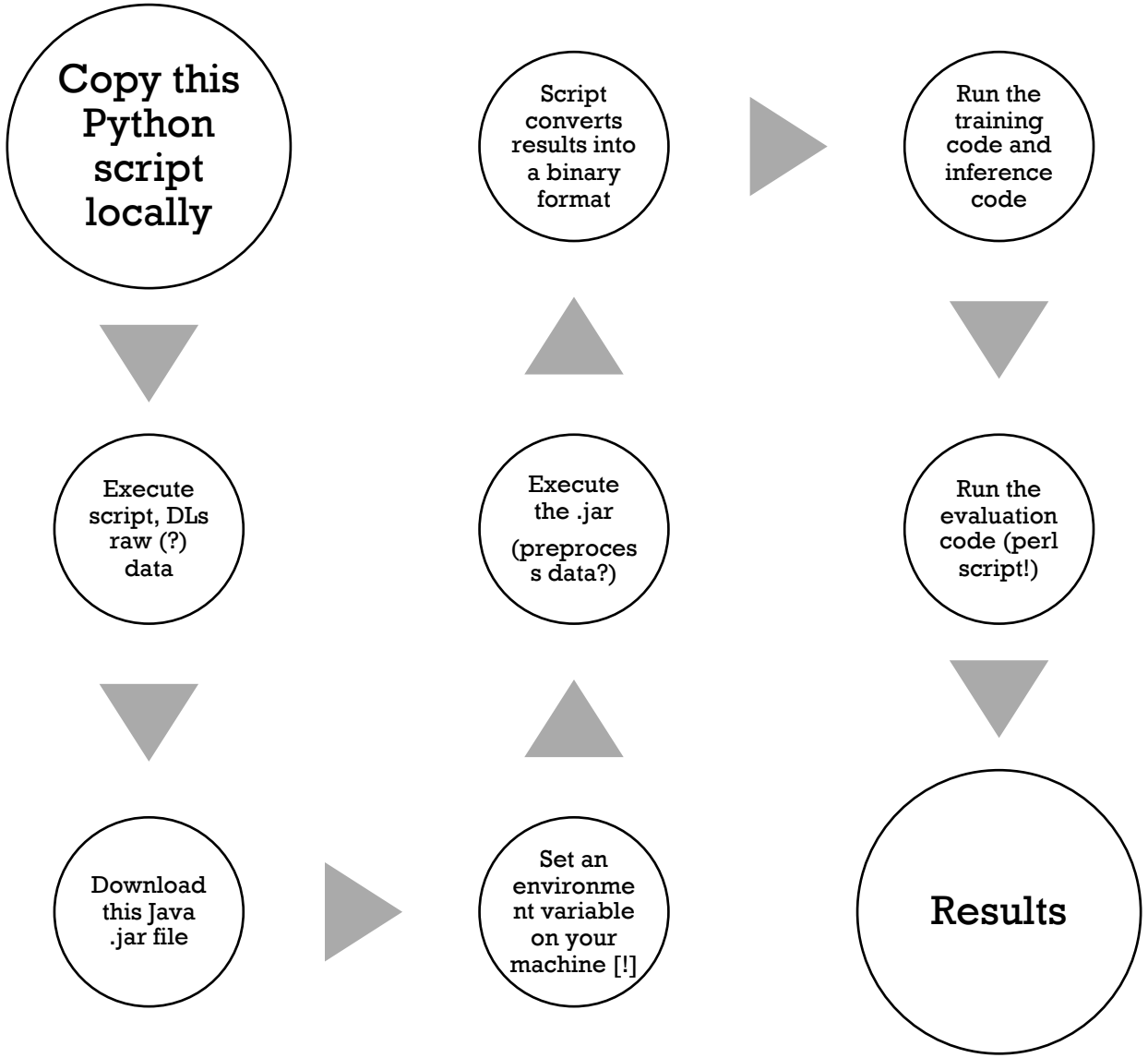
Change our process

Change our output

Real Published Paper Reproduction Instructions...

Time to setup the pipeline
~ 2 days

Time to execute the code on the data:
~ 2 minutes



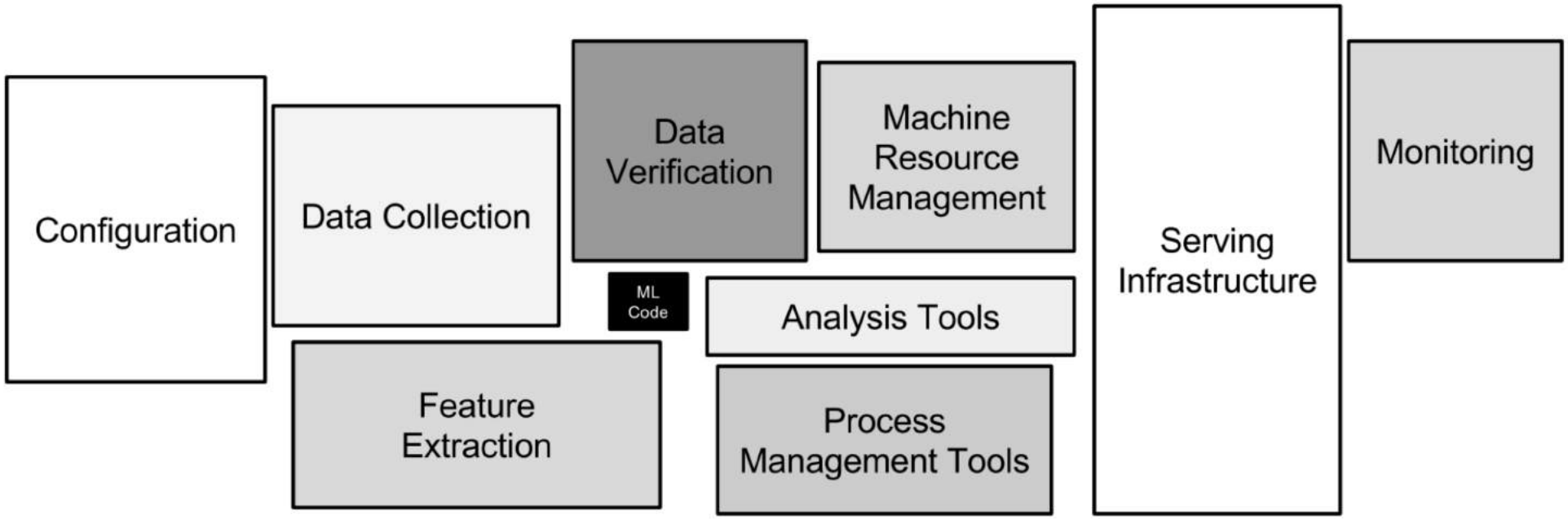
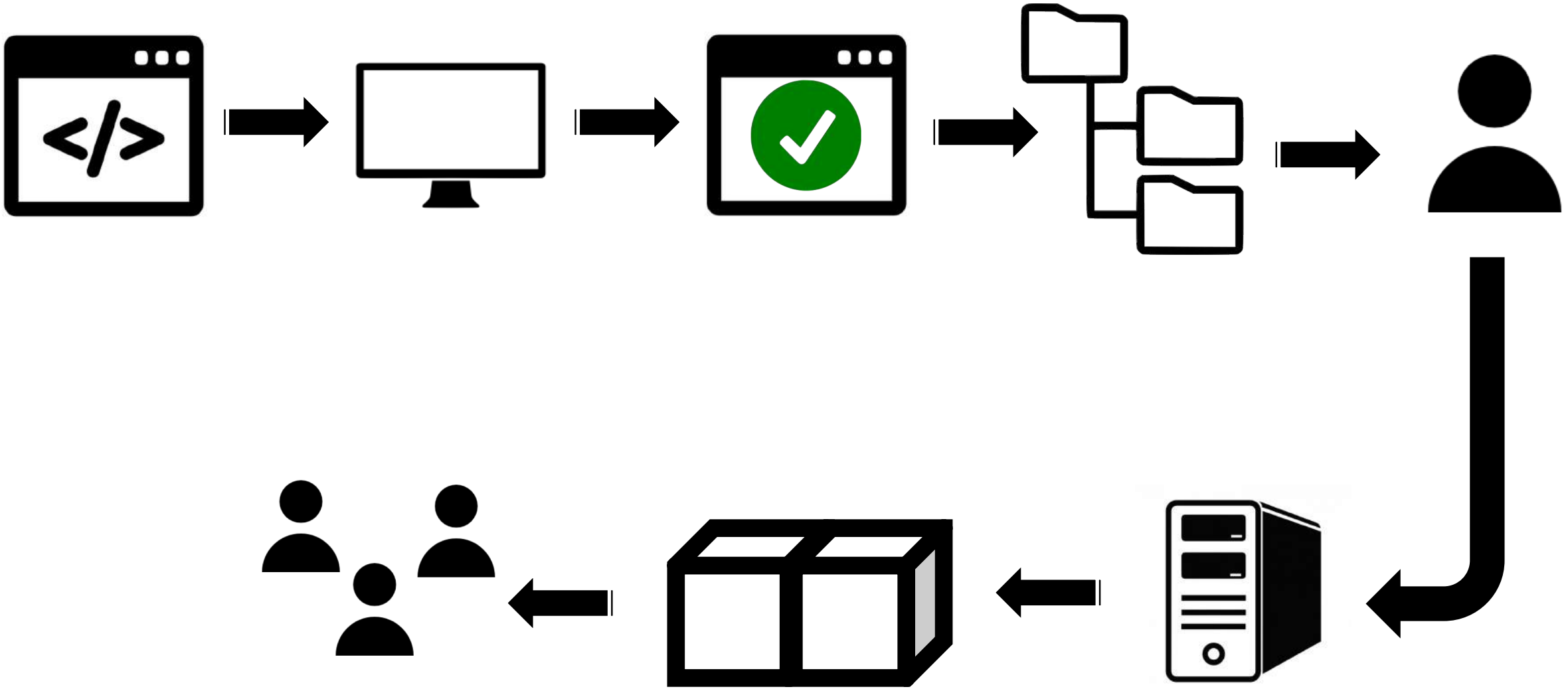


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

— *Sculley et. al., "Machine Learning: The High Interest Credit Card of Technical Debt"*

A typical software workflow



Process

Source control

- Version code for yourself and collaborators
- Backup your existing code
- Single flip of switch to make public when paper is published
- Updates can all happen in one place

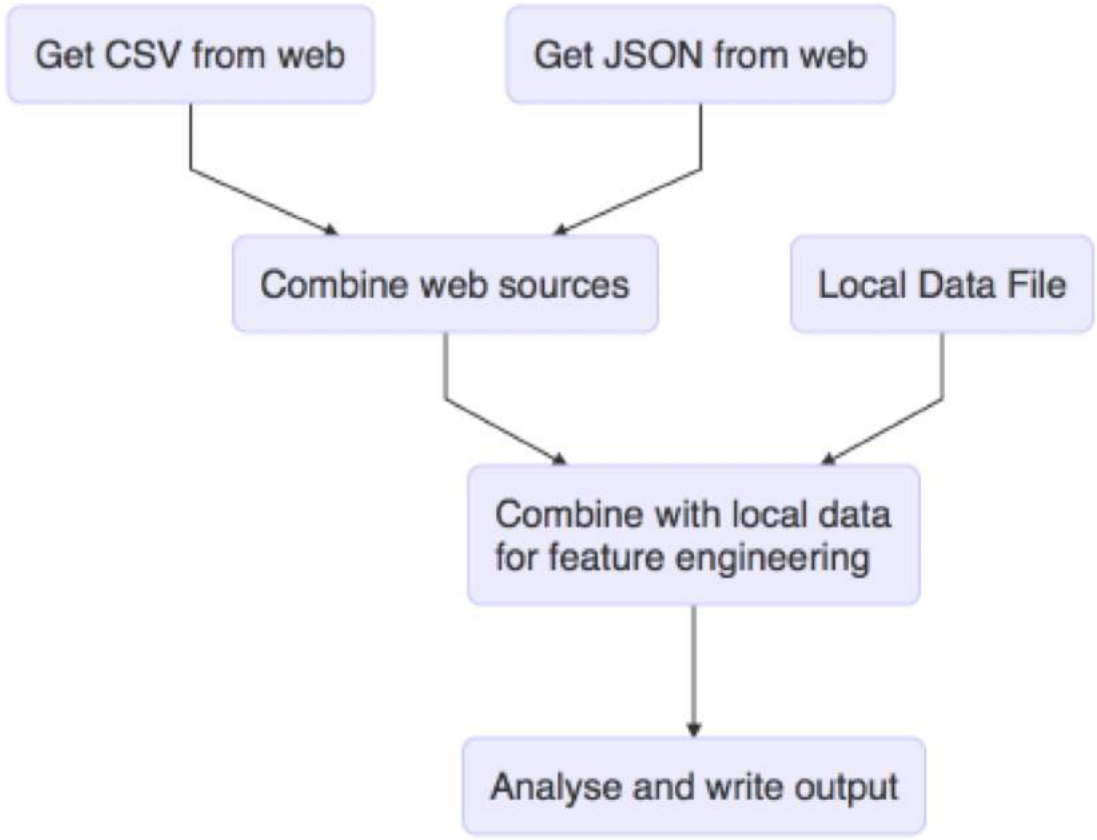
"Writing in 2018, [source control] is such an ingrained cultural norm on modern software development teams that the lack of version control is widely seen as *prima facie* evidence that a team or company is a burning dumpster fire and should be avoided."

— *Isaac Slavitt (preprint on reproducible data science)*

Process

Analysis is a DAG

- Everything comes from somewhere
- The raw data is immutable
- Code as documentation
- Think of your code like an argument



Process

Config separated from code

```
class xgboost.XGBRegressor(max_depth=3,
learning_rate=0.1, n_estimators=100, silent=True,
objective='reg:linear', booster='gbtree', n_jobs=1,
nthread=None, gamma=0, min_child_weight=1,
max_delta_step=0, subsample=1, colsample_bytree=1,
colsample_bylevel=1, reg_alpha=0, reg_lambda=1,
scale_pos_weight=1, base_score=0.5, random_state=0,
seed=None, missing=None, **kwargs)
```

Process

Config separated from code

Commit: 9e2f43bcc0ffeefa2f58fbf22a68b43a838deed5

config.yml

```
n_threads: 4
train_ratio: 0.5
log_level: debug

models:
 xgboost:
  max_depth:
   - 2
   - 5
   - 10
  N_estimators:
   - 50
   - 100
   - 150
   - 200
  ...
 random_forest:
  criterion:
    - gini
    - entropy
...

ensemble:
 voting_classifier:
  voting:
   - soft
   - hard
...
```

results_2018-09-03_14-13-23.log

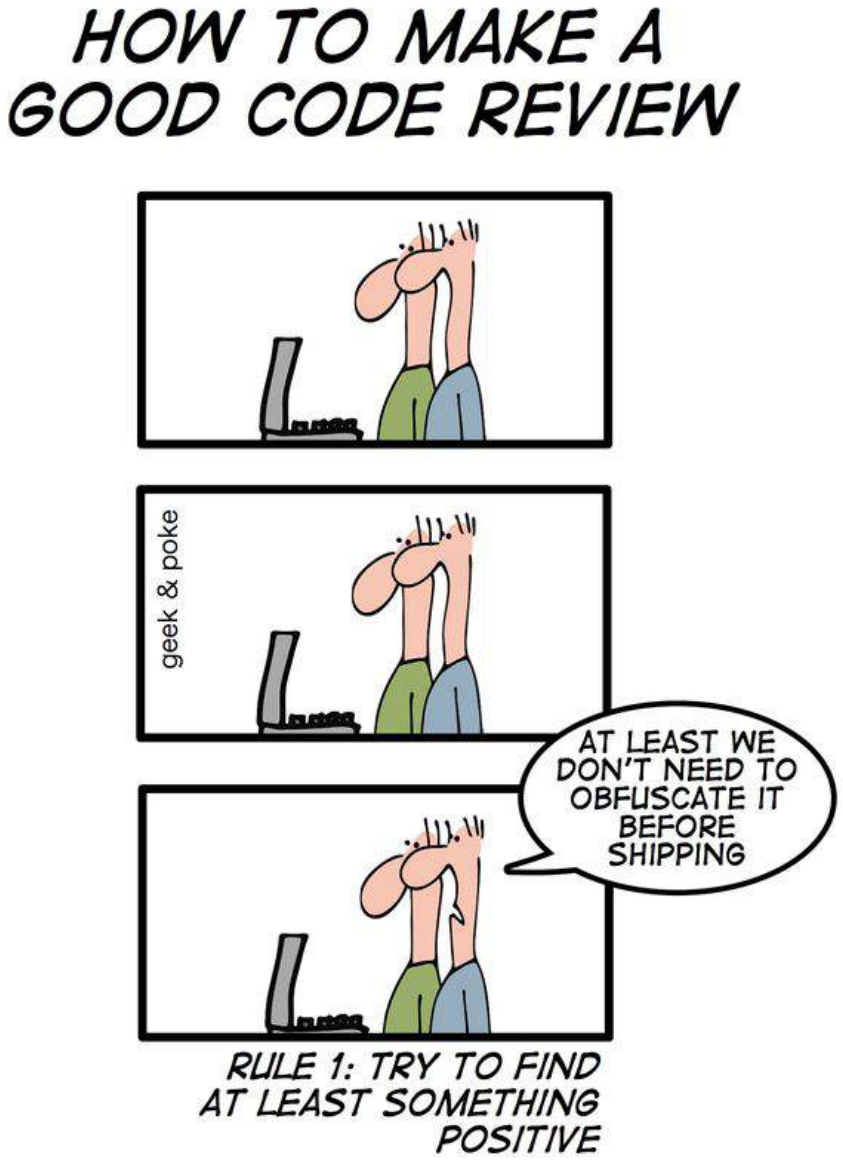
```
2018-09-03 14:10:02 DEBUG reading config file
2018-09-03 14:10:03 INFO reading in and merging data files
2018-09-03 14:10:39 INFO finished loading data
2018-09-03 14:10:39 INFO starting grid search CV using ...
2018-09-03 14:11:01 DEBUG ... 30/120
2018-09-03 14:11:46 DEBUG ... 60/120
2018-09-03 14:12:31 DEBUG ... 90/120
2018-09-03 14:13:03 DEBUG ... 120/120
2018-09-03 14:13:03 INFO training voting classifier on ensemble of 3 best models...
2018-09-03 14:13:19 INFO making predictions
2018-09-03 14:13:22 INFO ---- MODEL RESULTS ----
-----------------------------------------------------------
[ metrics ]
- precision: 0.9624
- recall:    0.9388
- f1-score:  0.9514
- support:   32124

[ best parameters ]
xgboost = {'max_depth': 5, ...snip...}
random_forest = {'max_depth': None, ...snip...}
adaboost = {'algorithm': 'SAMME.R', ...snip...}

[ time ]
real    3m20.157s
user    3m20.113s
sys     11m54.045s
-----------------------------------------------------------
2018-09-03 14:13:23 INFO writing predictions to output/results_2018-09-03_14-13-
23.csv
```

Process

Code review

- You review everything else, don't you?
- Catch problems early
- Share knowledge with colleagues
- Learn something yourself

(Protip: you can code review most data analysis code regardless of language for validitiy)



Output

Data is accessible

- AI research is suffers from a lack of diversity of quality data.
- If you're working on methods, collect, document, and release your data.
- Include a data-use license

EXPERT OPINION

Contact Editor: **Brian Brannon,** bbrannon@computer.org

The Unreasonable Effectiveness of Data

Alon Halevy, Peter Norvig, and Fernando Pereira, *Google*

Eugene Wigner's article "The Unreasonable Effectiveness of Mathematics in the Natural Sciences"[1] examines why so much of physics can be neatly explained with simple mathematical formulas such as $f = ma$ or $e = mc^2$. Meanwhile, sciences that involve human beings rather than elementary particles have proven more resistant to elegant mathematics. Economists suffer from physics envy over their inability to neatly model human behavior. An informal, incomplete grammar of the English language runs over 1,700 pages.[2] Perhaps when it comes to natural language processing and related fields, we're doomed to complex theories that will never have the elegance of physics equations. But if that's so, we should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data.

One of us, as an undergraduate at Brown University, remembers the excitement of having access to the Brown Corpus, containing one million English words.[3] Since then, our field has seen several notable corpora that are about 100 times larger, and in 2006, Google released a trillion-word corpus with frequency counts for all sequences up to five words long.[4] In some ways this corpus is a step backwards from the Brown Corpus: it's taken from unfiltered Web pages and thus contains incomplete sentences, spelling errors, grammatical errors, and all sorts of other errors. It's not annotated with carefully hand-corrected part-of-speech tags. But the fact that it's a million times larger than the Brown Corpus outweighs these drawbacks. A trillion-word corpus—along with other Web-derived corpora of millions, billions, or trillions of links, videos, images, tables, and user interactions—captures even very rare aspects of human behavior. So, this corpus could serve as the basis of a complete model for certain tasks—if only we knew how to extract the model from the data.

Learning from Text at Web Scale

The biggest successes in natural-language-related machine learning have been statistical speech recognition and statistical machine translation. The reason for these successes is not that these tasks are easier than other tasks; they are in fact much harder than tasks such as document classification that extract just a few bits of information from each document. The reason is that translation is a natural task routinely done every day for a real human need (think of the operations of the European Union or of news agencies). The same is true of speech transcription (think of closed-caption broadcasts). In other words, a large training set of the input-output behavior that we seek to automate is available to us *in the wild*. In contrast, traditional natural language processing problems such as document classification, part-of-speech tagging, named-entity recognition, or parsing are not routine tasks, so they have no large corpus available in the wild. Instead, a corpus for these tasks requires skilled human annotation. Such annotation is not only slow and expensive to acquire but also difficult for experts to agree on, being bedeviled by many of the difficulties we discuss later in relation to the Semantic Web. The first lesson of Web-scale learning is to use available large-scale data rather than hoping for annotated data that isn't available. For instance, we find that useful semantic relationships can be automatically learned from the statistics of search queries and the corresponding results[5] or from the accumulated evidence of Web-based text patterns and formatted tables,[6] in both cases without needing any manually annotated data.

A quick aside on trained models

You spent all that time and money training it and tuning your hyperparameters.

Pay it forward!

R0: Inference Reproducible: The published results of an experiment can be reproduced by running inference on the published test dataset and loading the published pre-trained model.

R1: Experiment Reproducible The results of an experiment are *experiment reproducible* when the execution of the same implementation of an AI method produces the same results when executed on the same data. This is often called *repeatability*.

R2: Data Reproducible The results of an experiment are *data reproducible* when an experiment is conducted that executes an alternative implementation of the AI method that produces the same results when executed on the same data. This is often called *replicability*.

R3: Method Reproducible The results of an experiment are *method reproducible* when the execution of an alternative implementation of the AI method produces consistent results when executed on different data. This is often called *reproducibility*.

Output

Project organization

- Learn from software projects
- All web frameworks have the same abstractions, and code lives in the same place
- Every linux file system has the same fundamental directory structure

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.testrun.org
```

Output

Specified dependencies

- Best case: machine readable
- Next best case: well-specified versions
- OK case: mentioned in the documentation
- Not great: mentioned in the paper
- Really not great: not mention of dependencies

```
Traceback (most recent call last):
  File "eval.py", line 12, in <module>
    encoder = torch.load('encoder.pt', map_location=lambda storage, loc: storage)
  File "/opt/anaconda/lib/python3.6/site-packages/torch/serialization.py", line 22
, in load
    return _load(f, map_location, pickle_module)
  File "/opt/anaconda/lib/python3.6/site-packages/torch/serialization.py", line 37
, in _load
    result = unpickler.load()
  File "/opt/anaconda/lib/python3.6/site-packages/torch/tensor.py", line 115, in _
setstate__
    self.set_(*state)
TypeError: set_ received an invalid combination of arguments - got (torch.FloatSto
age, int, tuple, tuple), but expected one of:
 * no arguments
 * (torch.cuda.FloatTensor source)
 * (torch.cuda.FloatStorage storage)
 * (torch.cuda.FloatStorage sourceStorage, int storage_offset, int ... size)
      didn't match because some of the arguments have invalid types: (torch.FloatS
orage, int, tuple, tuple)
 * (torch.cuda.FloatStorage sourceStorage, int storage_offset, torch.Size size)
 * (torch.cuda.FloatStorage sourceStorage, int storage_offset, torch.Size size, tu
le strides)
```

What are the themes that we have heard today across talks?
What is the single biggest challenge to addressing the theme?
What is the single best existing solution from what you heard or your experience?

Source control

Source control is basic professionalism for anyone working with code

Analysis is a DAG

Automate testing the entire pipeline by formally specifying your DAG; treat raw data as immutable

Config separated from code

The parameters of your experiments should be separated from your code versions

Code review

Don't you have others read your paper before submitting?

Data is accessible

Downloadable, as close to raw as possible, licensed; include trained models

Project Organization

Following a common template reduces cognitive overheard for yourself and others

Specified dependencies

Hardware, operating system, libraries, version of your code

High-quality, reproducible code

A byproduct of the changes to process, and forethought about the output

Questions?

My question for the audience:

How do we change the institutions, structures, processes, cultural norms, tools, and mindsets to enable reproducibility in AI research?