W01D4

Miniproject Introductions

Instructor: Eric Elmoznino

Outline for today

- Python/APIs review
- Value of miniprojects
- Strategies for miniprojects
- Project presentations

Python/APIs review

Value of miniprojects

Projects in data science

- Primary source of knowledge/experience
- Most important part of job interviews
- Benchmarking a new technique
    - Example: you develop a new classification algorithm and want to compare it to existing ones
    - Use popular publicly available benchmark datasets (e.g. from Kaggle)
- New use-case for existing technique
    - Example: you apply a product-recommendations algorithm to your business's order history

Projects at Lighthouse Labs

- Question/answer-based or open-ended
  - Come up with some of your own problem statements and goals to frame your work
  - Not limited to *just* answering the questions
- Miniprojects
  - APIs - Python data structures (individual)
  - Databases (SQL) - Pandas (individual)
  - Feature engineering - Dimensionality reduction - Unsupervised learning (pairs)
  - Supervised learning - Deployment (individual)
  - Deep Learning - NLP (individual)
- Open-ended final project

Individual and group work

- Two minds do not necessarily code twice as fast!
- Typical scenario: team of data scientists each with their own project
- Parallelization
  - Different sub-tasks
  - Different files
  - No pair-coding (unless to help someone)!
- Code reviews of each-other’s work
- GitHub: push only at working milestones (no errors)

This week's miniproject

- Part 1: Transport of London API
  - Example: Plan the journey from Heathrow Airport to Tower Bridge using Public Transport, Taxi or Bike? Which way is the fastest?
- Part 2: The Movie Database API (stretch)
  - Example: Find top 5 trending movies
- Challenges:
  - Working with difficult documentation (poor descriptions of what input/return values are)
  - Parsing complex data structures (nested lists/dictionaries)
- 7 minute presentation (plus 1 minute feedback)

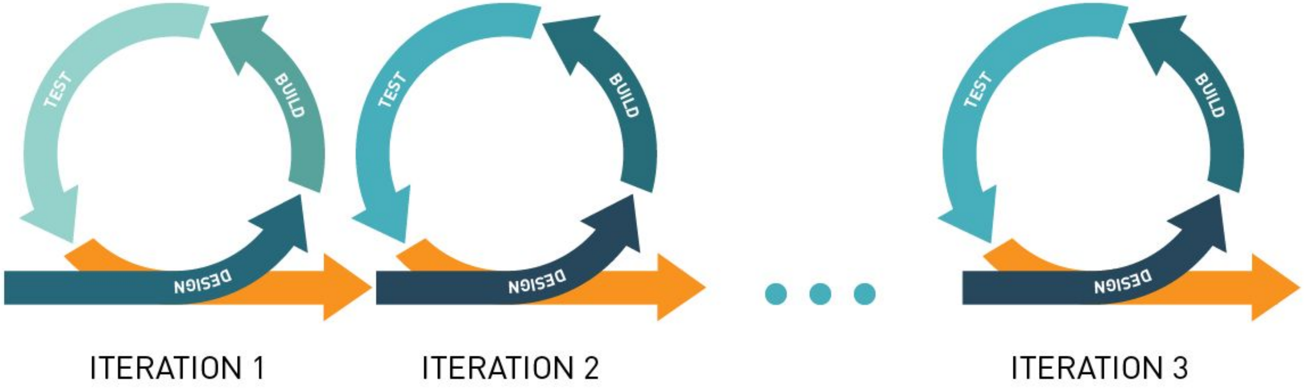
Strategies for miniprojects

Code

- Define functions and/or classes whenever possible (e.g. get_transit(url))
    - Any time you find yourself writing similar code again and again, make a function
- Save trained models and only retrain when needed
- Save GET request results and only fetch when needed (e.g. function that checks if data has been fetched, fetches data at a URL, then saves it)
- Jupyter code blocks should have a clearly defined goal.
- Periodically refactor code (i.e. clean up, reorganize, consolidate)

Iterative progress and difficulty

- Make a minimum viable product (MVP) early
- Dataset difficulty (e.g. one company's stock before arbitrary company)
- Model complexity (e.g. autoregressive before LSTM)
- Task complexity (e.g. past stock before adding sentiment)



Explore the dataset, API, and other tools

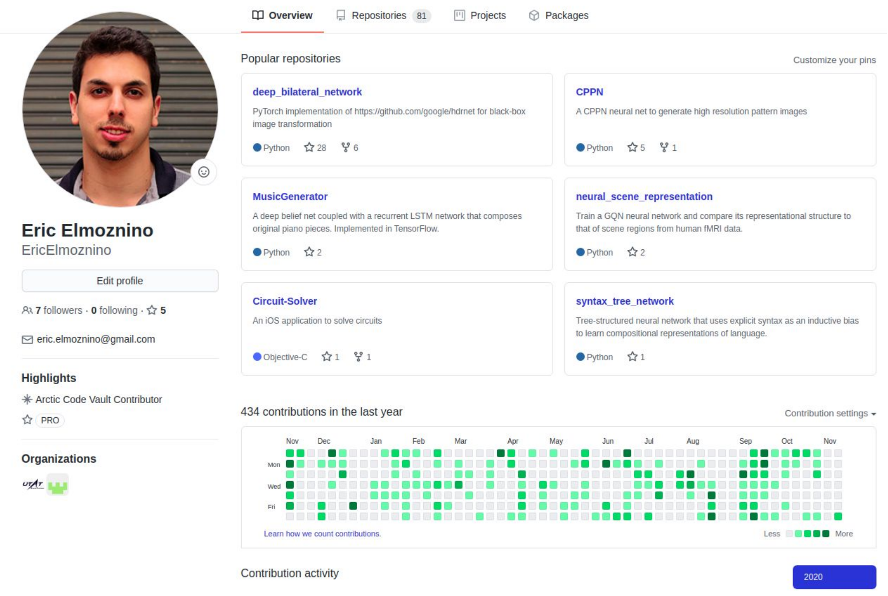
- Play with the dataset
  - What do the variables mean?
  - Which variables are categorical, ordinal, and continuous?
  - Range/variance of each variable?
  - Plot the dataset!!!
- Try out the API functions and explore the returned structure
  - For json, print using JSON(response)!

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	CF
1	Flow ID	Source IP	Source Pc	Destinati	Destinati	Protocol	Timestam	Flow Dura	Total Fwd	Total Bacl	Total Leng	Total Len	Fwd Pack	Fwd Pack	Fwd Pack	Fwd Pack	Label
2	10.42.0.15	10.42.0.15	58063	104.46.62.	443	6	13/06/201	209484	1	2	913	309	913	913	913	0	BENIGN
3	10.42.0.15	10.42.0.15	58063	104.46.62.	443	6	13/06/201	31308	2	0	85	0	85	0	42.5	60.10408	BENIGN
4	137.116.1	10.42.0.15	36864	137.116.1	443	6	13/06/201	13333	2	0	85	0	85	0	42.5	60.10408	BENIGN
5	137.116.1	137.116.1	443	10.42.0.15	36864	6	13/06/201	45	2	0	0	0	0	0	0	0	BENIGN
6	172.217.2.	10.42.0.15	34482	172.217.2.	443	6	13/06/201	22164	1	1	0	0	0	0	0	0	BENIGN
7	172.217.2.	10.42.0.15	56284	172.217.2.	443	6	13/06/201	22194	1	1	0	0	0	0	0	0	BENIGN
8	172.217.2.	10.42.0.15	34483	172.217.2.	443	6	13/06/201	22171	1	1	0	0	0	0	0	0	BENIGN
9	172.217.6.	10.42.0.15	43824	172.217.6.	443	6	13/06/201	36804	1	1	0	0	0	0	0	0	BENIGN
10	172.217.1	10.42.0.15	39937	172.217.1	443	6	13/06/201	118922	1	1	0	0	0	0	0	0	BENIGN
11	10.42.0.15	10.42.0.15	35047	104.41.20	443	6	13/06/201	64883	2	0	85	0	85	0	42.5	60.10408	BENIGN
12	10.42.0.15	104.41.20	443	10.42.0.15	35047	6	13/06/201	6928	2	0	0	0	0	0	0	0	BENIGN
13	172.217.0.	10.42.0.15	45711	172.217.0.	443	6	13/06/201	22358	1	1	0	0	0	0	0	0	BENIGN
14	172.217.0.	10.42.0.15	35363	172.217.0.	443	6	13/06/201	22358	1	1	0	0	0	0	0	0	BENIGN
15	172.217.3.	10.42.0.15	55798	172.217.3.	443	6	13/06/201	36704	1	1	0	0	0	0	0	0	BENIGN
16	172.217.0.	10.42.0.15	37583	172.217.0.	443	6	13/06/201	22465	1	1	0	0	0	0	0	0	BENIGN
17	172.217.0.	10.42.0.15	52935	172.217.0.	443	6	13/06/201	22461	1	1	0	0	0	0	0	0	BENIGN
18	172.217.2.	10.42.0.15	45286	172.217.2.	443	6	13/06/201	22476	1	1	0	0	0	0	0	0	BENIGN

{} JSON
  [] navigations
    {} 0
      disp_order : "1"
      menu_id : "25266"
      [] nodes
        {} 0
          disp_order : "2"
          menu_id : "18951"
          type : "DOC"
      type : "MENU"
    {} 1
      disp_order : "20"
      menu_id : "25204"
      [] nodes
        {} 0
          disp_order : "1"
          menu_id : "10295"
          type : "DOC"
        {} 1
          disp_order : "10"

Why use git?

- **Public**: Employers can see all the projects you've worked on
- **Versioned**: You will have a history and can roll back to old commits
- **Server deployment**: Just git pull to any new machine
- **Teamwork**: Everyone can work on their own copy and working versions to the master copy



Important git commands

- `git init`: Create a local repository in the current folder
- `git remote add origin [GitHub repo URL]`: Connect your local repository to the remote on you created on GitHub
- `git add [filename]`: Add a file to the local repository
- `git commit -am [commit message]`: Store commits (i.e. changes) in the local repository
- `git push`: Send local changes to the remote repository

Project presentations

General pointers

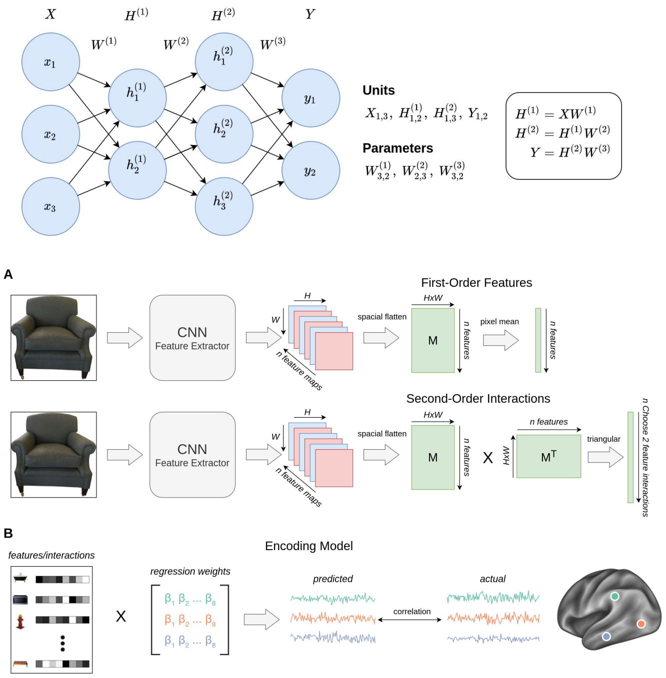
- Present as if to a client (who has some data science knowledge)
- Make it a story
    - What is the problem?
    - What is the dataset?
    - How did you analyze the dataset?
    - What were the findings?
- You can walk through code, but only as a chronological reference for explaining how you analyzed the data
    - However, I recommend having *no code at all*

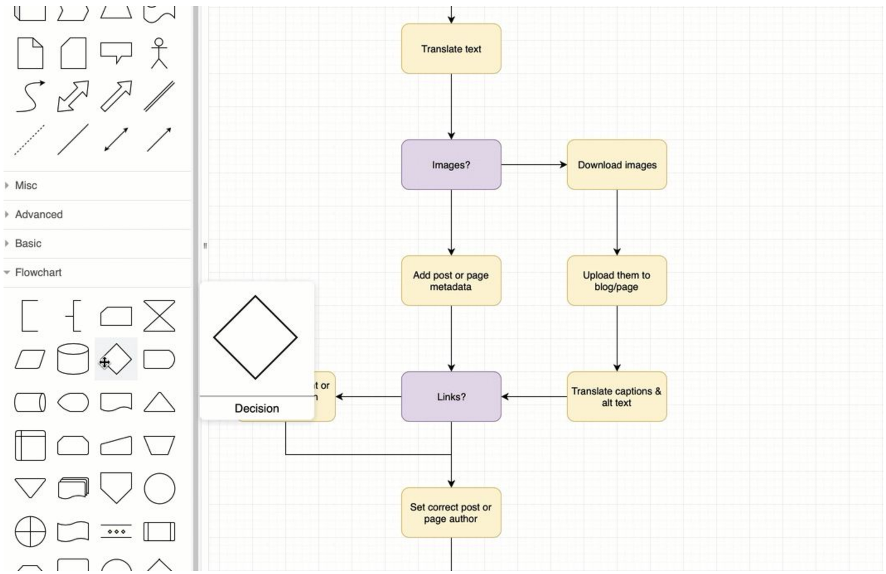
Presentation structure

- **Motivation**: What is the problem? Why is it important (either business, public good, or research perspective)?
- **Task**: Problem from a technical perspective. Description of the dataset, algorithm inputs/outputs, analyses done using model
- **Modeling**: *Important* aspects of your approach. How did you process the data or engineer features? What model did you use? Use schematics!
- **Results**: Visuals! Show metrics and experiments. Demo (if any)
- **Conclusions**: What worked? What didn't (and why)? How are we better off? Where could the project go next?

Figures: draw.io

- Good for schematics, model diagrams, shapes, math typesetting, etc.





Figures: python plotting libraries

- Good for displaying information about your dataset and results