FRAMING THE ML PROBLEM

What is ML?

The process of training a piece of software, called a model, to make useful predictions using a data set.

- The model can then make predictions from new unforeseen data

- Inputs = features
- Outputs = labels

The ML Process

- Hypothesis
- Data
  - Obtain
  - Explore
  - Clean
- Model
  - Train
  - Evaluate
  - Repeat
- Build the data pipeline
- Run the model
  - Monitor
  - Evaluate
  - Adapt

The ML Mindset

- Traditional software development
  - Waterfall
    - Requirements -> Design -> Implementation -> Testing -> Release
  - Agile
    - Preliminary Requirements -> Prototype -> User Feedback -> Repeat (quickly) until finished/satisfied
- ML mindset
  - **Experiment** until you find a working (useful) model
  - Produces bugs that are difficult to debug
    - E.g. skewed data, unexpected data interpretations

Scientific Method

Step	Example
1. Set the research goal.	I want to predict how heavy traffic will be on a given day.
2. Make a hypothesis.	I think the weather forecast is an informative signal.
3. Collect the data.	Collect historical traffic data and weather on each day.
4. Test your hypothesis.	Train a model using this data.
5. Analyze your results.	Is this model better than existing systems?
6. Reach a conclusion.	I should (not) use this model to make predictions, because of X, Y, and Z.
7. Refine hypothesis and repeat.	Time of year could be a helpful signal.

ML Problem Spectrum

- Supervised Learning
  - The model is provided with **labeled** training data (e.g. examples)
    - The data contains features (inputs) and their corresponding labels
    - The relationship between the two is the model
  - *Any potential issues you see applying this method?*
- Unsupervised Learning
  - the goal is to identify meaningful patterns in the data
  - the machine must learn from an **unlabeled** data set
  - model has no hints how to categorize each piece of data
- Think of potential ML problems lying in the **spectrum of supervision** (from supervised to unsupervised)

Reinforcement Learning (RL)

- A different type of ML
- No data needed!
  - You don’t collect labeled examples
- In RL
  - You tell the model (agent) the goal
  - During training, the agent receives a reward when reaching the goal (the reward function)
- Sound tempting?
  - Can be difficult to come up with a good reward function
  - RL models are less stable and predictable than supervised models
  - You need a way for agent to interact with the environment of interest
- Examples: https://neptune.ai/blog/reinforcement-learning-applications
- Why RL is hard and maybe unnecessary for many problems: https://www.alexirpan.com/2018/02/14/rl-hard.html

Type of ML Problems

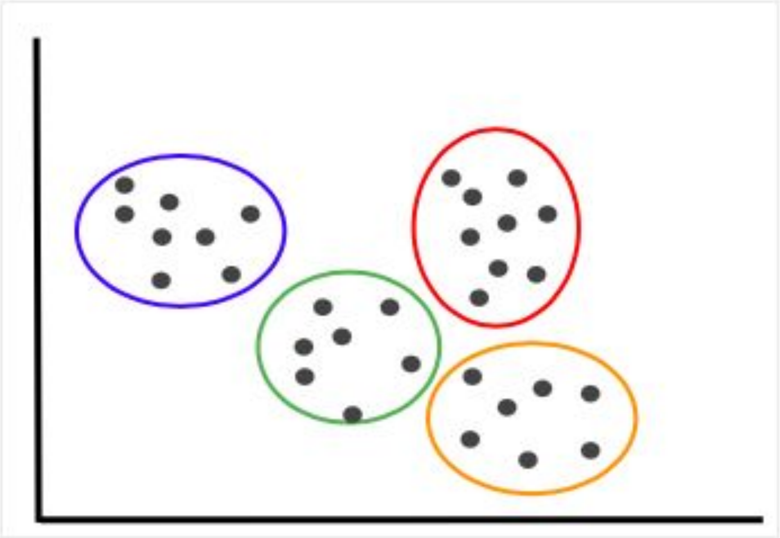
Type of ML Problem	Description	Example
Classification	Pick one of N labels	Cat, dog, horse, or bear
Regression	Predict numerical values	Click-through rate
Clustering	Group similar examples	Most relevant documents (unsupervised)
Association rule learning	Infer likely association patterns in data	If you buy hamburger buns, you're likely to buy hamburgers (unsupervised)
Structured output	Create complex output	Natural language parse trees, image recognition bounding boxes
Ranking	Identify position on a scale or status	Search result ranking

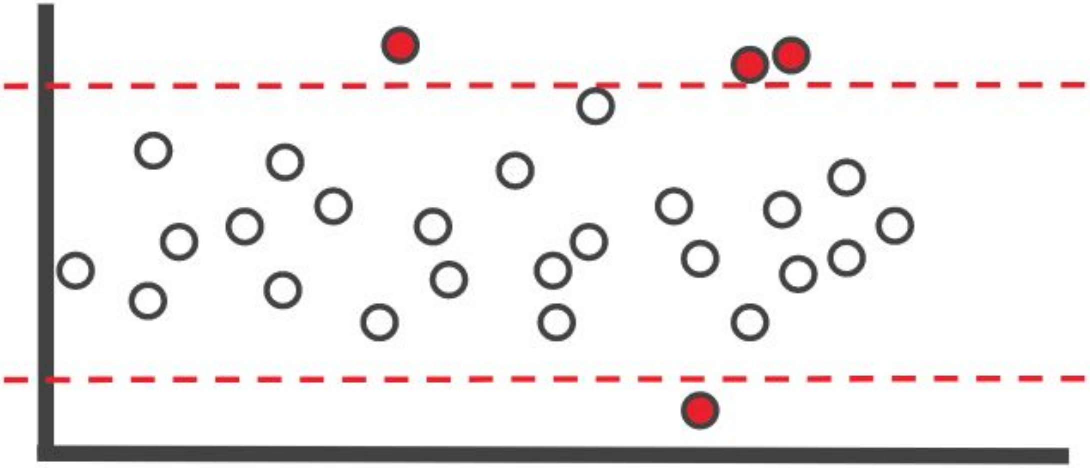
Identifying Good Problems for ML

- Clear use case
- Know the Problem Before Focusing on the Data
- Log data is plentiful
- Predictive power
  - Your features (inputs) contain predictive power.
- Predictions vs Decisions
  - **Aim to make decisions, not just predictions.**
  - Your product should take (useful) action on the output of the model. “Insights” are not a good use case ML.

Hard ML Problems

- Clustering
- Anomaly detection
- Causation
- No existing data





Deciding on ML

- Start Clearly and Simply
  - In plain terms, what would you like your ML model to do?
- What is Your Ideal Outcome?
- Success and Failure Metrics
  - How will you know if your system has succeeded or failed?
  - Are the Metrics Measurable?
- What Output Would You Like the ML Model to Produce?
  - which type of output are you looking for: a number, a label, a cluster, or something else?
- **How might you solve your problem without ML?**

Formulate Your Problem as an ML Problem

- Articulate your problem.
- Start simple.
- Identify Your Data Sources.
- Design your data for the model.
- Determine where data comes from.
- Determine easily obtained inputs.
- Ability to Learn.
- Think About Potential Bias.

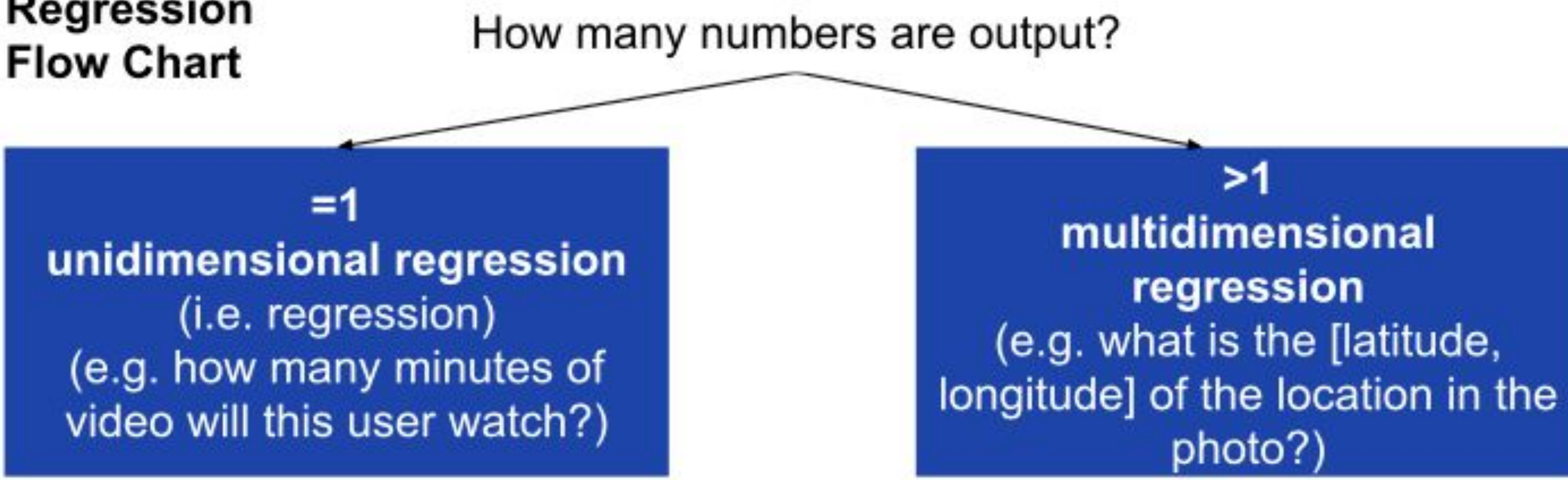
Articulate your problem

- Our problem is best framed as …
  - Binary classification
  - Unidimensional regression
  - Multi-class single-label classification
  - Multi-class multi-label classification
  - Multidimensional regression
  - Clustering (unsupervised)
  - Other (translation, parsing, bounding box id, etc.)
- Which predicts …

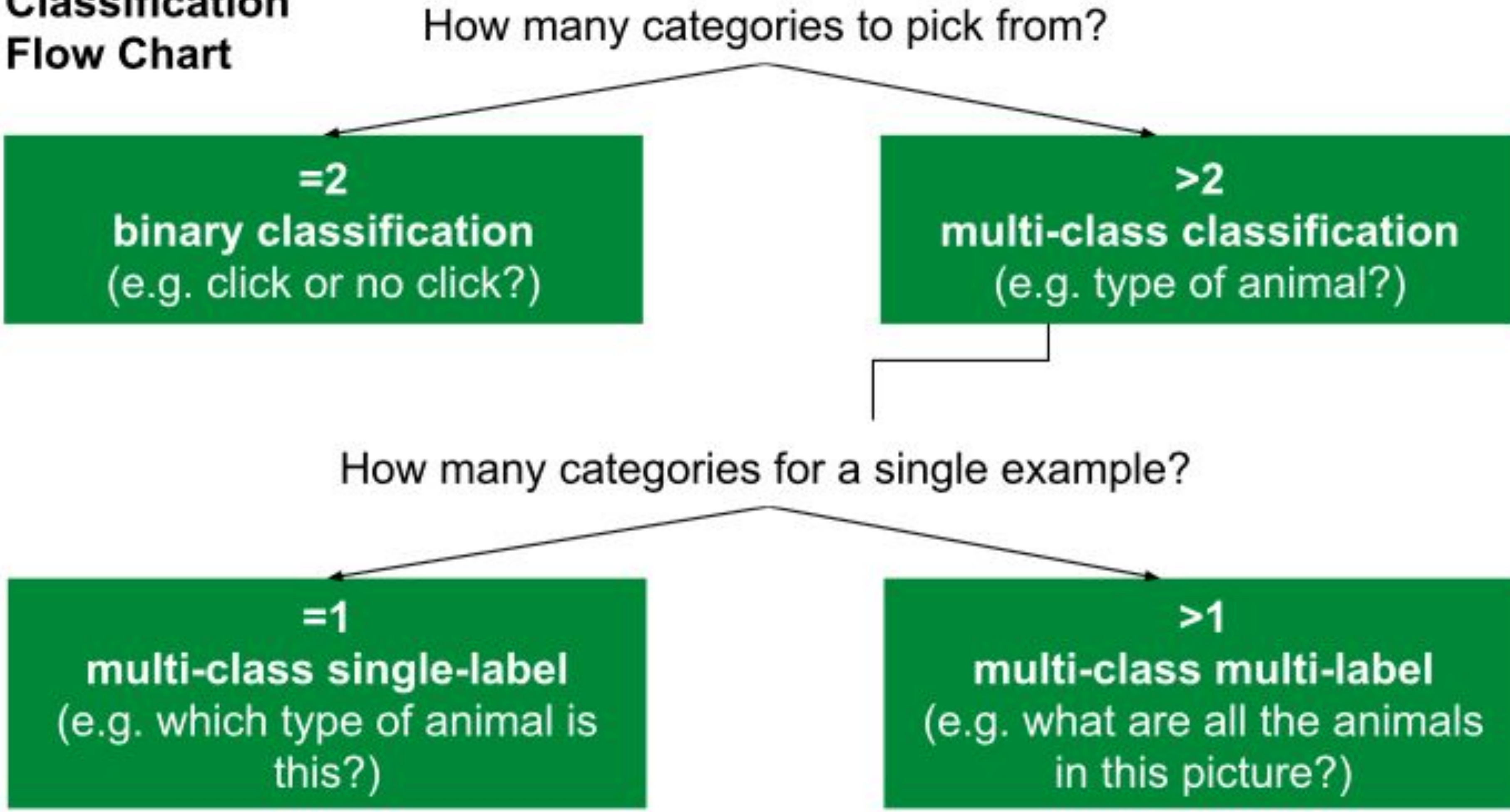
Start Simple

- Can you simplify your problem?
  - Simplify your model
    - Simple model easier to understand, implement, and iterate
    - Full data pipeline for complex models is harder than iterating on simpler models
  - Always try a unidimensional regression problem or binary classification first
    - Well defined, low ambiguity, high tooling
  - If neither fits, try other model types (from model types slide)
- Once you start with simplest model, iterate to improve
  - Can be your baseline model
  - Can help you decide if a more complex model is even needed

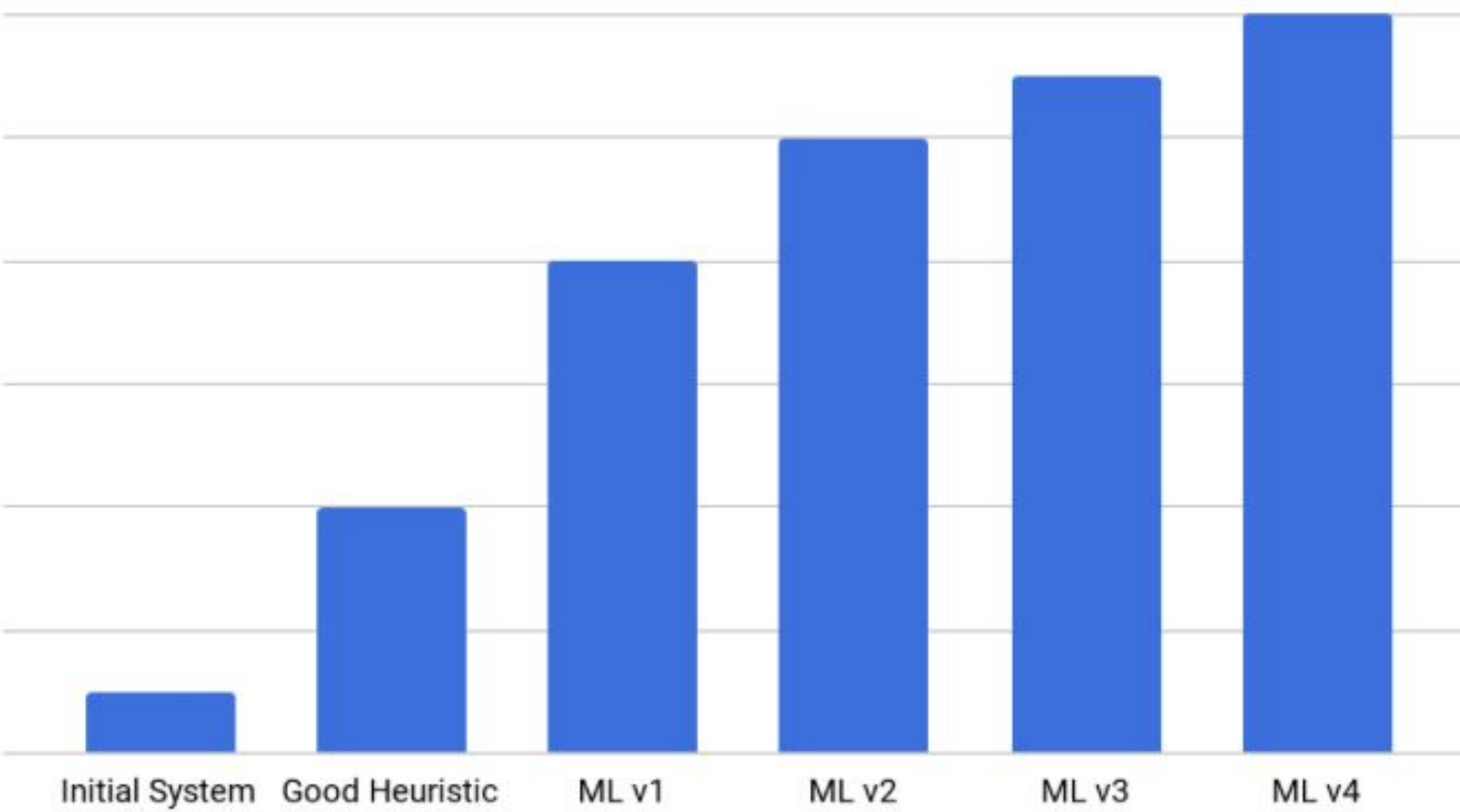
Regression Flow Chart



Classification Flow Chart



Biggest Gain in ML is First Launch



Identify Your Data Sources

- Provide answers to the following questions about your labels:
  - How much labeled data do you have?
  - What is the source of your label?
  - Is your label closely connected to the decision you will be making?

Design your Data for the Model

- Identify the data that your ML system should use to make predictions (input -> output)
- Provide a sample data table. Example:
  - Each row constitutes one piece of data for which one prediction is made.
  - Only include information that is available at the moment the prediction is made.
  - Each input can be a scalar or a 1-dimensional (1D) list of integers, floats, or bytes (including strings).
    - If not 1D list of integers, consider splitting into separate inputs
    - Exceptions: audio, image and video data, where a cell is a blob of bytes.

Determine Where Data Comes From

- Assess how much work it will be to develop a data pipeline to construct.
- Make sure all your inputs are available at prediction time in exactly the format you've written down.
- If it will be difficult to obtain certain feature values at prediction time, omit those features from your model.

Determine Easily Obtained Inputs

- Pick 1-3 inputs that are easy to obtain and that you believe would produce a reasonable, initial outcome.
- Which inputs would be useful for implementing heuristics mentioned previously?
- Consider the engineering cost to develop a data pipeline to prepare the inputs, and the expected benefit of having each input in the model.
- Focus on inputs that can be obtained from a single system with a simple pipeline. Start with the minimum possible infrastructure.

Ability to Learn

- Will the ML model be able to learn?
- List aspects of your problem that might cause difficulty learning.
- For example:
  - The data set doesn't contain enough positive labels.
  - The training data doesn't contain enough examples.
  - The labels are too noisy.
  - The system memorizes the training data, but has difficulty generalizing to new cases.

Think About Potential Bias

- Many datasets are biased in some way. These biases may adversely affect training and the predictions made.
- For example:
  - A biased data source may not translate across multiple contexts.
  - The training sets may not be representative of the ultimate users of the models and may therefore provide them with a negative experience.
  - Watch out for “popular”
    - May reinforce unfair or unbiased views
- *Any examples you know of that had bad bias consequences?*
  - Your model results are only as good as your assumptions, data, model, computing.

In 2016, Microsoft's Racist Chatbot Revealed the Dangers of Online Conversation › The bot learned language from people on Twitter—but it also learned values

BY OSCAR SCHWARTZ | 25 NOV 2019 | 4 MIN READ

Data Exploration

Data Storage Needs

- Data set size will dictate storage and processing choices
- Laptop Hard drive
- Server storage
- File bucket
- Database

Data Size Considerations

- # atoms in the universe?
  - ~ 10^80 (range is 10^78 – 10^82)
- # of stars in the observable universe?
  - ~ 7x10^22 (2003 estimate)
  - Equal to number of H_2O molecules in 10 drops of water
- # of grains of sand on Earth
  - ~ 7.5x10^18
- # of neurons in the brain?
  - **~ 10^11 (100 billion) neurons***
  - Number of synapses (10x # of neurons): ~ 10^12 (trillions)
  - Number of support (glial) cells: 10^12
  - Consumes 20% of body energy but only 2% of body mass
    - The energy demand during brain development is even more striking; it has been estimated that the newborn human brain, which represents about 13% of lean body weight, is consuming around 60% of the body's daily requirement

* Generally accepted numbers, but newer research suggest otherwise

The Human Brain in Numbers: A linearly scaled up primate brain

Table 2 | Expected values for a generic rodent and primate brains of 1.5 kg, and values observed for the human brain (Azevedo et al., 2009).

	Generic rodent brain	Generic primate brain	Human brain
Brain mass	1500 g	1500 g	1508 g
Total number of neurons in brain	12 billion	93 billion	86 billion
Total number of non-neurons in brain	46 billion	112 billion	85 billion
Mass, cerebral cortex	1154 g	1412 g	1233 g
Neurons, cerebral cortex	2 billion	25 billion	16 billion
Relative size of the cerebral cortex	77% of brain mass	94% of brain mass	82% of brain mass
Relative number of neurons in cerebral cortex	17% of brain neurons	27% of brain neurons	19% of brain neurons
Mass, cerebellum	133 g	121 g	154 g
Neurons, cerebellum	10 billion	61 billion	69 billion
Relative size of the cerebellum	9% of brain mass	8% of brain mass	10% of brain mass

Notice that although the expected mass of the cerebral cortex and cerebellum are similar for these hypothetical brains, the numbers of neurons that they contain are remarkably different. The human brain thus exhibits seven times more neurons than expected for a rodent brain of its size, but 92% of what would be expected of a hypothetical primate brain of the same size. Expected values were calculated based on the power laws relating structure size and number of neurons (irrespective of body size) that apply to average species values for rodents (Herculano-Houzel et al., 2006) and primate brains (Herculano-Houzel et al., 2007), excluding the olfactory bulb.

Suzana Herculano-Houzel
Front Hum Neurosci. 2009; 3: 31. Prepublished online 2009 Aug 5. Published online 2009 Nov 9. doi: 10.3389/neuro.09.031.2009

What data size problems can we handle currently?

- What do we mean by “Big Data”?
  - Any problem whose data size cannot be handled in a single computer.
- Current limits
  - Single SSD hard drive: 20TB (= 1.6x10^14 bits)
  - Workstation: HP Z8 PC: 3TB RAM, 48TB storage
  - Server:
    - AWS: u-24tb1.metal, 448 CPUs, 24TB RAM (1.92x10^14 bits)
    - GCP: M2 VM, 12TB RAM

Data Exploration, why do it?

- Test the waters before jumping in
  - Get a sense of what you are getting into
  - Data Science
    - Data cleansing/prep can take more than half your time!

Jupyter Notebooks - What are they?

- Computational notebook
- Multi-language support
- Open source
  - https://jupyter.org
- Hosted
  - Amazon AWS
  - Google Colab
  - Others
- For our class
  - Anaconda version

Jupyter Notebooks – why use them?

- Good for data exploration
- Good for presentations
  - Model evaluation
  - Model results
- Not for “production” purposes

Jupyter notebooks – why use them?

- Able to provide context alongside code
- Can create visualizations
- Can work with remote data sets
- Less daunting than other IDEs (?)

Running in production (Linux)

- You don’t run a Jupyter notebook in production. You just run the code.
- Interpreted vs compiled languages
- Creating the script
  - vim myprog.py
- Passing the file to interpreter via command line argument
  - type python # to see path
  - python myprog.py
- Or can specify the interpreter (from type command)
  - #!/usr/bin/python in first line of program file
- Making the file executable
  - chmod 700 myprog.py
- Running the file
  - ./myprog.py or myprog.py if $PATH=$PATH:.

rsilva — -bash — 72×16

~ — -bash

```
(base) Renzos-MacBook-Air:~ rsilva$ type python
python is /Users/rsilva/opt/anaconda3/bin/python
(base) Renzos-MacBook-Air:~ rsilva$ vim myprog.py
(base) Renzos-MacBook-Air:~ rsilva$ cat myprog.py
#!/Users/rsilva/opt/anaconda3/bin/python

print('Hello ML class!')
(base) Renzos-MacBook-Air:~ rsilva$ python myprog.py
Hello ML class!
(base) Renzos-MacBook-Air:~ rsilva$ chmod 700 myprog.py
(base) Renzos-MacBook-Air:~ rsilva$ ./myprog.py
Hello ML class!
(base) Renzos-MacBook-Air:~ rsilva$
```

WarGames Movie [1983]: Turn Your Key, Sir!

- Premise:
  - AI for controlling missile launches to replace manual decision making
  - Manual solution requires two humans to coordinate actions
- How would you frame this ML problem?
- What data would you need?
- What potential issues do you see? Biases?
- Are decisions easily reversible?
  - Search “amazon two way door” principle
- If you see the movie, how would you categorize the AI solution?
- **Would you recommend an AI solution for this problem?**
- **How would you frame this ML problem?**

WarGames [1983]: Turn Your Key, Sir!

Frame this Problem as an ML Problem

- Articulate your problem.
  - What would you like your ML model to do?
  - Start simple.
  - Ideal outcome?
  - Success and failure metrics?
- Data Sources.
  - Determine where data comes from.
  - Determine easily obtained inputs.
  - Think About Potential Bias.
- What Output Would You Like the ML Model to Produce?
- How might you solve your problem without ML?
  - Why is an ML solution the best solution?