Data Science for Cybersecurity: Hands-on Labs for Basic Data Science Skills

Dr. Thomas P. Scanlon

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

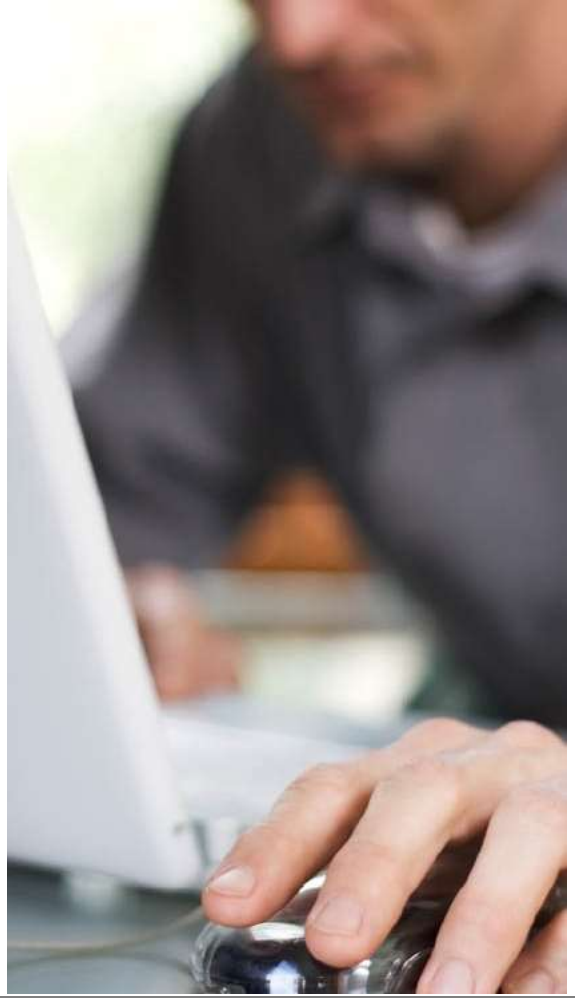
DM22-0839

Lab Workshop Objectives

What should I be able to take home from today's tutorial?

- What **digital forensics** and **data science** are
- Techniques to approach a data-driven problem
  - Exploring data
  - Developing and testing hypotheses
  - Checklist for data-do's and don'ts
- Practical applications
- Familiarity with Jupyter notebooks
- Understanding applying data science to cybersecurity

Jupyter Notebooks and Python



Why bother with a new tool?

Researchers have worked very hard recently to bridge the gap between **decision makers** and **analysts**

Tools like jupyter notebooks, markdown, Quarto, Rmarkdown, etc. provide

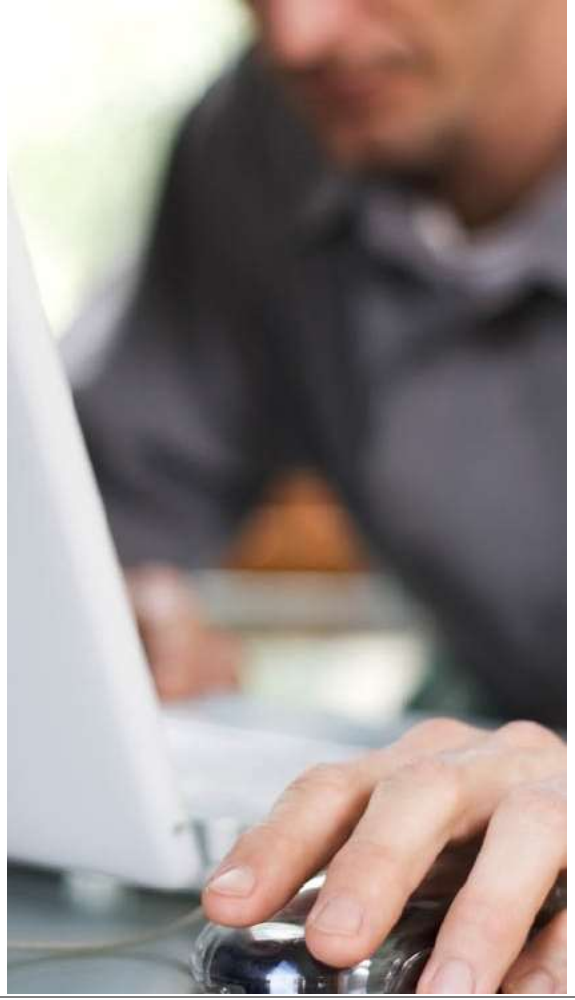
- Reproducibility for analyses
- Accessibility to both high and low level detail
- Responsiveness and flexibility to dynamic needs
- Portability (slides, html, word doc, etc.)

Takeaway: The analysis is the document

LAB 1

Getting started with Python and Jupyter Notebooks

Data Science for Digital Forensics



Can You Spot the Fake?

This Person Does Not Exist…

https://thispersondoesnotexist.com/

https://thisxdoesnotexist.com/

Schedule

4x1.5 hour sessions (30-60 minute instruction / 30-60 minute lab)

Part I: Introduction

Lab 1: Getting familiar with Jupyter Notebooks and Python

Part II: Exploratory Data Analysis

Lab 2: Reading, cleaning, and visualizing data

Part III: Hypothesis Testing

Lab 3: Fit a model from analyst perspective; examine from decision-maker perspective

Part IV: Mis-, Dis-, and Mal- Information

Lab 4: MDM lab

Digital Forensics

Digital Forensics Continued

Executive Summary

Every interaction with a digital device has the potential to leave a trail of what we did, who we did it with, where we were, and when the event took place. This trail is made up of digital artifacts, which are created in the routine operation of a digital device. This trail can assist an investigator to discover and explain what happened. Computers generate many artifacts, most of which do not contribute to understanding what happened. The challenge is finding useful information and separating it from irrelevant information. Digital investigation techniques can extract this information and construct a narrative of the events. The analysis of digital devices for investigative purposes is widely practiced and, as this report shows, there are at least 11,000 digital forensic laboratories in the United States.

From NISTIR 8354-DRAFT Digital Investigation Techniques: A NIST Scientific Foundation Review

Digital Forensics Continued x2

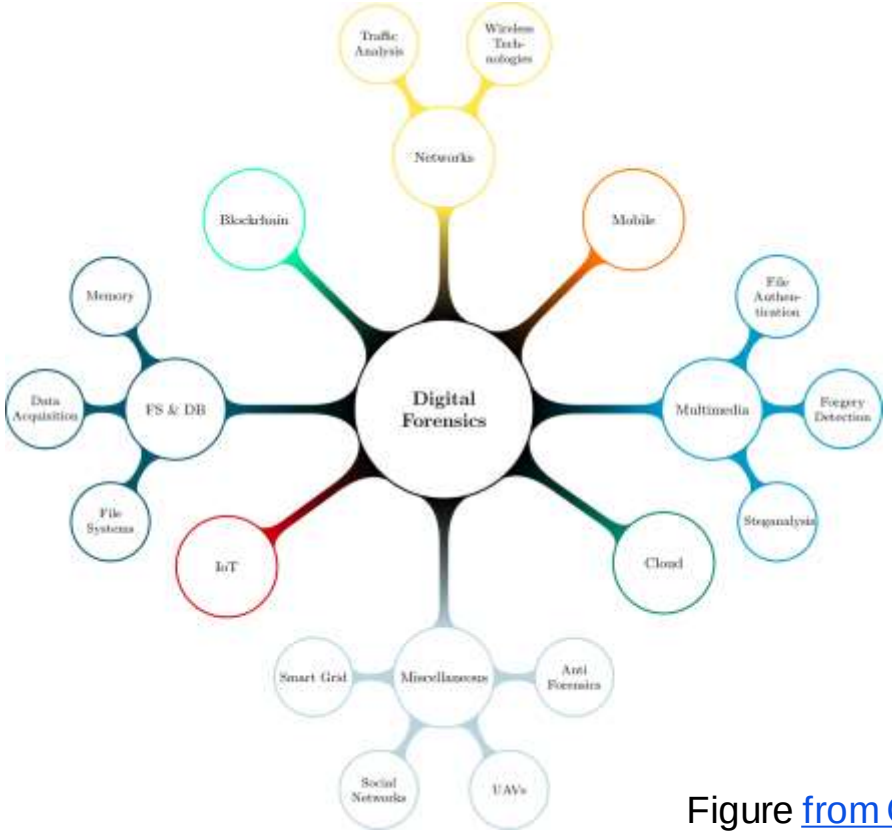
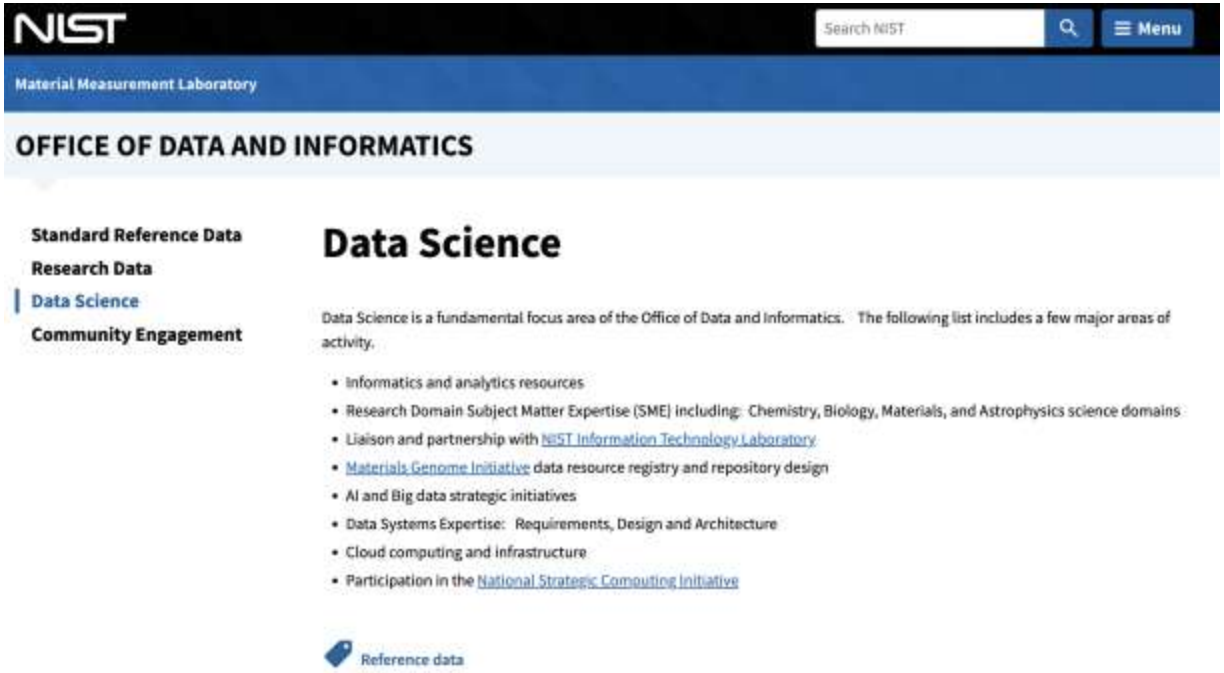
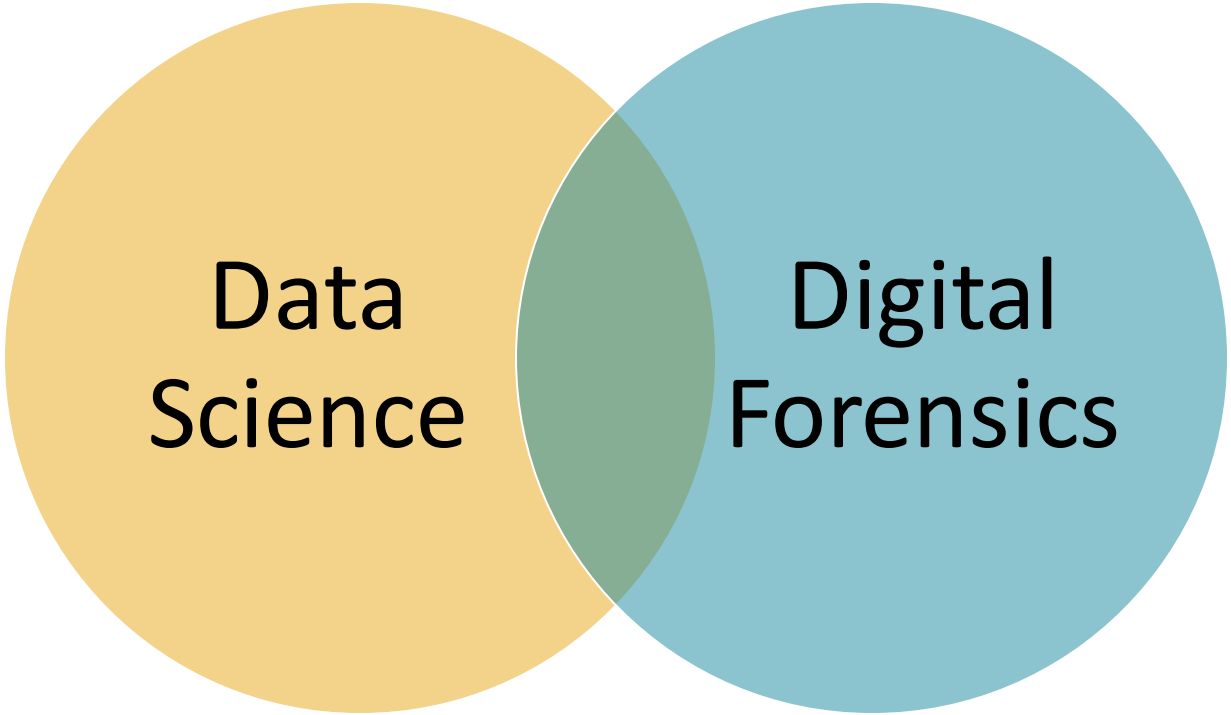


Figure [from Casino et al. (2022)]

Data Science



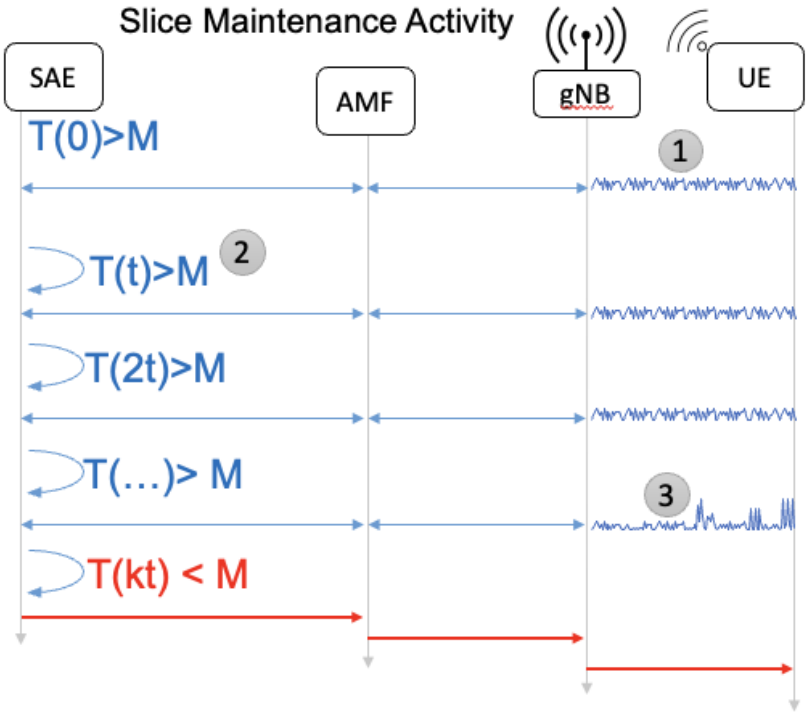
What's the difference?



Example 1: Deepfakes

Is this a real image? (Answer: no)

Example 2: Trust (of digital devices)



20 billion devices are expected to be connected in the next few years

- Which ones can we trust?

Example 3: Malware analysis

- Malware Identification
- Spread of viruses
- Natural language processing
- Cyber attacks

Data Science Do's and Don't's

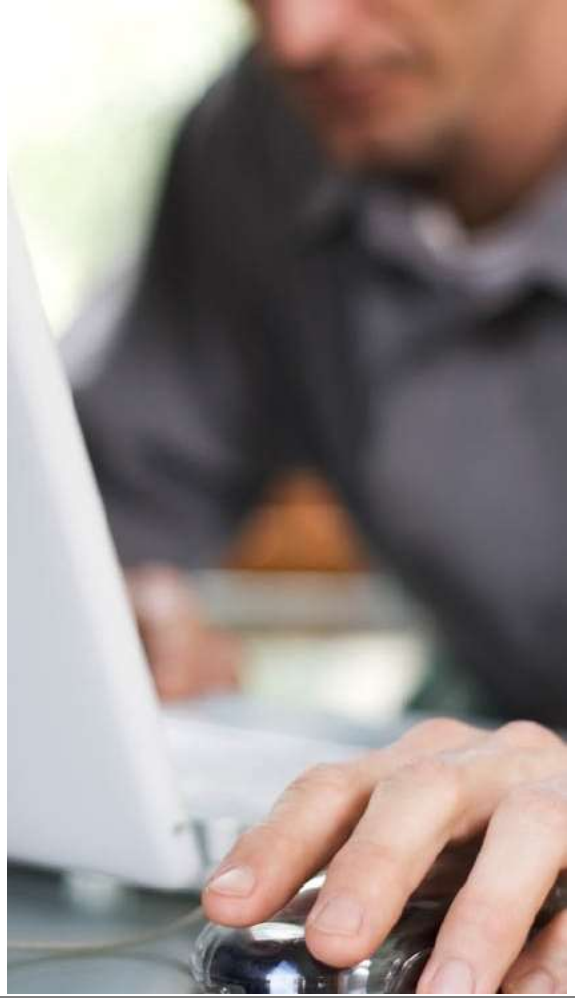
Do

- Have a question in mind
- Utilize your subject experts
- Think of what data you **need** vs. **have**
- Interrogate your data
- Document all collection, cleaning, and transformation steps
- Justify models used
- Interrogate your model(s)
- Be ready for 'negative' results

Don't

- Force your data to fit your hypothesis
- Forsake model interpretability to do a 'machine learning / AI model'
- Overfit
- Overinterpret

Exploratory Data Analysis



What are data?

From NIST:

Definition(s):

Distinct pieces of digital information that have been formatted in a specific way.

Source(s):

NIST SP 800-86 under *Data*

Pieces of information from which "understandable information" is derived.

Source(s):

NIST SP 800-88 Rev. 1 under *Data*

A subset of information in an electronic format that allows it to be retrieved or transmitted.

Source(s):

NIST SP 1800-10B under Data from CNSSI 4009-2015

NIST SP 1800-25B under Data

NIST SP 1800-26B under Data from CNSSI 4009-2015

A representation of information as stored or transmitted.

Source(s):

NISTIR 4734 under *Data*

A representation of information, including digital and non-digital formats.

Source(s):

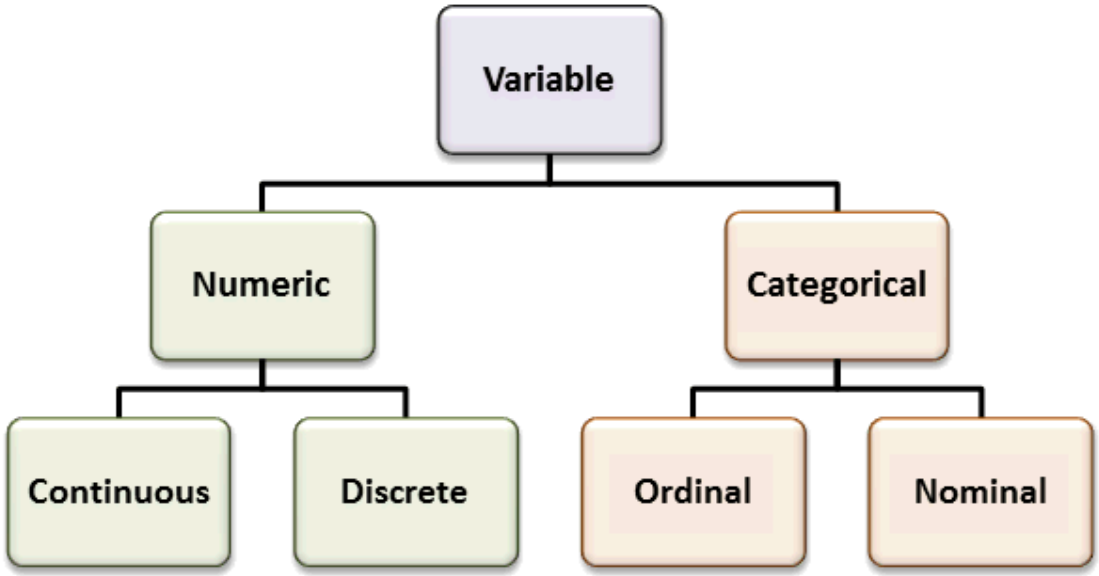
NIST Privacy Framework Version 1.0 under *Data*

Information in a specific representation, usually as a sequence of symbols that have meaning.

Source(s):

CNSSI 4009-2015 [Superseded] from IETF RFC 4949 Ver 2

Types of data



Data Do's and Don't's

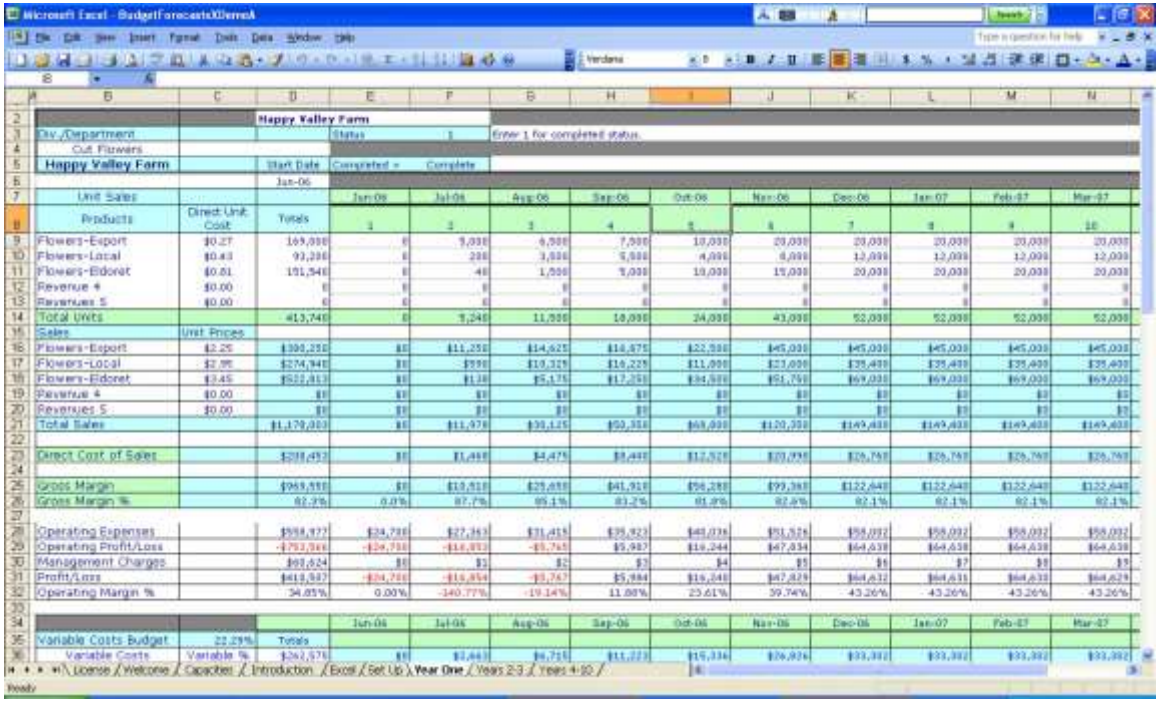
Do

- Have a question in mind
- Utilize your subject experts
- Think of what data you **need** vs. **have**
- **Interrogate your data**
- Document all collection, cleaning, and transformation steps
- Justify models used
- Interrogate your model(s)
- Be ready for 'negative' results

Don't

- Force your data to fit your hypothesis
- Look at your data, make a hypothesis, and then test that hypothesis on the same data
- Forsake model interpretability to do a 'machine learning / AI model'
- Overfit
- Overinterpret

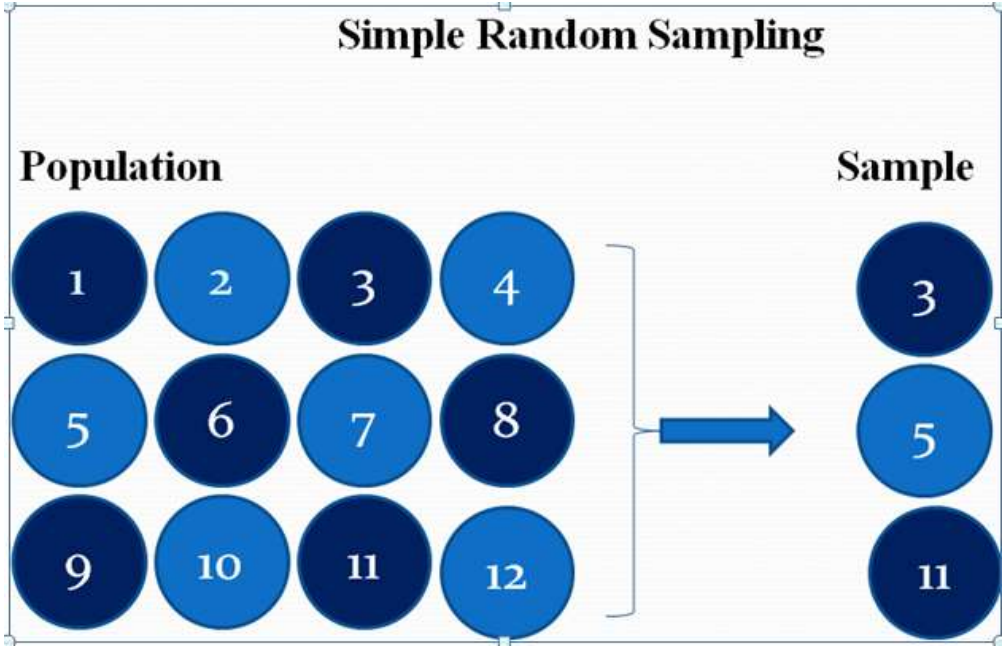
What are some reasons we need to interrogate our data?



EDA checklist

1. Check integrity of data
2. Count size of data and size of subgroups
3. Examine features one-by-one
4. Examine relationships between features
5. Transform features

Interrogate your data: how was it sampled/collected?



1. Check integrity of data

- Are data **representative** of a population in which we are interested?
- Are we interested in making **predictions** about future observations, or are we interested in making **inferences** about why things happen?

Interrogate your data: provenance

1. **Check integrity of data**

   Source?

   Version control?

Interrogate your data: shape

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston Uniersity	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

X = our feature matrix, e.g. ”data frame”

Here, an **observation** is one row

1. **Check integrity of data**

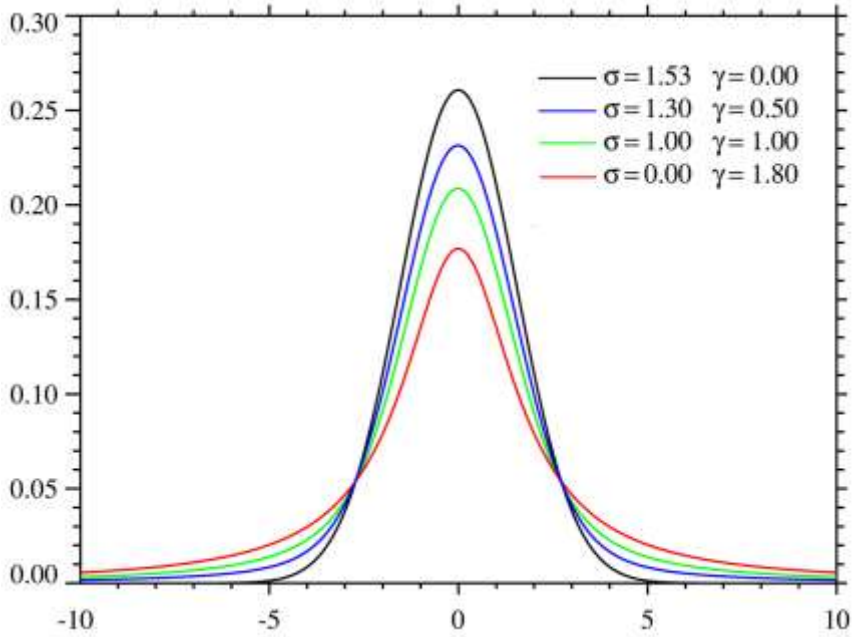
	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston Uniersity	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

Interrogate your data: sample size (n)

2. Count size of data

- Want n to be ‘large’
  - Perhaps you remember the rule of thumb for n=30 for a continuous variable to be have approximately a normal distribution
- The sample size will (usually) limit what type of models we can use to analyze the data
- Usually* want n >> p
- Want to split the data into training and testing sets

*There is a large set of problems where this does not happen and many ways to handle this



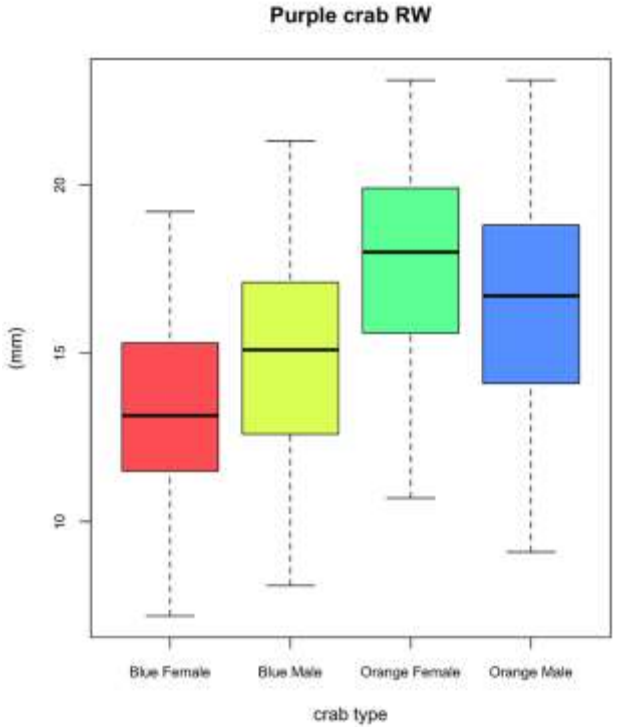
Interrogate your data: type

3. Examine features one-by-one

- **Numeric**
  - Discrete (e.g. number of characters)
  - Continuous ($ spent)
- **Categorical**
  - Ordered (e.g. small-medium-large)
  - Nominal (e.g. red-yellow-blue)

Examine features 1-by-1

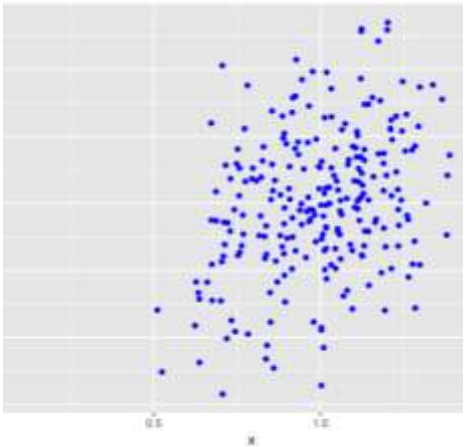
- Determine type of feature
  - Categorical, ordered, numeric
- Check range and distribution of responses
  - There are nearly always ‘mistakes’ in data entry
- Determine # of missing values
- Note which features your experts think are useful



We tend to believe that some feature **Y** is dependent on some other features **X, e.g.**

- **Higher temperature -> more ice cream sales**
- **Spread of virus is associated with computers with less security (e.g. firewalls, updates, encryption capabilities)**
- **Image with non-matching earrings -> more likely to be a fake photo**

4. Examine relationships between features



Transform features

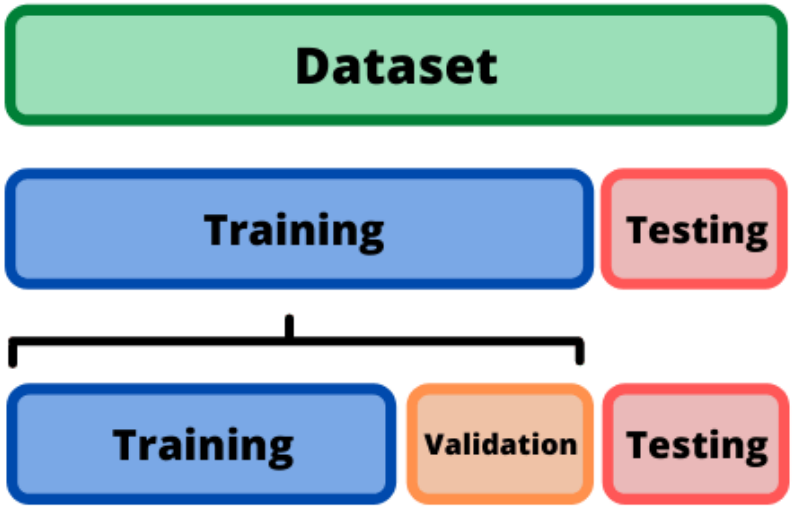
5. Transform features

- Some features work better on a log scale
- Merge categories together
- “Normalize” features

Split data into training/test/validation sets

- Defense against bias
- Sacrifice some ‘power’
- Depends on your ‘n’
- 50/50 splits often common
- Sometimes 70/30, 80/20, 90/10

4. If possible, randomly split data into train/test/validation sets



LAB 2

Exploratory Data Analysis

Hypothesis Testing

Data Do's and Don't's

Do

- Have a question in mind
- Utilize your subject experts
- Think of what data you **need** vs. **have**
- Interrogate your data
- Document all collection, cleaning, and transformation steps
- Justify models used
- Interrogate your model(s)
- Be ready for 'negative' results

Don't

- Force your data to fit your hypothesis
- Look at your data, make a hypothesis, and then test that hypothesis on the same data
- Forsake model interpretability to do a 'machine learning / AI model'
- Overfit
- Overinterpret

Hypothesis Testing

1. Have a question you want to answer (a hypothesis)
2. Randomly split data into train/validation/test sets
3. Develop model(s) to test hypothesis
4. Fit model(s) on training data
5. Tune model(s) on validation data
6. Fit final model(s) on testing data
7. Interpret results

1. Have a question you want to answer (a hypothesis)

Context: SEI held a coding course where the participants had to write code to and complete 10 modules. Data collected in this class include the individual IDs, minutes spent on each module, number of lines of code written, years of programming experience, and whether the individual completed the course or not.

My hypothesis: I think that the more years of coding experience someone has, the less time s/he spends on a coding practice module, after adjusting for the lines of code written, and whether s/he completed the course.

2. Randomly split data into train/validation/test sets

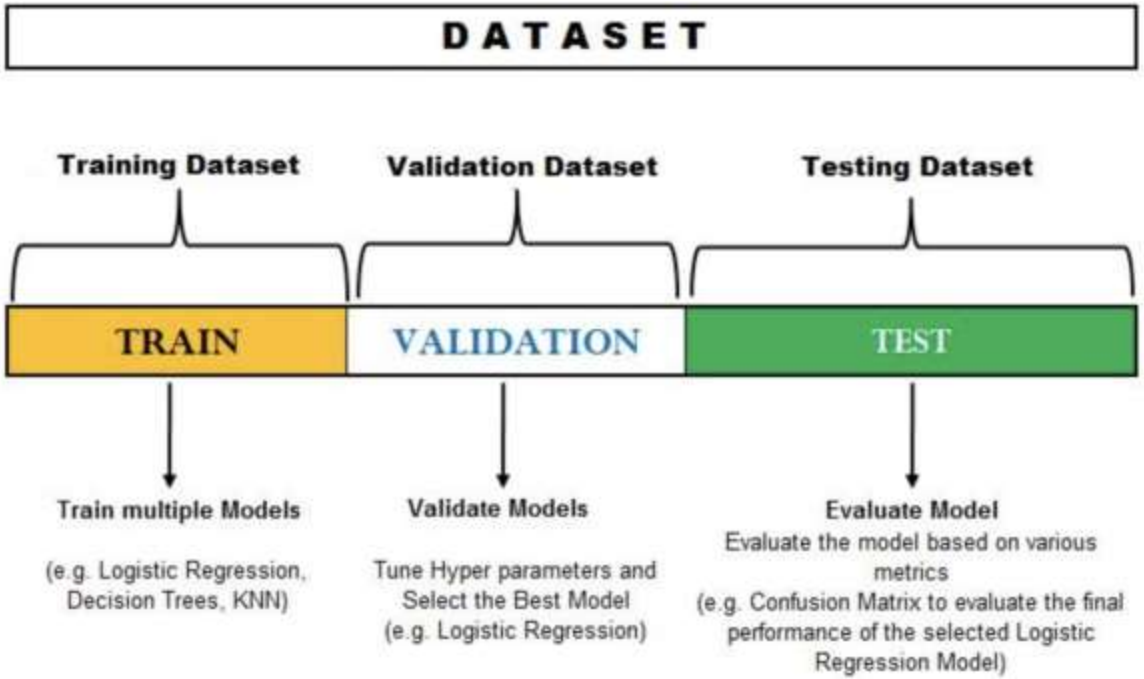


Figure from [this link]

Why do we split our data?

- Usually we want to fit more than one model
- But training a model and then testing it on the same data set leads to bias (e.g. ‘double dipping’)
- Randomly splitting our data eliminates this bias
  - Does sacrifice some ‘statistical power’
  - If we have a large number of observations (n), we’re usually fine with that tradeoff

3. Develop model(s) to test hypothesis

Y = output we care about (e.g. # of minutes taken in course)
X = input data (e.g. lines of code, years of experience, course completed)
ϵ = noise term
f = function of the data

$$Y = f(X) + \epsilon$$

There are **so many** models to choose from, e.g. linear regression, logistic regression, decision trees, neural nets, etc.
We want ours to **make sense** and **answer the question**

Hypothesis in words to hypothesis in math

Expected # of minutes is a linear function of years of coding experience, lines of code of written in the modules, and whether the participant completed the course

$$E[\#\,\boldsymbol{minutes}] = \beta_0 + \beta_1 X_{years} + \beta_2 X_{LOC} + \beta_3 X_{comp}$$

My hypothesis: more years means fewer minutes

$$H_0: \beta_1 \geq 0$$
$$H_A: \beta_1 < 0$$

4-6. Fit/Tune/Fit

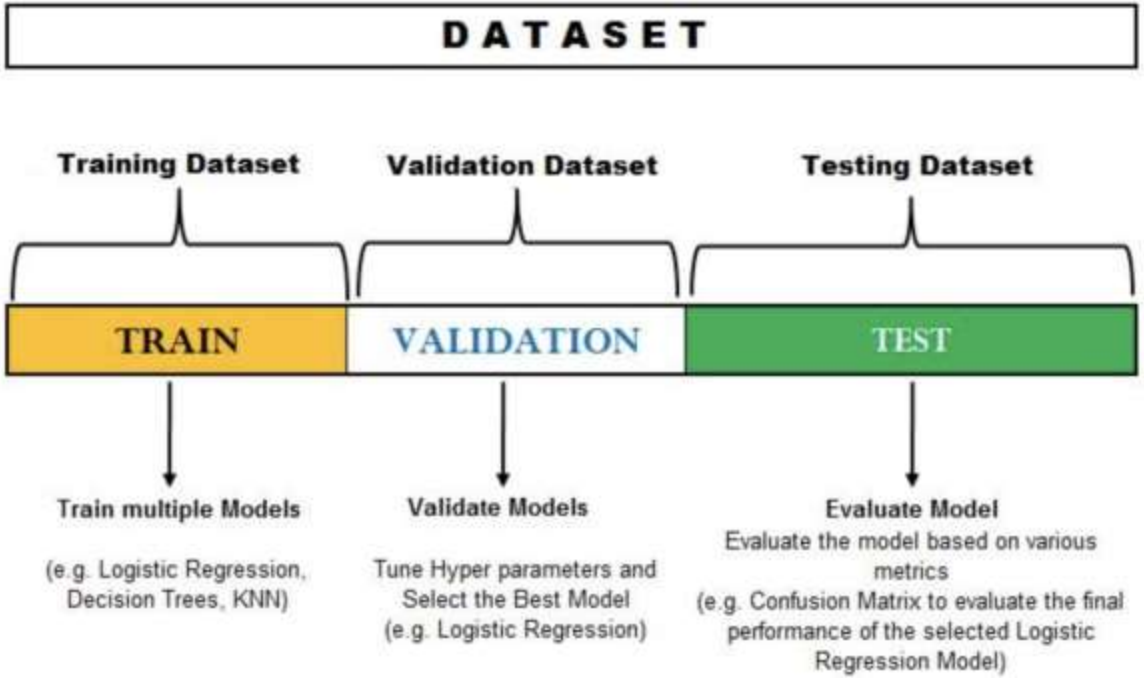


Figure from [this link]

7. Interpret results in context

Caution!

- Were model assumptions met?
- Were all data used in analysis?
- Were continuous variables transformed into discrete variables?
- Was sensitivity analysis performed?
- Do we understand limitations of model?

If yes, interpret with care.

p-values. They're tricky

1 "Don't" Is Not Enough

There's not much we can say here about the perils of *p*-values and significance testing that hasn't been said already for decades (Ziliak and McCloskey 2008; Hubbard 2016). If you're just arriving to the debate, here's a sampling of what not to do:

- Don't base your conclusions solely on whether an association or effect was found to be "statistically significant" (i.e., the *p*-value passed some arbitrary threshold such as $p < 0.05$).
- Don't believe that an association or effect exists just because it was statistically significant.
- Don't believe that an association or effect is absent just because it was not statistically significant.
- Don't believe that your *p*-value gives the probability that chance alone produced the observed association or effect or the probability that your test hypothesis is true.
- Don't conclude anything about scientific or practical importance based on statistical significance (or lack thereof).

Don't. Don't. Just...don't. Yes, we talk a lot about don'ts. The *ASA Statement on p-Values and Statistical*

From Wasserstein, Schirm, and Lazar (2018)

p-values. Still tricky. (From Wasserstein, Schirm, and Lazar (2018))

Accept Uncertainty

Thoughtful

Open

Modest

Data Do's and Don't's

Do

- Have a question in mind
- Utilize your subject experts
- Think of what data you **need** vs. **have**
- Interrogate your data
- Document all collection, cleaning, and transformation steps
- Justify models used
- Interrogate your model(s)
- Be ready for 'negative' results

Don't

- Force your data to fit your hypothesis
- Look at your data, make a hypothesis, and then test that hypothesis on the same data
- Forsake model interpretability to do a 'machine learning / AI model'
- Overfit
- Overinterpret

LAB 3

Hypothesis Testing

Mis-, Dis-, and Mal- Information (MDM)

What is MDM?

DHS CISA defines MDM as information activities intended to cause chaos, confusion, and division.

Mis-, **D**is-, **M**al-information

- Misinformation: false information that is shared without intent to harm
- Disinformation: false information deliberately created to mislead of cause harm
- Mal-information: information based on truths but purposefully used out of context to mislead or cause harm

MDM Examples

Mis-, **D**is-, **M**al-information

- Misinformation: Betsy Ross sewed the first American flag
- Disinformation: Operation INFEKTION
- Mal-information: 80% of dentists recommend Colgate

Disinformation and Mal-information are often shared as misinformation

What Is a Deepfake?

- Deepfake = ‘deep-learning’ + ‘fake.’
- ‘deepfake’ originates from a Reddit user, who, in 2017, claimed to have created the method.
- A deepfake can be audio, video, an image, or multimodal.
- It is not the same as using Photoshop.
- Deepfakes are considered disinformation.
  - Or they are combined with disinformation (e.g., profile with deepfake images).

A deepfake is a media file, typically videos, images, or speech representing a human subject, that has been modified deceptively using deep neural networks to alter a person’s identity. Advances in machine learning have accelerated the availability and sophistication of tools for making deepfake content. As deepfake creation increases, so too do the risks to privacy and security.

Deepfake Creation

Main Deepfake Types

- Face Swap
- Lip syncing
- Puppeteering
- Synthetic

Common Deepfake Techniques

- Auto-encoder
- GAN

Deepfake Creation Process

- Extraction
  - -Data collection (source data)
- Training
- Conversion / Generation

Deepfake Creation Process - Extraction

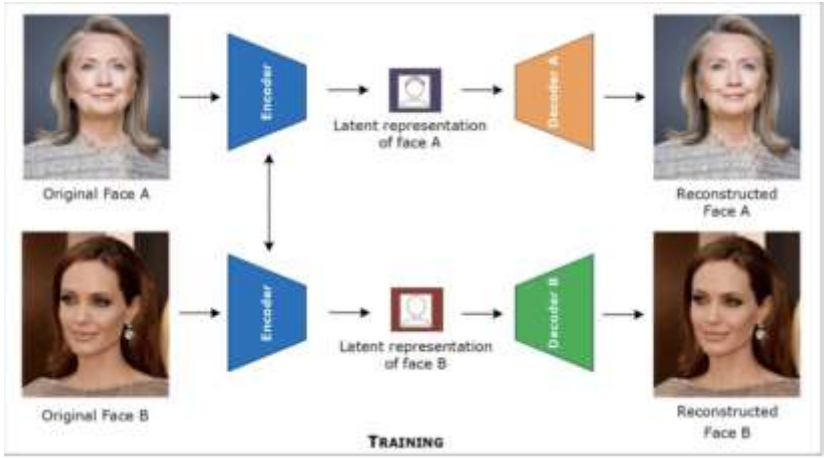
- As a practical matter, need to consider what data sources will provide this data
- A lot of training data is needed
- For images, thousands of images may be necessary
- Just a few video clips can replace thousands of images
- Extraction is the process of extracting individual frames from from video source, identifying faces and aligning them.
- Will need images of source (subject we want to embed) and destination (subject we want to override)

This is different from feature extraction

Deepfake Creation Process - Extraction

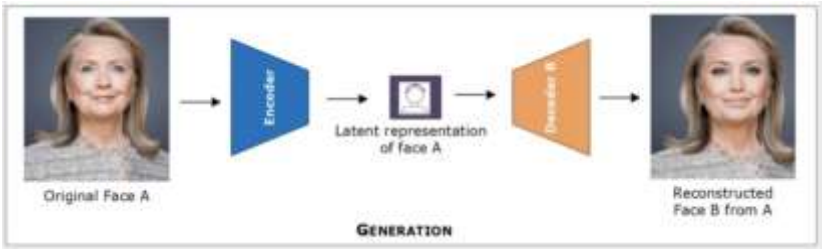
*resizing, normalization, augmentation, etc.

Deepfake Creation Process - Training



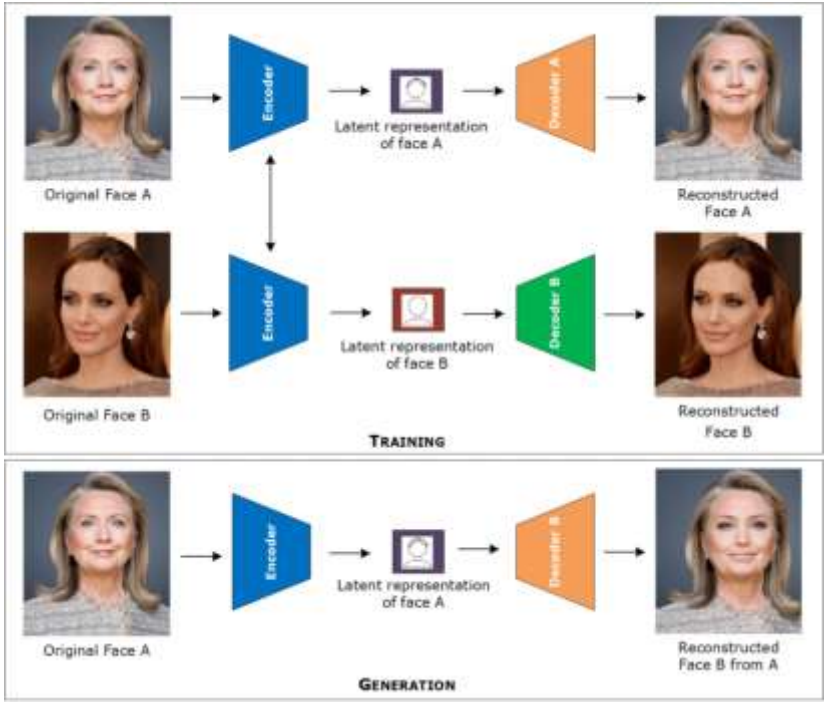
Masood, Momina & Nawaz, Marriam & Malik, Khalid & Javed, Ali & Irtaza, Aun. (2021). Deepfakes Generation and Detection: State-of-the-art, open challenges, countermeasures, and way forward.

Deepfake Creation Process - Generating



Masood, Momina & Nawaz, Marriam & Malik, Khalid & Javed, Ali & Irtaza, Aun. (2021). Deepfakes Generation and Detection: State-of-the-art, open challenges, countermeasures, and way forward.

Deepfake Creation Process – Auto-encoder



Masood, Momina & Nawaz, Marriam & Malik, Khalid & Javed, Ali & Irtaza, Aun. (2021). Deepfakes Generation and Detection: State-of-the-art, open challenges, countermeasures, and way forward.

Deepfake Detection

- Deepfake detectors *discriminate* between real and deepfake images
  - Detector = discriminator
- To develop models that automatically discriminate, we can:
  - **Humans** feed the model with features associated with differences
  - **Computers** 'learn' useful features on their own
  - Combine the above two approaches

Humans determine features

What *describable* features make deepfakes different?

- 'Obvious' errors (e.g. two heads)
- Facial boundaries blurring into background
- Asymmetries (e.g. earrings not matching)
- Inconsistent light sources
- Odd color frequencies
- Irregular 'heartbeats'
- Discontinuity between frames
- Lack of variation

Obvious features: conjoined heads

Image from
thispersondoesnotexist.com

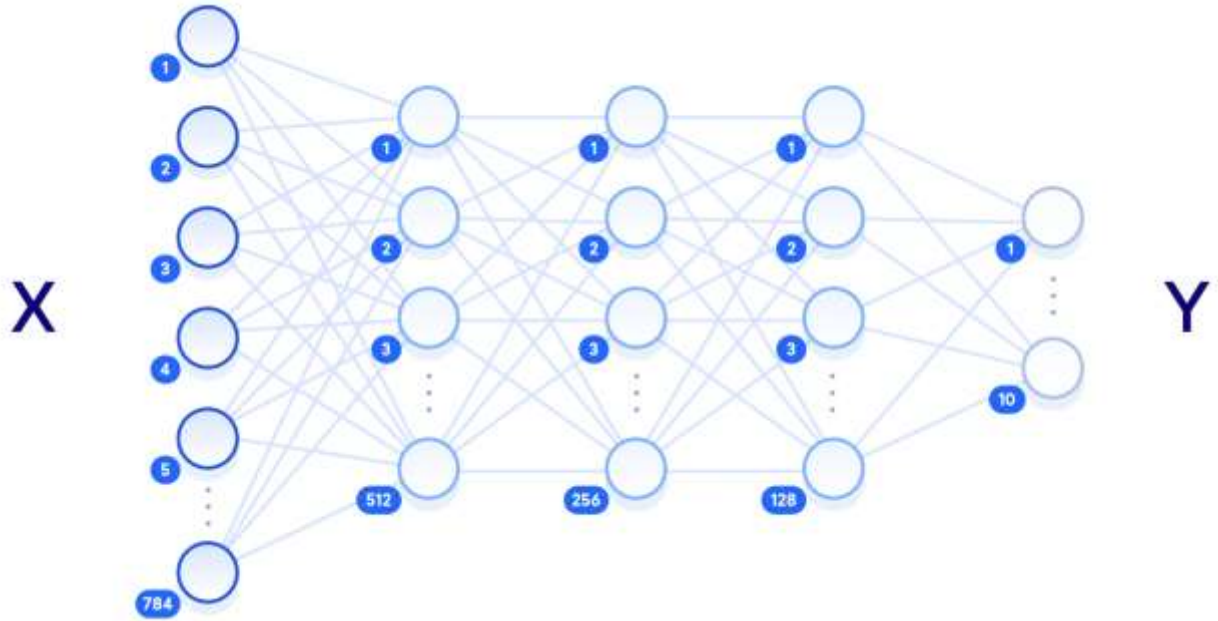
Advantages of feeding the model ourselves

- Fortunately, photo forensics and image analysis have been active fields long before deepfakes came along
  - Hany Farid: physical objects follow physical laws
    - Light, shadow, weight, specularities, lenses, etc.
- Computer vision –huge strides in detecting human facial features
- Explainability
- Generalizability

Disadvantages of feeding the model ourselves

- Intuition can be difficult to transform into digital representation
  - 'affine transformation' problems: scaling, rotating, sliding, mirroring
  - Unclear boundaries: e.g. what if a hand is covering the face?
- Hard to capture all describable differences in one model

Computers determine features: Neural nets



Advantages of computer extracted features

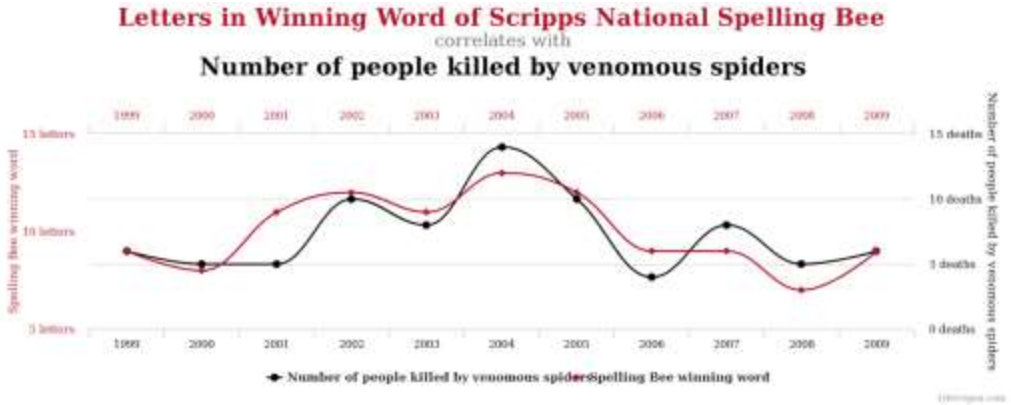
- Work well in practice (e.g. image detection)
- (Relatively) easy to code
- Can find real, *latent* features that are associated with differences that humans cannot easily find

pytorch

tensorflow

Disadvantages of computer extracted features

- Find lots of *spurious* features that *seem* to be associated with differences but are just random noise
- Hard to interpret
- Can be difficult to implement and train
- Generalizability



The generator-detector game

1. Adversary generates fake images
2. We make detector with high accuracy to discriminate real/fake
3. Adversary introduces improved fake images
4. We make improved detector
5. …

Normally, this game takes a huge amount of effort and time by both the adversary and us. Researchers are automating this process.

Is this a never ending game?

Conclusions

- It is difficult, but not impossible to make high-quality deepfakes with enough time, effort, skill, and GPU.
  - Source and destination identities should be as similar as possible
  - Significant post-processing

- Detection is a cat-and-mouse game
  - Human-recognizable vs. computer-recognizable features
  - Potential scale of deepfakes makes automatic detection necessary

LAB 4

Examining Images for Deepfakes

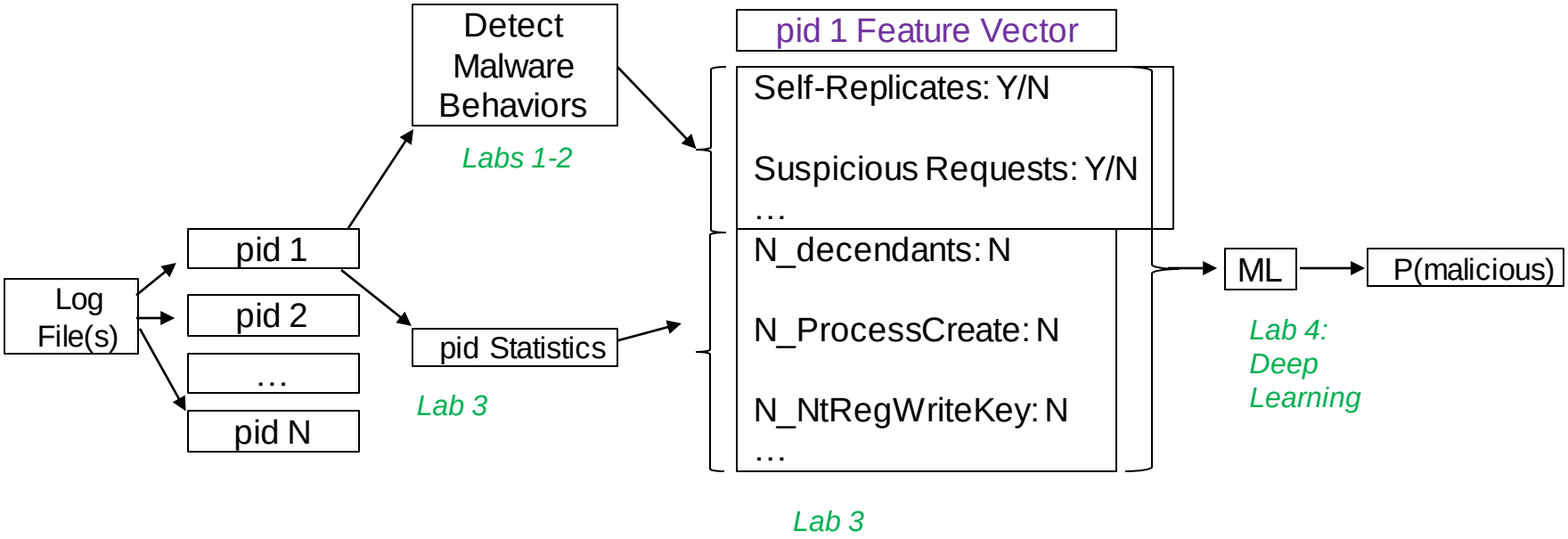
Data Science for Cybersecurity Use Case: Feature Engineering for Malware Identification

Our Data

Malicious / Benign.exe → monitor → Log file

```
Process-memory-write-delete-access-attempt, target pid: 501478248, writer pid: 924, desired access: PROCESS_ALL_ACCESS
Process-memory-write-accessattempt, target pid: 501498728, writer pid: 3284, desired access: PROCESS_VM_WRITE
Process-terminate-attempt, target pid: 501498728, terminator pid: 3284, desired access: PROCESS_TERMINATE
Process-memory-write-delete-access-attempt, target pid: 501498728, writer pid: 3284, desired access: PROCESS_ALL_ACCESS
Process-memory-write-accessattempt, target pid: 501773160, writer pid: 4244, desired access: PROCESS_VM_WRITE
Process-memory-write-accessattempt, target pid: 501797736, writer pid: 3220, desired access: PROCESS_VM_WRITE
Process-terminate-attempt, target pid: 501773160, terminator pid: 4244, desired access: PROCESS_TERMINATE
Process-terminate-attempt, target pid: 501797736, terminator pid: 3220, desired access: PROCESS_TERMINATE
Process-memory-write-delete-access-attempt, target pid: 501773160, writer pid: 4244, desired access: PROCESS_ALL_ACCESS
Process-memory-write-delete-access-attempt, target pid: 501797736, writer pid: 3220, desired access: PROCESS_ALL_ACCESS
Process-memory-write-accessattempt, target pid: 555344744, writer pid: 600, desired access: PROCESS_VM_WRITE
Process-terminate-attempt, target pid: 555344744, terminator pid: 600, desired access: PROCESS_TERMINATE
Process-memory-write-delete-access-attempt, target pid: 555344744, writer pid: 600, desired access: PROCESS_ALL_ACCESS
Process-memory-write-accessattempt, target pid: 501838696, writer pid: 1352, desired access: PROCESS_VM_WRITE
Process-terminate-attempt, target pid: 501838696, terminator pid: 1352, desired access: PROCESS_TERMINATE
Process-memory-write-delete-access-attempt, target pid: 501838696, writer pid: 1352, desired access: PROCESS_ALL_ACCESS
Process-memory-write-accessattempt, target pid: 501912424, writer pid: 5004, desired access: PROCESS_VM_WRITE
Process-terminate-attempt, target pid: 501912424, terminator pid: 5004, desired access: PROCESS_TERMINATE
Process-memory-write-delete-access-attempt, target pid: 501912424, writer pid: 5004, desired access: PROCESS_ALL_ACCESS
Process-memory-write-accessattempt, target pid: 502076264, writer pid: 5132, desired access: PROCESS_VM_WRITE
Process-terminate-attempt, target pid: 502076264, terminator pid: 5132, desired access: PROCESS_TERMINATE
```

Does anything seem suspicious?



Malware Behaviors: A Data Science Perspective

Self-replication

P(malware) ~ small

P(malware | self replication) >> P(malware)

Takeaway: Use subject matter expertise!

Coding a Self-Replication Detector

1. **Create**
2. **Read**
3. **Write**

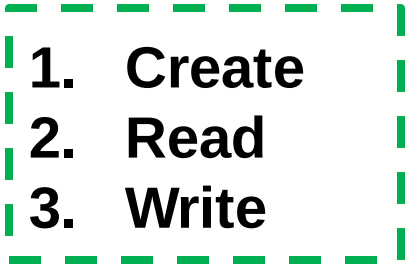
IRP_MJ_READ: The process reads *from* a file *to* a memory destination.

IRP_MJ_WRITE: The process writes *from* a memory address *to* another file.

The key to finding self-replication is to track what's in memory.

Identifying Self-Replication

1. Find process creation index, and store the source file.
2. Find pid IRP_MJ_READ, match the source file, and store the mem address.
3. Find pid IRP_MJ_WRITE, and match the mem write address.



['True-ProcessCreated', {'pid': 3416, 'ppid': 3980, 'source image file 01': '...\\Virus.Win32.Neshta.a.exe', 'process': 'FFFFE00170779780'}]

['IRP_MJ_READ', {'owner pid': 3416, 'destination address': '0x00195d40', 'Source file': '...\\Virus.Win32.Neshta.a.exe'}]

['IRP_MJ_WRITE', {'owner pid': 3416, 'source address': '0x00195d40', 'destination file': '\\Device\\HarddiskVolume1\\Windows\\TEMP\\3582-490\\Virus.Win32.Neshta.a.exe'}]

Parsing Log Files

For Line in File

parsed_log[Line] = [process, {keys: values}]

e.g., [‘Process-memory-write-accessattempt’, {‘target pid’: 12345, ‘writer pid’: 666, ‘desired access’: ‘PROCESS_VM_WRITE’}]

parsed_log[0][0]

>>> ‘Process-memory-write-accessattempt’

list(Parsed_log[0][1].keys())

>>> [‘target pid’, ‘writer pid’, ‘desired access’]

list(parsed_log[0][1].values())

>>> [12345, 666, ‘PROCESS_VM_WRITE’]

Other Needed Functions

get_pid_2_idx(parsed_log):

returns {pid_1: [index_1, index_2, …], pid_2: [index_1, index_2, …], …}

search_for_process(parsed_log, type_of_process, lines=None):

returns [index_1, index_2…]

Function 1: Create

Pseudocode

```
pid_2_idx = get_pid_2_idx(parsed_log)
PID_lines = pid_2_idx[PID] # pid: indices
PID_process_creations = search_for_process(parsed_log, 'True-ProcessCreated', lines=PID_lines)
line_of_creation = None
for line in PID_process_creations:
    if line[1]['pid'] == PID:
        line_of_creation = line
```

Function 2: Self-Reads

Pseudocode

```
# assume we have pid, line_of_creation from previous slide
image_file = line[1]['source image file 01']
read_lines = search_for_process(parsed_log, 'IRP_MJ_READ', lines=PID_lines)
self_reads = []
for line in read_lines:
    if line['Source file'] == image_file:
        self_reads.append(line)
```

Self-Replication

Pseudocode

```
self_replicates=False
for pid in log:
    line_of_creation = find_line_of_creation(log, pid)
    self_reads = find_self_reads(log, pid, line_of_creation)
    self_writes = find_self_writes(log, pid, self_reads)
    if len(self_writes) > 0:
        self_replicates=True
```

Feature Vector So Far…

Feature Description	Value
Self-replicates?	1 or 0

Suspicious Requests

Anything with PROCESS_VM_WRITE or PROCESS_ALL_ACCESS is considered suspicious.

It is particularly suspicious if a target pid and writer pid have no ancestry relationship.

Suspicious Process Requests

The pids match.

['Process-memory-write-access-attempt', {'target pid': 4920, 'writer pid': 4920, 'desired access': 'PROCESS_VM_WRITE'}]

The pids don't match.

['Process-memory-write-delete-access-attempt', {'target pid': 524354408, 'writer pid': 1080, 'desired access': 'PROCESS_ALL_ACCESS'}]

You must check ancestry.

Example Feature Vector

Feature Description	Count
Number of suspicious requests (i.e., has the same pid)	20
Number of suspicious requests (i.e., has different pids; has ancestry)	6
Number of suspicious requests (i.e., has different pids and **no** ancestry)	2

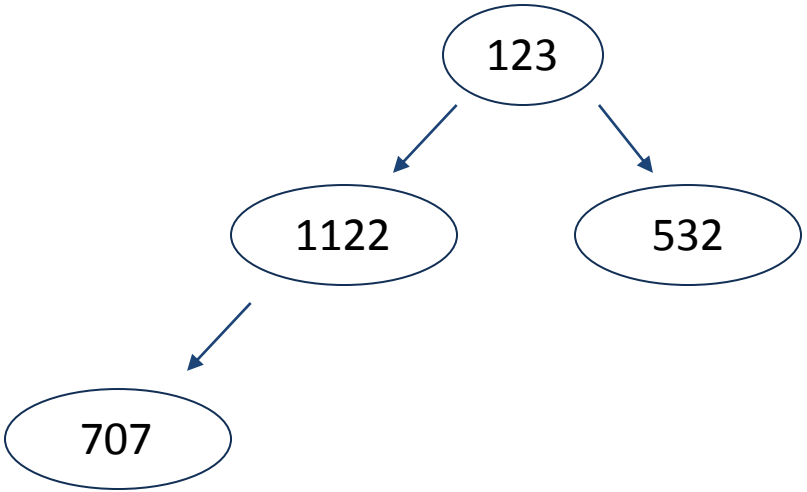
Parent-Child Relationship –1

```
['True-ProcessCreated’,
{'pid': 1080,'ppid':5040,
'source image file 01':‘some_file’,
'process': 'FFFFE0011F705080'}]
```

```
['image-loaded’,
{'loaded full image name': '\\Windows\\SysWOW64\\imm32
.dll’,
'target pid': 5040,
'injector pid’: 1234,
'image base address': '0x74820000'}]
```

(Ignore if the injector and target pids are the same.)

Parent-Child Relationship –2



Find Immediate Descendants

Pseudocode

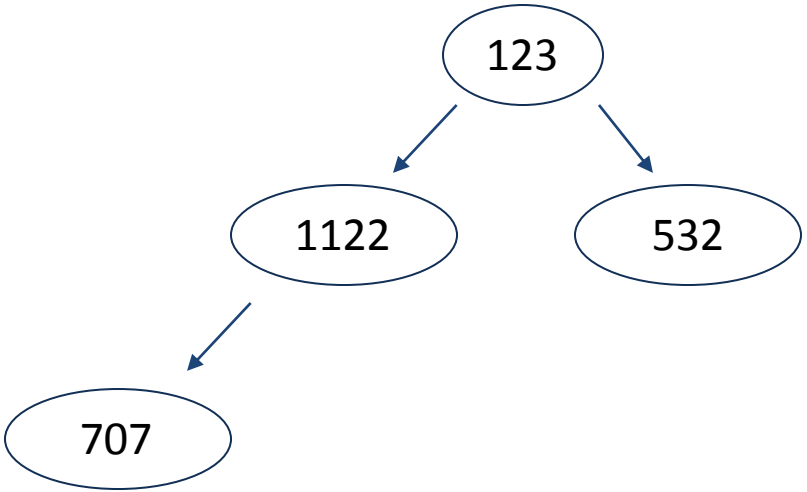
```
def get_children(self, pid):

    searchable_lines = np.intersect1d(self.pid_2_idx[pid], self.process_creation_lines)
    kids = []
    kids_index = []
    for i in searchable_lines:
        line = self.parsed_log[i][1]

        if line['ppid'] == pid:
            kids.append(line['pid'])
            kids_index.append(i)

    self.pid_nodes[pid].children = kids
    return kids, kids_index
```

Parent-Child Relationship –2



Detecting Ancestry with Recursion

```
kids = self.pid_nodes[pid].children # pid.get_children() was called already
if len(kids) == 0:
    return []
else:
    decendants.extend(kids)
    for kid in kids:
        kid_decendants = self.get_decendants(kid)
        decendants.extend(kid_decendants)
```

Feature Vector So Far…

Feature Description	Value
Self-replicates?	True
Number of suspicious requests (i.e., has the same pid)	20
Number of suspicious requests (i.e., has different pids; has ancestry)	6
Number of suspicious requests (i.e., has different pids and **no** ancestry	2
Number of children	3
Number of parents	1
Number of descendants	12
Number of ancestors	1

What Next…

- Identify as many feature as possible
- Inspect, clean and prep the data as in the labs
- Create training and test sets
- Train a model to predict whether a file is malicious or benign
- Exercise the model against real data
- *Investigate and take action for potentially malicious files*
- Retrain the model as needed

LAB 5

Applying Data Science to Network Traffic Analysis (Multi-part Lab)