Data centric AI

Slides from https://dcai.csail.mit.edu/

Traditional Machine Learning is model-centric

- When you learn ML in school…
  - A dataset is given to you
  - Usually fairly clean & well-curated with no mislabeling.
- Your goal…
  - Produce the best model for this dataset (model-centric AI)
- In traditional (model-centric) AI, you learn:
  - Data preprocessing – mainly based on feature extraction
  - Different types of models or architectures
  - Tuning their hyperparameters

But in real life…

- The dataset is not fixed!
  - The company/user doesn't care about the ML tricks you've used to produce accurate predictions on highly curated data
  - Real-world data tends to be messy
    - **So consider fixing issues in the data**
    - Ten of the most-used ML test sets have pervasive label errors
      - https://labelerrors.com/ (Northcutt, Athalye, & Mueller, NeurIPS, 2021)
- Experimented data scientist
  - It's more worthwhile to invest in exploring & fixing the data than tinkering with models
  - But this process is time consuming / boring for large datasets

What is data-centric AI?

- Data-centric AI often takes one of two forms:
  - AI algorithms that understand data and use that information to improve models.
    - Curriculum learning, Bengio et al., ICML, 2009
    - “Humans and animals learn much better when the examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts.”
  - AI algorithms that modify data to improve AI models.
    - Confident learning, Northcutt et al., JAIR, 2021
    - “Confident learning (CL) is an alternative approach which focuses instead on label quality by characterizing and identifying label errors in datasets, based on the principles of pruning noisy data.”

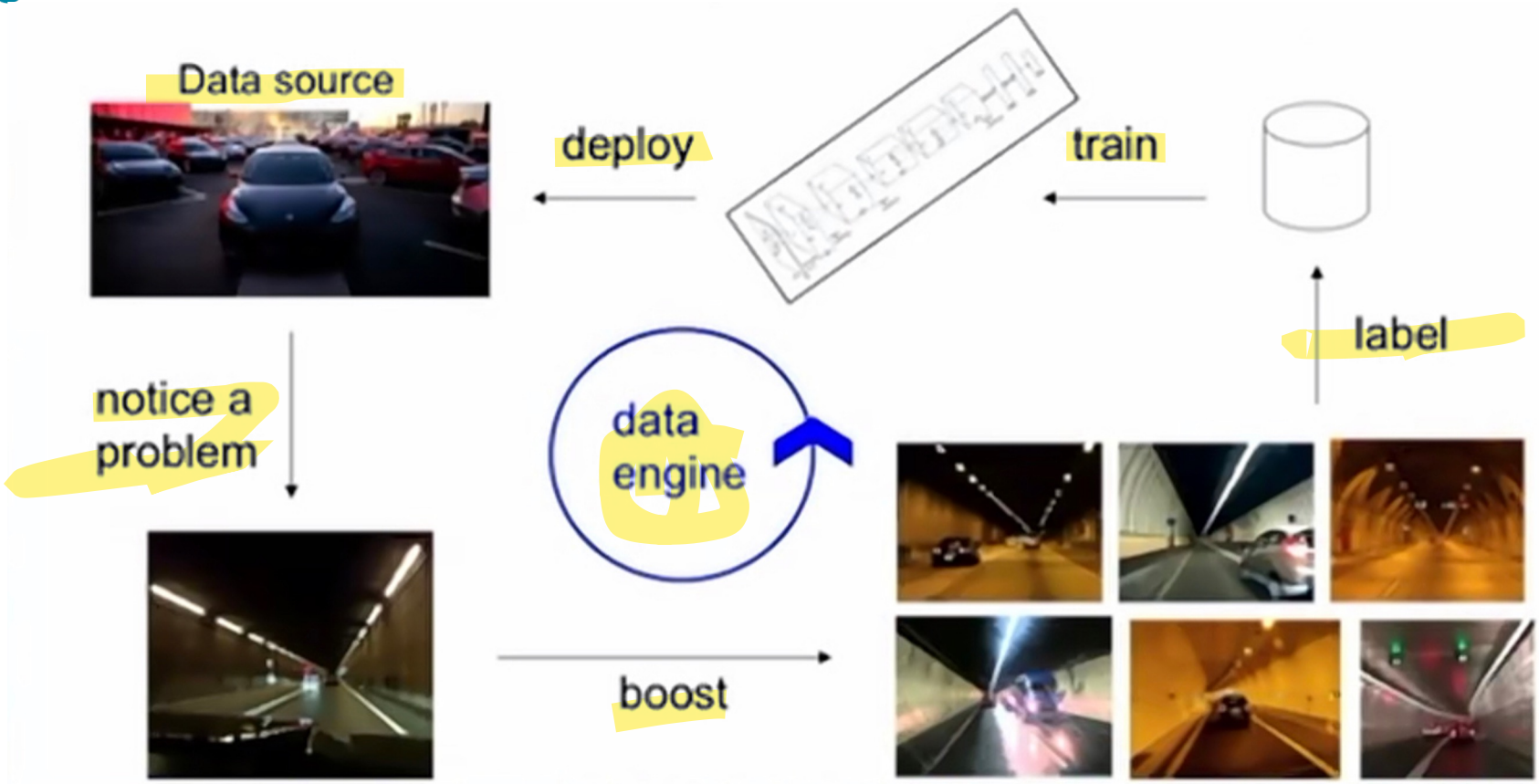
FOCUSES ON LEVEL ERRORS

Model-centric AI vs Data-centric AI

- Model-centric AI:
  - Given a dataset, try to produce the best model
  - Change the model to improve performance on an AI task
- Data-centric AI:
  - Given any model, try to improve the training dataset
  - Systematically/algorithmically change the dataset to improve performance on an AI task
- Examples of Data-centric AI
  - Outlier Detection and Removal (handling abnormal examples in dataset)
  - Error Detection and Correction (handling incorrect values/labels in dataset)
  - Data Augmentation (adding examples to data to encode prior knowledge)
  - Feature Engineering and Selection (manipulating how data are represented)
  - Establishing Consensus Labels (determining true label from crowdsourced annotations)
  - Active Learning (selecting the most informative data to label next)
  - Curriculum Learning (ordering the examples in dataset from easiest to hardest)

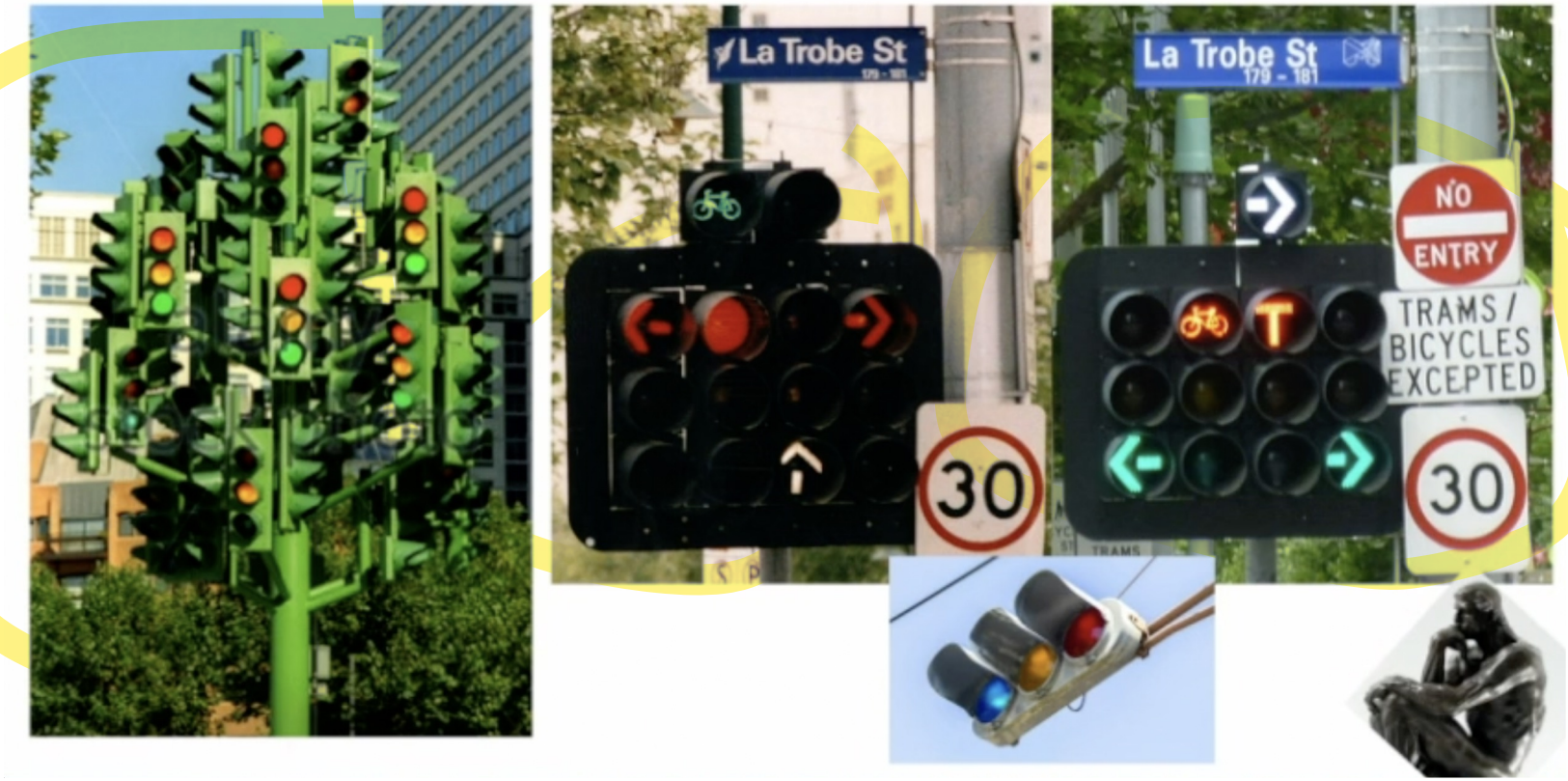

"To conclude my talk, I will show that our method finds a label error in Yann's MNIST dataset."
Jun 17, 2016
Fri, 2:19 PM GMT-04:00
- Hinton (@FAIR, NYC)
MNIST Given Label:
3

Tesla Data Engine: use model outputs to improve training dataset



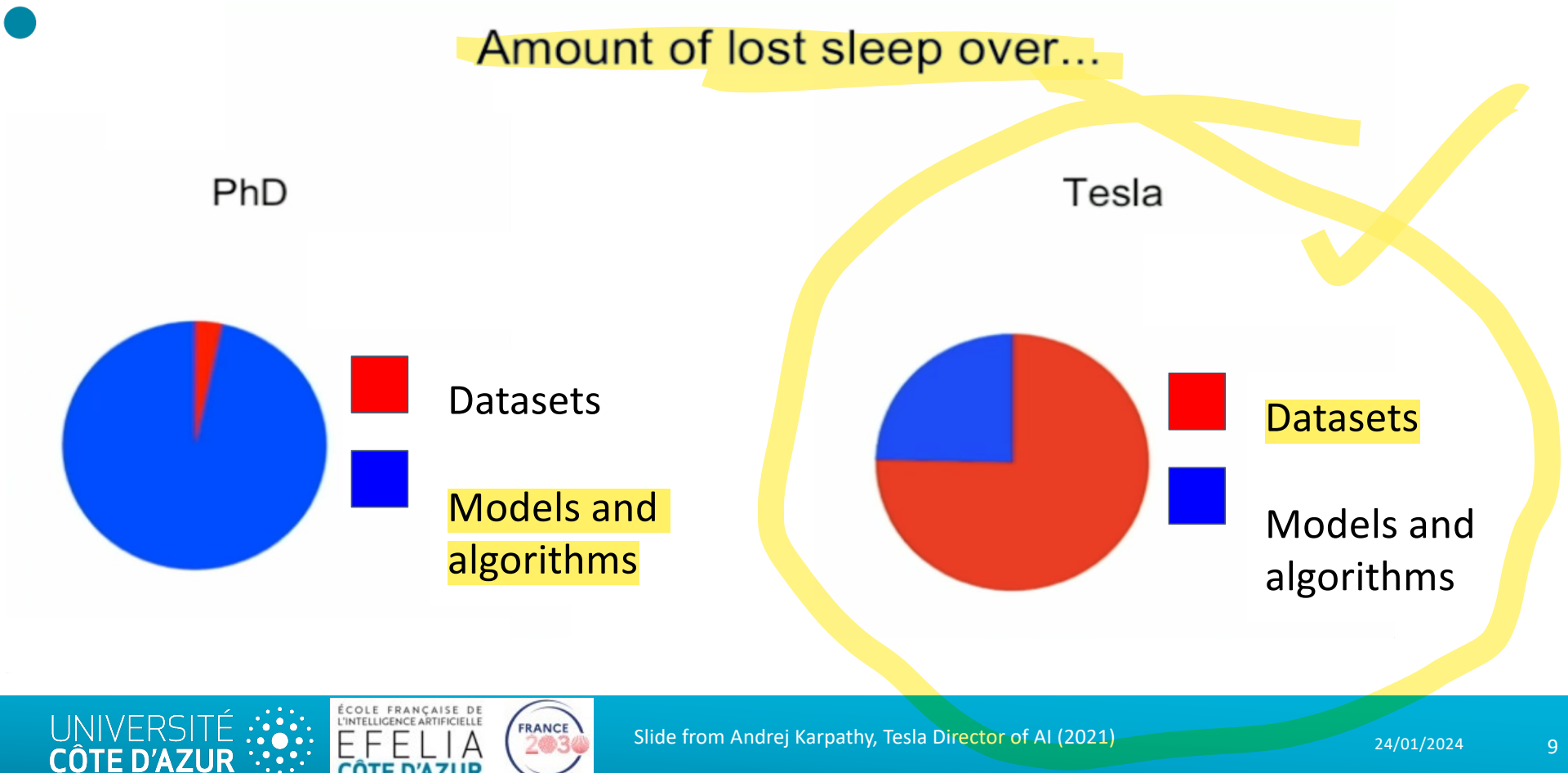
Slide from Andrej Karpathy, Tesla Director of AI (2021)

Tesla Data Engine: use model outputs to improve training dataset



Slide from Andrej Karpathy, Tesla Director of AI (2021)

Tesla Data Engine: use model outputs to improve training dataset



Slide from Andrej Karpathy, Tesla Director of AI (2021)

Data-centric vs model-centric for learning with noisy labels

Compare Accuracy: Learning with 40% label noise in CIFAR-10

Northcutt et al., JAIR, 2021
"Confident Learning"

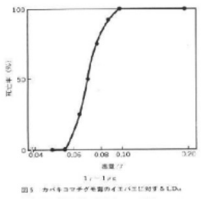
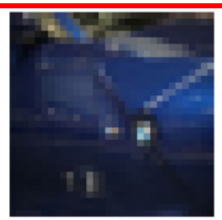
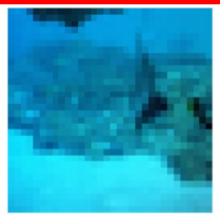
		Fraction of zeros in the off-diagonals of $p(\tilde{y}\|y^*)$: 0	0.6 ← More realistic (e.g. ImageNet)
Baseline (remove prediction != label)	Data-centric: Train with errors removed. "*Change the dataset*"	83.9	84.2
Confident learning methods		84.8	86.2
		86.7	86.9
		87.1 → Same perf	**87.2**
		87.1	**87.2**
INCV (Chen et al., 2019)		84.4	73.6
Mixup (Zhang et al., 2018)		76.1	59.8
SCE-loss (Wang et al., 2019)	Model-centric: Train with errors. "*adjust the loss*"	76.3 → Perf drop-off	58.3
MentorNet (Jiang et al., 2018)		64.4	61.5
Co-Teaching (Han et al., 2018)		62.9	58.1
S-Model (Goldberger et al., 2017)		58.6	57.5
Reed (Reed et al., 2015)		60.5	58.6
Baseline		60.2	57.3

How to improve dataset quality

- We relied on mostly-human-powered solutions to improve dataset quality:
  - Spend more $ for higher quality data or more labels
  - Build custom tools to evaluate specific data (e.g. Tesla data quality platform)
  - Fixing data inside a Jupyter notebook
- Data-centric AI
  - Systematizing these approaches to be more reliable, accurate, and generally usable on many datasets.

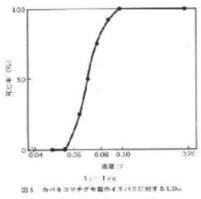
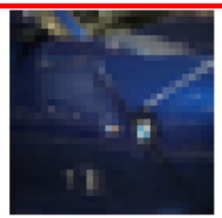
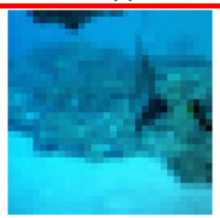
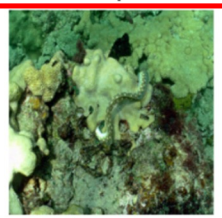
Label errors

Examples from
https://labelerrors.com/

	MNIST	CIFAR-10	CIFAR-100	Caltech-256	ImageNet	QuickDraw
correctable	 given: 8 corrected: 9	 given: cat corrected: frog	 given: lobster corrected: crab	 given: dolphin corrected: kayak	 given: white stork corrected: black stork	 given: tiger corrected: eye
multi-label	(N/A)	(N/A)	 given: hamster also: cup	 given: laptop also: people	 given: mantis also: fence	 given: wristwatch also: hand
neither	 given: 6 alt: 1	 given: deer alt: bird	 given: rose alt: apple	 given: house-fly alt: ladder	 given: polar bear alt: elephant	 given: pineapple alt: raccoon
non-agreement	 given: 4 alt: 9	 given: automobile alt: airplane	 given: dolphin alt: ray	 given: yo-yo alt: frisbee	 given: eel alt: flatworm	 given: bandage alt: roller coaster

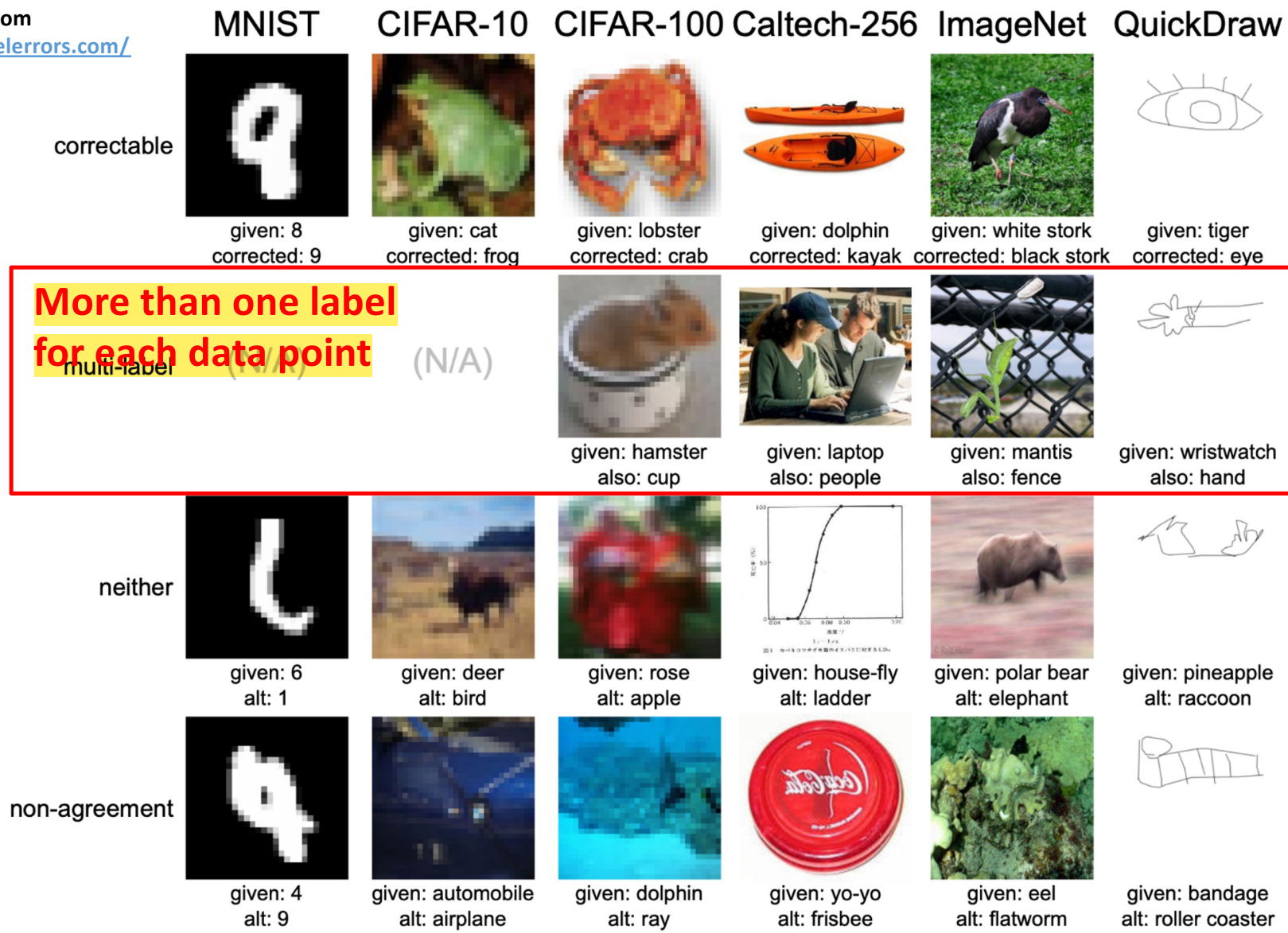
'Hard' Examples

Examples from
https://labelerrors.com/

	MNIST	CIFAR-10	CIFAR-100	Caltech-256	ImageNet	QuickDraw
correctable	 given: 8 corrected: 9	 given: cat corrected: frog	 given: lobster corrected: crab	 given: dolphin corrected: kayak	 given: white stork corrected: black stork	 given: tiger corrected: eye
multi-label	(N/A)	(N/A)	 given: hamster also: cup	 given: laptop also: people	 given: mantis also: fence	 given: wristwatch also: hand
neither	 given: 6 alt: 1	 given: deer alt: bird	 given: rose alt: apple	 given: house-fly alt: ladder	 given: polar bear alt: elephant	 given: pineapple alt: raccoon
non-agreement	 given: 4 alt: 9	 given: automobile alt: airplane	 given: dolphin alt: ray	 given: yo-yo alt: frisbee	 given: eel alt: flatworm	 given: bandage alt: roller coaster

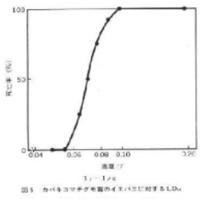
Potentially out of distribution

Examples from
https://labelerrors.com/



One correct label

Focus of this lecture.

	MNIST	CIFAR-10	CIFAR-100	Caltech-256	ImageNet	QuickDraw
correctable	 given: 8 corrected: 9	 given: cat corrected: frog	 given: lobster corrected: crab	 given: dolphin corrected: kayak	 given: white stork corrected: black stork	 given: tiger corrected: eye
	(N/A)	(N/A)	 given: hamster also: cup	 given: laptop also: people	 given: mantis also: fence	 given: wristwatch also: hand
neither	 given: 6 alt: 1	 given: deer alt: bird	 given: rose alt: apple	 given: house-fly alt: ladder	 given: polar bear alt: elephant	 given: pineapple alt: raccoon
non-agreement	 given: 4 alt: 9	 given: automobile alt: airplane	 given: dolphin alt: ray	 given: yo-yo alt: frisbee	 given: eel alt: flatworm	 given: bandage alt: roller coaster

Examples from https://labelerrors.com/

How did you find misslabeled data ?

- Sorting label errors by loss ?
  - Sure you can sort examples by loss, but what's the cut-off? How are you supposed to know how many label errors there are in the dataset without checking the errors by hand? How do you automate this for large datasets?
- Confident learning
  - is a framework of theory and algorithms for:
    - Finding label errors in a dataset
    - Ranking data by likelihood of being a label issue
    - Learning with noisy labels
    - Complete characterization of label noise in a dataset
- With confident learning, you can use ANY model's predicted probabilities to find label errors
  - Data-centric
  - Model-agnostic

Where do noisy labels come from?

- Where do noisy labels come from? Manual or Automatic annotation
  - Clicked the wrong button (upvote/downvote, 1 sta
  - Mistakes
  - Mismeasurement
  - Incompetence
  - Another ML model's bad predictions
  - Corruption and a million other places

$C_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	100	40	20
$\tilde{y}=fox$	56	60	0
$\tilde{y}=cow$	32	12	80

- All of these result in labels being flipped to other labels.
- Examples of label flippings:
  - Image of a Dog is labeled Fox
  - Tweet "Hi welcome to the team!" is labeled Toxic language
- Generally, label noise is not symmetric
  - labeling errors are more often made in one direction than the other

Confident Learning assume label noise

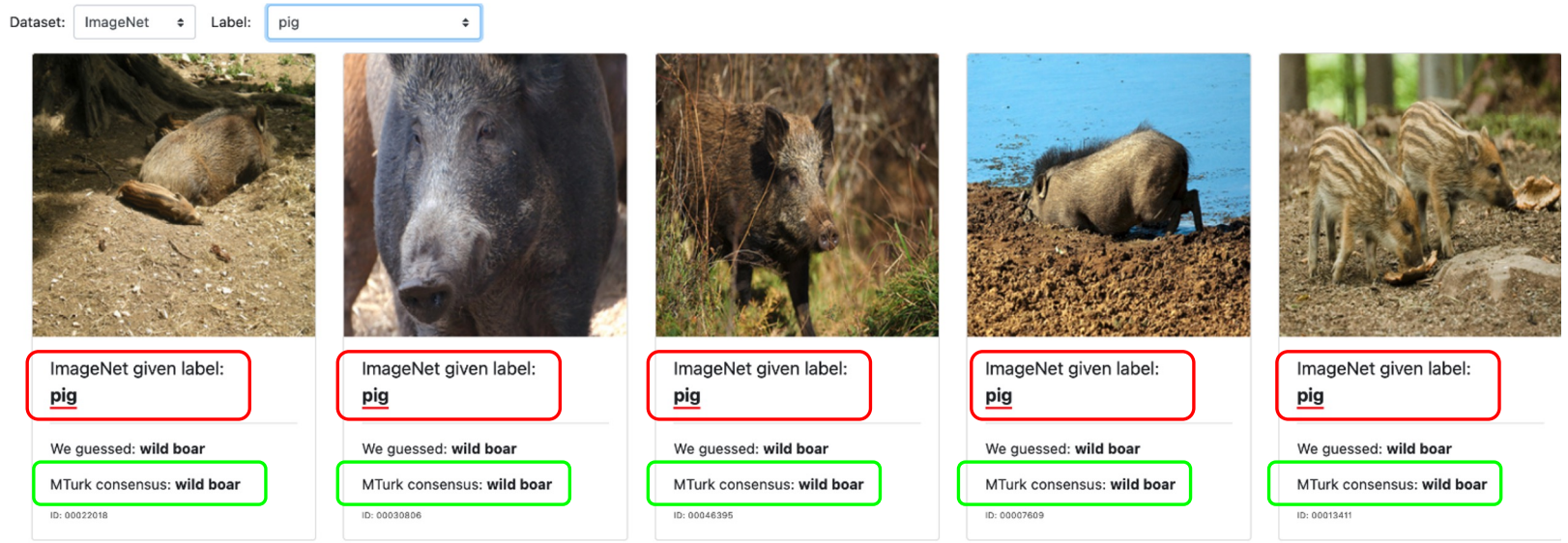
- Confident Learning **assume** labels are flipped based on an unknown transition matrix $p(\tilde{y}|y^*)$ that depends only on pairwise noise rates between classes, not the data x
  - $p(\tilde{y}|y^*;x) = p(\tilde{y}|y^*)$
  - $\tilde{y}$ is the noisy observed label
  - y^* is the unobserved, latent, correct label

This assumption is reasonable for real-world data. Let's look at some...

https://docs.aws.amazon.com/AWSMechTurk/latest/RequesterUI/mechanical-turk-concepts.html

In real-world images, lots of “boars” were mislabeled as “pigs”

But no “missiles” or “keyboards” were mislabeled as “pigs”

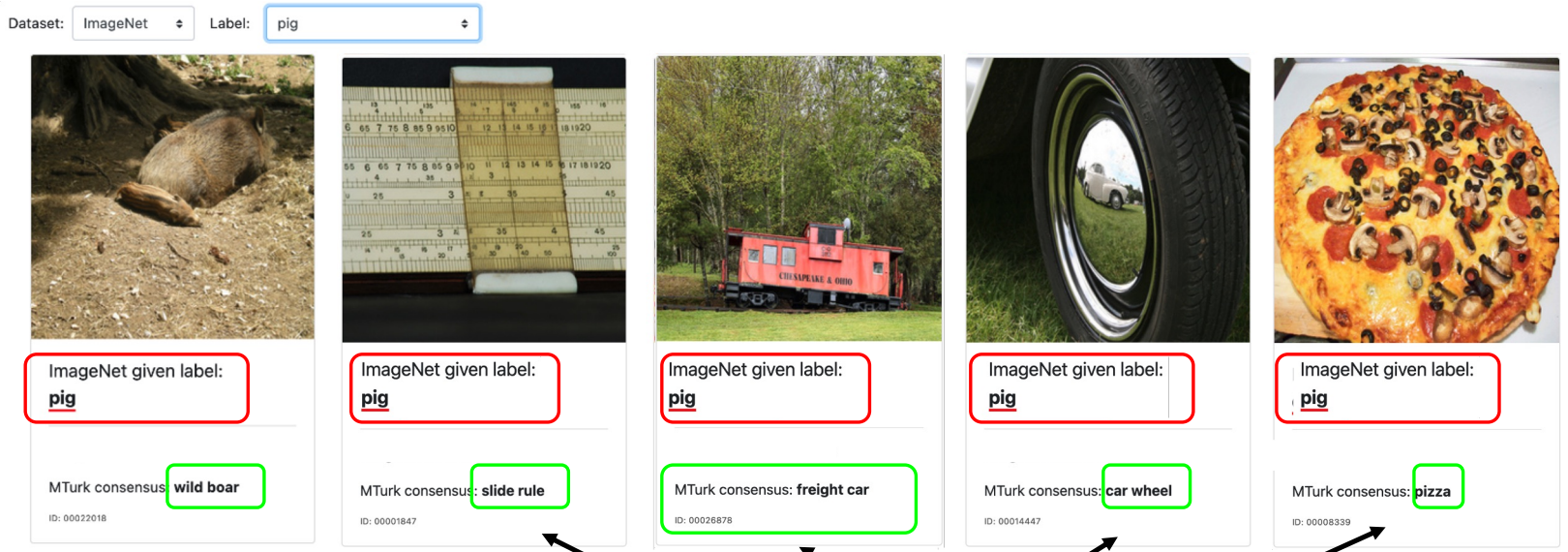


This “class-conditional” label noise depends on the class, not the image data (wh $\boldsymbol{x}$ the pig looks like)

Given its realistic nature, we choose to solve for “class-conditional noise” in CL.

What does uniform label noise look like?

Goldberger and BenReuven (2017) Arazo et al. (2019)



Fictitious examples
(**not** naturally occurring)

Does label noise matter? Deep learning is robust to label noise… right?

(Jindal et al. ICDM 2016), (Krause et al. ECCV 2016) suggest that “with enough data, learning is possible with arbitrary amounts of uniformly random label noise”

These results assume uniformly random label noise and usually don’t apply to **real-world settings**.

Types of Noise that we will NOT cover

Noise in Data

Blurry images, adversarial examples, typos in text, background noise in audio

CL assumes **labels** are noisy, not data.

Annotator Label Noise

1 Annotation: Sports Car

2 Annotation: Toy Car

3 Annotation: Toy Car

Dawid and **Skene** (1979)

CL assumes **one** annotation per example

Types of methods for Learning with Noisy Labels

Model-Centric Methods

"Change the Loss"

- Use loss from another network
  - Co-Teaching (Han et al., 2018)
  - MentorNet (Jiang et al., 2017)
- Modify loss directly
  - SCE-loss (Wang et al., 2019)
- Importance reweighting
  - (Liu & Tao, 2015; Patrini et al., 2017; Reed et al., 2015; Shu et al., 2019; Goldberger & Ben-Reuven, 2017)

You'll see in another course

Data-Centric Methods

"Change the Data"

- Find label errors in datasets
- Then learn with(out) noisy labels by providing cleaned data for training
  - (Pleiss et al., 2020; Yu et al., ICML, 2019; Li et al., ICLR, 2020; Wei et al., CVPR, 2020, Northcutt et al., JAIR, 2021)

This lecture

How does confident learning work?

$p(\tilde{y}|y^*)$

$p(y^*)$

$p(y^*|\tilde{y})$

$p(\tilde{y}, y^*)$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0.25	0.1	0.05
$\tilde{y}=fox$	0.14	0.15	0
$\tilde{y}=cow$	0.08	0.03	0.2

Off-diagonals tell you what fraction of your dataset is mislabeled.

Example -- "3% of your cow images are actually foxes"

How does confident learning work?

To estimate $p(\tilde{y}, y^*)$and find label errors, confident learning requires two inputs:

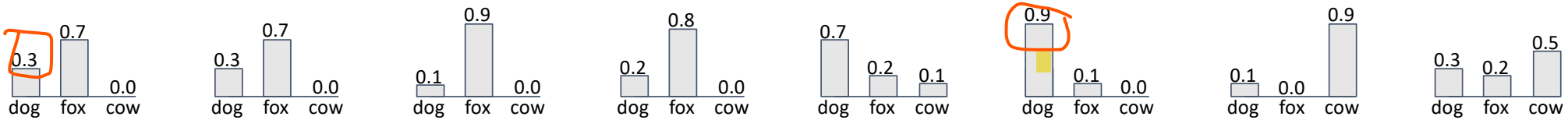
- Noisy labels: $\tilde{y}$
- Predicted probabilities: $\hat{p}(\tilde{y} = i; x, \theta)$

Key idea: First we find thresholds as a proxy for the machine's self-confidence, on average, for each class j

$$t_j = \frac{1}{|\boldsymbol{X}_{\tilde{y}=j}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=j}} \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta})$$

How does confident learning work?

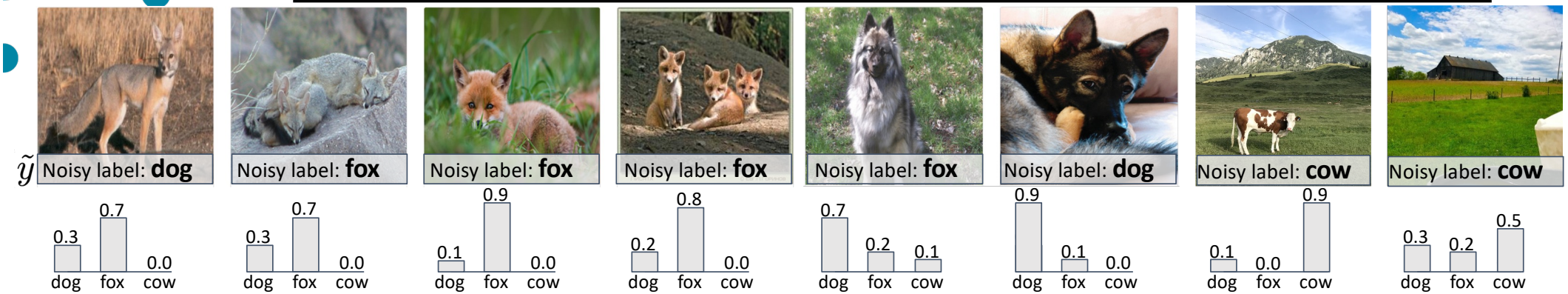
Before confident learning starts, a model is trained on this data using cross-validation, to produce $\hat{p}(\tilde{y} = i; x, \theta)$, the out-of-sample predicted probabilities



We then calculate the thresholds that represent the confidence of the model for each class.

$$t_{dog} = \frac{1}{|X_{\tilde{y}=dog}|} \sum_{x \in X_{\tilde{y}=dog}} p(\tilde{y} = dog; x, \theta) = \frac{1}{2}(0.3 + 0.9) = 0.6$$

How does confident learning work?



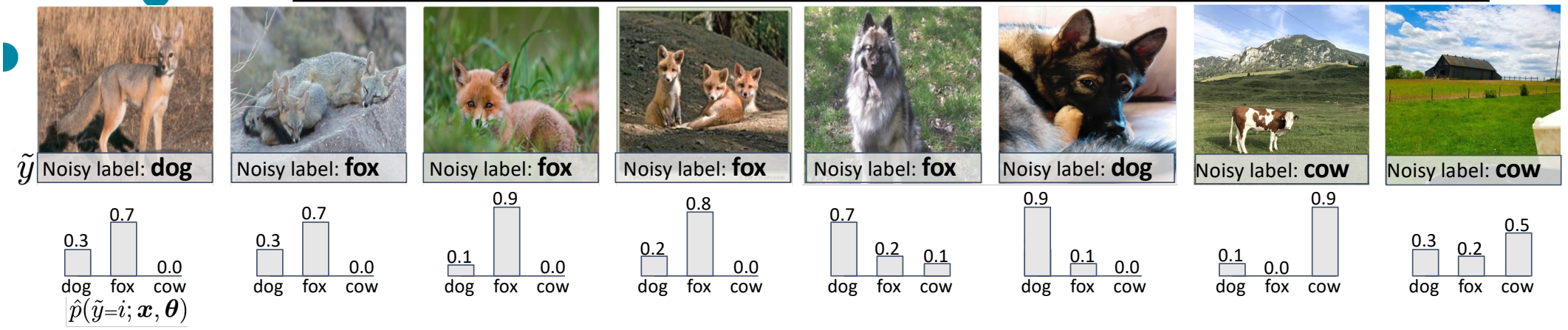
We then calculate the thresholds that represent the confidence of the model for each class.

$$t_{dog} = \frac{1}{|X_{\tilde{y}=dog}|} \sum_{x \in X_{\tilde{y}=dog}} p(\tilde{y} = dog; x, \theta) = \frac{1}{2}(0.3 + 0.9) = 0.6$$

$$t_{fox} = \frac{1}{4}(0.7 + 0.9 + 0.8 + 0.2) = 0.65$$

$$t_{cow} = \frac{1}{2}(0.9 + 0.5) = 0.7$$

How does confident learning work?



CL estimates the labeling errors for each pair of labels.(noisy label i, true label j)

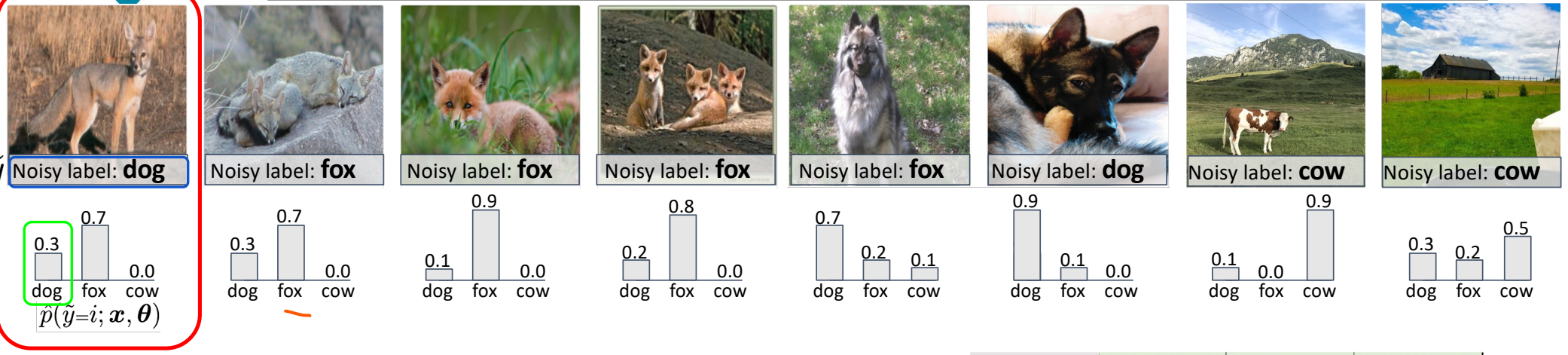
$$\hat{X}_{\tilde{y}=i,y^*=j} = \{x \in X_{\tilde{y}=j}: \hat{p}(\tilde{y}=j; x, \theta) \geq t_j\}$$

The confident joint $C_{\tilde{y}.y^*}$, counts the size of each set

→ $C_{\tilde{y},y^*}[i][j] = |\hat{X}_{\tilde{y}=i,y^*=j}|$

$C_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	Creating a matrix of counts to estimate the unnormalized joint distribution		
$\tilde{y}=fox$			
$\tilde{y}=cow$			

How does confident learning work?



t_j

$t_{dog} = 0.6$

$t_{fox} = 0.65$

$t_{cow} = 0.7$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} =$$

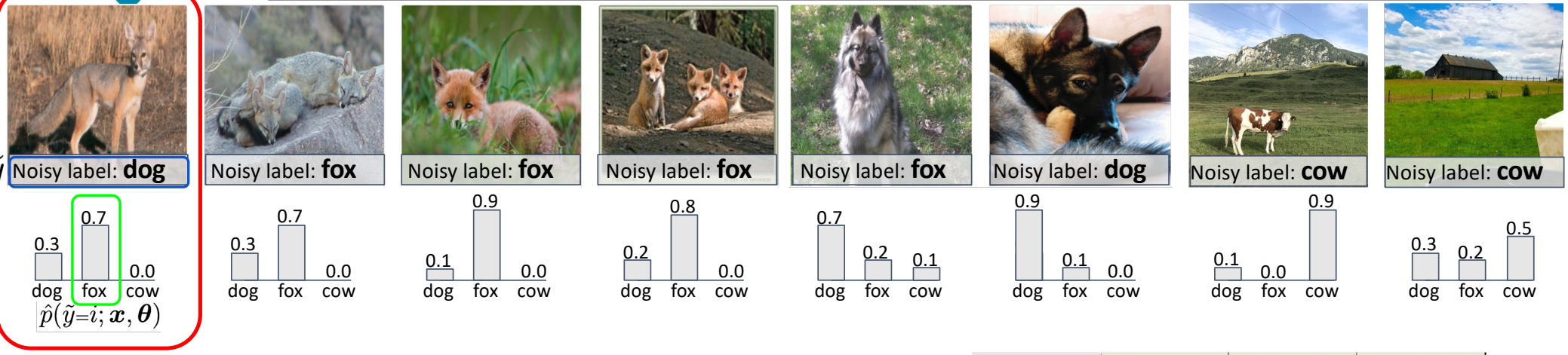
$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

$0.3 \not\geq 0.6$

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0	0	0
$\tilde{y}=fox$	0	0	0
$\tilde{y}=cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?



$$t_j$$

$$t_{dog} = 0.6$$

$$t_{fox} = 0.65$$

$$t_{cow} = 0.7$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

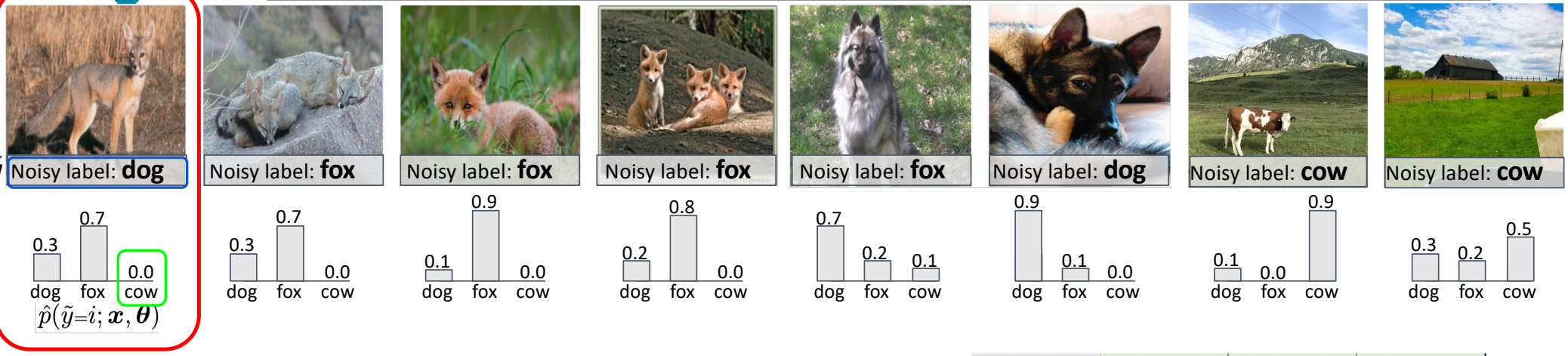
✓

$0.7 \geq 0.65$

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0	1	0
$\tilde{y}=fox$	0	0	0
$\tilde{y}=cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?



$$t_j$$

$$t_{dog} = 0.6$$
$$t_{fox} = 0.65$$
$$t_{cow} = 0.7$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} =$$

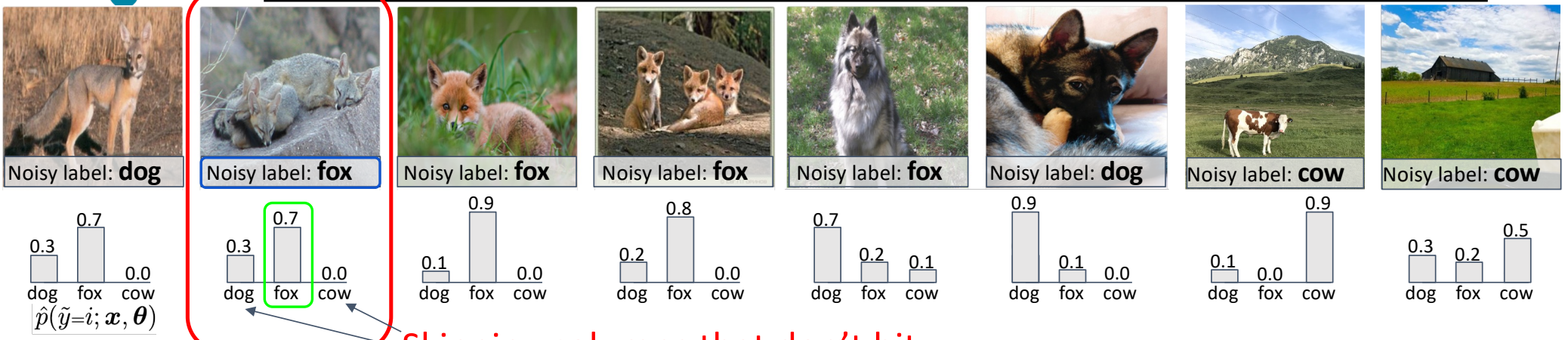
$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

0.0 ≱ 0.9

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0	1	0
$\tilde{y}=fox$	0	0	0
$\tilde{y}=cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j}|$$

How does confident learning work?



Skipping columns that don't hit threshold

$$t_j$$

$$t_{dog} = 0.6$$

$$t_{fox} = 0.65$$

$$t_{cow} = 0.7$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

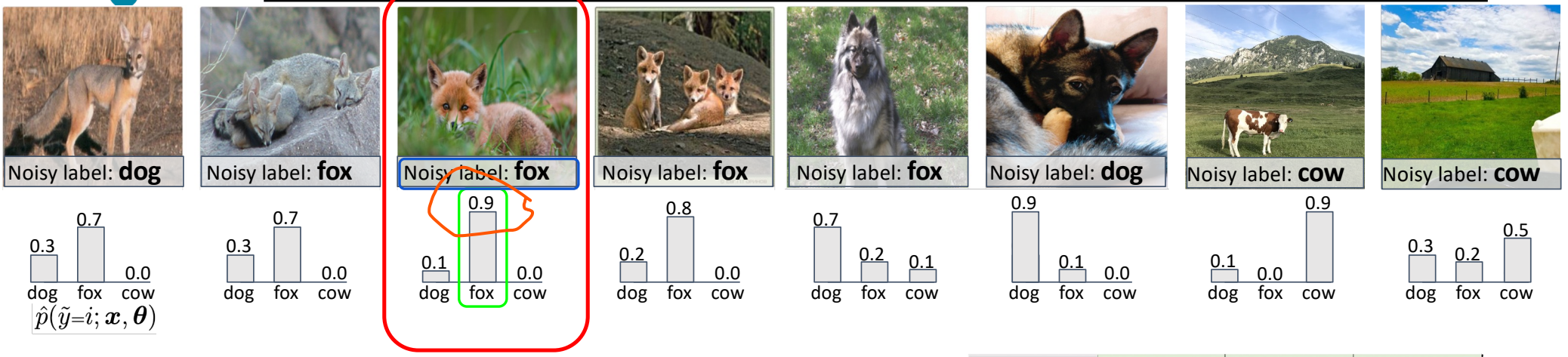
✓

$0.7 \geq 0.65$

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0	1	0
$\tilde{y}=fox$	0	1	0
$\tilde{y}=cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?



t_j

$t_{dog} = 0.6$

$t_{fox} = 0.65$

$t_{cow} = 0.7$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

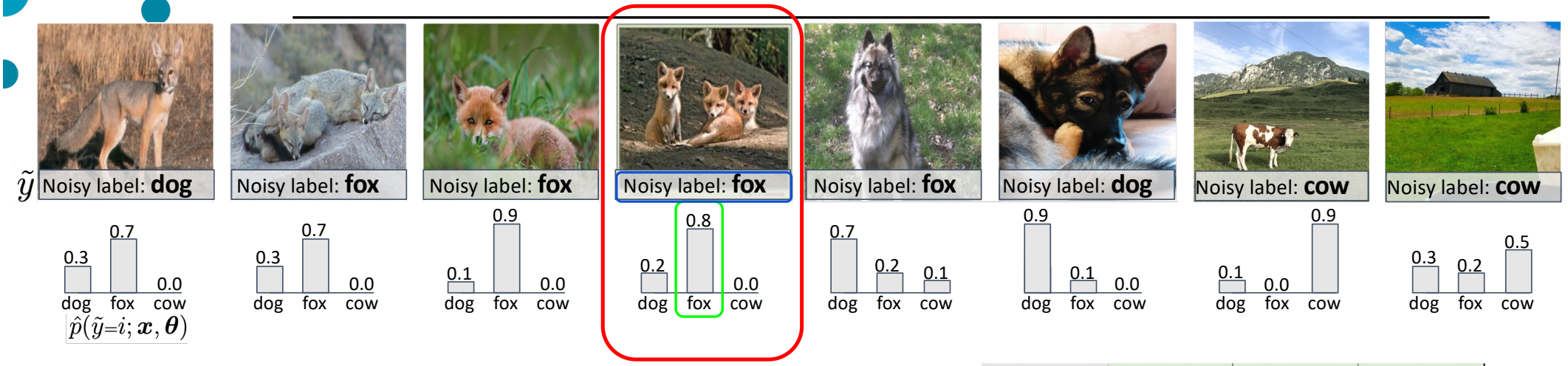
✓

0.9 ≥ 0.65

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0	1	0
$\tilde{y}=fox$	0	2	0
$\tilde{y}=cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?



$$t_j$$

$$t_{dog} = 0.6$$

$$t_{fox} = 0.65$$

$$t_{cow} = 0.7$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

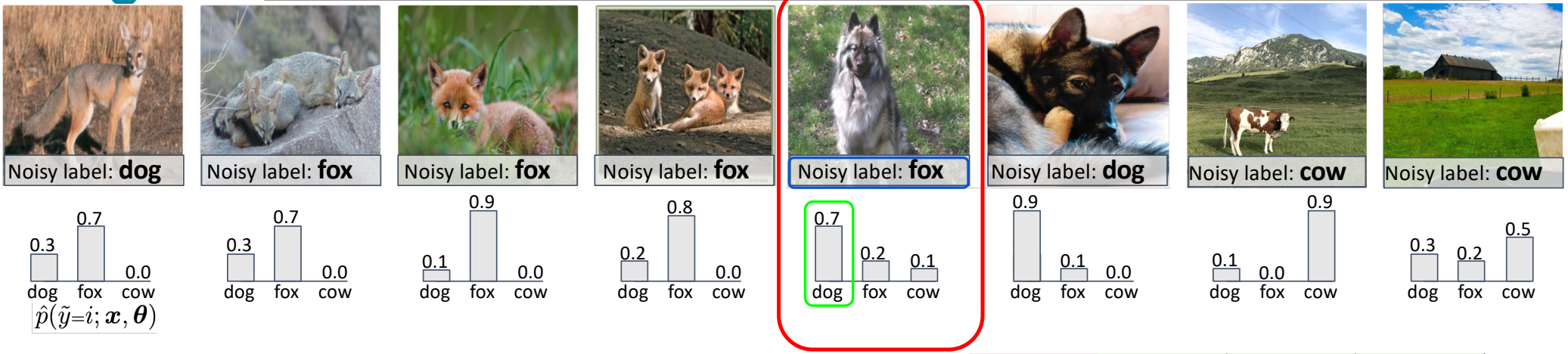
✓

0.8 ≥ 0.65

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0	1	0
$\tilde{y}=fox$	0	3	0
$\tilde{y}=cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?



$$t_j$$

$$t_{dog} = 0.6$$

$$t_{fox} = 0.65$$

$$t_{cow} = 0.7$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

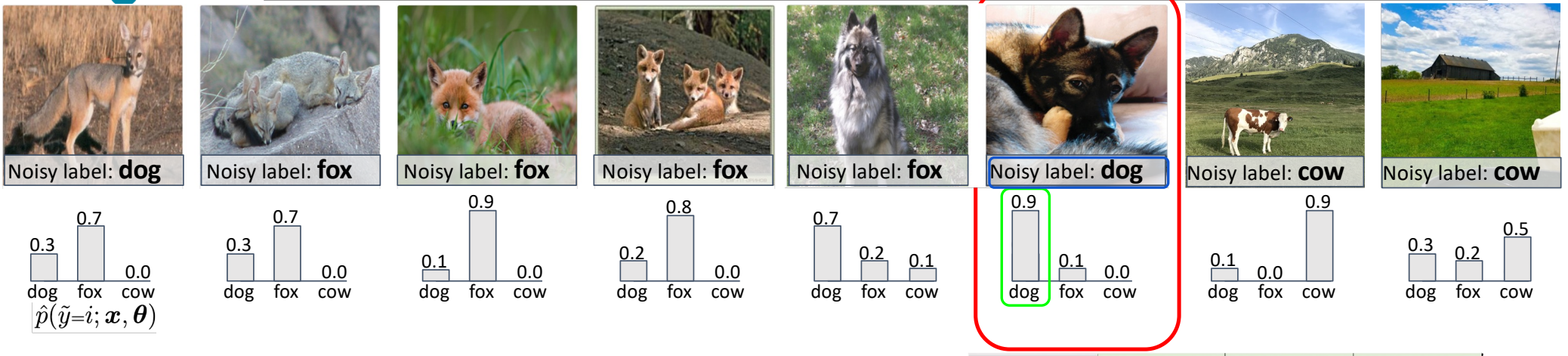
✓

$0.7 \geq 0.65$

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0	1	0
$\tilde{y}=fox$	1	3	0
$\tilde{y}=cow$	0	0	0

$$C_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?



$$t_j$$

$$t_{dog} = 0.6$$
$$t_{fox} = 0.65$$
$$t_{cow} = 0.7$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

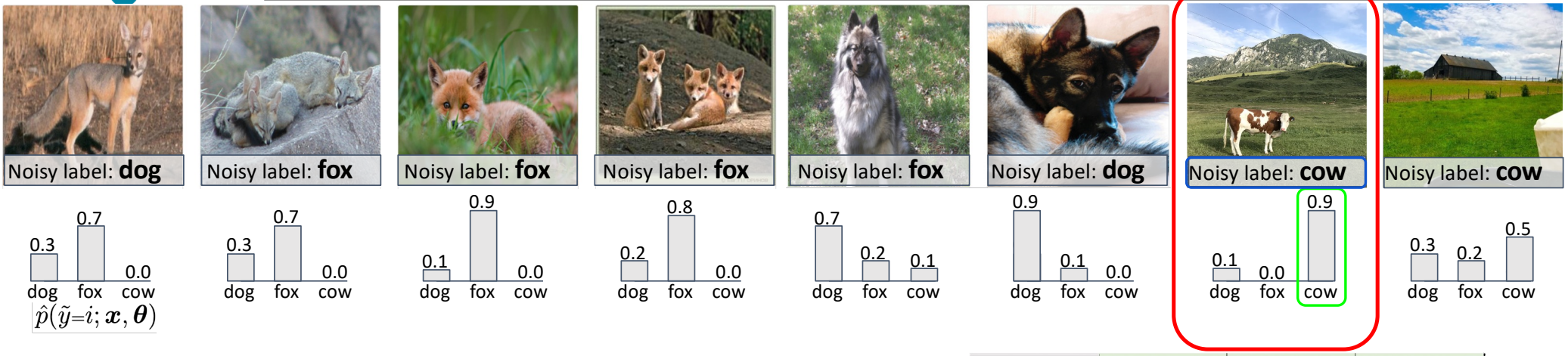
✓

$0.9 \geq 0.6$

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	1	1	0
$\tilde{y}=fox$	1	3	0
$\tilde{y}=cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?



$$t_j$$

$$t_{dog} = 0.6$$
$$t_{fox} = 0.65$$
$$t_{cow} = 0.7$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

✓

0.9 ≥ 0.7

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	1	1	0
$\tilde{y}=fox$	1	3	0
$\tilde{y}=cow$	0	0	1

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?

$$t_j$$

$$t_{dog} = 0.6$$
$$t_{fox} = 0.65$$
$$t_{cow} = 0.7$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} =$$

$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

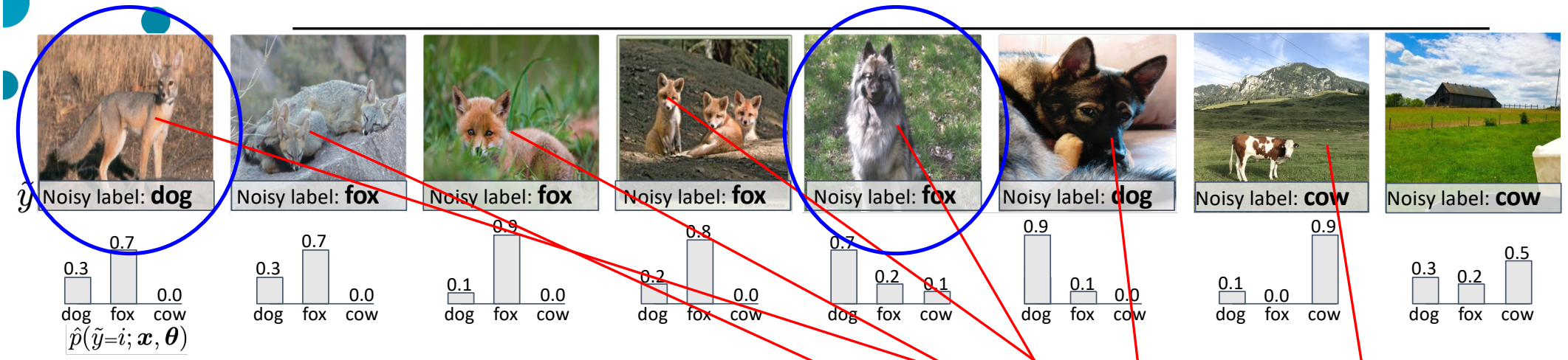
$0.5 \not\geq 0.7$

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	1	1	0
$\tilde{y}=fox$	1	3	0
$\tilde{y}=cow$	0	0	1

Out of distribution

$$C_{\tilde{y},y^*}[i][j] = |\hat{X}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?



$$t_j$$

$$t_{dog} = 0.6$$
$$t_{fox} = 0.65$$
$$t_{cow} = 0.7$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

Off diagonals are CL-guessed label errors

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	1	1	0
$\tilde{y}=fox$	1	3	0
$\tilde{y}=cow$	0	0	1

$$C_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?

- After looking through the entire dataset (400 images), we have:

$C_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	100	40	20
$\tilde{y}=fox$	56	60	0
$\tilde{y}=cow$	32	12	80

- From $C_{\tilde{y},y^*}$ we obtain the joint distribution of label noise

Estimated

$\hat{p}(\tilde{y},y^*)$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0.25	0.1	0.05
$\tilde{y}=fox$	0.14	0.15	0
$\tilde{y}=cow$	0.08	0.03	0.2

How to use confident learning

- Choose a model
  - `lr = LogisticRegression(max_iter=5000)`
- Fit the model with cross_val_predict
  - `nb_folds = 3  # for efficiency; values like 5 or 10 will generally work better`
  - `pred_probs = cross_val_predict(lr, X, y, cv=num_ nb_folds, method="predict_proba")`
- Build the 'confident' list and the 'label issues' list
  - All elements on the diagonal are on the confident list
  - Element that are not on the diagonal have some label issues
  - You can implement the algorithm
  - Or use cleanlab
    - from cleanlab.filter import find_label_issues
    - ranked_label_issues = find_label_issues(y, pred_probs, return_indices_ranked_by="self_confidence")
- Examine the label for all elements that are not in the confident list
  - With this list, you can decide whether to fix label issues or prune some of these examples from the dataset.

Example with MNIST

- Confident list length: more than 1000 elements

Original Data

	precision	recall	f1-score	support
0	0.96	0.96	0.96	6903
1	0.95	0.97	0.96	7877
2	0.91	0.90	0.91	6990
3	0.90	0.89	0.90	7141
4	0.93	0.93	0.93	6824
5	0.88	0.88	0.88	6313
6	0.94	0.95	0.95	6876
7	0.93	0.93	0.93	7293
8	0.89	0.87	0.88	6825
9	0.89	0.90	0.90	6958
acc			0.92	70000
macro	0.92	0.92	0.92	70000
weighted	0.92	0.92	0.92	70000

With removing all elements with label issue

	precision	recall	f1-score	support
0	0.97	0.98	0.97	6867
1	0.97	0.98	0.97	7853
2	0.94	0.93	0.94	6758
3	0.93	0.92	0.93	6944
4	0.94	0.94	0.94	6742
5	0.91	0.91	0.91	6077
6	0.96	0.97	0.96	6825
7	0.95	0.95	0.95	7207
8	0.92	0.91	0.91	6583
9	0.92	0.93	0.92	6784
acc			0.94	68640
macro	0.94	0.94	0.94	68640
weighted	0.94	0.94	0.94	68640

Some examples

ID: 24798
Label: 4

ID: 59915
Label: 4

ID: 59915
Label: 5

You can see that even widely-used datasets like MNIST contain problematic labels. Never blindly trust your data! You should always check it for potential issues, many of which can be easily identified by cleanlab.

CleanLab

- https://github.com/cleanlab/cleanlab
- pip install cleanlab

```
from cleanlab.classification import CleanLearning
from cleanlab.filter import find_label_issues

# Option 1 - works with sklearn-compatible models - just input the data and labels ツ
cl = CleanLearning(clf=sklearn_compatible_model)
label_issues_info = cl.find_label_issues(data, labels)

# Option 2 - works with ANY ML model - just input the model's predicted probabilities
ordered_label_issues = find_label_issues(
    labels=labels,
    pred_probs=pred_probs,  # out-of-sample predicted probabilities from any model
    return_indices_ranked_by='self_confidence',
)
```

More on preprocessing

Most common causes of outliers

- The most common or familiar type of outlier is the observations that are far from the rest of the observations or the center of mass of observations.
  - Anomalous events occur relatively infrequently
  - However, when they occur, their impact can be significant and often in a bad way
- Outliers causes
  - Data entry errors (human errors)
  - Measurement errors (instrument errors)
  - Data processing errors (data manipulation or data set unintended mutations)
  - Sampling errors (extracting or mixing data from wrong or various sources)
  - **Natural (not an error, novelties in data)**
    - **Model drift detection → retrain the model with new data**

Why removing outliers could be important

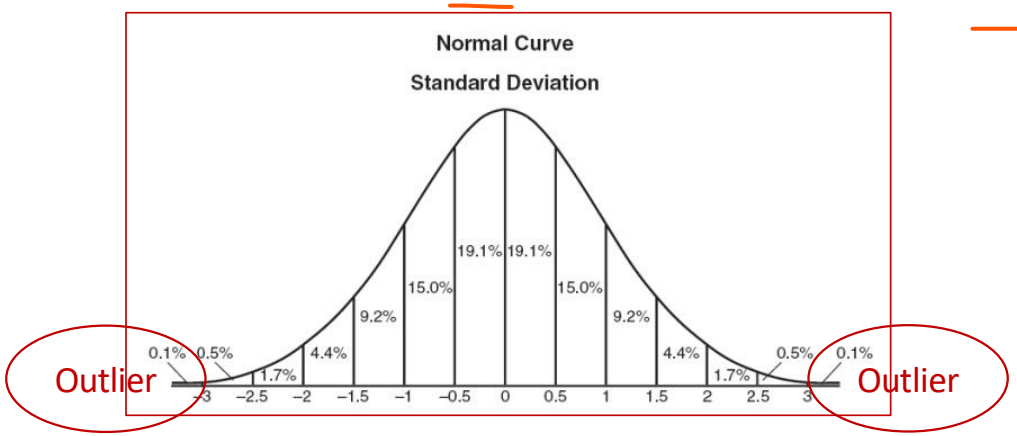
- It can be important to identify and remove outliers from data
  - Eliminating outliers allows for more accurate predictions.
  - Outliers distort statistical measures and data distributions
  - Outliers provide a misleading representation of the data.

- Generally, outlier detection is an unsupervised problem
  - We do not know which observations should be considered as "abnormal"

- Outliers could be:
  - **Univariate** can be found by examining a distribution of values in a single feature space
  - **Multivariate** can be found in a n-dimensional space
- Outliers could be
  - Point: single data
  - Contextual: noise in the data
  - Collective: subset of novelties in data

z-score (or standard score)

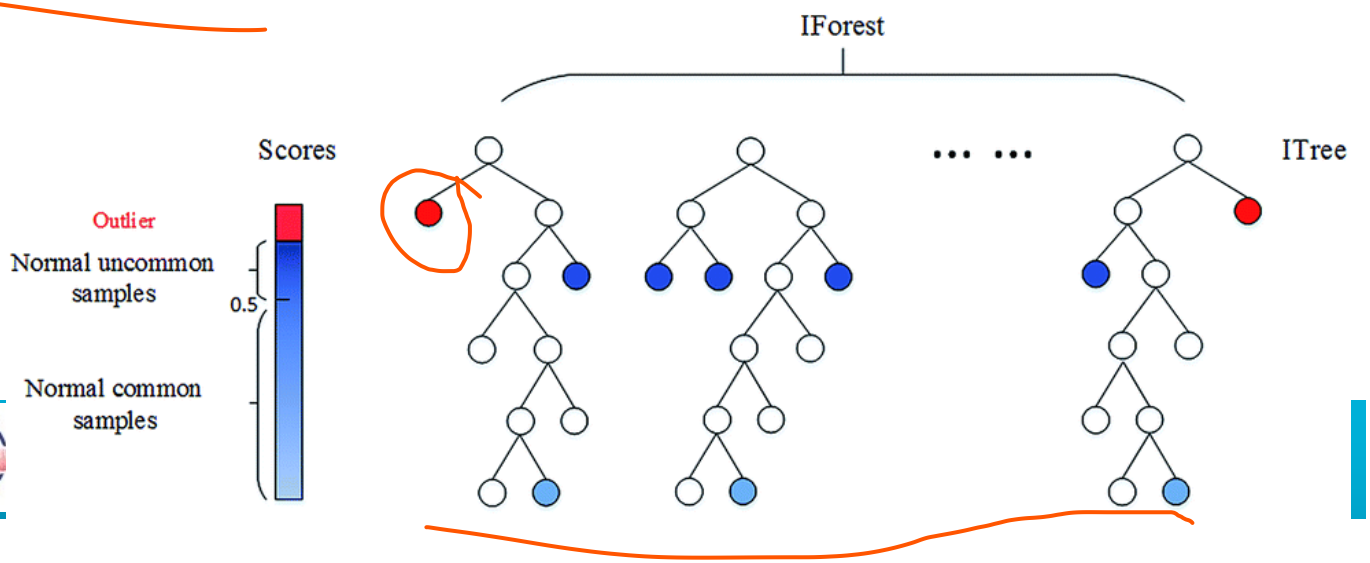
- Remove all data with $|\text{z-score}| > 3$, $z - score = \frac{x-\mu}{\sigma}$



- For univariate
  - →compute z-score for each observation
  - →Remove each observation with $|\text{z-score}| > 3$
- For multivariate
  - Compute the distance between each observation and the mean
  - We now consider the distance as a univariate characteristic

Isolation forest

1. Build a tree
   - We build a decision tree by choosing successively, in a random way, a characteristic and for this characteristic a random division value $\in [min, max]$
2. Build a forest: build randomly many trees
3. The predicted value is the mean of the path length between the root and the leaf
4. Find outlier: the outliers have a shorter path length than the other observations



sklearn.ensemble.IsolationForest

- **n_estimators**, *default=100*
- **max_samples**
  - The number of samples to draw from X to train each base estimator.
  - If int, use max_samples
  - If float, use max_samples * n_samples
  - If “auto”, use max_samples=min(256, n_samples)
- **Contamination,** *default=‘auto’*
  - The amount of contamination of the data set, i.e. the proportion of outliers in the data set.
  - Used when fitting to define the threshold on the scores of the samples.
  - If ‘auto’, the threshold is determined as in the original paper.
  - If float, the contamination should be in the range (0, 0.5].
- **max_features**, *default=1.0*
  - The number of features used to fit each tree
  - If int, then use max_features features.
  - If float, then use max_features * n_features.

- fit(X[, y]), Fit estimator.
- predict([X]), Predict the labels (1 inlier, -1 outlier)

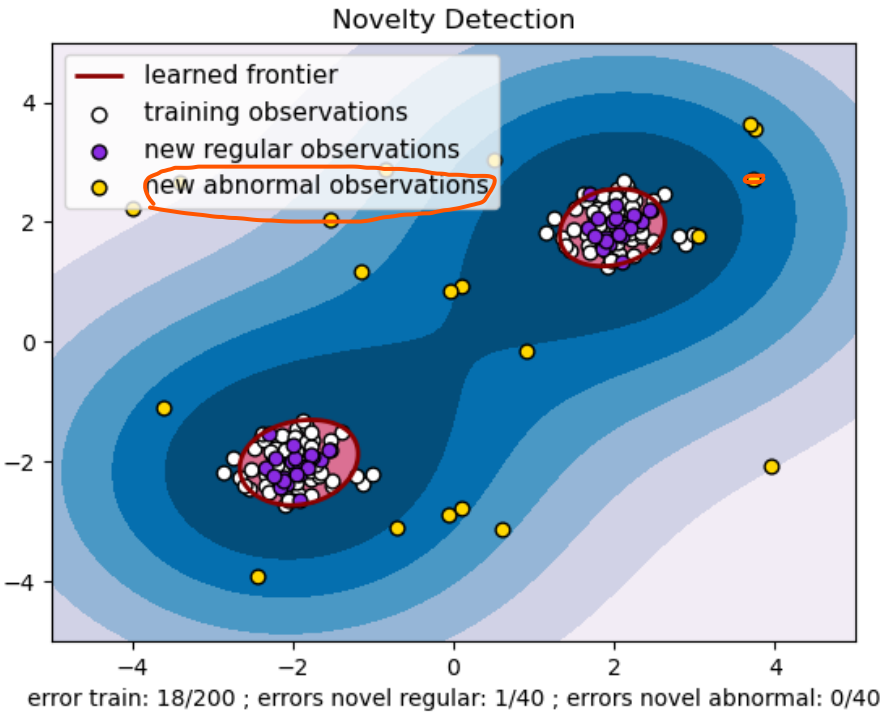
Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. “Isolation forest.” Data Mining, 2008. ICDM’08.

One-class SVM

- SVMs are max-margin methods,
  - SVM separates two classes using a hyperplane with the largest possible margin.
  - One-Class SVM uses a hypersphere to encompass all of the instances.
    - ”Margin" as referring to the outside of the hypersphere
    - "the largest possible margin", we mean "the smallest possible hypersphere".
  - Maps input data into a high dimensional feature space
  - Iteratively finds the maximal margin in the hyperplane which best separates the training data from the origin
    - The origin is the only original member of the second class i.e. outliers
  - Solves optimization problem to find rule f with maximal margin
    - $f(\mathbf{x})=\langle \mathbf{w},\mathbf{x}\rangle+b$
    - If $f(\mathbf{x})<0$, label $\mathbf{x}$ as anomalous

Bernhard Scholkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, Robert C. Williamson, "Estimating the Support of a High-Dimensional Distribution", Neural Computation, Volume 13, Issue 7, July 2001, pp. 1443–1471.

One-class SVM

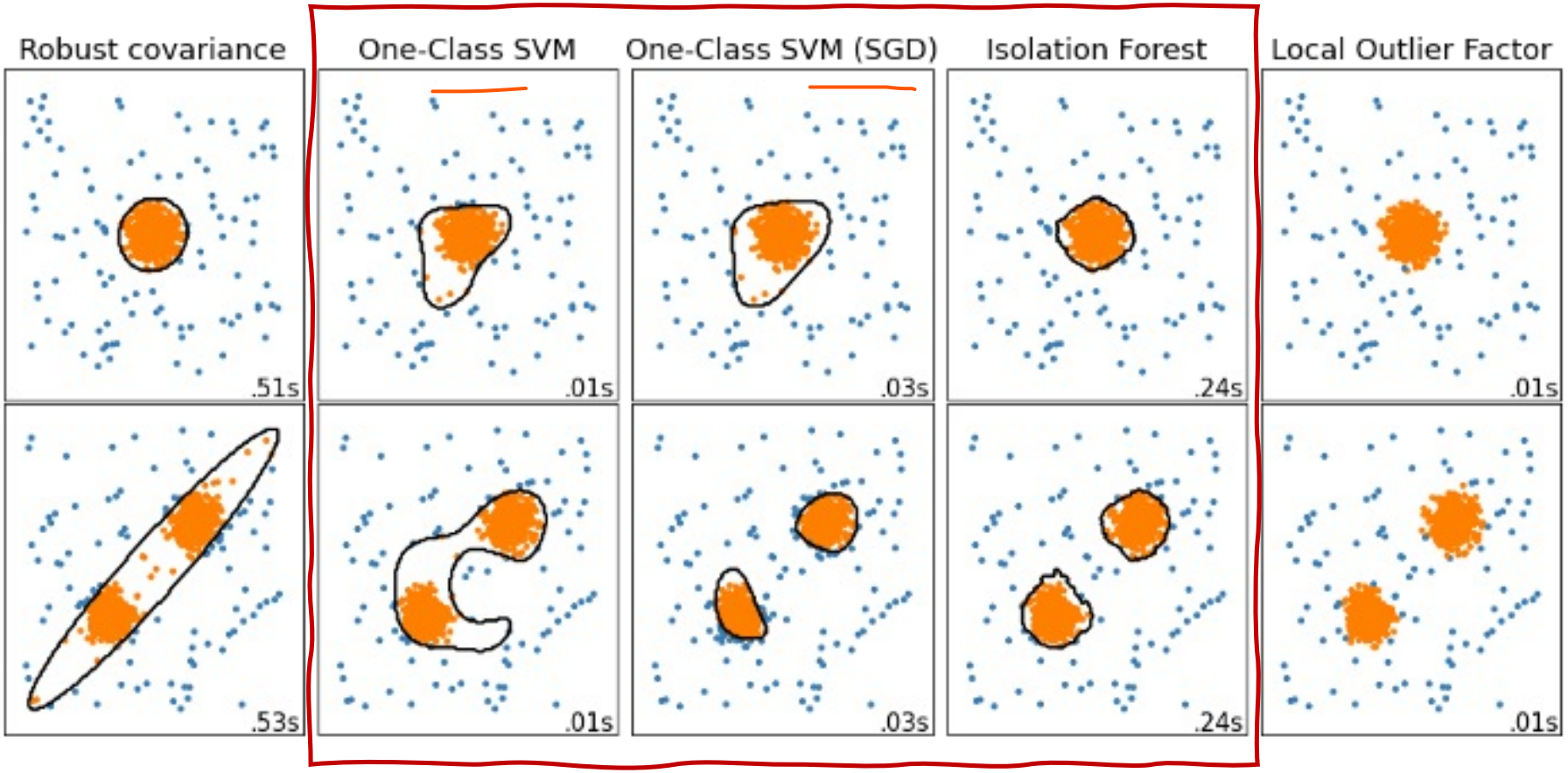


sklearn.svm.OneClassSVM

- **Kernel**, *default='rbf'*
  - Specifies the kernel type to be used in the algorithm. I
- **Gamma**, *default='scale'*
  - if gamma='scale' (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
  - if 'auto', uses 1 / n_features.
- **nu**, *default=0.5*
  - An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors.
  - Should be in the interval (0, 1].

- fit(X[, y]), Fit estimator.
- predict([X]), Predict the labels (1 inlier, -1 outlier)

Benchmarking



MISSING DATA IMPUTATION

- Missing data are observations that were originally planned but were not made

- Missing data can occur for many reasons:
  - Participants may not respond to questions (legitimately or illegitimately),
  - Collection or recording equipment malfunctioned,
  - Subjects may withdraw from studies before they are completed, data entry oversights may occur.

- Missing data indicator matrix (n x d)
  - $M = (m_{ij})$ with $m_{ij} = 1 \; if \; x_{ij}$ is missing, else 0
    - n: observations
    - d: features

Imputation strategy in python
Single imputation strategy

- **Impute missing data by a new one, based on « columns » observation**

- from sklearn.impute import SimpleImputer
  - **Strategy,** The imputation strategy.
    - “mean”, replace missing values using the mean along each column. Can only be used with numeric data.
    - “median”, replace missing values using the median along each column. Can only be used with numeric data.
    - “most_frequent”, replace missing using the most frequent value along each column. Can be used with strings or numeric data. If there is more than one such value, only the smallest is returned.
    - “constant”, replace missing values with fill_value. Can be used with strings or numeric data.
  -

Imputation methods in python
Multiple imputation strategy

- Modeling each variable with missing values as a function of other variables in a round-robin mode.
  - **MAR case:** missing data is random but there is a systematic relationship between it and the other data,

- from sklearn.impute import IterativeImputer
  - **Estimator:** estimator to use at each step of the round-robin imputation
  - **n_nearest_features:** number of other features to use to estimate the missing values of each feature column.
  - **initial_strategy:** Which strategy to use to initialize the missing values. Same as the strategy parameter in SimpleImputer.
  - **imputation_order**
    - 'ascending': From features with fewest missing values to most.
    - 'descending': From features with most missing values to fewest.
    - 'roman': Left to right.
    - 'arabic': Right to left.
    - 'random': A random order for each round.
-

Imputation methods in python
Multiple imputation strategy

Another solution: K_Nearest Neighbor Imputation

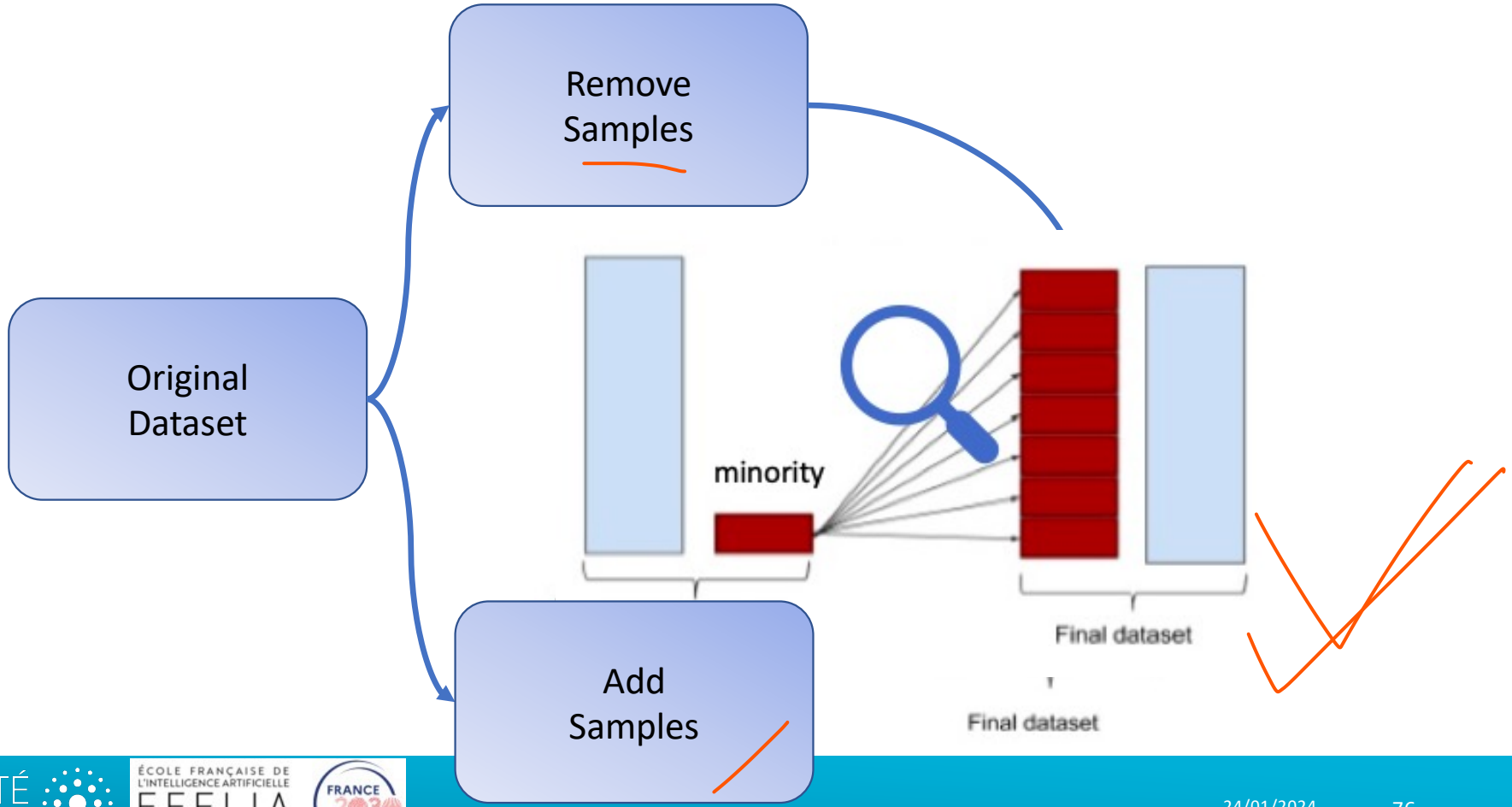
- The KNN algorithm helps to impute missing data by finding the closest neighbors
  - use Euclidean distance metric
  - Imputs missing data with non-missing values from neighbors.

- from sklearn.impute import KNNImputer
  - **n_neighbors:** Number of neighboring samples to use for imputation.
  - **Weights***:* Weight function used in prediction.
    - 'uniform' : All points in each neighborhood are weighted equally.
    - 'distance' : weight points by the inverse of their distance.

Unbalanced dataset

- The models tend to favor the majority class.
- If a gradient descent algorithm is used
  → the majority class dominates the gradient
- And in most cases, the likelihood is dominated by the majority class.
- It is therefore important to "counterbalance" these effects.
- The strategies focus on:
  - Modifying the dataset
  - Modifying the cost function
- They are generally not used at the same time.
- Whatever the approach, a suitable metric must be chosen

Dataset strategies



Basic dataset strategies
Random Under Sampling

- **Remove samples from the majority class: RUS [Kubat97]**
  - Sample the data records from majority class
  - Usually results in a general loss of information

```
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(sampling_strategy= ???)
X_res, y_res = rus.fit_resample(X, y)
```

- Sampling strategy
  - Float: specify the ratio $\frac{minority\ class}{majority\ class}$ (only for binary classification)
  - String: the number of samples in the different classes will be equalized
    - 'majority': resample only the majority class
    - 'all': resample all classes
    - 'auto'

Basic dataset strategies
Random Over Sampling

- **Over-sampling the rare class: ROS** [Ling98]
  - Make the duplicates of the rare events until the data set contains as many examples as the majority class => balance the classes
  - Does not increase information but increase misclassification cost

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(sampling_strategy= ???)
X_res, y_res = ros.fit_resample(X, y)
```

- Sampling strategy
  - Float: specify the ratio $\frac{minority\ class}{majority\ class}$ (only for binary classification)
  - String: the number of samples in the different classes will be equalized
    - **'minority'**: resample only the minority class
    - **'all'**: resample all classes
    - **'auto'**

Basic dataset strategies

Generating artificial observation

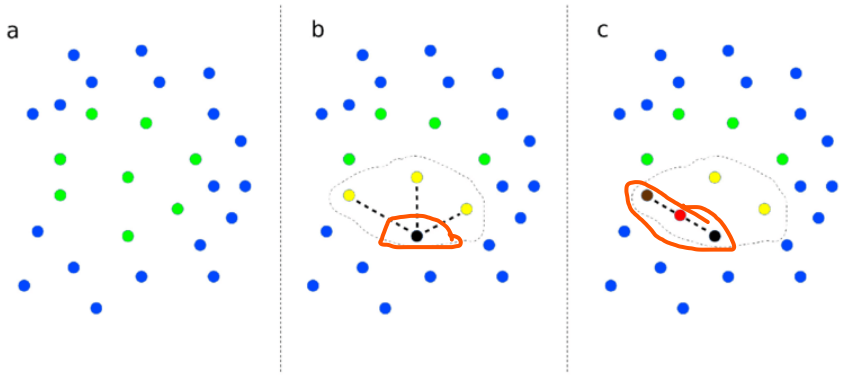
- SMOTE (Synthetic Minority Over-sampling TEchnique) [Chawla02]
  - New rare class examples are generated inside the regions of existing rare class examples

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy=???, k_neighbors=???)
X_res, y_res = smote.fit_resample(X, y)
```

- Sampling strategy
  - Float: specify the ratio $\frac{minority\ class}{majority\ class}$ (only for binary classification)
  - String: the number of samples in the different classes will be equalized
    - **'minority'**: resample only the minority class
    - **'all'**: resample all classes
    - **'auto'**
- Neighbor
  - number of nearest neighbors to used to construct synthetic samples

Chawla, SMOTE: Synthetic Minority Over-sampling Techniqu, 2002 (https://arxiv.org/abs/1106.1813)

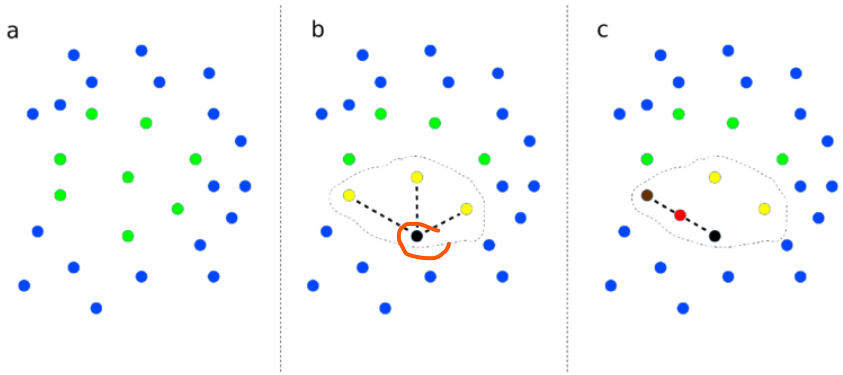
SMOTE



Supervised method

Starts ~~from a set of~~ positive (green) and negative (blue) points

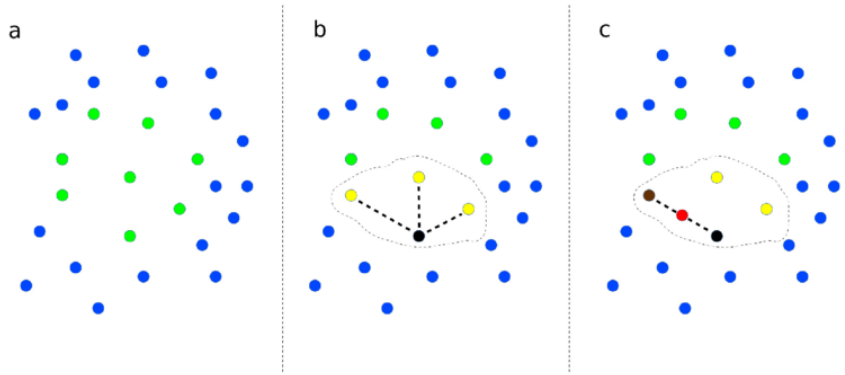
SMOTE



Selects a positive example (black)
Selects its k nearest neighbors among the positives (yellow, k = 3),

SMOTE

- Select one of the k nearest neighbours (brown)
- Add a new synthetic positive example (red), alond the straight line that connects the black and brown points.



Repeats for all the positives

- Adding each time a new synthetic example
- It's similar in an Euclidean sense

Algorithm strategies

- Class weights:
  - Instead of repeating samples (or removing samples)
    - Weigth the loss function
  - Works for most models:
    - logistic regression, SVM, Random Forest, Neural Network
  - Doesn't work for some models:
    - Knn, NB,
- Same effect as over-sampling (add samplers)
  - But not as expensive → dataset remain the same
  - Generally `class_weight` parameter of the model
    - 'balanced': adjust weights inversely proportional to class frequencies
    - {class_label: weight}
- Main advantage
  - Does not change the data and calculation time

Others algorithm strategies

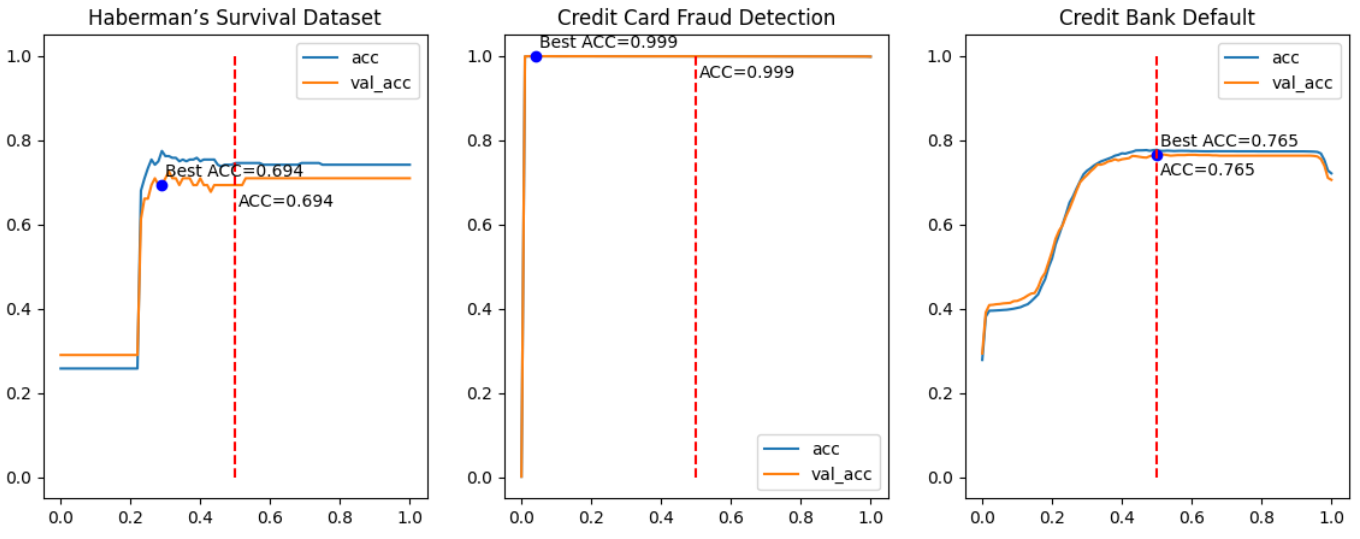
- **Not All Classification Errors Are Equal**
  - **Cancer Diagnosis Problem**:
    - A doctor wants to determine whether a patient has cancer or not.
    - It is better to diagnose a healthy patient with cancer and follow-up with more medical tests than it is to discharge a patient that has cancer.
    - The cost of a false positive might be the monetary cost of performing complementary test
    - **FN must be very small**
  - **Fraud Detection Problem**:
    - An insurance company wants to determine whether a claim is fraudulent.
    - Identifying a customer as fraudulent when they are not is damaging to the company's image
    - **FPs should be very small**
  - Use `class_weight` parameter in order to penalize more FP as FN

https://machinelearningmastery.com/cost-sensitive-learning-for-imbalanced-classification/

More on postprocessing

Threshold adjustment

- binary-class problem
  - clf.predict(X) = np.argmax(model.predit(X), axis=1) →[0, 1]
  - clf.predict_proba(X) → [0.7, 0.3], [0.4, 0.6]
- But ACC (or F1 score) varies according to the threshold
  - Select best threshold on X_train
  - Predict on X_test



Threshold adjustment with F1 score

- Select best threshold on X_train
- Predict on X_test