IMEN891K / IDSC501 / AXGS-501 · DATA ENGINEERING · FALL 2026

5.

Pre-processing II: Transformation & Reduction

Sangwoo Mo · POSTECH

Industrial & Management Engineering

Where We Are

PART I · Collection and Preparation | **PART II** | **PART III**

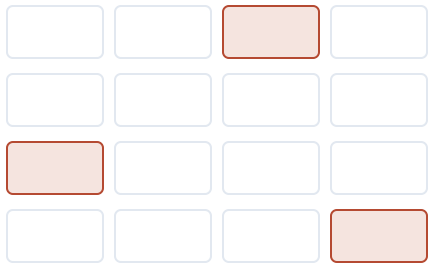


preprocessing changes the representation; some choices also remove information

Recap: Four Stages of Preprocessing

01

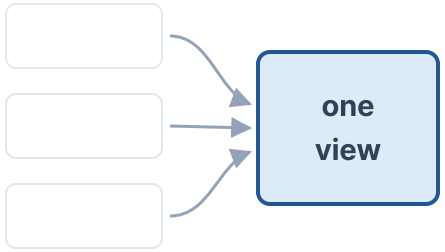
Cleaning



Wrong values in one table

02

Integration



Tables that disagree

03

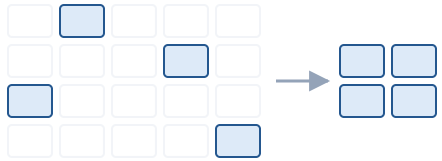
Transformation



Values a model can read

04

Reduction



Less data, same signal

Transformation

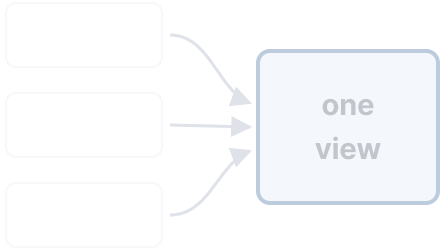
01

Cleaning

Wrong values in one table

02

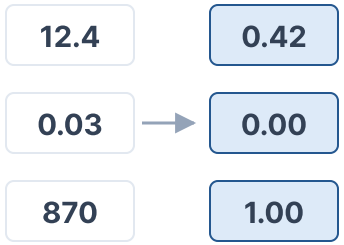
Integration



Tables that disagree

03

Transformation



Values a model can read

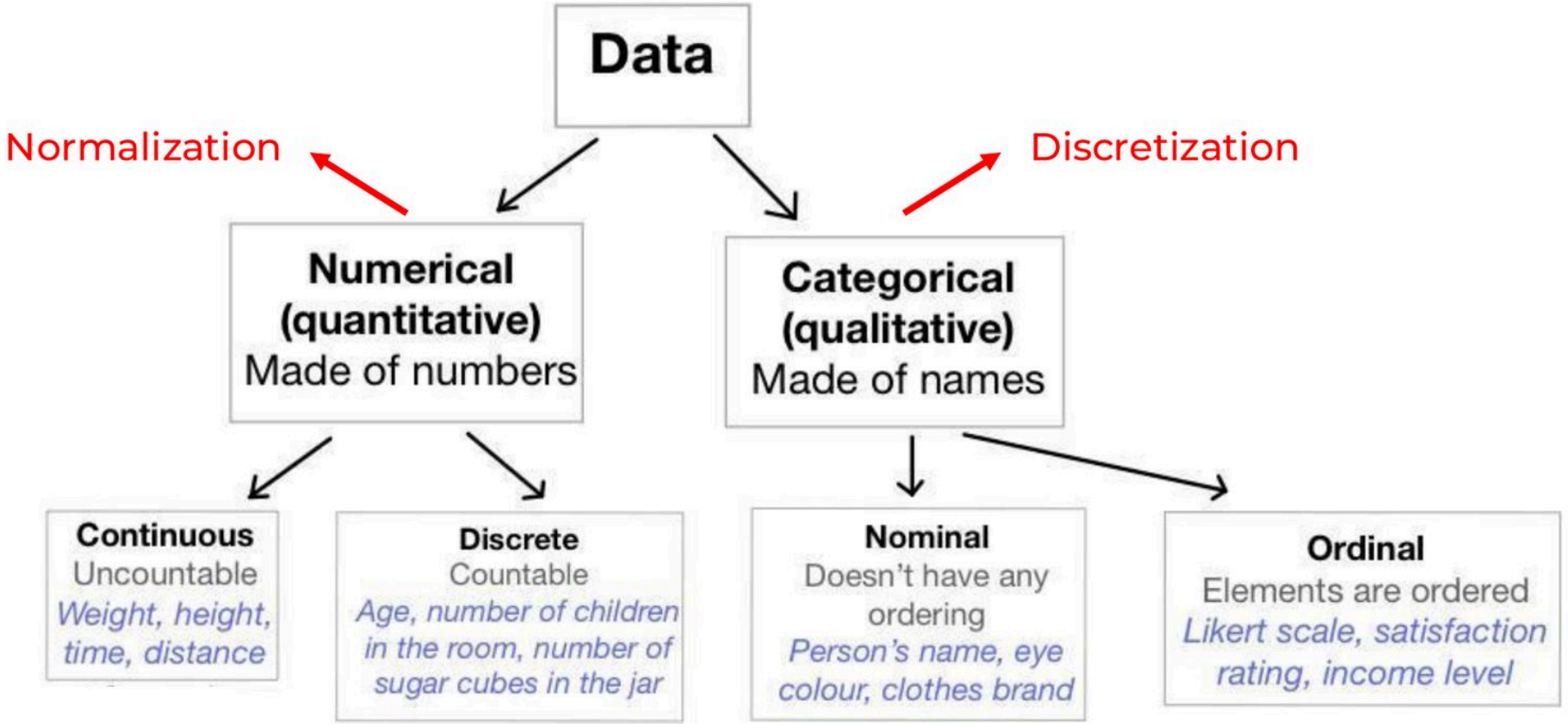
04

Reduction

Less data, same signal

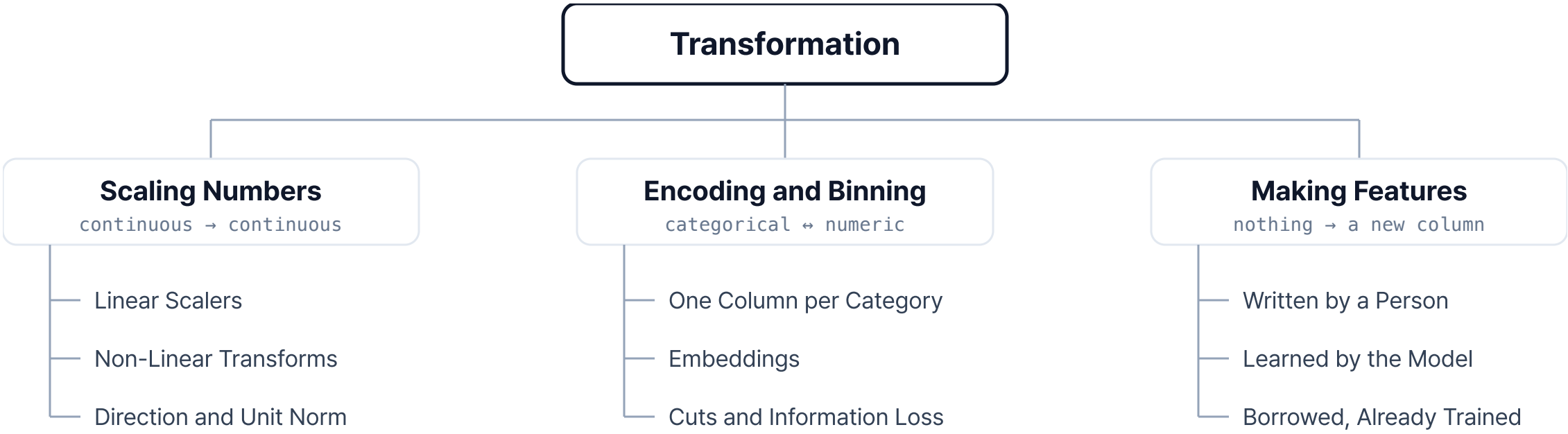
Recap: Data Types

Numbers get rescaled; discretization turns numbers into categories

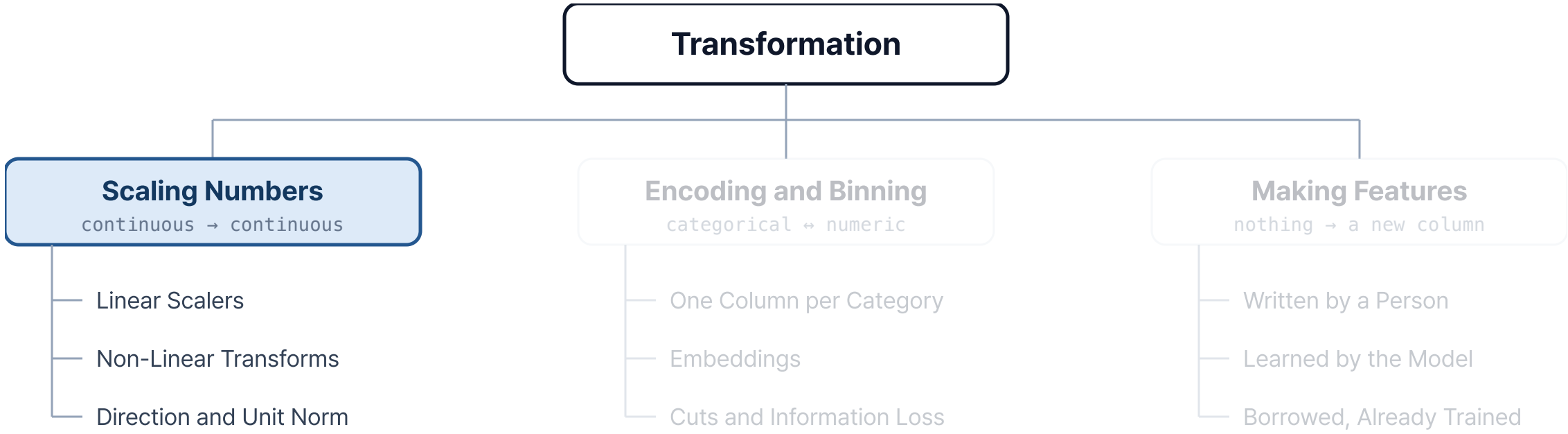


The type of a column **decides the tool**

Three Branches of Transformation



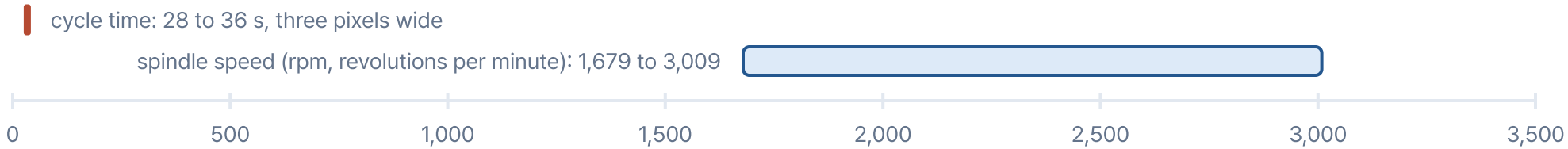
Scaling Numbers



Why Scale at All

Distance-based models add up raw units, so the widest column decides

two columns from one machine, on one raw axis



how far apart two rows can be, column by column



In raw Euclidean distance, the rpm difference **dominates** the cycle-time difference

The Evaluation Must Match Deployment

Define what will be new at prediction time before splitting the records

Deployment question	Useful split	What a random row split can hide
Predict future events	Earlier train; later test	Future information reaches training
Predict new machines	Hold out whole machines	Same-machine patterns occur on both sides
Predict new customers	Hold out whole customers	Repeated identity makes the test easier

A held-out row can still share an entity or future information with training

Caveat: One Machine on Both Sides

Random rows

Machine	Train	Test
M1	80 sampled cycles	20 held-out cycles
M2	80 sampled cycles	20 held-out cycles

New-machine evaluation

Machine	Role
M1	Train
M2	Test

Testing new machines requires machines that training has never seen

Fitted Transformation State

Fit learns parameters; transform reuses the saved parameters

Step	Fit on train	Transform new rows
Scaling	min, max or mean, SD	reuse the same numbers
Encoding	category vocabulary	reuse the same columns
PCA	center and axes	project onto the same axes

Fit once on training rows, then apply the saved state

Linear Scalers

Three scalers, one shape: subtract a center, divide by a spread

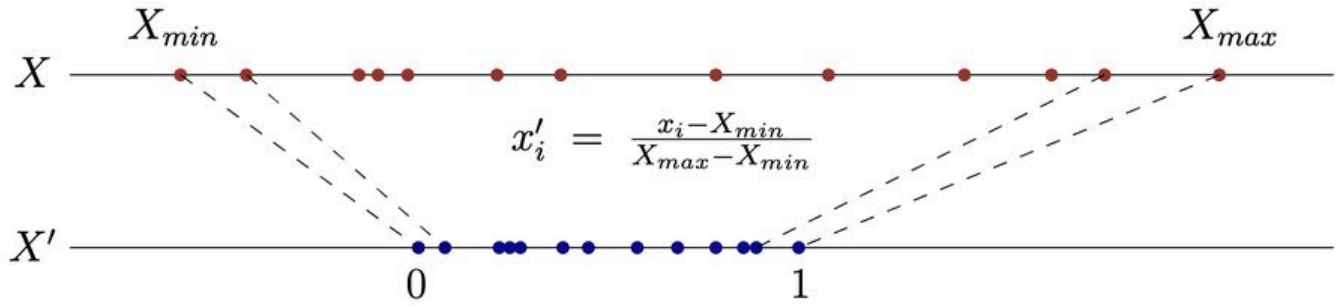
SCALER	THE SAME FORMULA, TWO ESTIMATORS	BREAKDOWN POINT
Min-max	(x − **min**) / **max − min**	0%
Standardization	(x − **mean**) / **standard deviation**	0%
Robust	(x − **median**) / **interquartile range**	25%

Picking a scaler is picking how many bad rows it takes to move the answer

Min-Max Scaling

x′ = (x − min) / (max − min) maps the training range onto [0, 1]

$$x_i' = \frac{x_i - X_{min}}{X_{max} - X_{min}}$$



The minimum maps to 0, the maximum to 1. **Fit both on train only**

Demo: Fit on Train, Apply to Test

Train cycle times stay at 28, 32 and 36 seconds

1. Can a transformed test value exceed 1? 2. Does refitting change the training rows?

Test value 40 s include test when fitting

Split	Raw seconds	Transformed
train	28	0.000
train	32	0.500
train	36	1.000
test	40	1.500

fit rows: train only · min 28 · max 36 · train 36 becomes 1.000

Answer: Reusing a Scaler

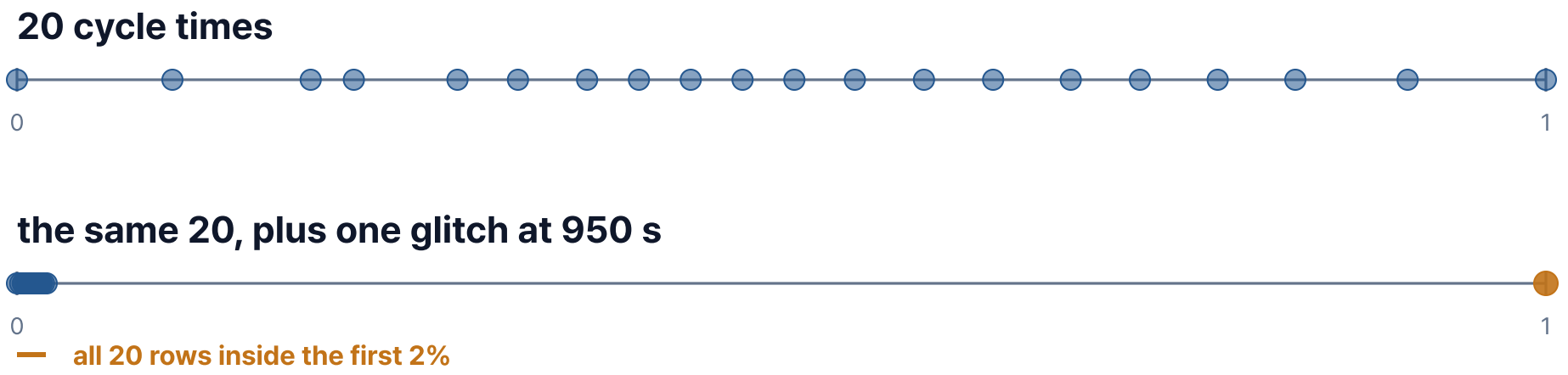
At test = 40, the saved training range produces 1.5

Fitting rows	min / max	train 36 →	test 40 →
Train only	28 / 36	1.000	1.500
Train + test	28 / 40	0.667	1.000

New observations can lie outside the fitted training range

Caveat: Min-Max and Outliers

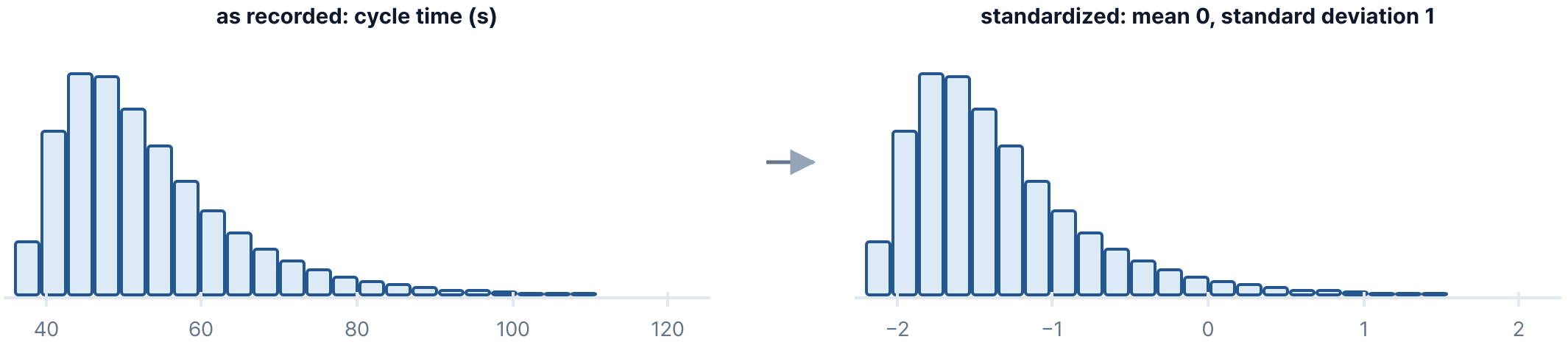
One outlier is enough to squash every other value



The maximum defines the range, so the maximum defines everything

Z-Score Standardization

Each training column gets mean 0 and standard deviation 1



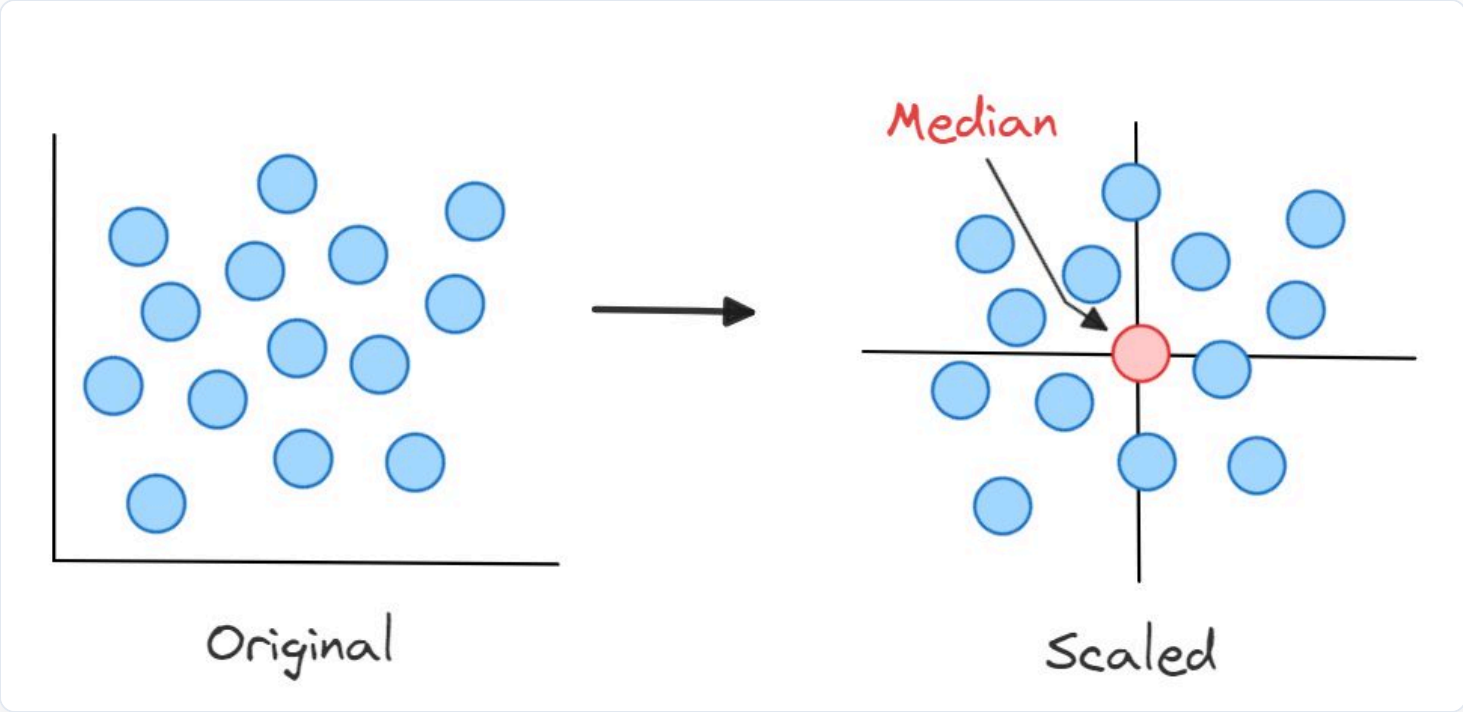
the bars are identical; only the numbers under them changed

$$z = (x - \mu) / \sigma$$ the shape is untouched; only the axis is relabeled

A right tail stays a right tail. **Standardizing does not make a column normal**

Robust Scaling

The median and the IQR barely move when one row goes wrong



$$x' = (x - \text{median}) / \text{IQR}$$ IQR = interquartile range

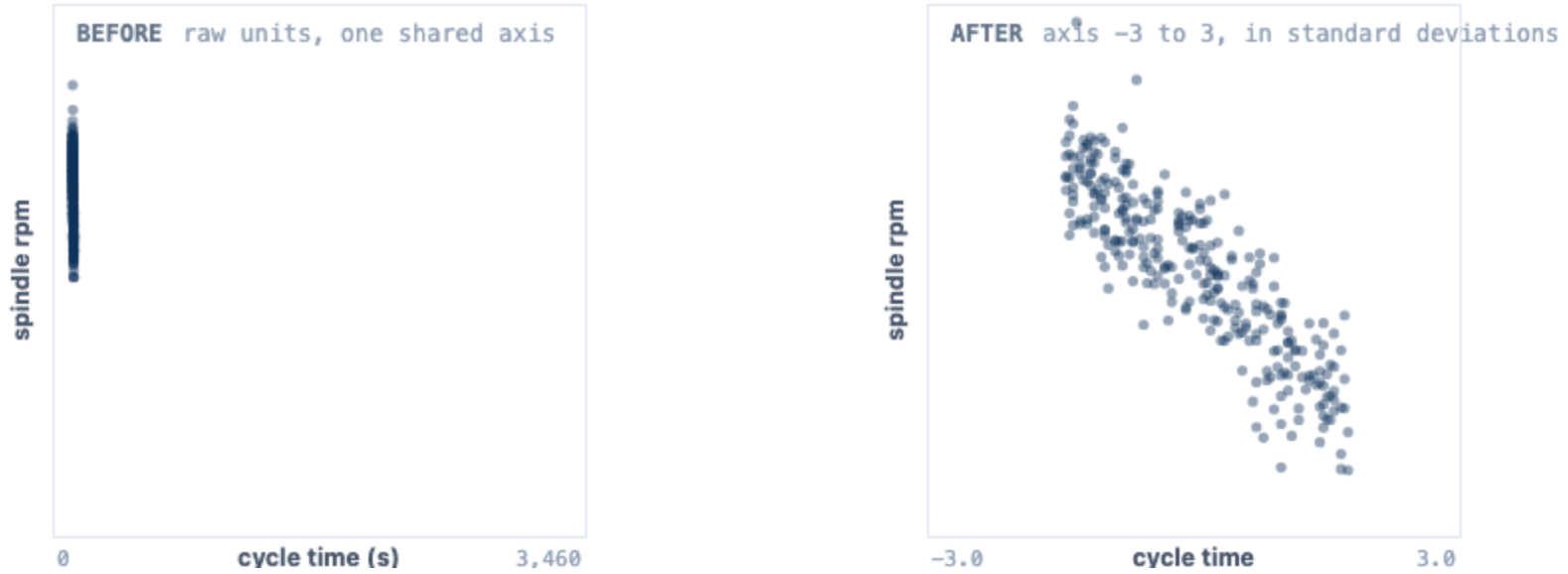
The median moves to zero and **the outliers stay far out. That is intended**

Demo: Feature Scalers

300 cycles from one computer numerical control (CNC) machine, both columns on one axis

1. Compare scalers with a mistyped rpm. 2. Did scaling repair the error or change the reference statistics?

Scaler StandardScaler ☐ one mistyped rpm (2,451 → 24,510)



raw: rpm spans **166×** the cycle-time range · after StandardScaler, the rows fill **89%** of the axis

StandardScaler is the default. RobustScaler when outliers are real. MinMaxScaler only when the bounds are known

both columns now on one scale

Answer: Scaling and Outliers

Scaling changes the reference statistics, not the recorded event

Scaler	Reference statistics	Effect of an extreme value
MinMax	min and max	clean values compress into a narrow range
Standard	mean and SD	center and spread can move
Robust	median and IQR	resists some extremes; keeps the outlier

Scaling changes representation; it does not validate or remove a value

Log Transform

Log turns ratios into equal steps

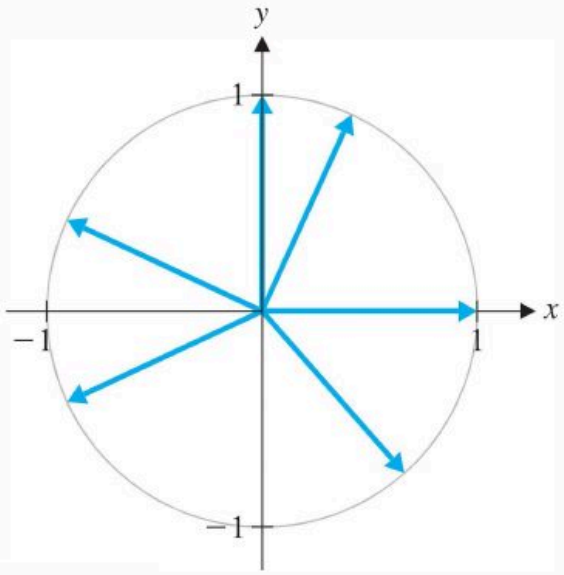


$$\log_2(2x) - \log_2(x) = 1 \quad x > 0$$

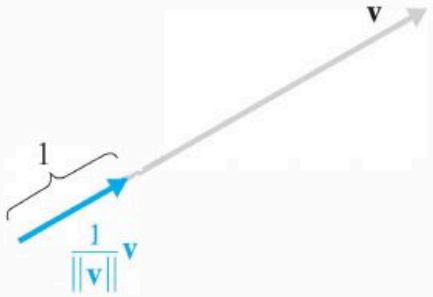
Each doubling becomes the **same gap**. Normality is not guaranteed

Unit Vector Normalization

Scale each row, not each column, to length 1



Unit vectors in $\mathbb{R}^2$

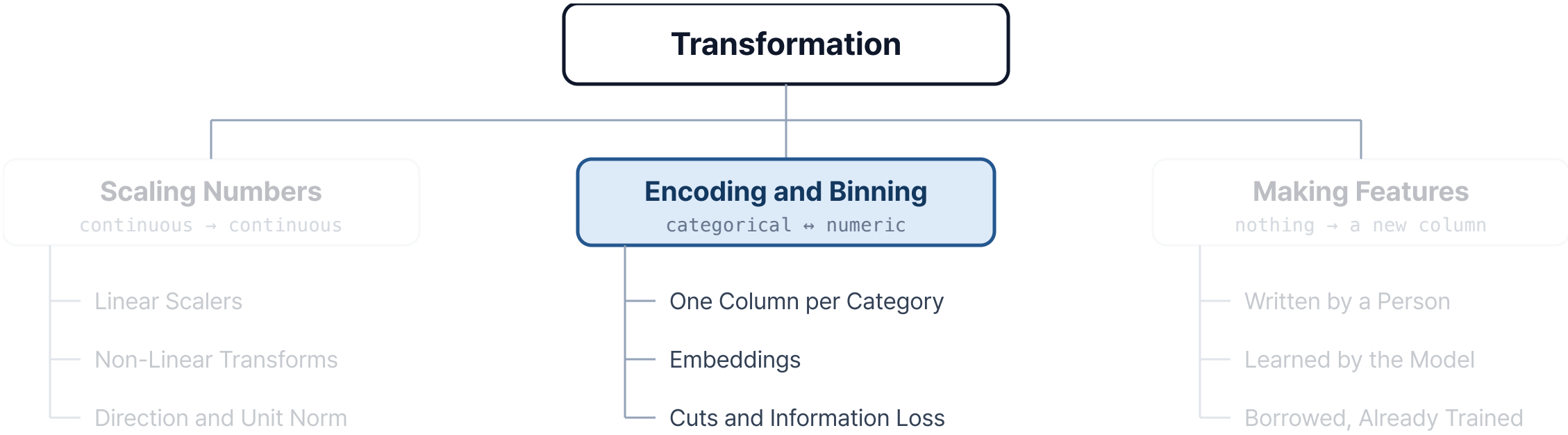


Normalizing a vector

$x' = x / \|x\|_2$ then $x'^T y'$ is exactly the cosine similarity

Every row lands on the unit circle; **the dot product is now cosine similarity**

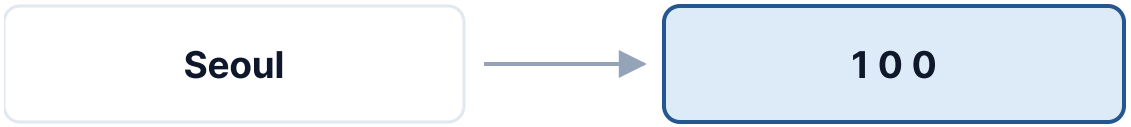
Encoding and Binning



Encoding vs Binning

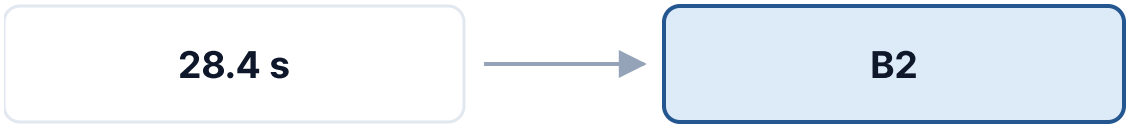
One turns categories into numbers, the other turns numbers into categories

ENCODING a model cannot multiply a name



one-hot · label · count · binary · embedding

BINNING a continuous column rarely repeats a value



equal width · equal frequency · clustering

What the model can use decides which way you cross

Nominal and Ordinal Categories

A numeric code can introduce order and spacing into the model

Nominal: no known order

Color	Arbitrary code
red	1
green	2
blue	3

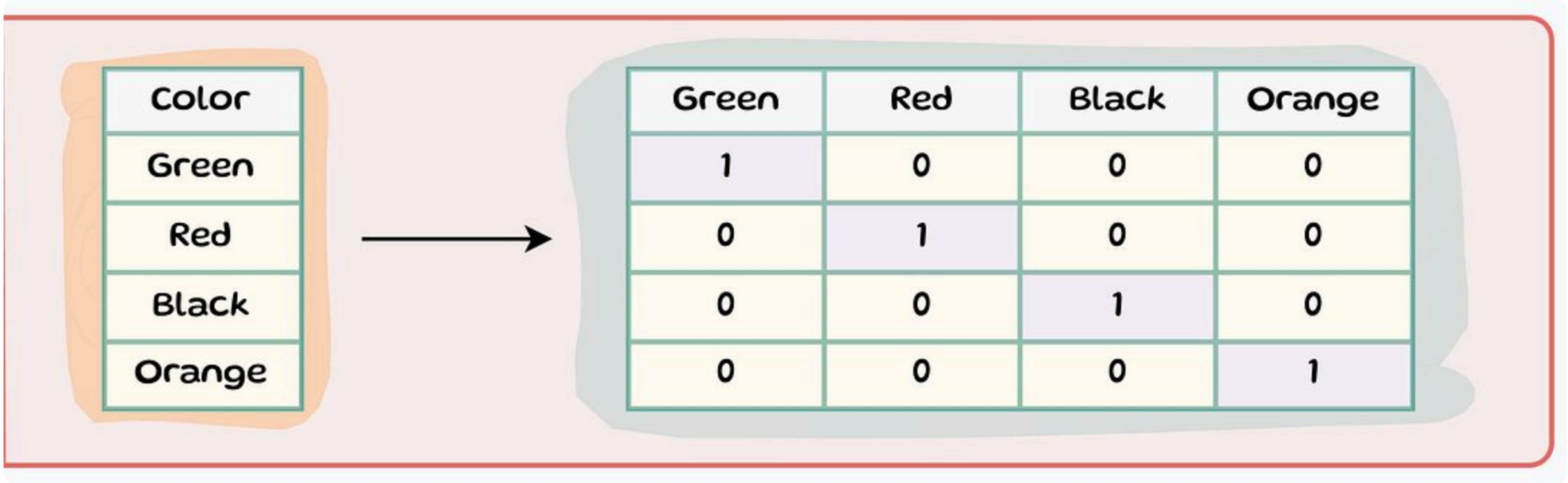
Ordinal: a known order

Size	Ordered code
small	1
medium	2
large	3

Color codes invent an order; size codes express an existing order

One-Hot Encoding

One indicator column per category, so no order is invented



Color
Green
Red
Black
Orange

Green	Red	Black	Orange
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

$$x \to \left(\mathbb{1}[x = c_1], \ldots, \mathbb{1}[x = c_k] \right)$$ k categories become k columns whose sum is always 1

With an intercept, **k one-hot columns are linearly dependent**. Dropping one removes this dependency

Choosing an Encoding

Order, the number of categories and the model decide the encoding

DATA MEANING	STARTING CHOICE	CHECK
Unordered categories	One-hot	Width and unknown values
Known ordered levels	Ordinal codes	The supplied order matches the meaning
Text or many related categories	Learned embedding	Vocabulary, model version and validation

Start with the category meaning, then check width and held-out behavior

Demo: Categorical Encoders

Six training rows; reuse their category vocabulary for test

1. Compare one-hot with integer codes. 2. Apply the fitted vocabulary to an unseen test color.

Encoder: One-hot

Width estimate: total categories: 4

☐ add unseen test color: Blue

BEFORE

COLOR
Green
Red
Black
Orange
Green
Red

AFTER · fitted one-hot · unknown → zeros

BLACK	GREEN	ORANGE	RED
0	**1**	0	0
0	0	0	**1**
1	0	0	0
0	0	**1**	0
0	**1**	0	0
0	0	0	**1**

with **4** distinct categories: one-hot **4** columns · binary **2** · label / count **1**

no order invented, 4 columns for 4 categories

Six training rows stay fixed; Blue is test only. The slider estimates width

Answer: An Unseen Category

A fitted encoder needs an explicit policy for a new value

Encoder policy	Blue at test time	Consequence
One-hot, ignore unknown	[0, 0, 0, 0]	keeps four fitted columns
Integer codes, unknown = -1	-1	unknown remains distinct
Add a test-only Blue column	five columns	changes the fitted representation

Keep the fitted vocabulary; define how unknown values are represented

Vector Embedding

A learned model maps each category to a short real vector



d(Seoul, Busan) = 0.35 · d(Seoul, Tokyo) = 0.83

Distances now differ between pairs. **In one-hot, every pair of categories is equally far apart**

An Encoding Cannot Restore a Binned Value

Binning loses within-bin differences; encoding represents the resulting category

Measured number

```
cycle time = 28.4 s
```

→

Bin category

```
B2: [25 s, 30 s)
```

→

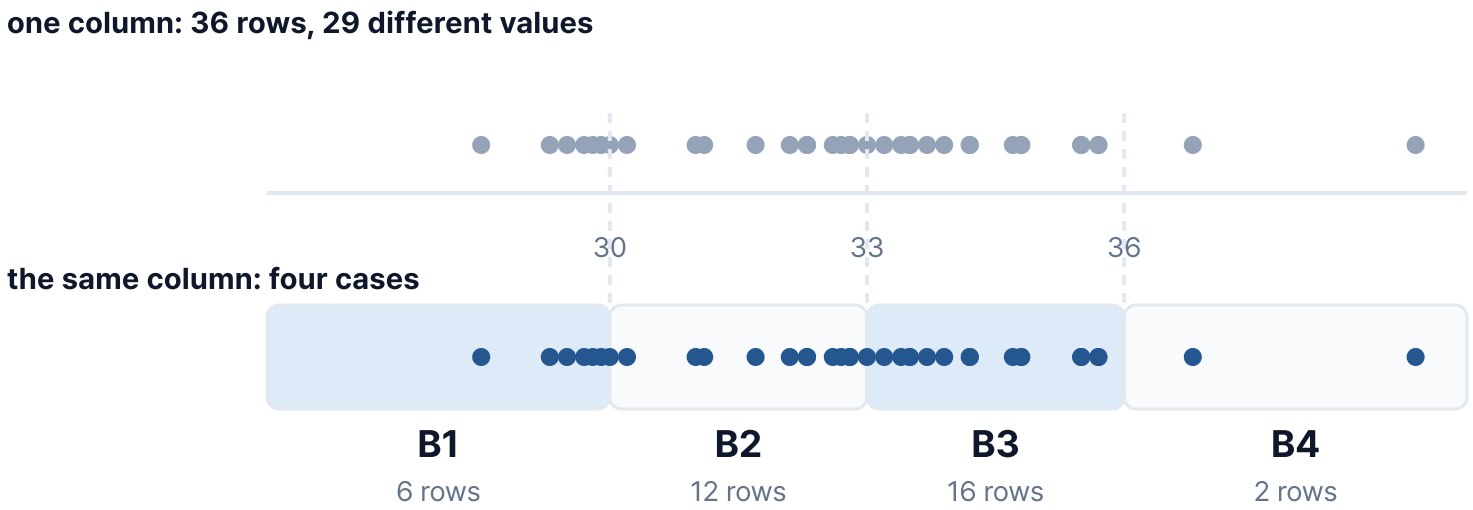
One-hot representation

```
B1 B2 B3
 0  1  0
```

A numeric output does not imply that the original measurement was preserved

Why Discretize

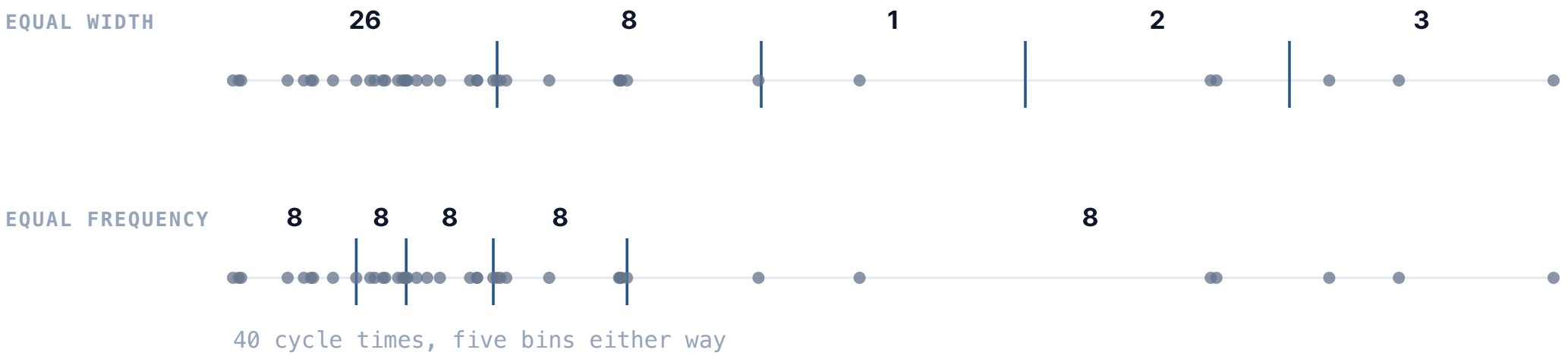
Continuous values give almost every row its own case



Four bins replace 29 values. **Where the cuts go is the whole question**

Binning by Histogram

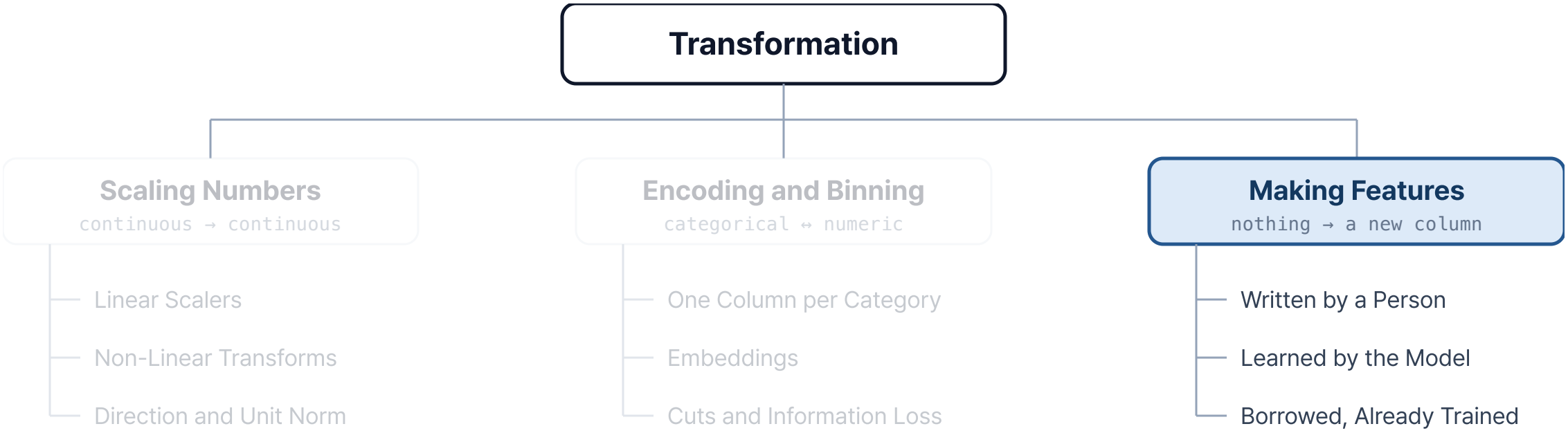
Equal width keeps the scale, equal frequency keeps the counts



`width = (max − min) / k` vs `count = n / k` the same k, two different things held constant

Equal width: **26, 8, 1, 2, 3 rows**. Equal frequency: 8 in every bin

Making Features



Hand-Crafted Features

No column holds it, so you compute one: TF-IDF (term frequency × inverse document frequency)

TERM IN ONE MAINTENANCE LOG	TF	DF OF 1,000 LOGS	LOG(N/DF)	WEIGHT
the	12	1,000	0.00	0.0
pump	4	300	1.20	4.8
cavitation	2	12	4.42	8.8

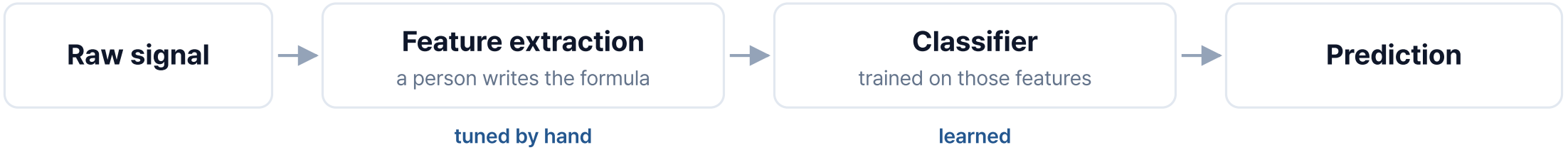
$$w = tf \cdot \log(N / df)$$ N = 1,000 logs · df = how many of them contain the term

The most frequent word scores zero. A person decided that is the right behavior

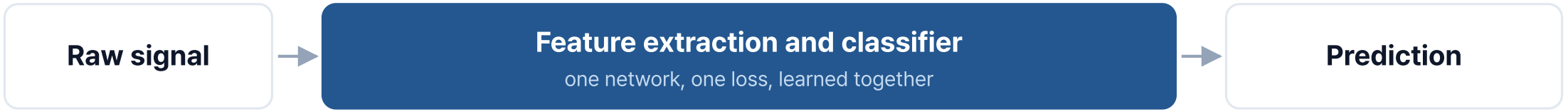
Learned Features

The network learns the features and the classifier together

CLASSIC MACHINE LEARNING



DEEP LEARNING



Fit your own encoder or reuse a versioned pretrained encoder

Case study: YouTube-8M

The dataset ships features, not videos



Feature extraction is also compression, **and it is irreversible**

Reduction

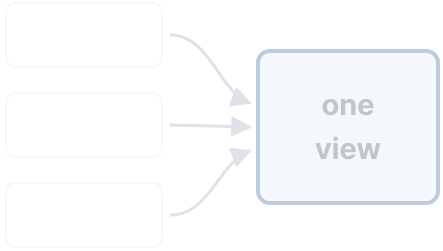
01

Cleaning

Wrong values in one table

02

Integration



Tables that disagree

03

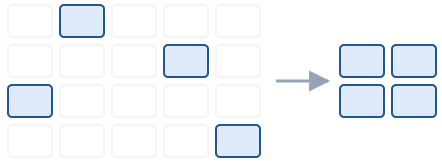
Transformation



Values a model can read

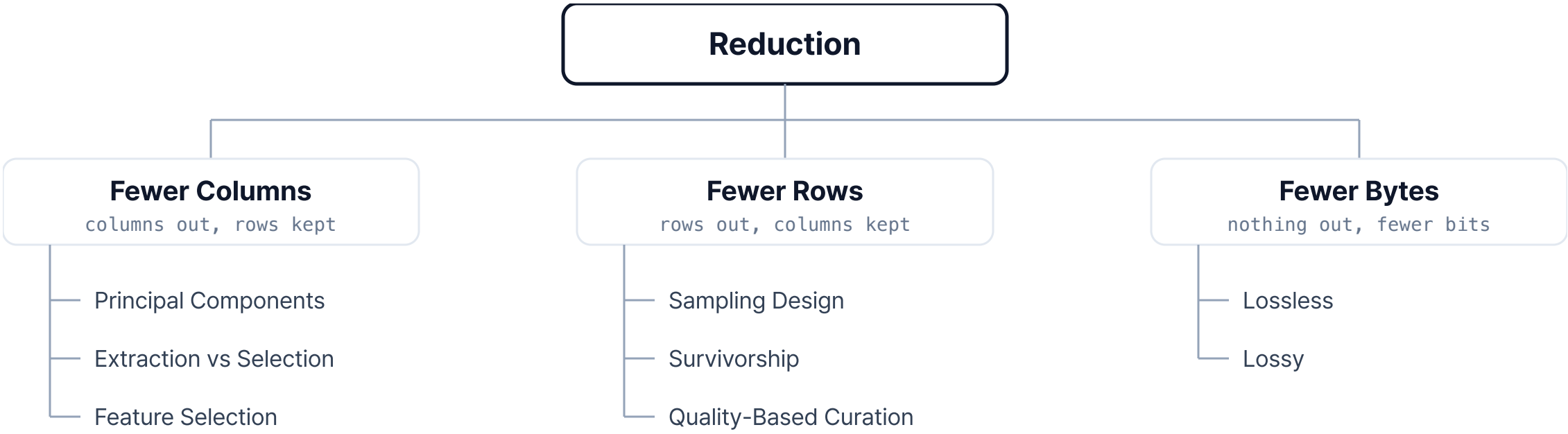
04

Reduction



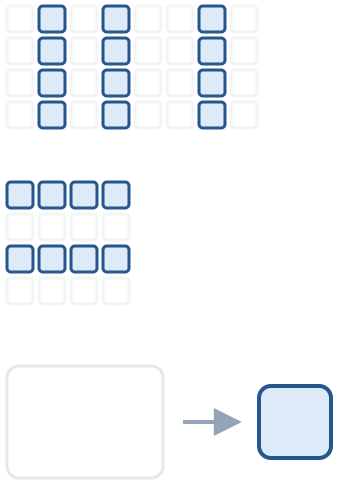
Less data, same signal

Three Branches of Reduction



Why Reduce at All

Columns, rows and bytes each cost something different



Fewer columns

Distances stop separating and the model fits noise

Fewer rows

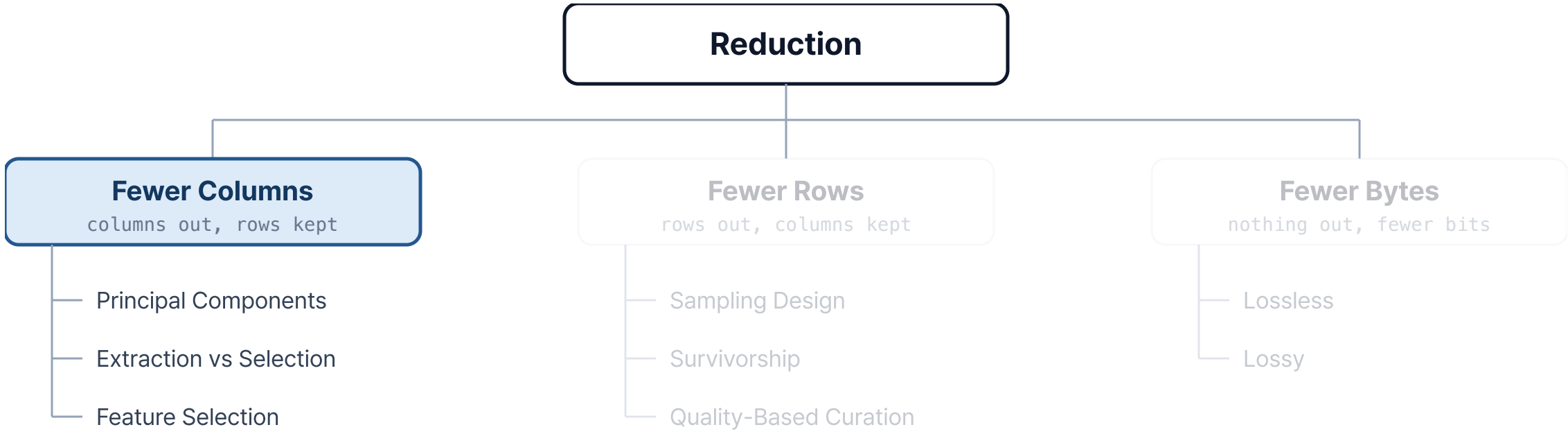
You pay to store, move and label every one of them

Fewer bytes

Storage and transfer are a bill that arrives every month

Removing rows or columns changes learning; lossless compression preserves the data

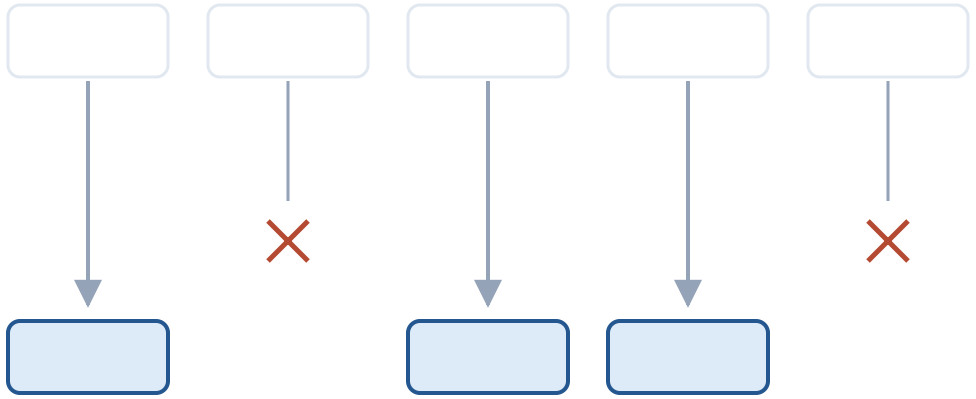
Fewer Columns



Extraction or Selection

Keep some original columns, or mix all of them into new ones

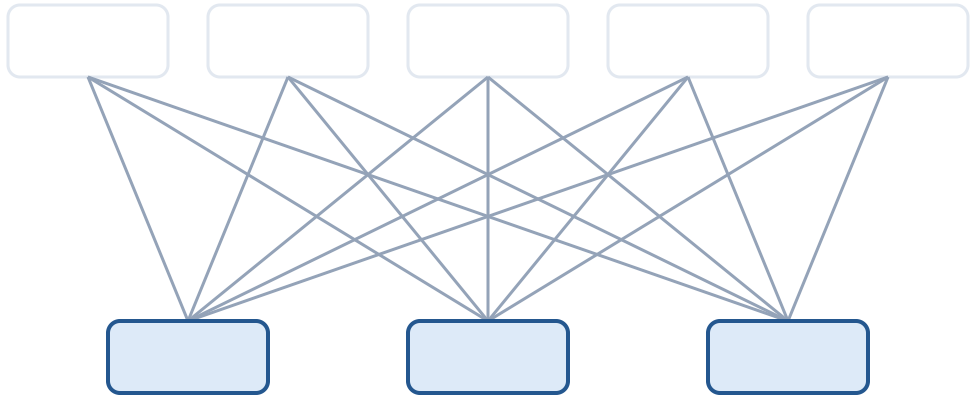
SELECTION keep some of the original columns



5 columns in, 3 of the same columns out

filter · wrapper · embedded

EXTRACTION make new columns from all of them



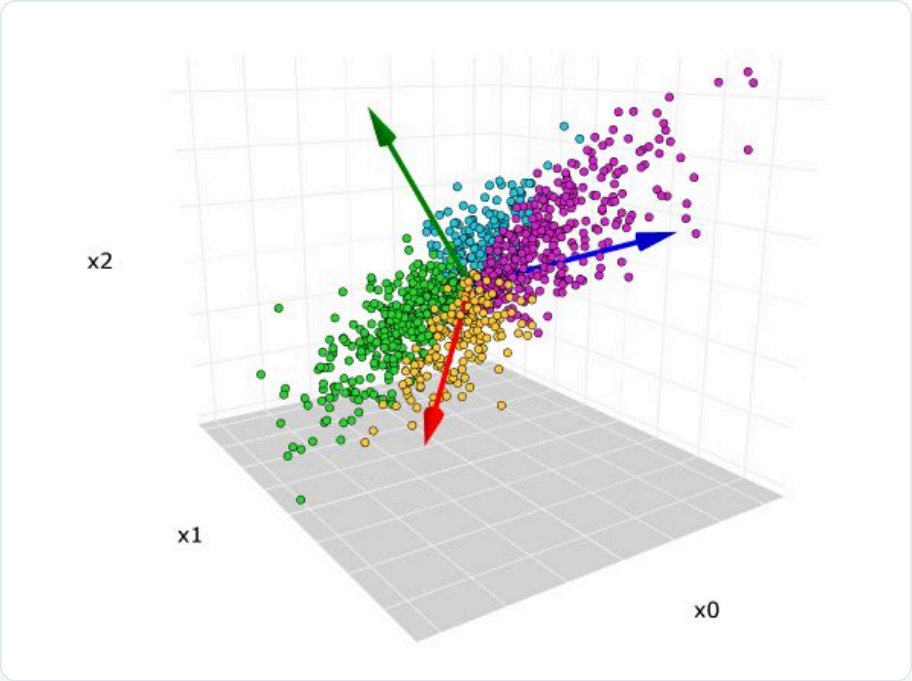
5 columns in, 3 new columns out

principal components (PCA) · random projection

Selection keeps names you can read. **Extraction keeps information you cannot name**

Recap: PCA as a Fitted Transform

Reuse the training center and axes when new observations arrive



$$z_{new} = (x_{new} - \mu_{train})\, V_{train,k}$$

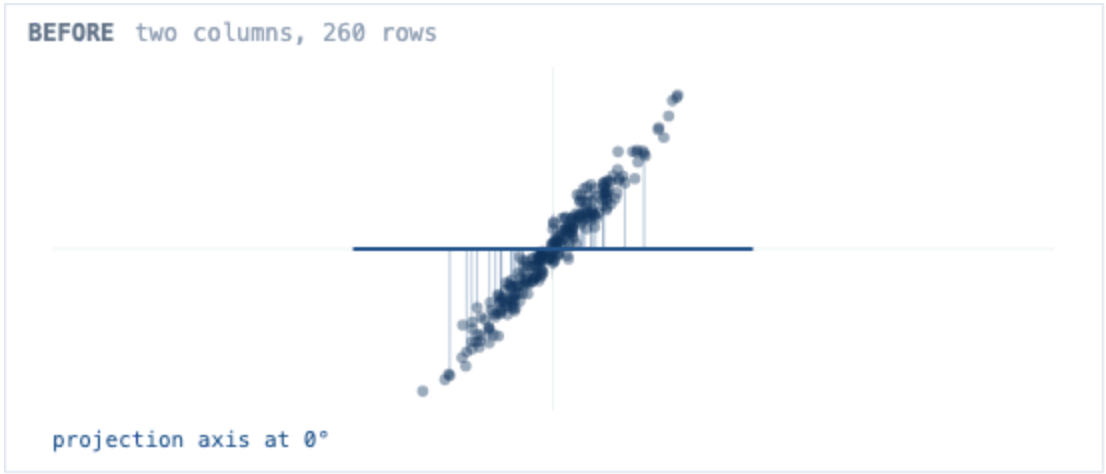
Projection keeps selected directions and discards the rest

Demo: PCA Projection Angle

Turn the axis and watch how much of the variation survives

1. Find a direction that keeps more variance. 2. Is retained variance enough to choose model features?

Projection angle 0° Use PCA





the wider these spread, the more of the original difference survives

angle **0°** keeps **43%** of the variation · PCA picks **49°** → **99%**

Center first; scale if units should count equally. Choose the component count using validation as well as retained variance

56 percentage points less variance than PCA

Answer: What PCA Keeps

Large retained variance does not guarantee useful predictive features

Question	Check
What survives?	Variation along the chosen training axes
What disappears?	Variation in discarded directions
Does the model still work?	Evaluate with held-out observations

Validate the retained representation, not only its variance

Three Families of Feature Selection

Compare model use and relative fitting cost within one selection procedure

FAMILY	WHAT IT LOOKS AT	MODEL FITS	BLIND SPOT
Filter	the columns alone, no model	none	a pair that only works together
Wrapper	the model’s score on a subset	many	the search itself overfits
Embedded	the model’s own coefficients	one	the answer is tied to that model

Compare cost, interactions and held-out performance before choosing a selection method

Wrapper Feature Selection

A wrapper compares feature subsets through repeated model fits

Candidate feature subset

```
temperature +
vibration
```

→

Fit and validate

```
Train on training
folds
Score on validation
folds
```

→

Compare another subset

```
temperature + current

Repeat within the same
protocol
```

Repeated subset search can overfit the validation evidence

Caveat: Features That Work Only Together

A univariate score can miss a predictive feature interaction

Four synthetic observations

valve A	valve B	defect
0	0	0
0	1	1
1	0	1
1	1	0

One column alone

A = 0: one pass, one defect

A = 1: one pass, one defect

A alone cannot separate them

The pair together

A ≠ B → defect

A = B → pass

Useful combinations can look uninformative when each column is scored alone

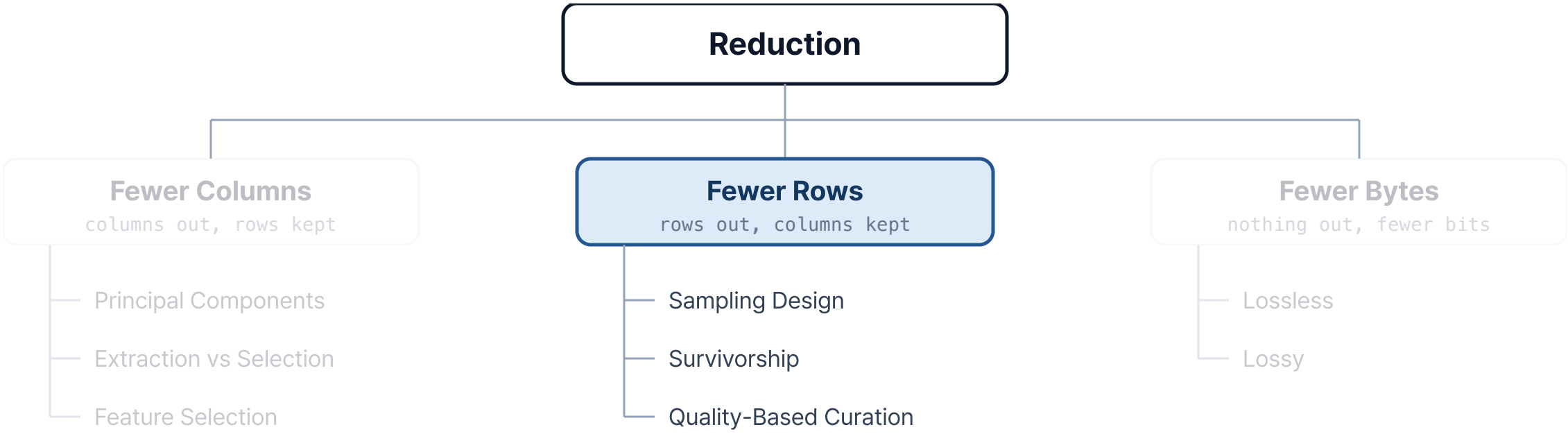
Embedded Methods

Some models select features during fitting; the result depends on the model and training data

Feature	Illustrative fitted coefficient	Interpretation
Vibration	0.8	Used by this fitted model
Temperature	0.3	Used by this fitted model
Humidity	0.0	Removed by this fitted model

A zero coefficient describes one fitted model, not the absence of a real relationship

Fewer Rows


Reduction
Fewer Columns
columns out, rows kept
Principal Components
Extraction vs Selection
Feature Selection
Fewer Rows
rows out, columns kept
Sampling Design
Survivorship
Quality-Based Curation
Fewer Bytes
nothing out, fewer bits
Lossless
Lossy

Sampling vs Curation

Both drop rows: sampling follows a design, curation follows quality

	SAMPLING	CURATION
Goal	A smaller set with known proportions	A better set: quality, consistency, coverage
Techniques	Random, systematic, stratified, cluster	Filtering for quality and coverage
Used for	Less computation, class imbalance	Training sets for large models, labeling budgets
Proportions	Kept, or changed on purpose (imbalance)	Changed by what the filters remove

Sampling decides **which rows to keep**; curation decides **how good they must be**

Sampling Design

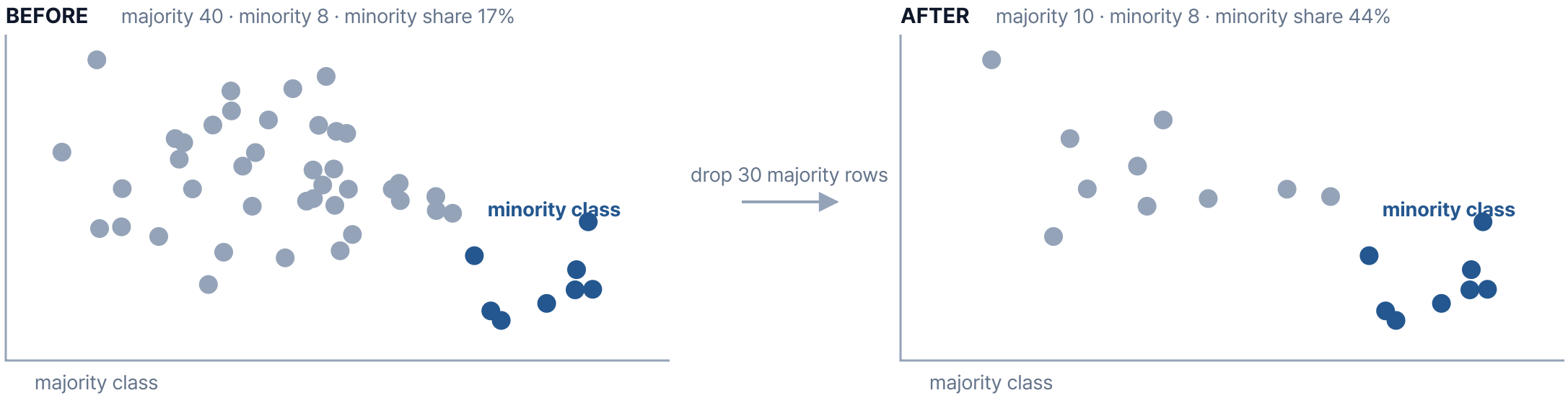
The selection rule determines which parts of the population survive

Method	How rows are chosen	Failure to check
Simple random	Draw rows at random	Small groups may be missed
Systematic	Every kth record	A periodic list can align with k
Stratified	Sample within each group	Keep the intended group proportions
Cluster	Select whole groups	One factory may not represent the rest

Check the population structure before choosing the sampling rule

Sampling to Fix Imbalance

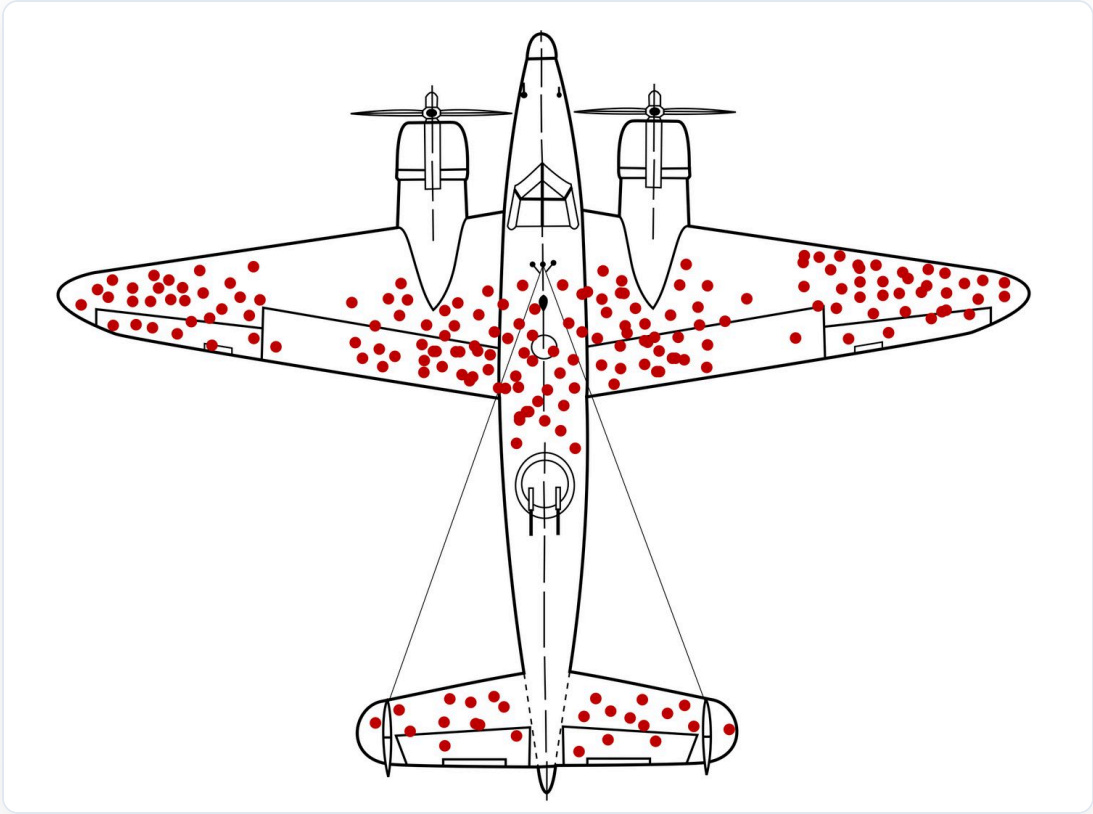
Dropping majority rows raises the minority share from 17% to 44%



Resampling changes class prevalence. **Evaluate and calibrate on the deployment distribution**

Caveat: Survivorship Bias

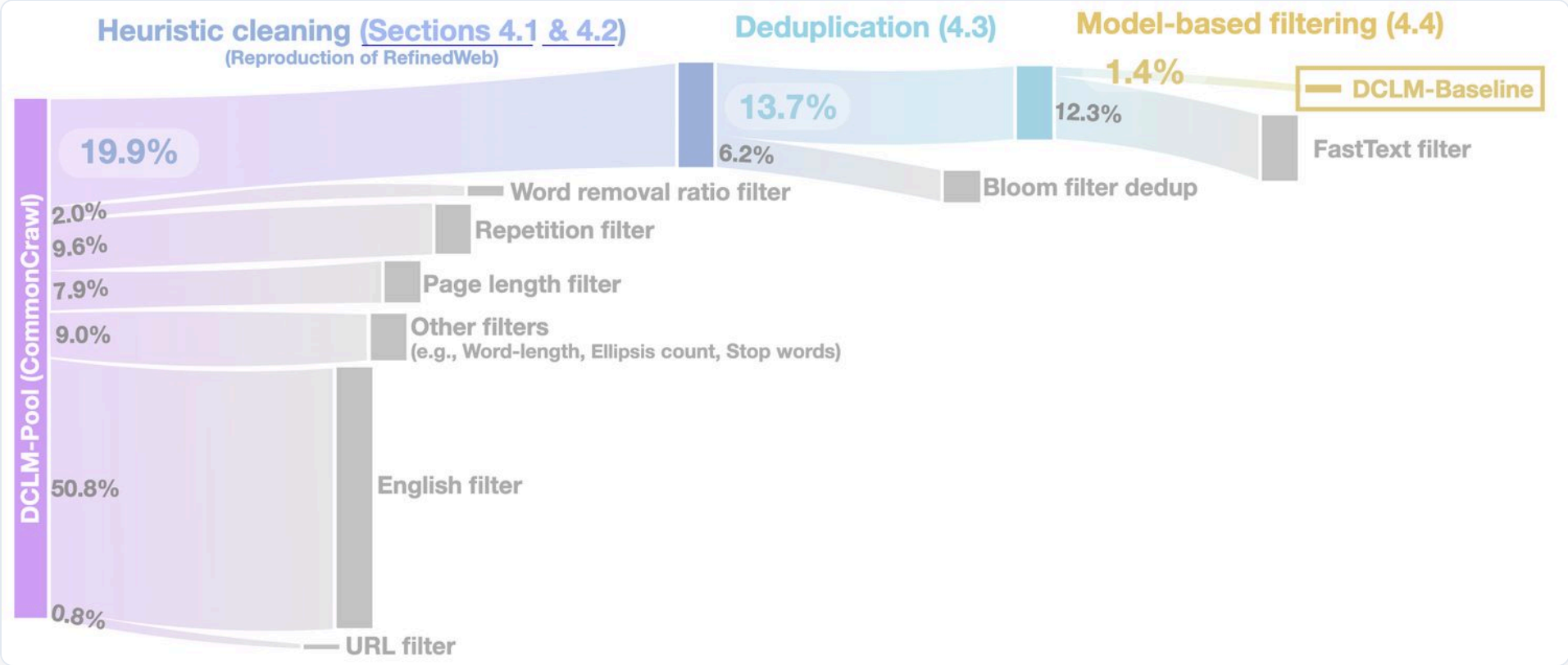
The rows you kept are not the rows you lost



This shows the bombers that came back. **Reinforce where there are no dots**

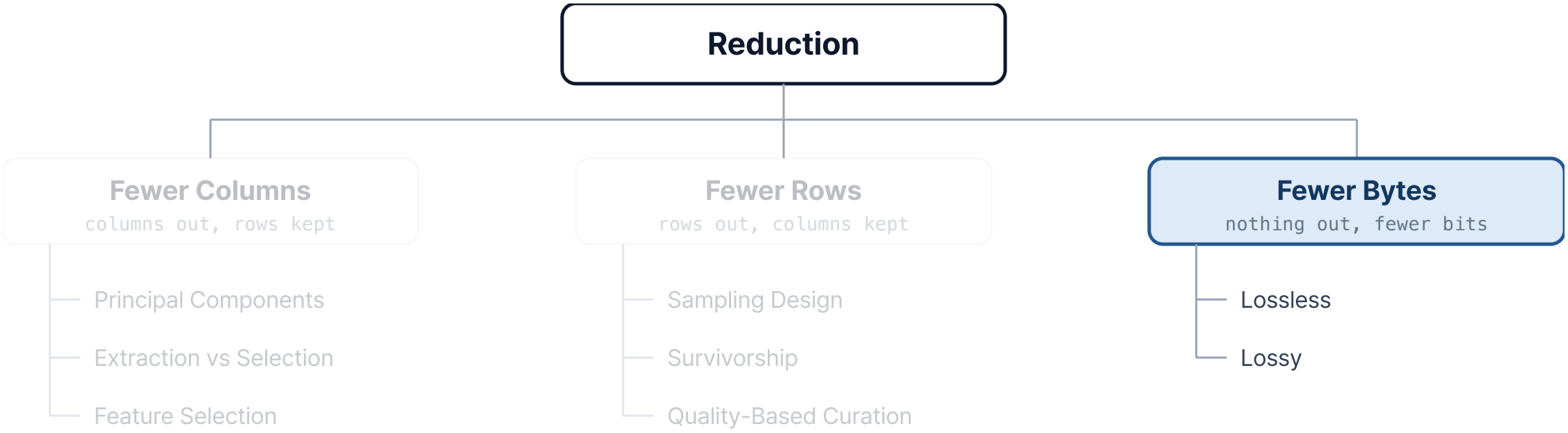
Data Curation

Curation filters for quality instead of sampling at random



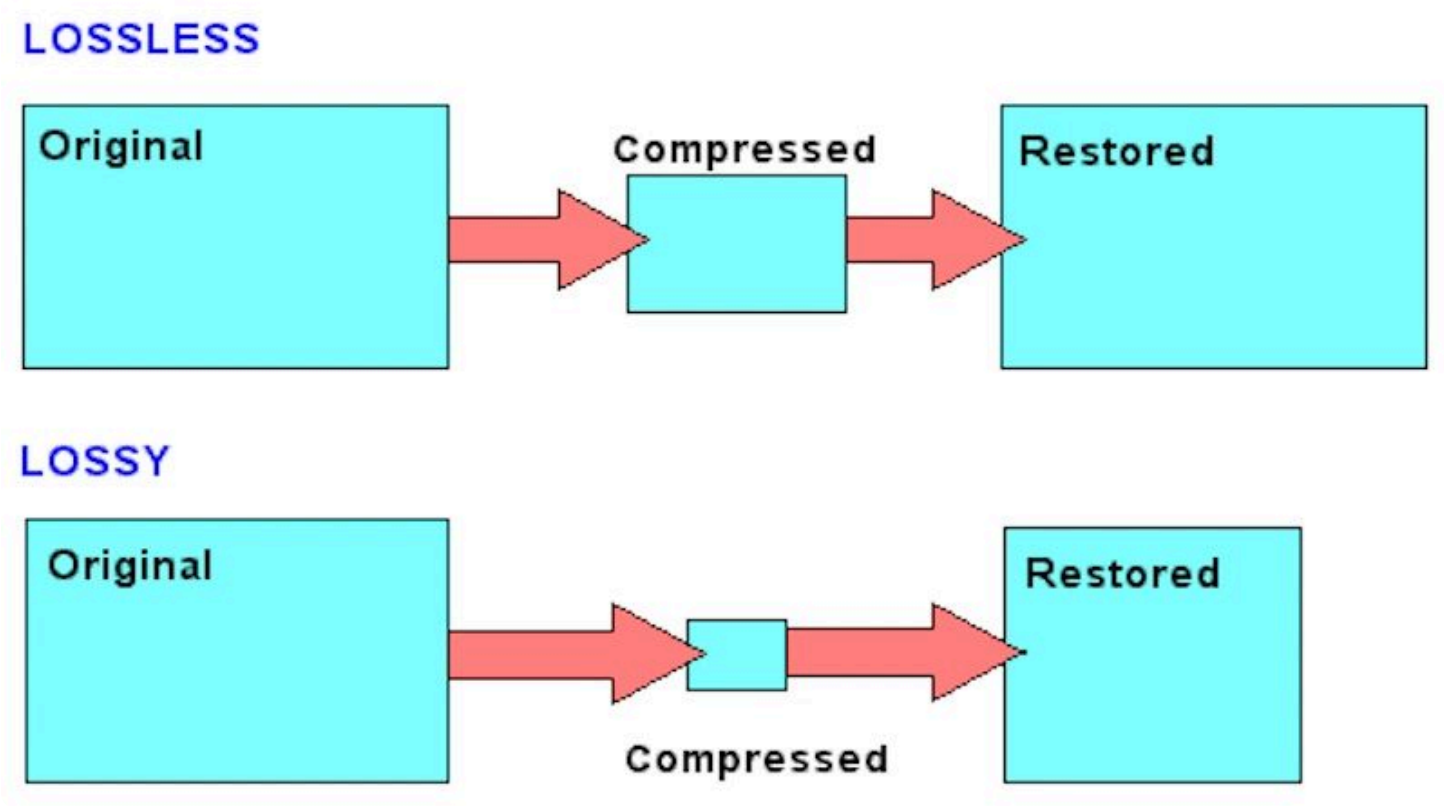
This pipeline keeps **1.4% of the crawl** as the training set

Fewer Bytes


Reduction
Fewer Columns
columns out, rows kept
Principal Components
Extraction vs Selection
Feature Selection
Fewer Rows
rows out, columns kept
Sampling Design
Survivorship
Quality-Based Curation
Fewer Bytes
nothing out, fewer bits
Lossless
Lossy

Lossless or Lossy

Compression restores the original bit for bit, or only an approximation



The lossy box is **smaller**, and **its changes stay**

Lossless Master Copy

Keep one lossless copy, because a lossy decision cannot be revisited

	LOSSLESS	LOSSY
Restores	The original, bit for bit	An approximation
Typical data	Text, code, logs, tabular	Images, audio, video
Formats	ZIP, PNG, GZIP, Parquet with Snappy or Zstd	JPEG, MP3; lossy H.264 / WebP
Ratio	Limited by entropy	Often much higher, tunable
The catch	Sometimes not small enough	The decision cannot be revisited

Text and logs stay lossless; **media is where lossy pays**

Fitting Preprocessing Steps

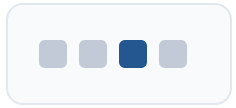
Fit on train, apply to test: every fitted step remembers something from the rows it saw

STEP	WHAT IT REMEMBERS	WHAT LEAKS IF YOU FIT ON EVERYTHING
Imputation	the column median	test rows move the median
Scaling	mean and standard deviation	the test spread enters every train row
Encoding	the list of categories	a category that exists only in test
Binning	the bin edges	edges drawn around test values
Feature selection	the columns kept	columns kept because they help on test
PCA	the components	axes rotated to fit test variance

$$z = (x - \mu_{train}) / \sigma_{train}$$ the same two numbers go to test and to every future row

Refit inside **every fold**; ship the final fit to testing and serving

Takeaway: Every transform and reduction is a decision



A scaler or an encoding decides what the model may compare

The center, the spread and any order you invent all end up inside the model



Know which reductions you can undo

Lossless compression restores; dropped rows and quantized values do not



Every fitted step carries state

Split for deployment, fit on train, and reuse the saved state