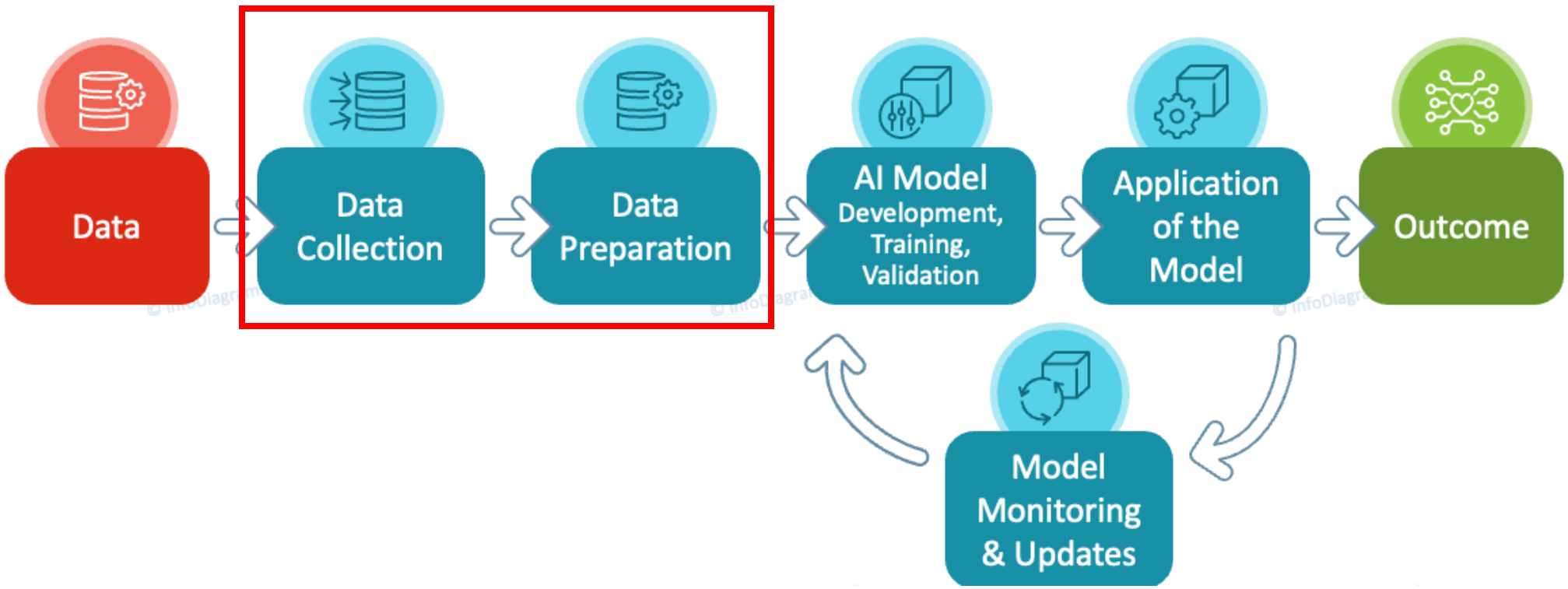
IMEN891K/IDSC501 Data Engineering

5. **Pre-processing II**: Transformation

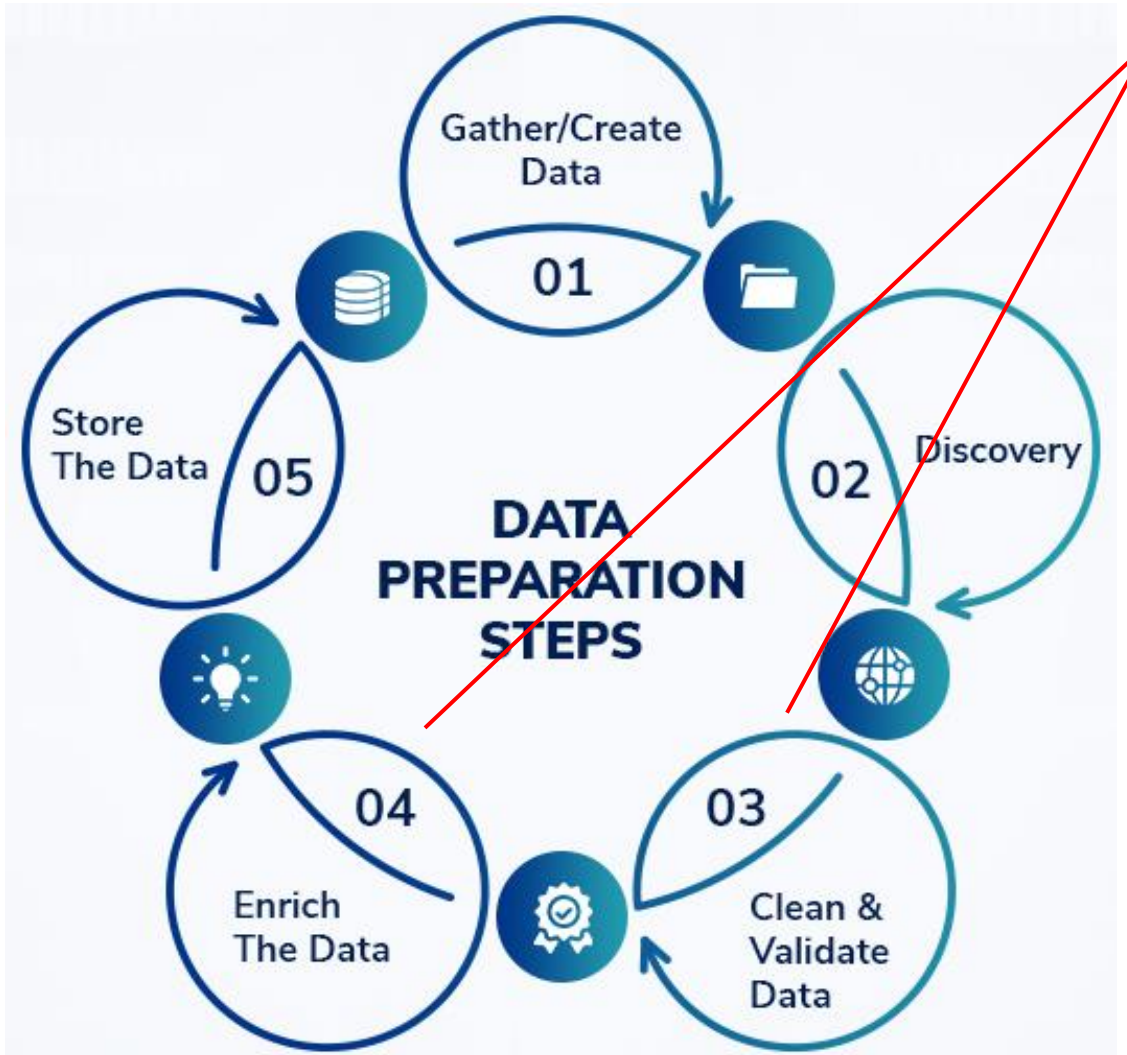
Fall 2025

Instructor: **Sangwoo Mo**

Recap: Course Overview

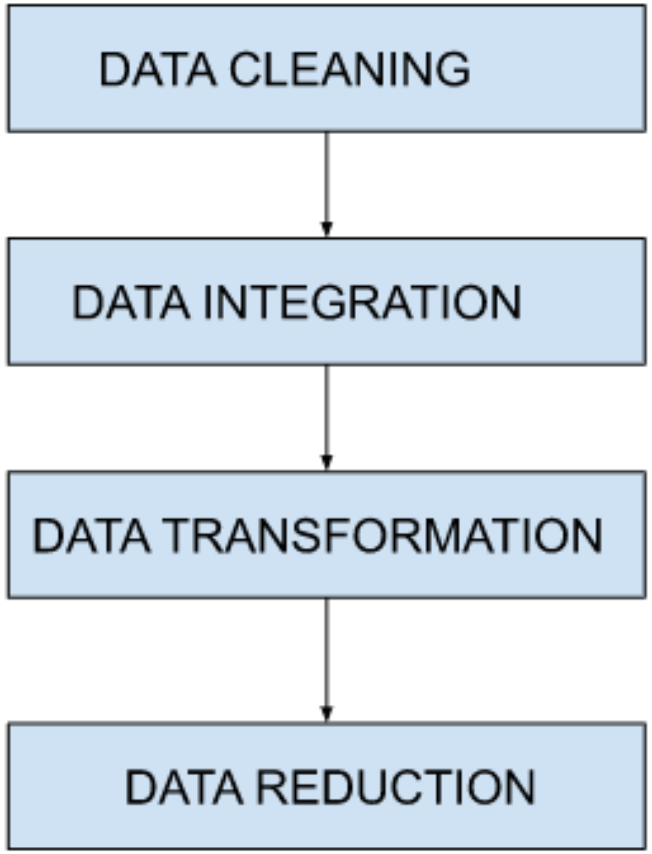


Recap: Course Overview



Lecture 4-6

Outline: Data Pre-processing



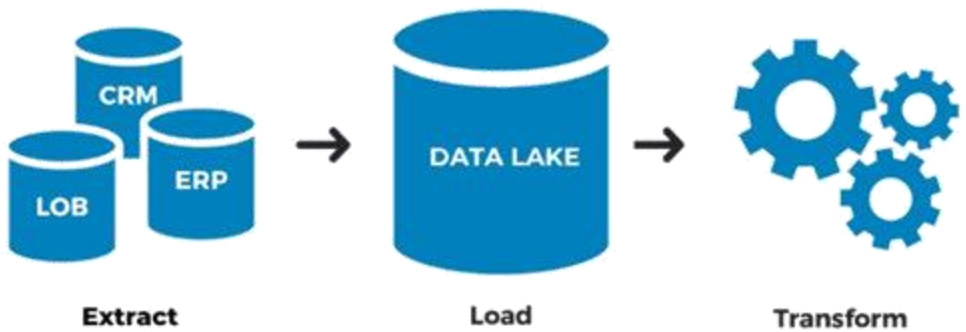
Recap: When to Transform?

ETL



vs.

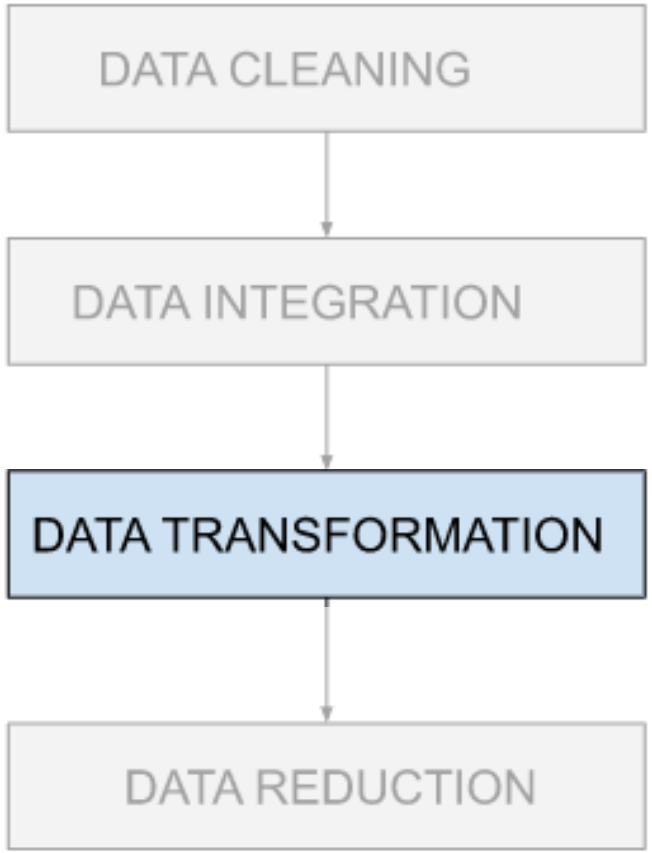
ELT



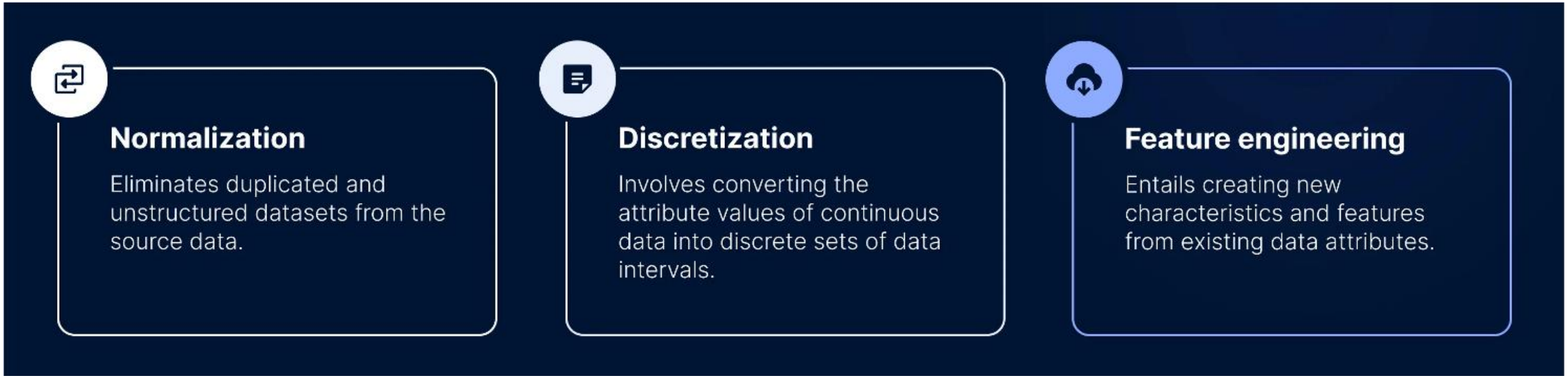
Recap: When to Transform?

Aspect	ETL (Extract → Transform → Load)	ELT (Extract → Load → Transform)
Transform timing	Transform **before** loading	Transform **after** loading
Best suited for	Traditional on-premise Data Warehouses	Cloud Data Warehouses / Big Data platforms
Advantages	- Only clean data stored in DW - Easier governance & compliance	- Scales well for large data - Raw data retained for reuse - Faster initial loading
Disadvantages	- Transformation server can be a bottleneck - Limited scalability	- Raw data in DW poses security/governance risks - Transformation can increase query/storage cost - Risk of “data swamp” if unmanaged

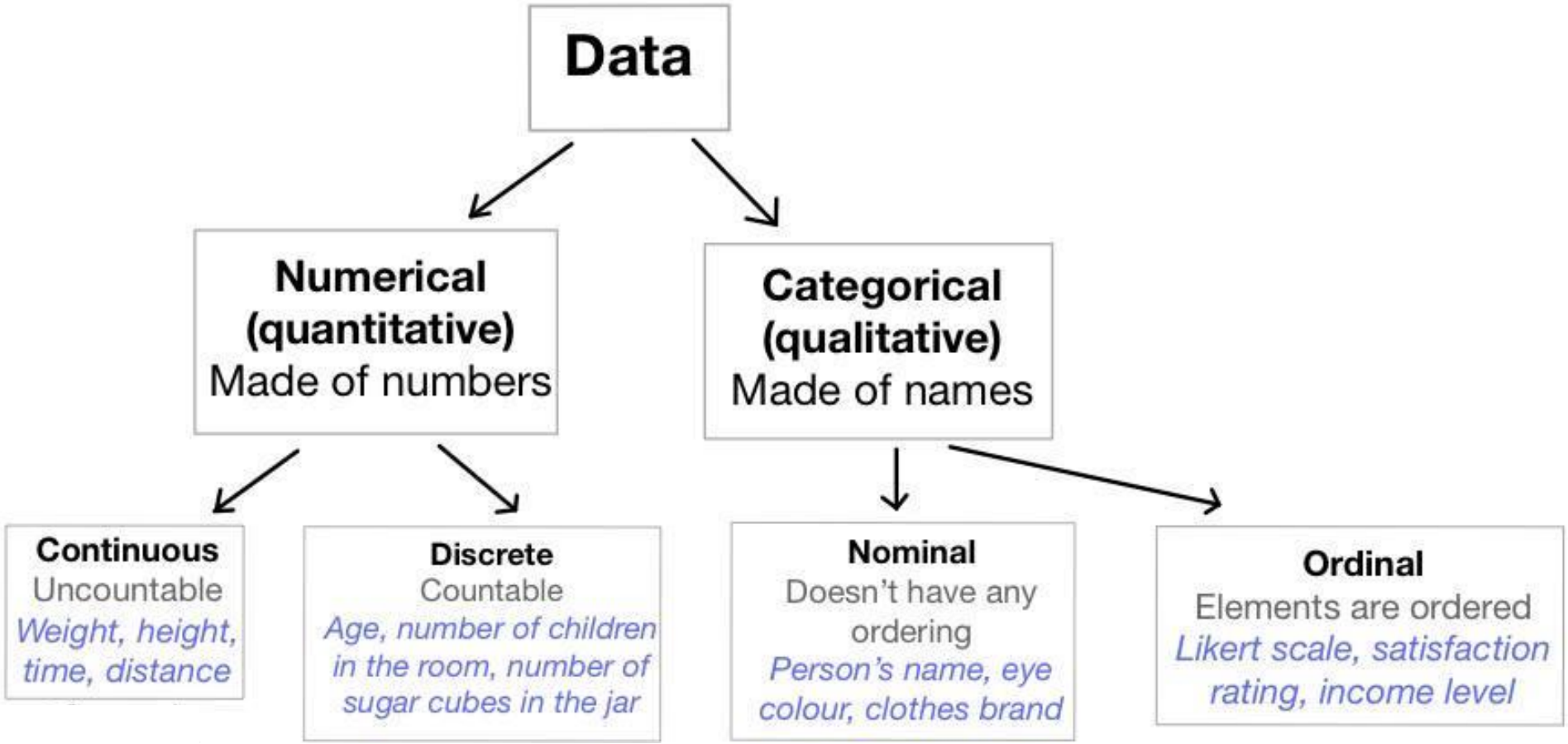
Outline: Data Pre-processing



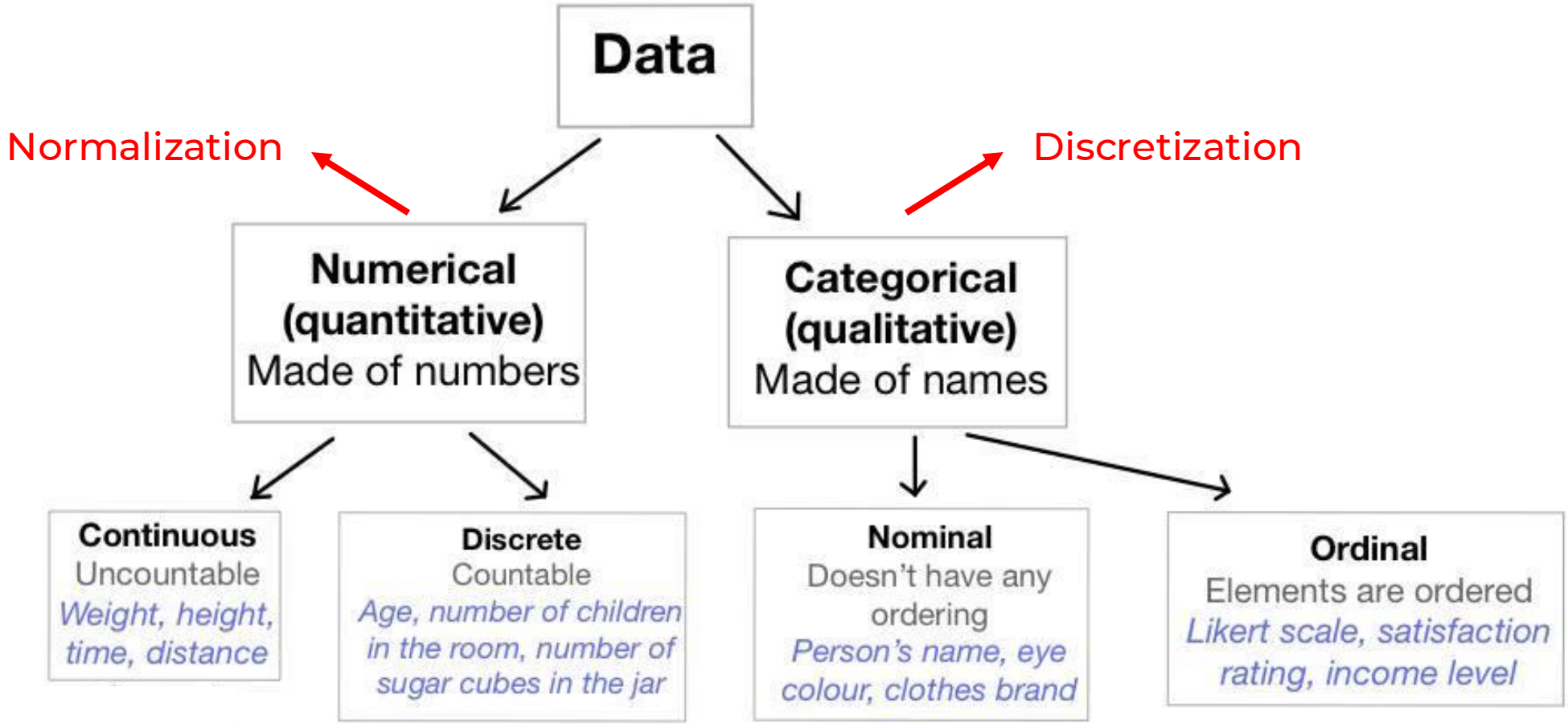
Data Transformation



Data Types


Data
Numerical (quantitative)
Made of numbers
Categorical (qualitative)
Made of names
Continuous
Uncountable
Weight, height, time, distance
Discrete
Countable
Age, number of children in the room, number of sugar cubes in the jar
Nominal
Doesn't have any ordering
Person's name, eye colour, clothes brand
Ordinal
Elements are ordered
Likert scale, satisfaction rating, income level

Data Types


Data
Normalization
Discretization
Numerical (quantitative)
Made of numbers
Categorical (qualitative)
Made of names
Continuous
Uncountable
Weight, height, time, distance
Discrete
Countable
Age, number of children in the room, number of sugar cubes in the jar
Nominal
Doesn't have any ordering
Person's name, eye colour, clothes brand
Ordinal
Elements are ordered
Likert scale, satisfaction rating, income level

Normalization



Why do we need Normalization?

age	income
20	100000
30	20000
40	500000



age	income
0.2	0.2
0.3	0.04
0.4	1

Not-normalized

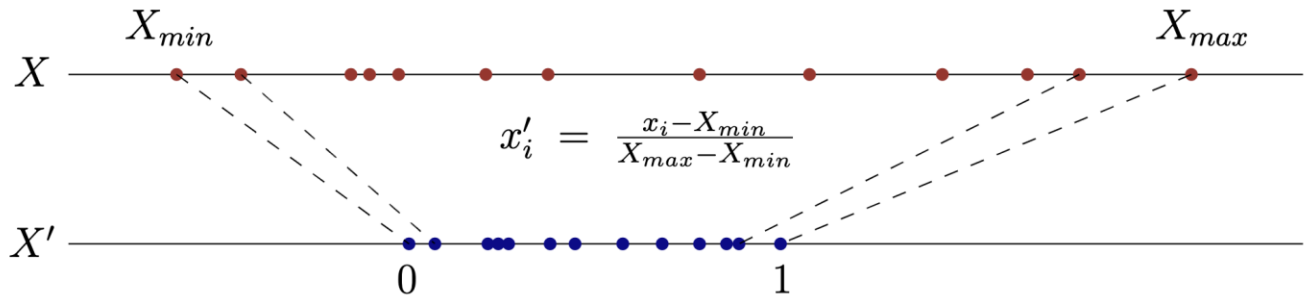
- "age" and "income" are in different range.
- hard to compare
- "income" will influence the result more

Normalized

- similar value range.
- similar intrinsic influence on analytical model.

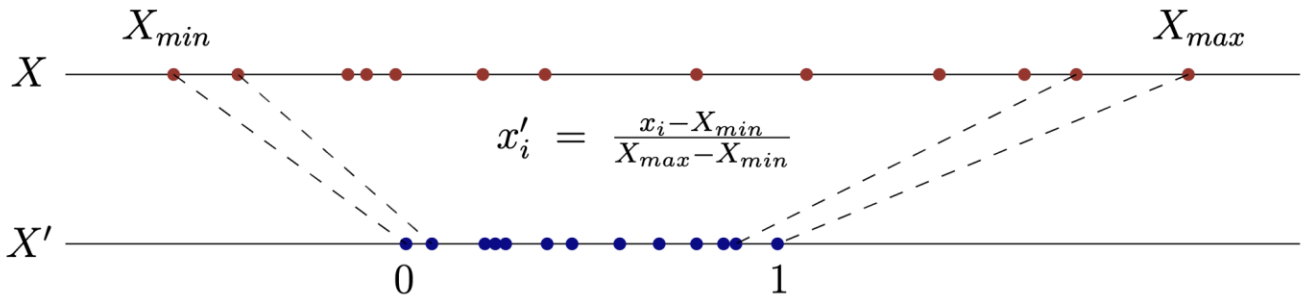
Normalization (= Max-Min Scaling)

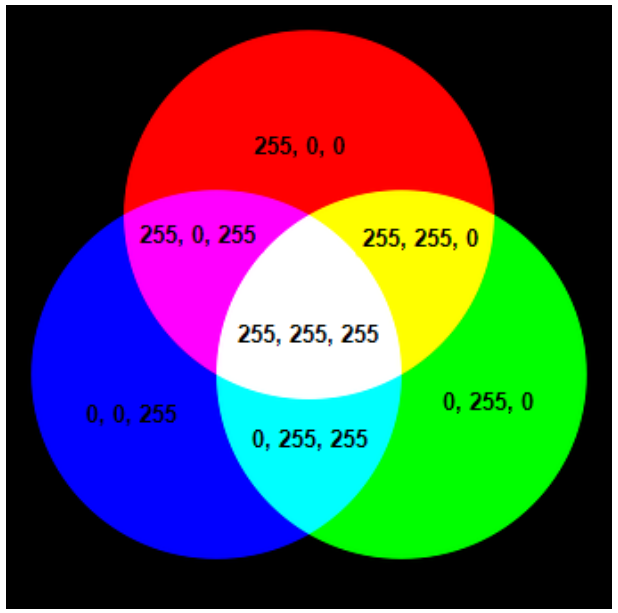
$$x_i' = \frac{x_i - X_{min}}{X_{max} - X_{min}}$$

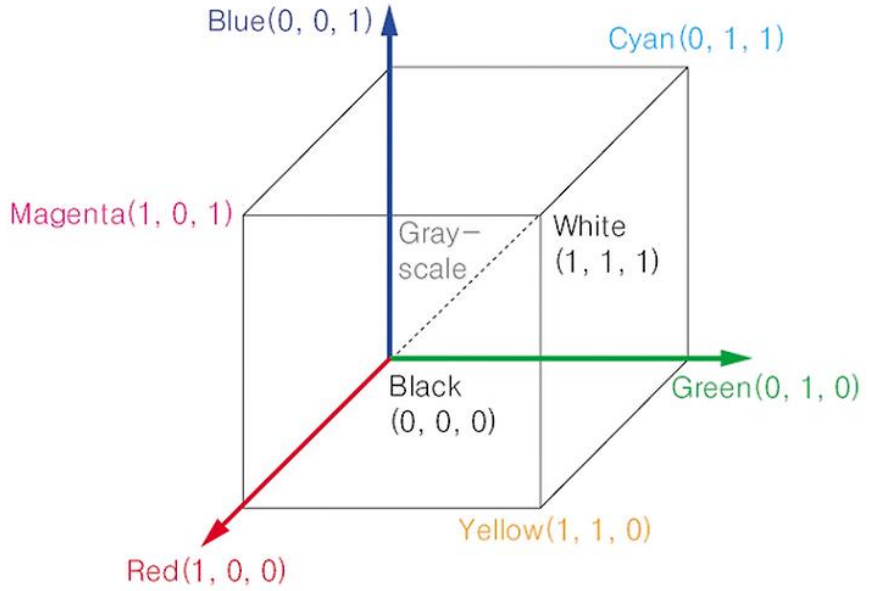


Normalization (= Max-Min Scaling)

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

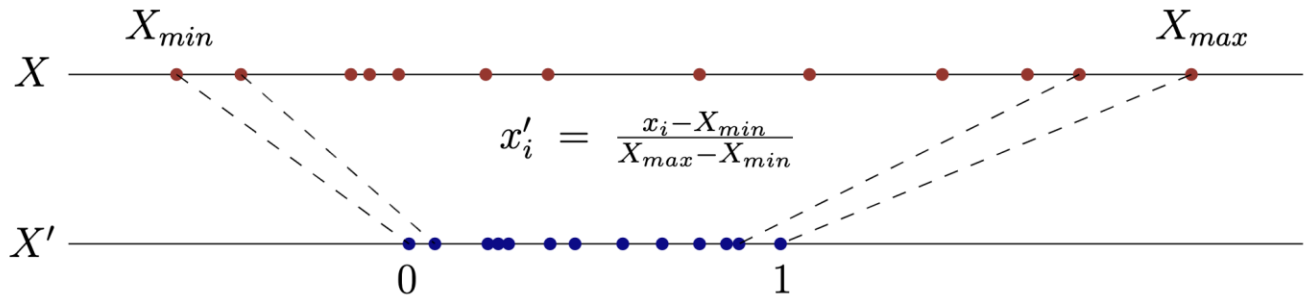


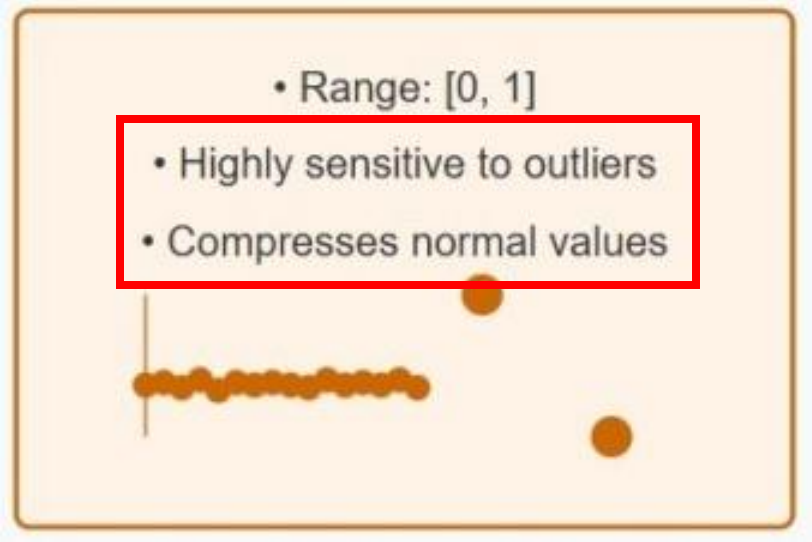




Issue of Max-Min Scaling

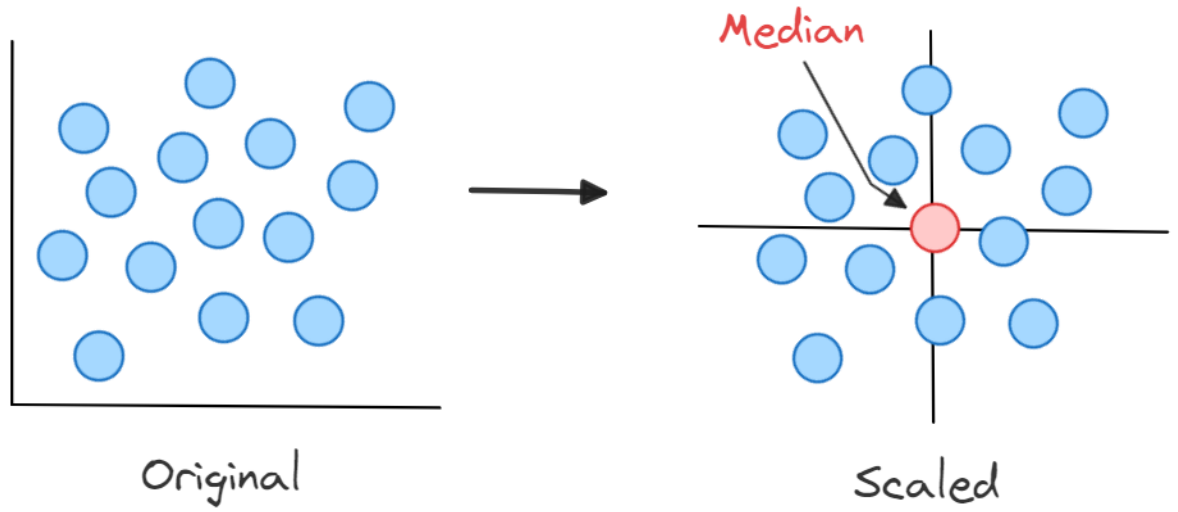
$$x_i' = \frac{x_i - X_{min}}{X_{max} - X_{min}}$$





Robust Scaling

$$X_{robust} = \frac{X - \text{Median}}{IQR}$$



Standardization (= Z-score Scaling)

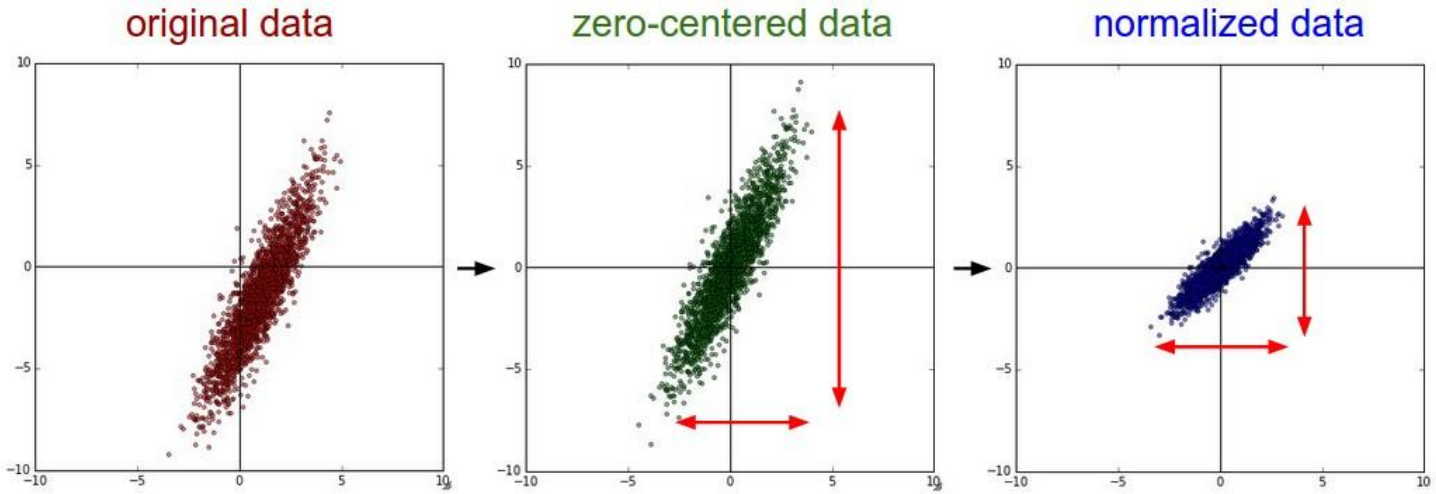
$$z = \frac{x - \mu}{\sigma}$$



Standardization (= Z-score Scaling)

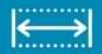
$$z = \frac{x - \mu}{\sigma}$$



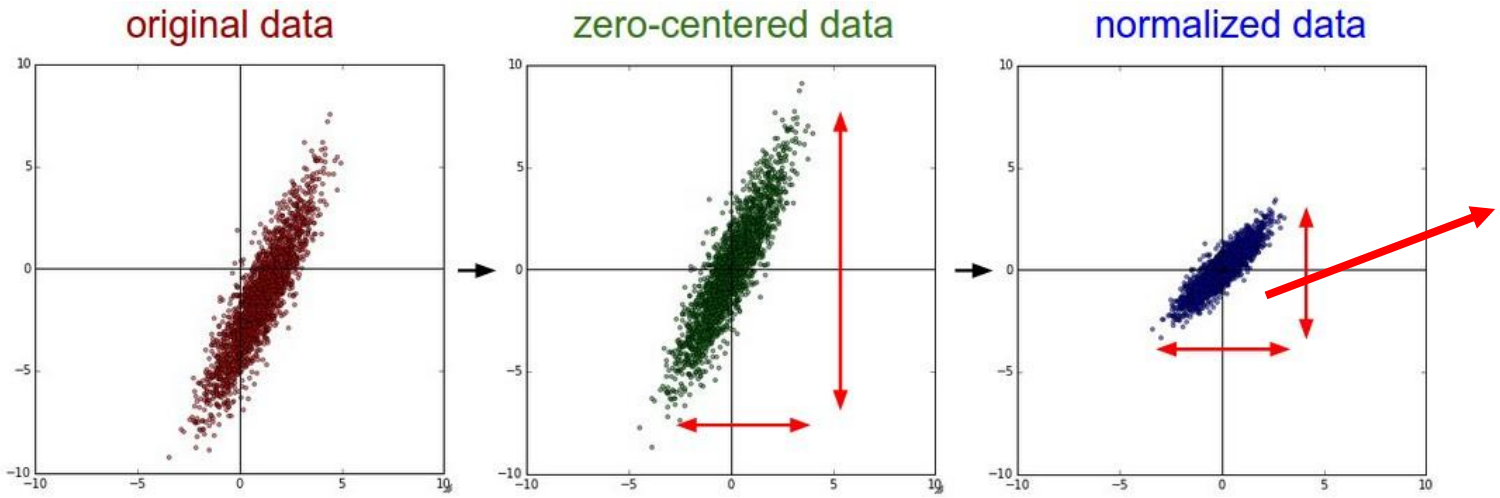


Normalization vs. Standardization

WHEN TO USE NORMALIZATION VS STANDARDIZATION

SCENARIO	USE NORMALIZATION OR STANDARDIZATION?
 You need to constrain the data within a specific **range**	**Normalization**
 You want to maintain the original unit **interpretation**	**Normalization**
 The distribution of the data is not necessarily **Gaussian**	**Normalization**
 You need to compare and analyze features with **different units and scales**	**Standardization**
 You want to reduce the impact of **outliers**	**Standardization**

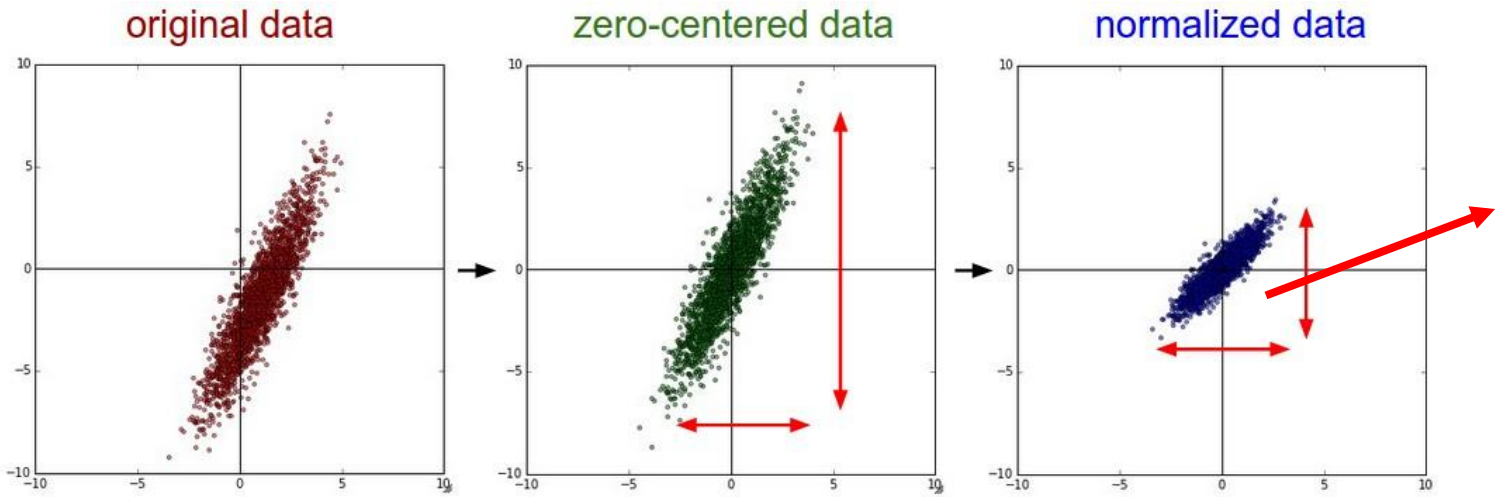
Whitening (= n-dim Standardization)



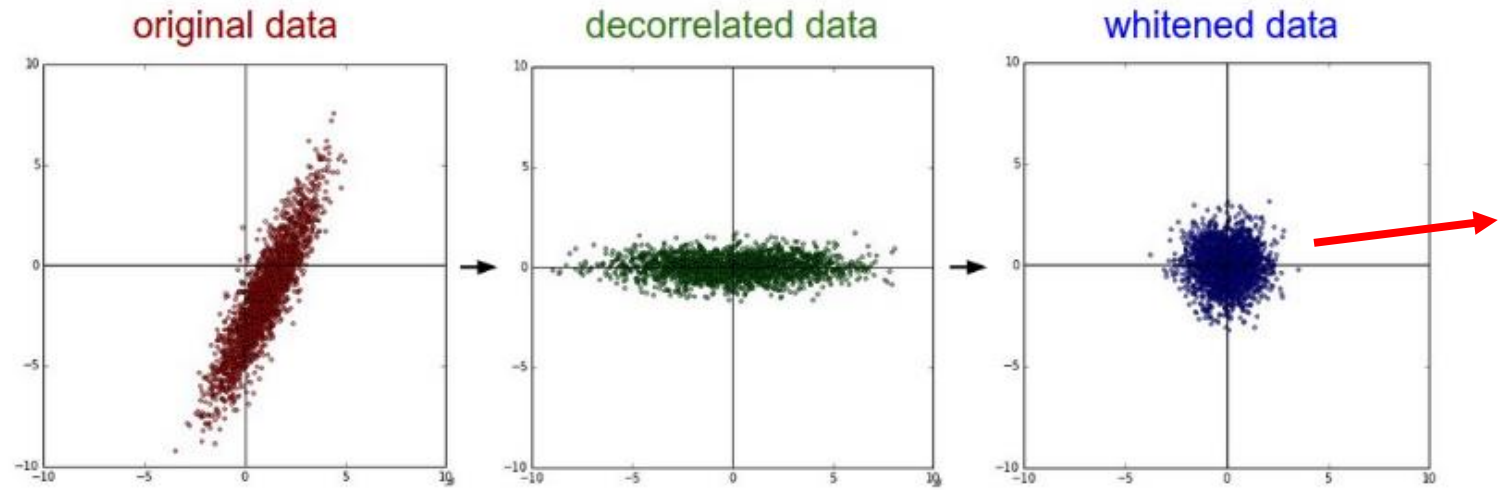
Standardization (1-dim):
Features are correlated

Whitening (= n-dim Standardiza

Whitening (= n-dim Standardization)



Standardization (1-dim):
Features are correlated



Whitening (n-dim):
Features are **de**correlated

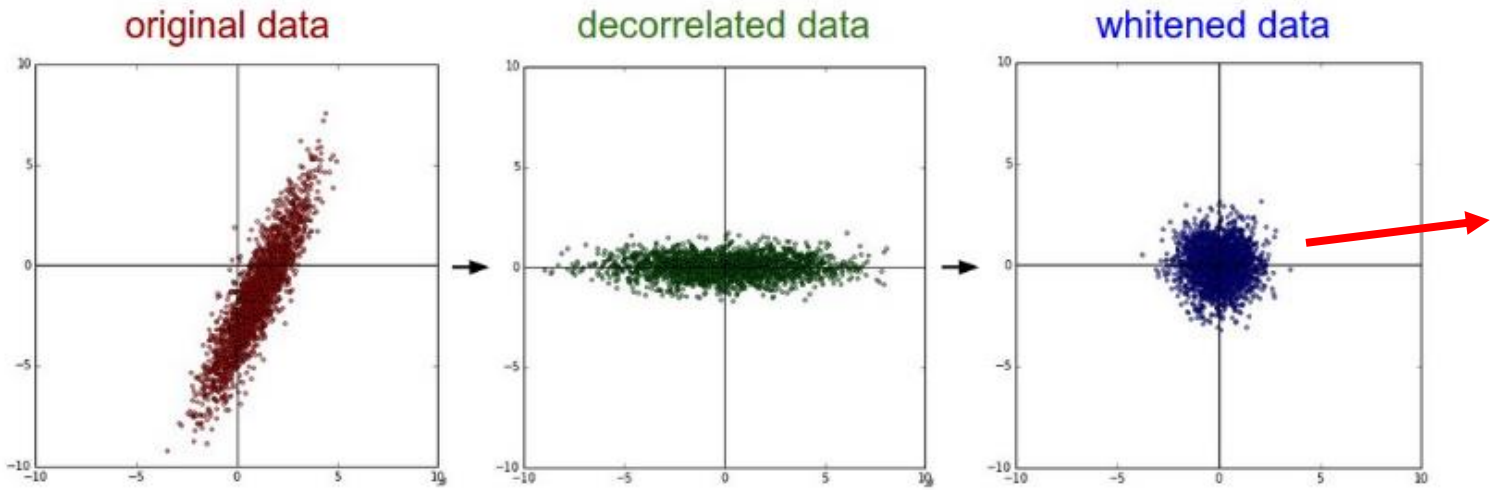
Whitening (= n-dim Standardization)

$$W_{PCA} = \Lambda^{\frac{-1}{2}} U^T X$$

X: Feature data

Λ: is the matrix for Eigenvalues (above is inverse square root)

U^T: is transpose of matrix for Eigenvectors



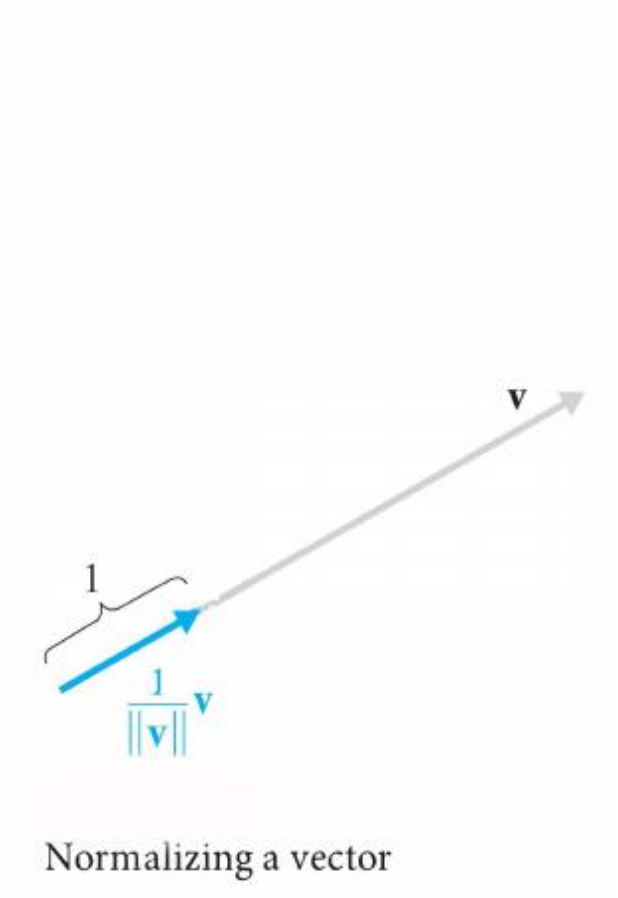
Whitening (n-dim):
Features are **de**correlated

Whitening (= n-dim Standardization)

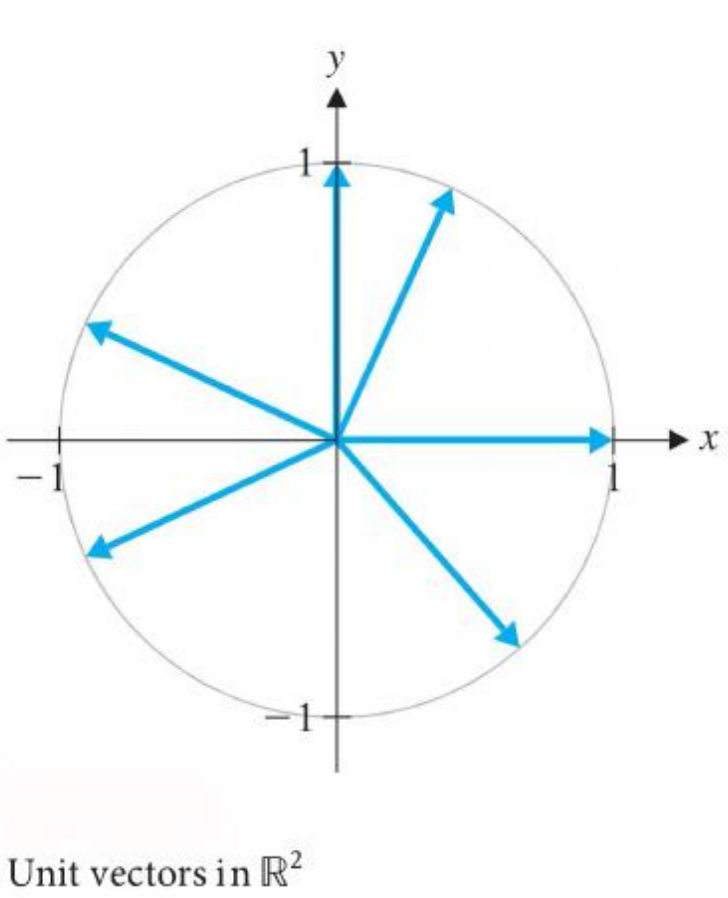
Data Decorrelation: When to Use vs When Not To

When to Use	When Not To Use
Model requirement: statistical models that assume feature independence (e.g., LDA, Gaussian Naive Bayes)	**Model robustness:** ML/DL models that handle correlations internally (trees, random forests, XGBoost, deep nets)
Data issue: high multicollinearity that destabilizes regression coefficients	**Data value:** when correlations themselves carry important information (finance, recommender systems, graph data)
Practical setting: classical signal/image preprocessing (PCA whitening, ICA)	**Practical limitation:** high-dimensional data where covariance decomposition is costly or unstable

Unit Vector Normalization



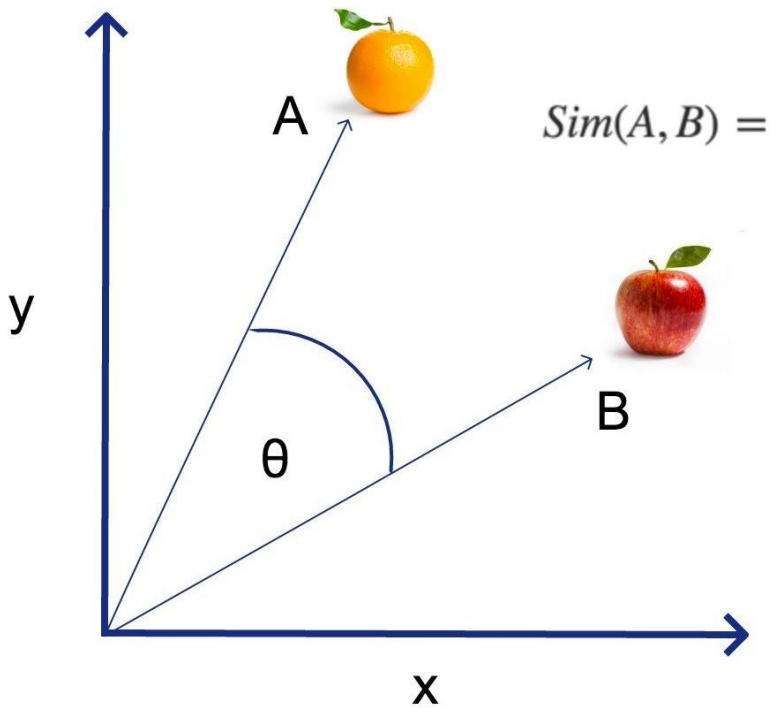
Normalizing a vector



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

Unit Vector Normalization

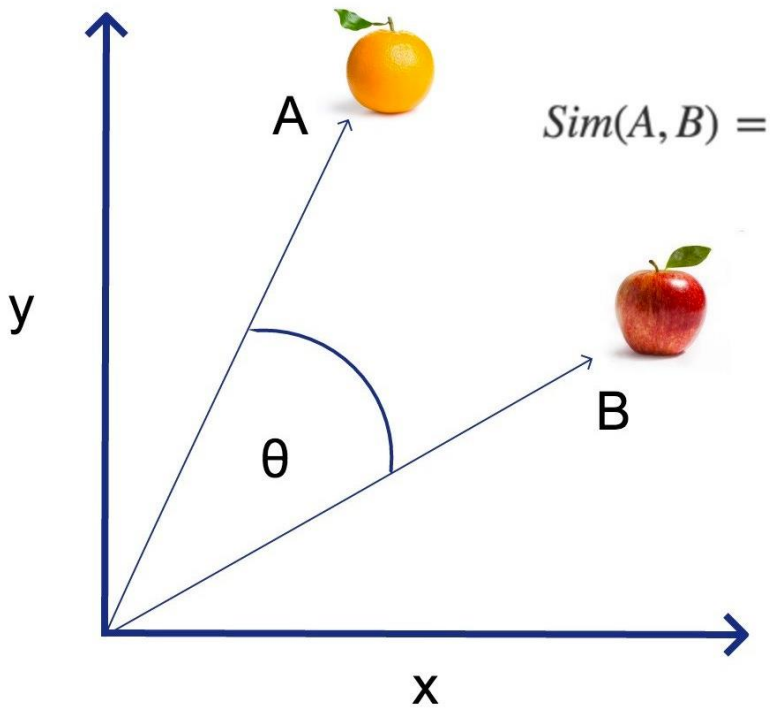
Cosine Similarity



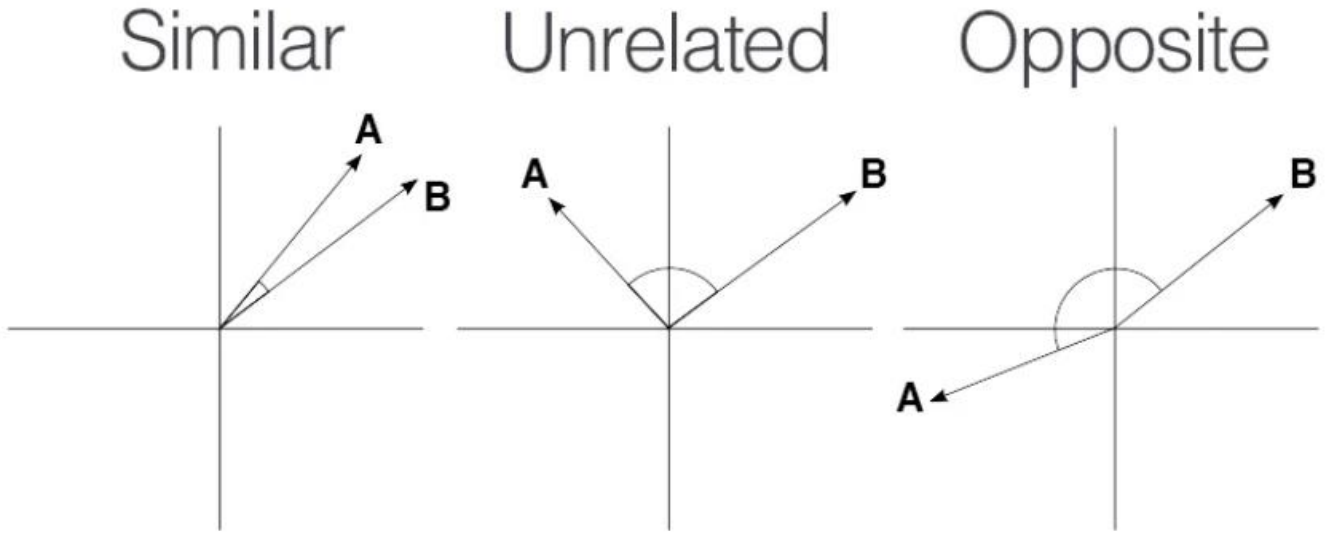
$$Sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

Unit Vector Normalization

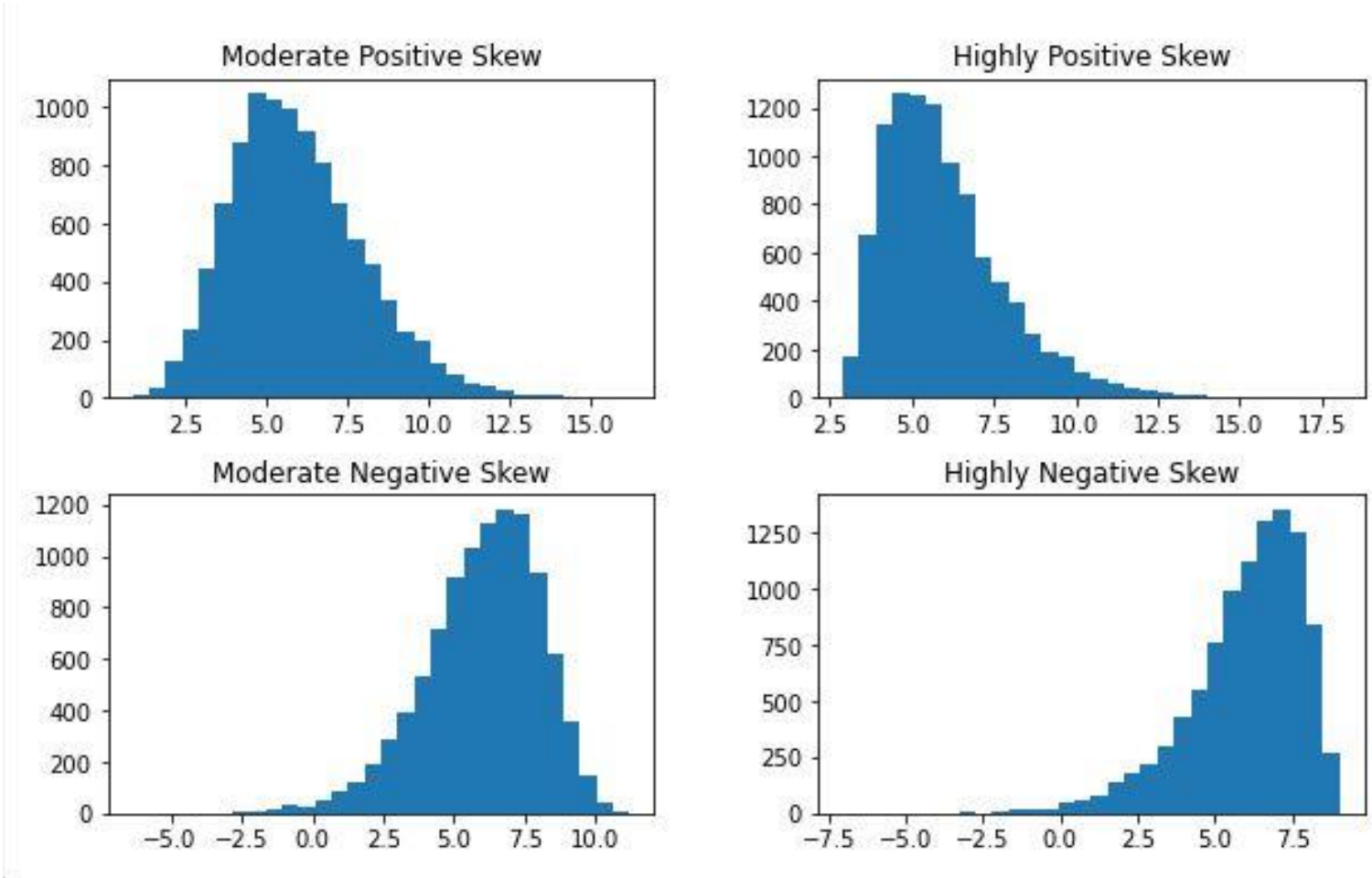
Cosine Similarity



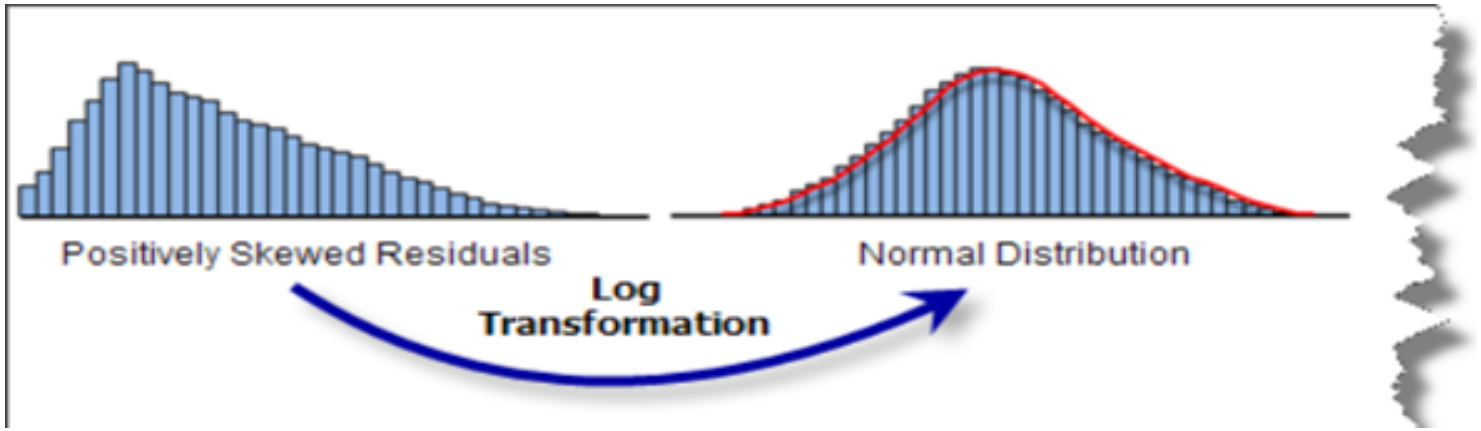
$$Sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$



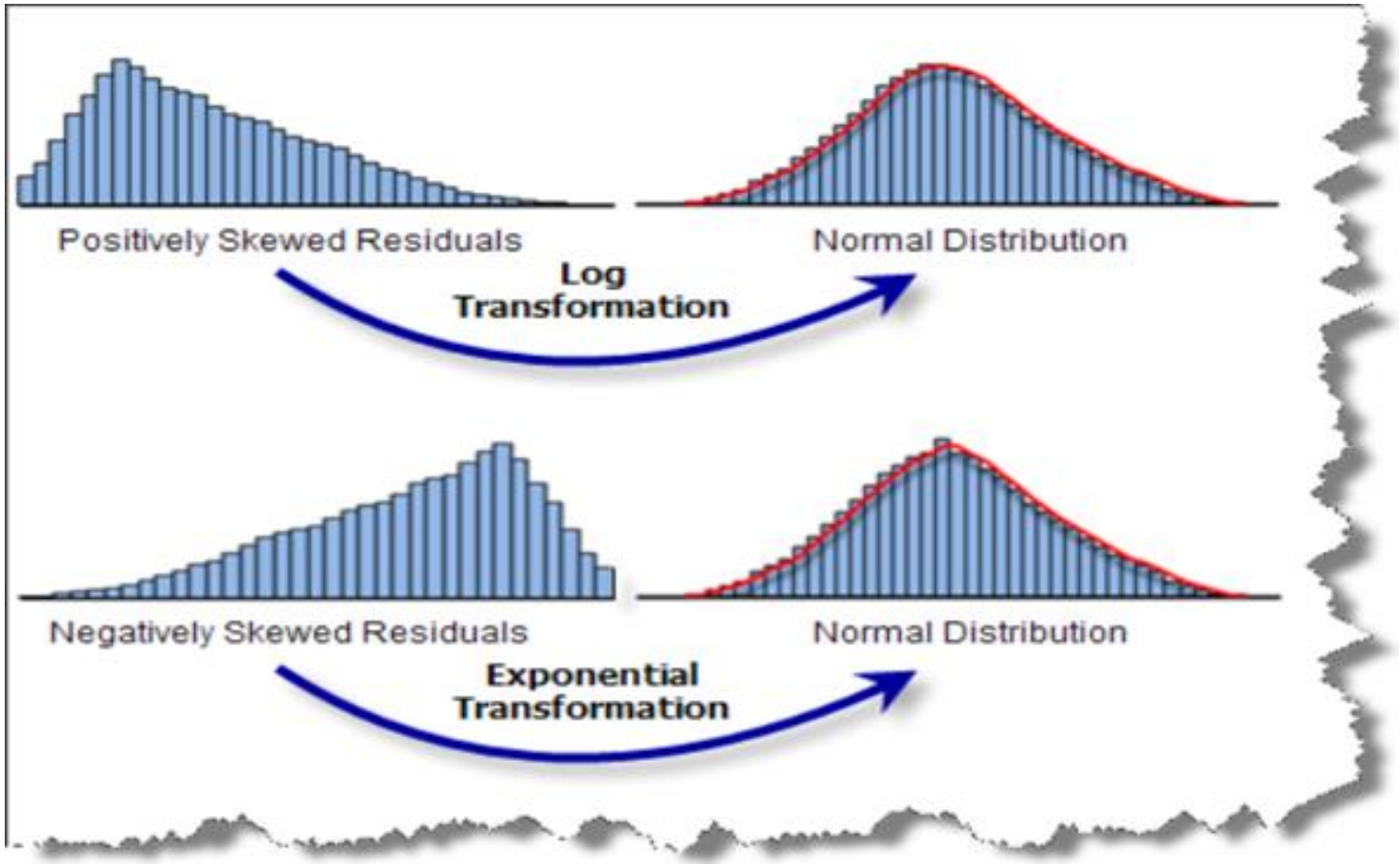
Recap: Skewness



Non-linear Transformations



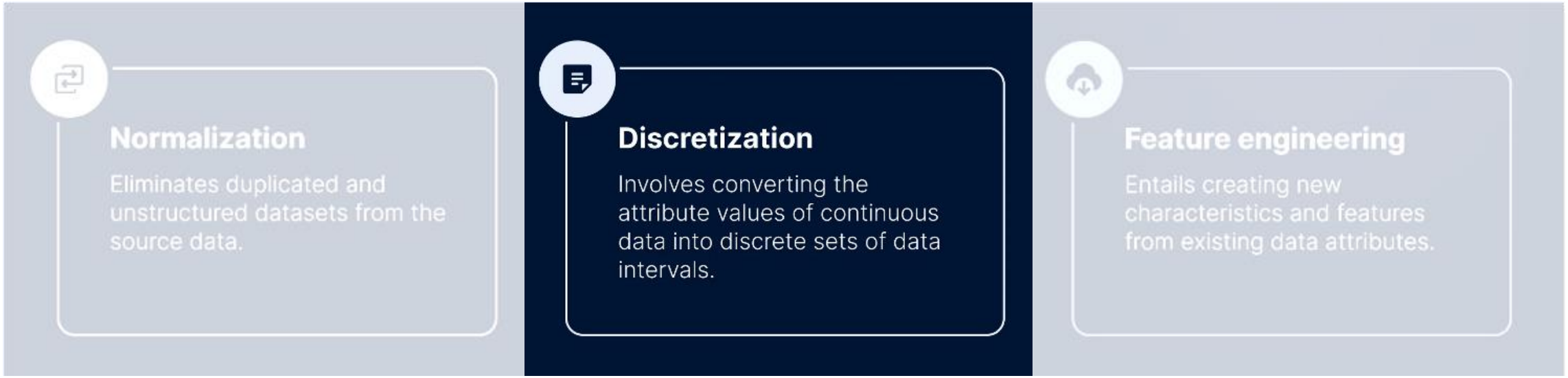
Non-linear Transformations



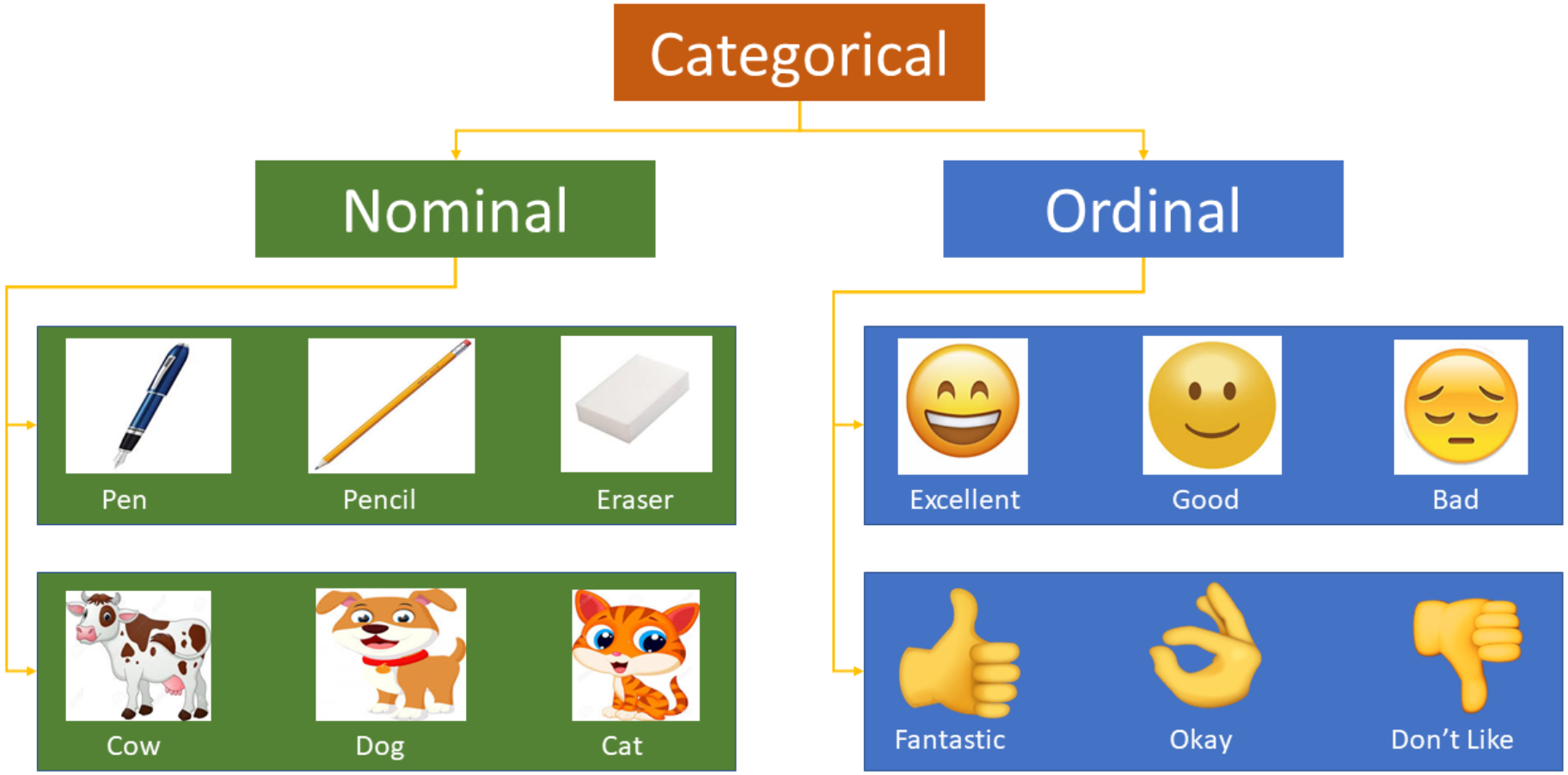
Non-linear Transformations

Transformation	Formula	Effect / Use Case
Logarithmic	$x' = \log(x+1)$	Reduces right skew, compresses large values
Square-root	$x' = \sqrt{x}$	Moderate skew reduction, stabilizes variance
Cube-root	$x' = \sqrt[3]{x}$	Useful for count data, mild skewness
Reciprocal	$x' = \frac{1}{x+\epsilon}$	Inverts scale, highlights small values
Negative reciprocal	$x' = -\frac{1}{x+\epsilon}$	Similar to reciprocal but reverses order

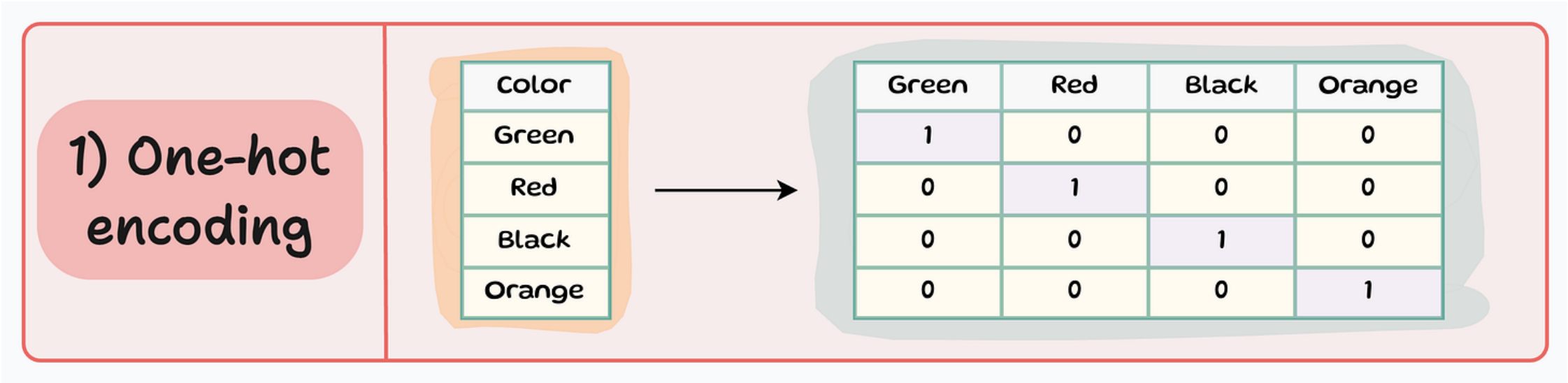
Discretization



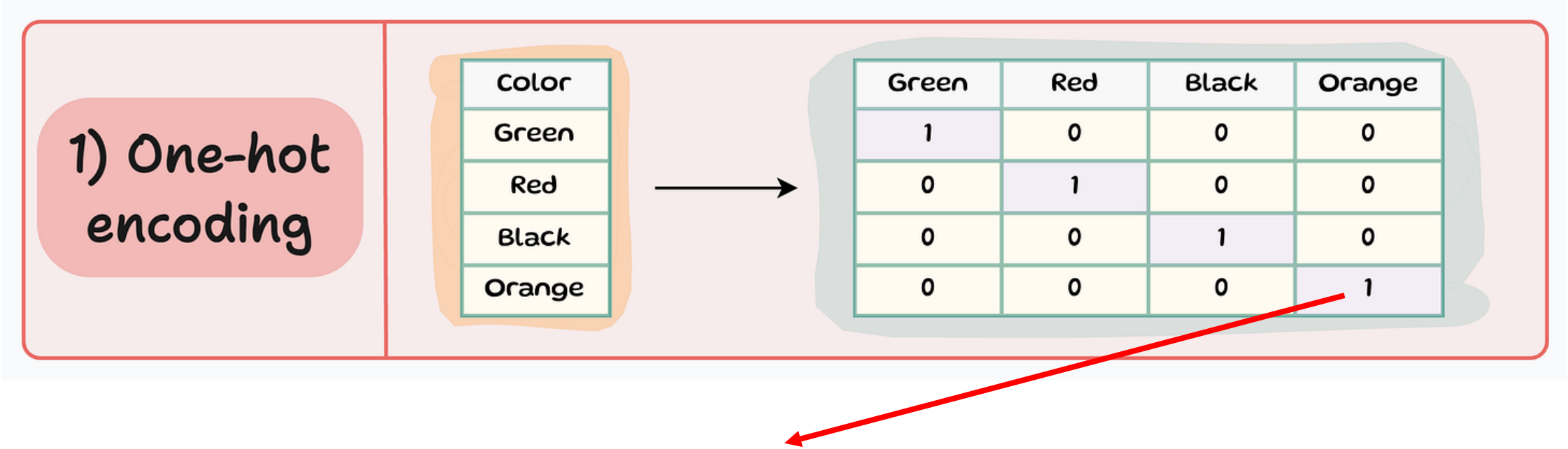
Recap: Nominal vs. Ordinal



One-hot Encoding

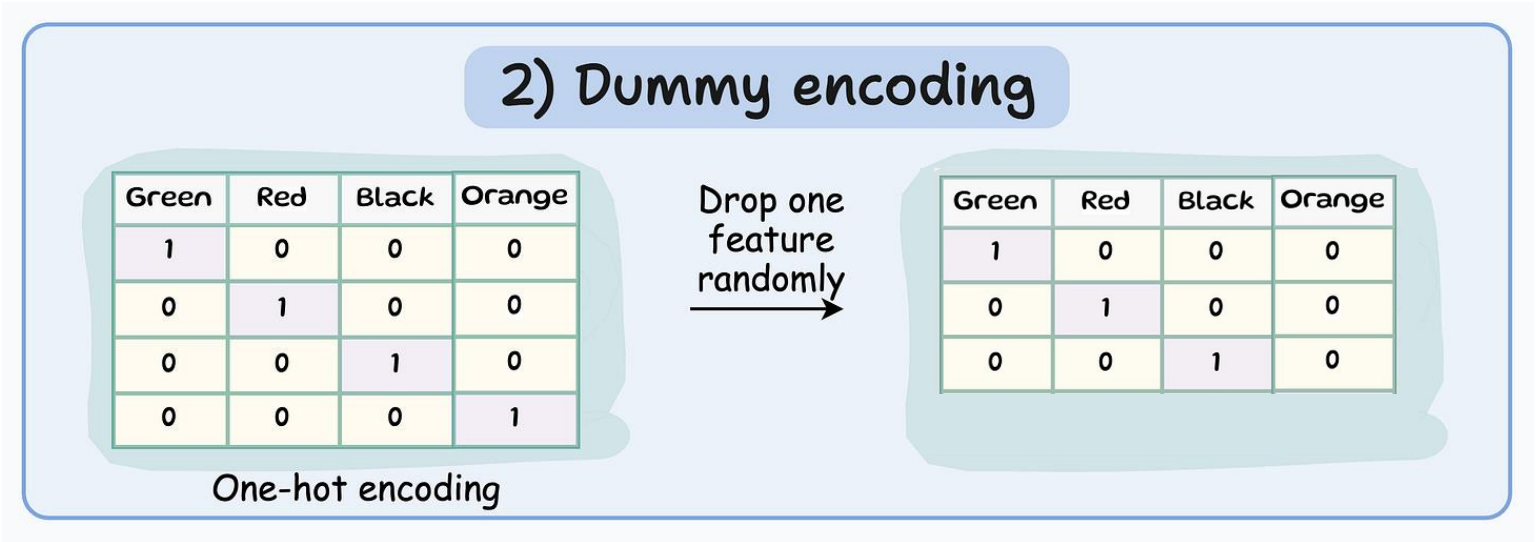


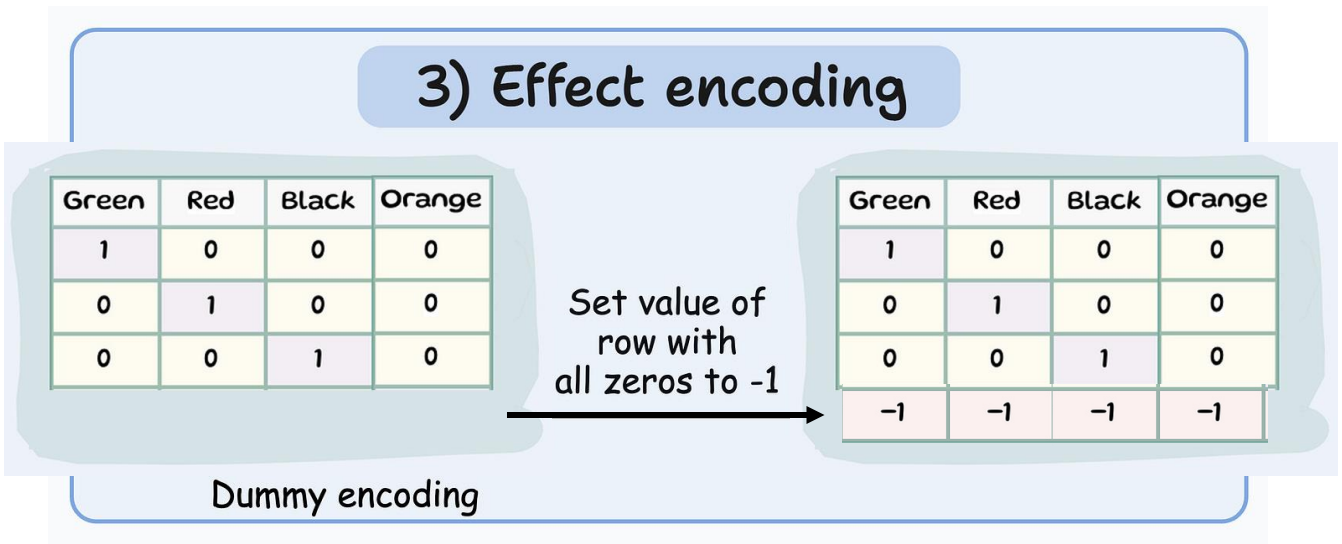
Issue of One-hot Encoding



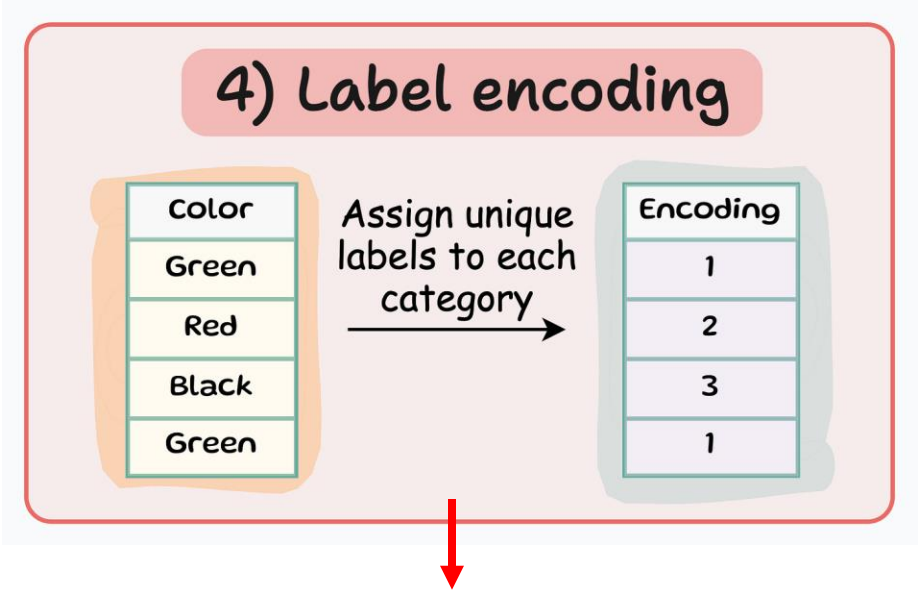
Multicollinearity: One variable is dependent on others
(We need only $k - 1$ features to represent k categories)

Dummy & Effect Encoding

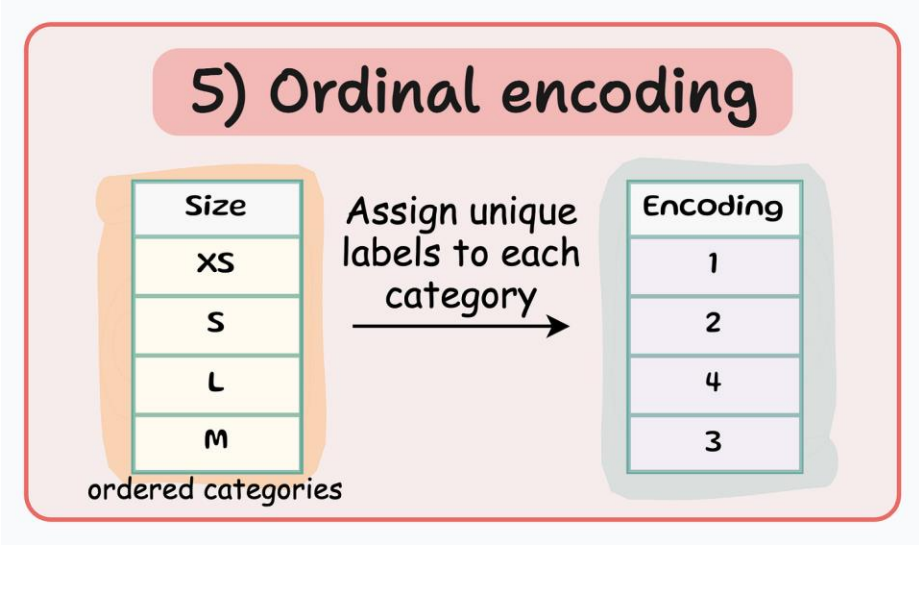




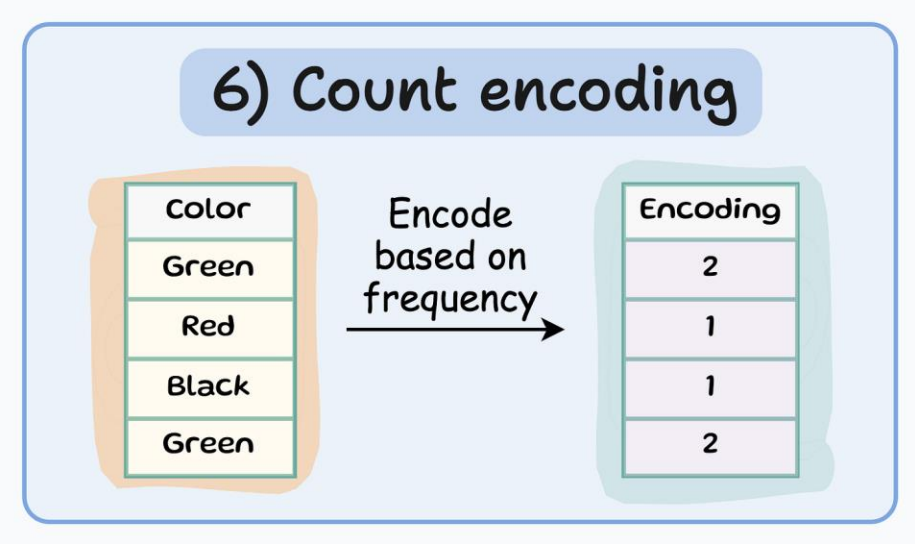
Label & Ordinal Encoding



Usually only used for ordinal data, except when applied to tree models

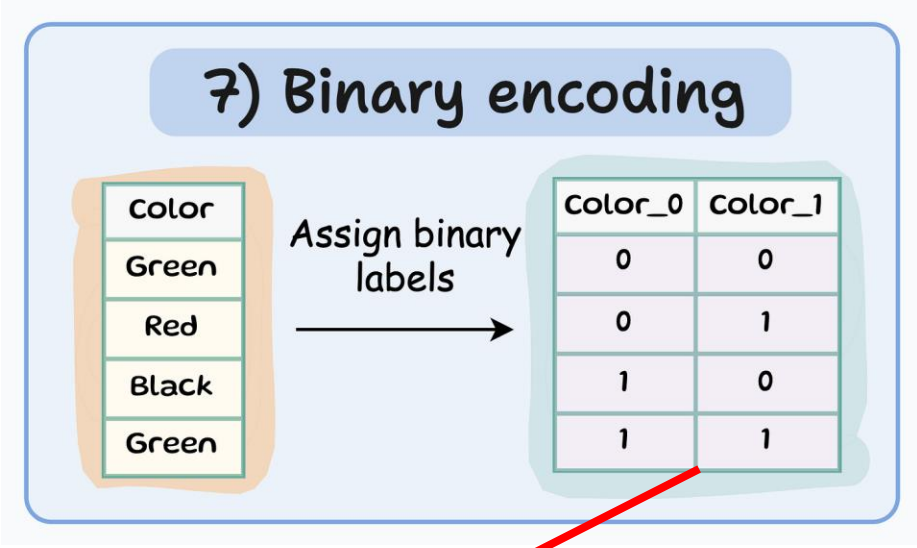


Count (Frequency) Encoding



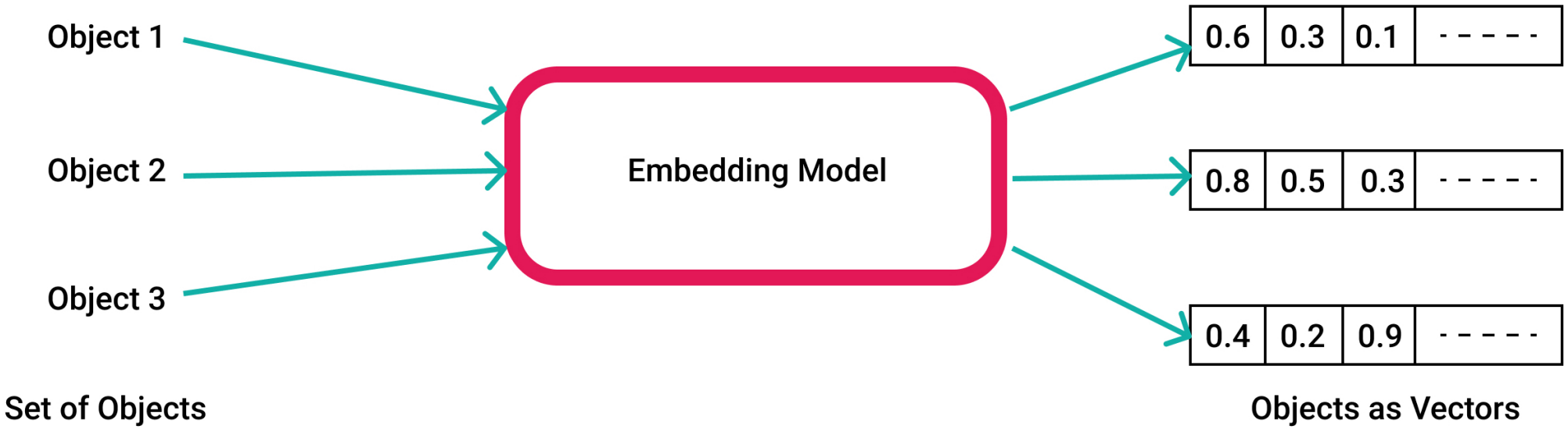
When count is an important feature
(e.g., product ID in e-commerce)

Binary Encoding

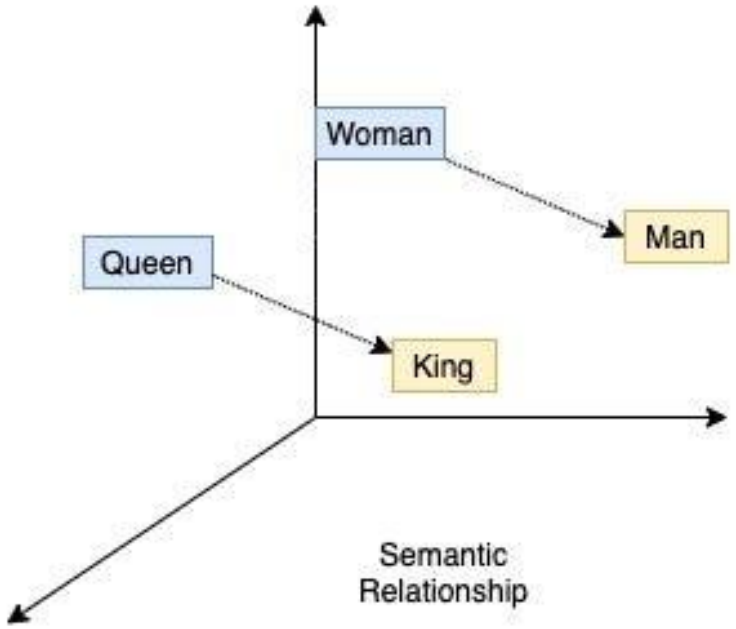


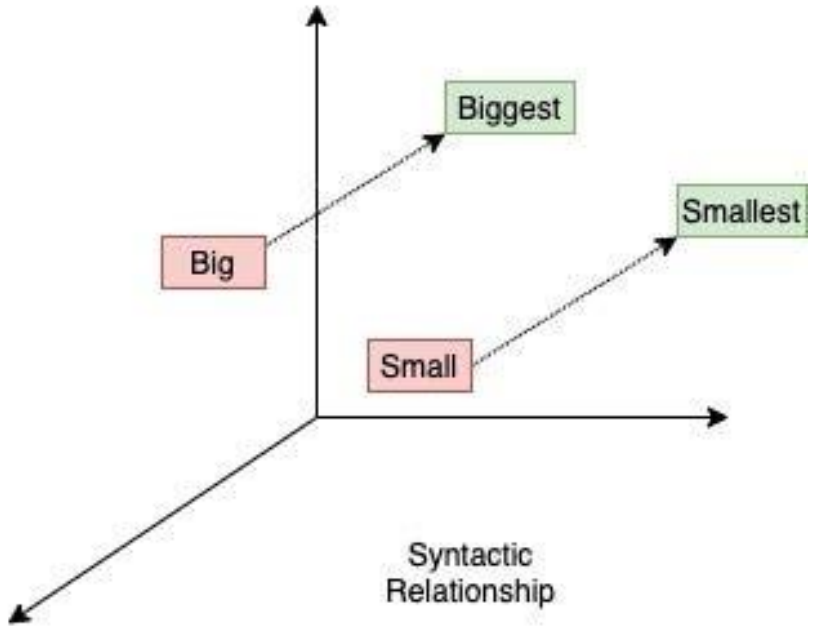
Need only $\log_2 k$ features
to represent k categories

Vector Embedding



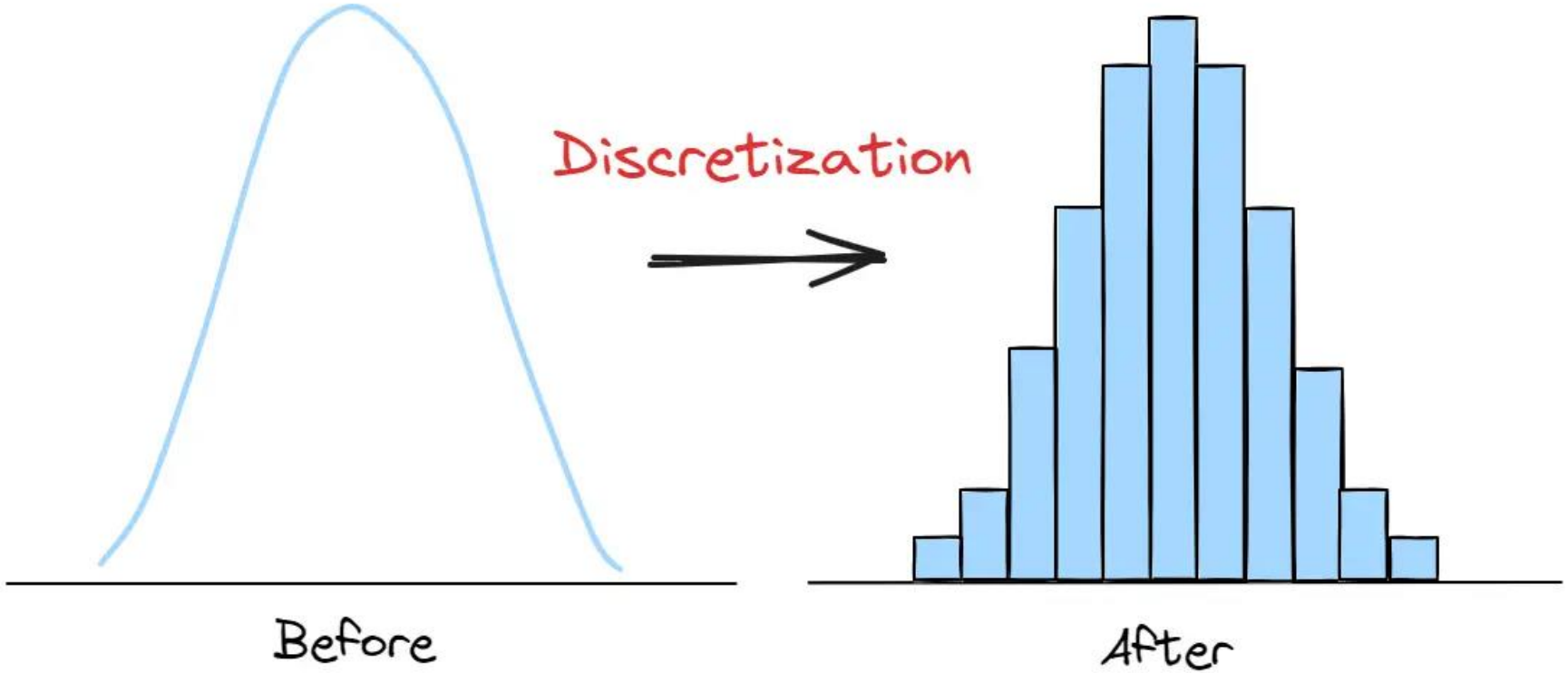
Case Study: Language Models (Word2Vec)





Efficient Estimation of Word Representations in Vector Space. ICMLW, 2013.

Converting Continuous to Discrete Values



Converting Continuous to Discrete Values

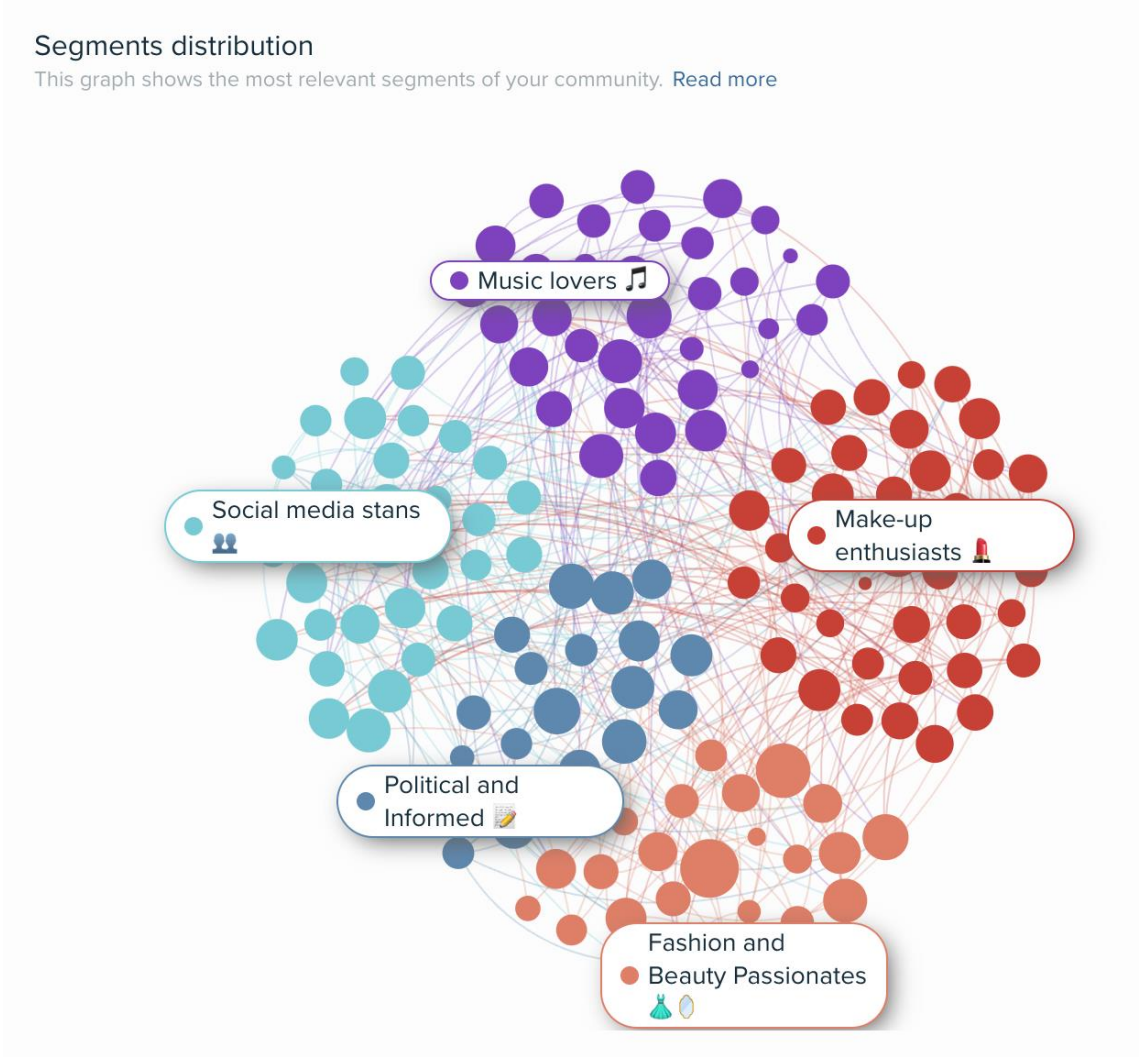
Why Discretization is Useful

Reason	Explanation	Example
Interpretability	Easier for humans to understand categories than raw numbers	Age 23 → “20s group”
Capture non-linear patterns	Binning can reveal effects that aren’t strictly linear	Purchase probability higher only in low/high income groups
Model requirements	Some algorithms require categorical inputs	Association rule mining
Outlier robustness	Extreme values have less influence once grouped	Age 120 → “100+”
Visualization & reporting	Simplifies data summaries and comparisons	“Sales by income bracket”

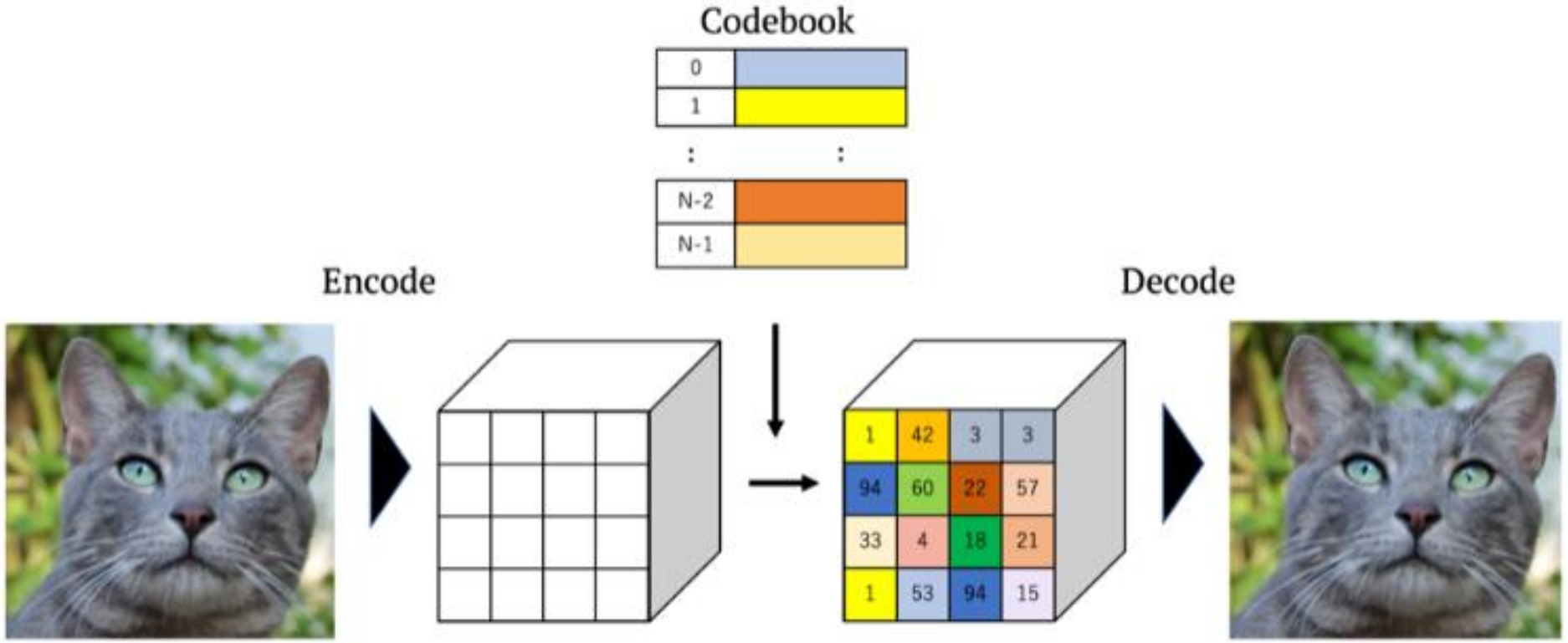
Binning by Histogram

```
> head(df,10)
   MyContinuous MySpecificBins
1            24         (15,25]
2            24         (15,25]
3            18         (15,25]
4            24         (15,25]
5            23         (15,25]
6            22         (15,25]
7            15      (-Inf,15]
8            36      (25, Inf]
9            15      (-Inf,15]
10           13      (-Inf,15]
```

Binning by Clustering



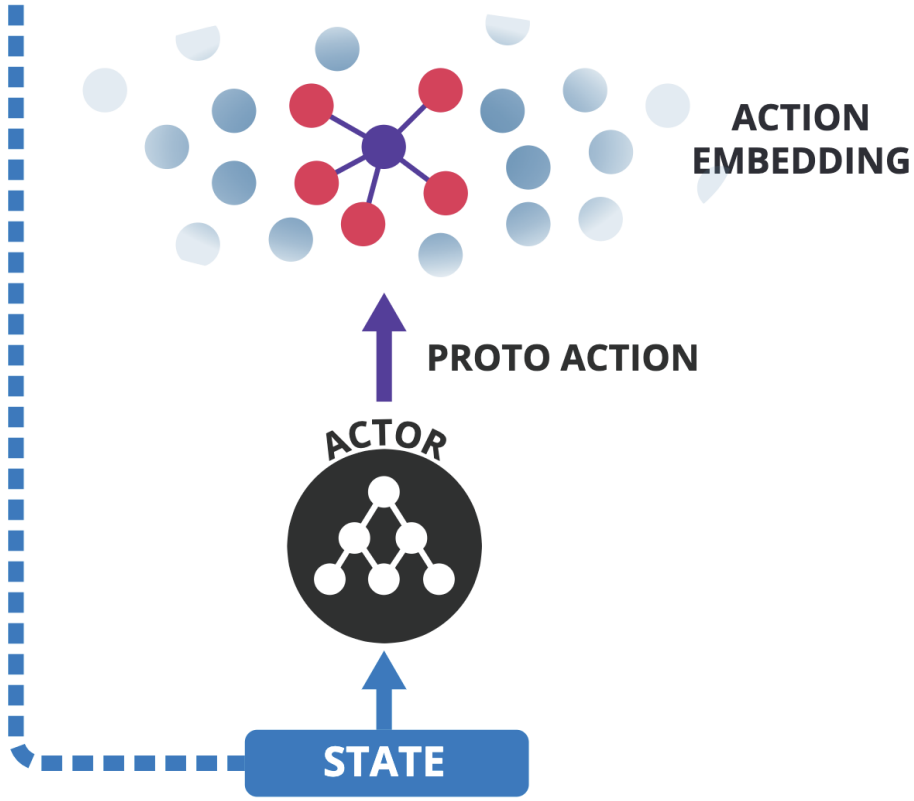
Case Study: Vector Quantization



Continuous data (e.g., image, audio) ≈ Sequence of discrete values

Neural Discrete Representation Learning. NeurIPS, 2017.

Case Study: Regression → Classification

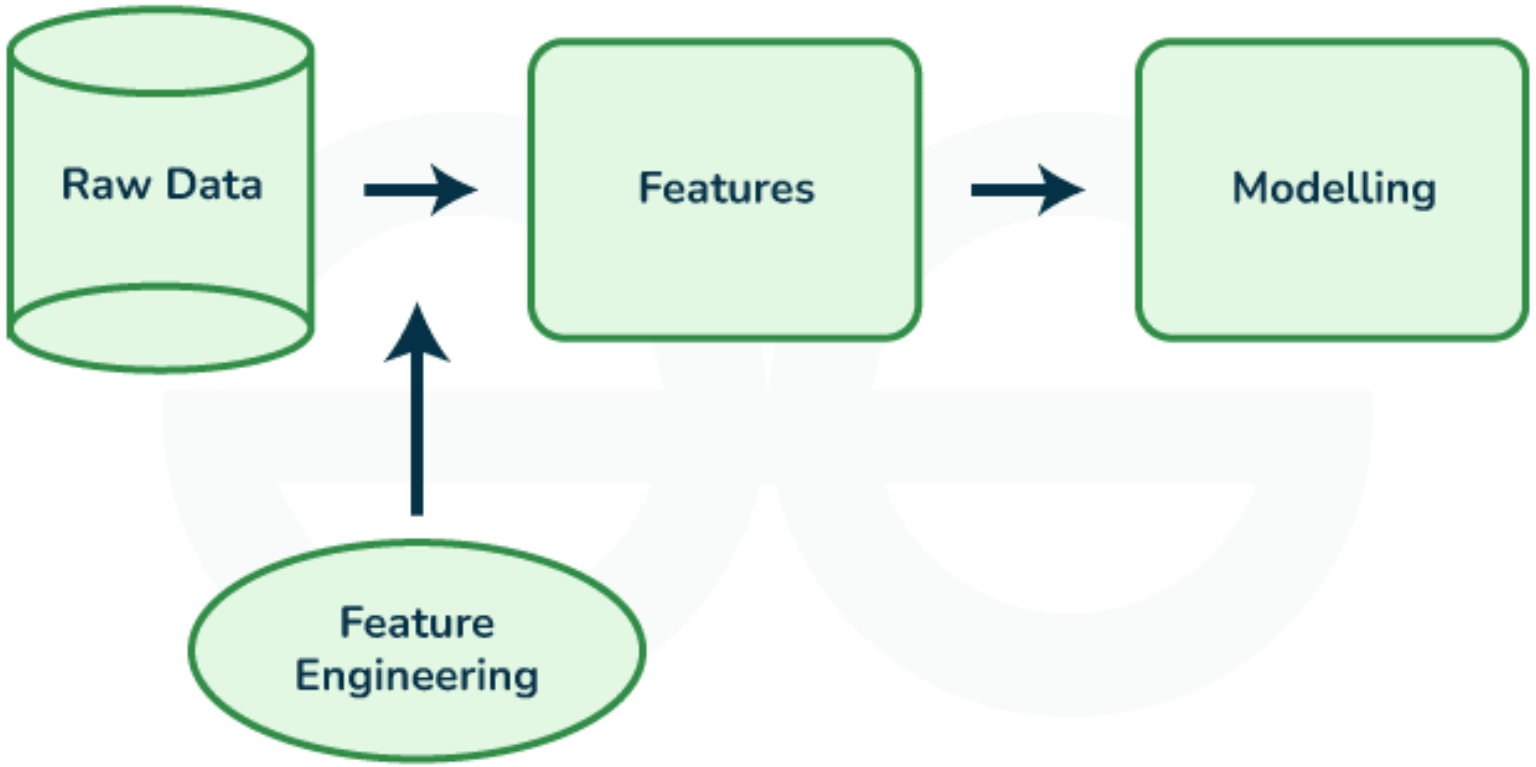


Deep Reinforcement Learning in Large Discrete Action Spaces. ICML, 2015.

Feature Engineering



Feature Engineering



Classic ML = Hand-crafted Features

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

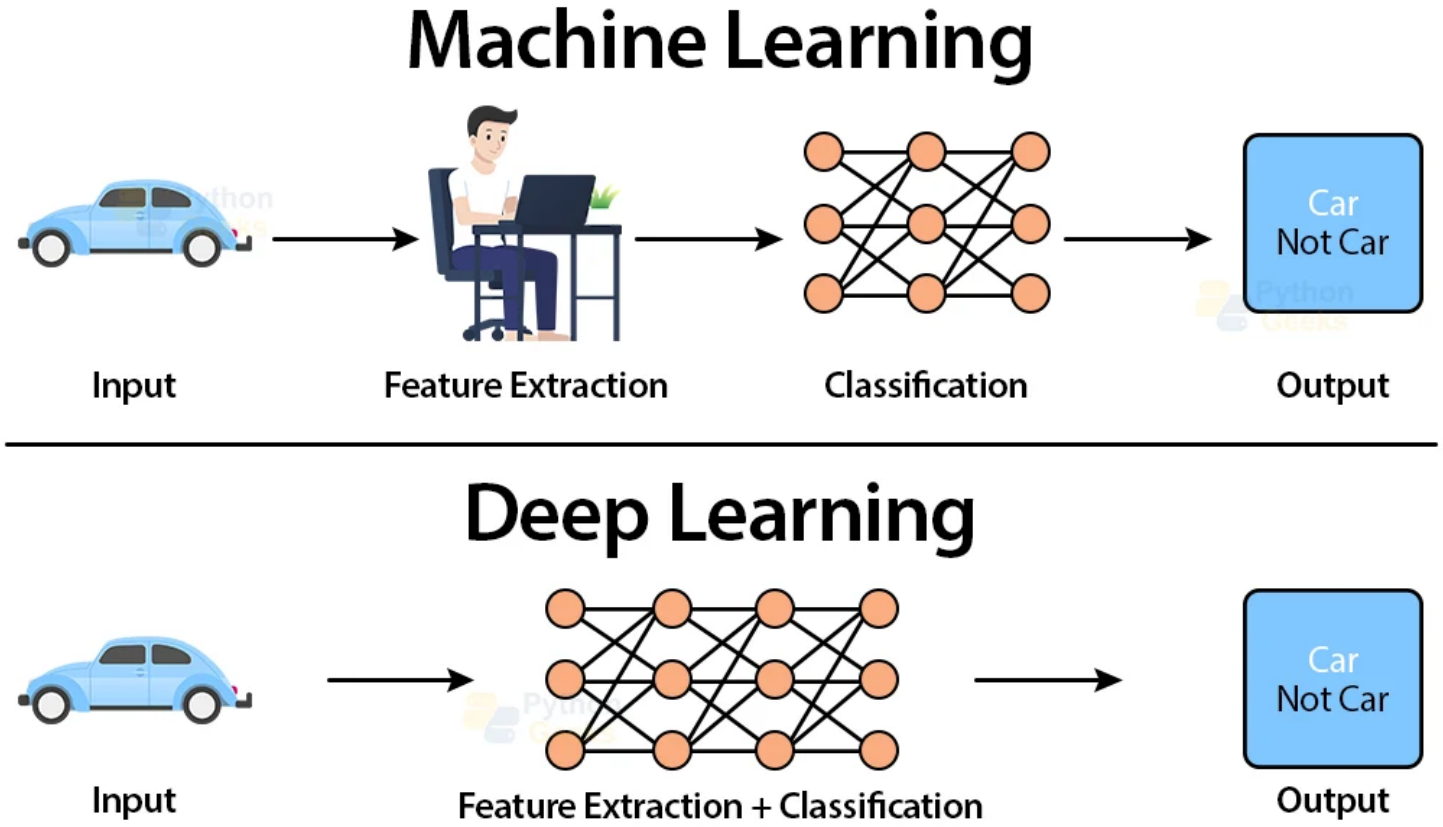
Term x within document y

$tf_{x,y}$ = frequency of x in y

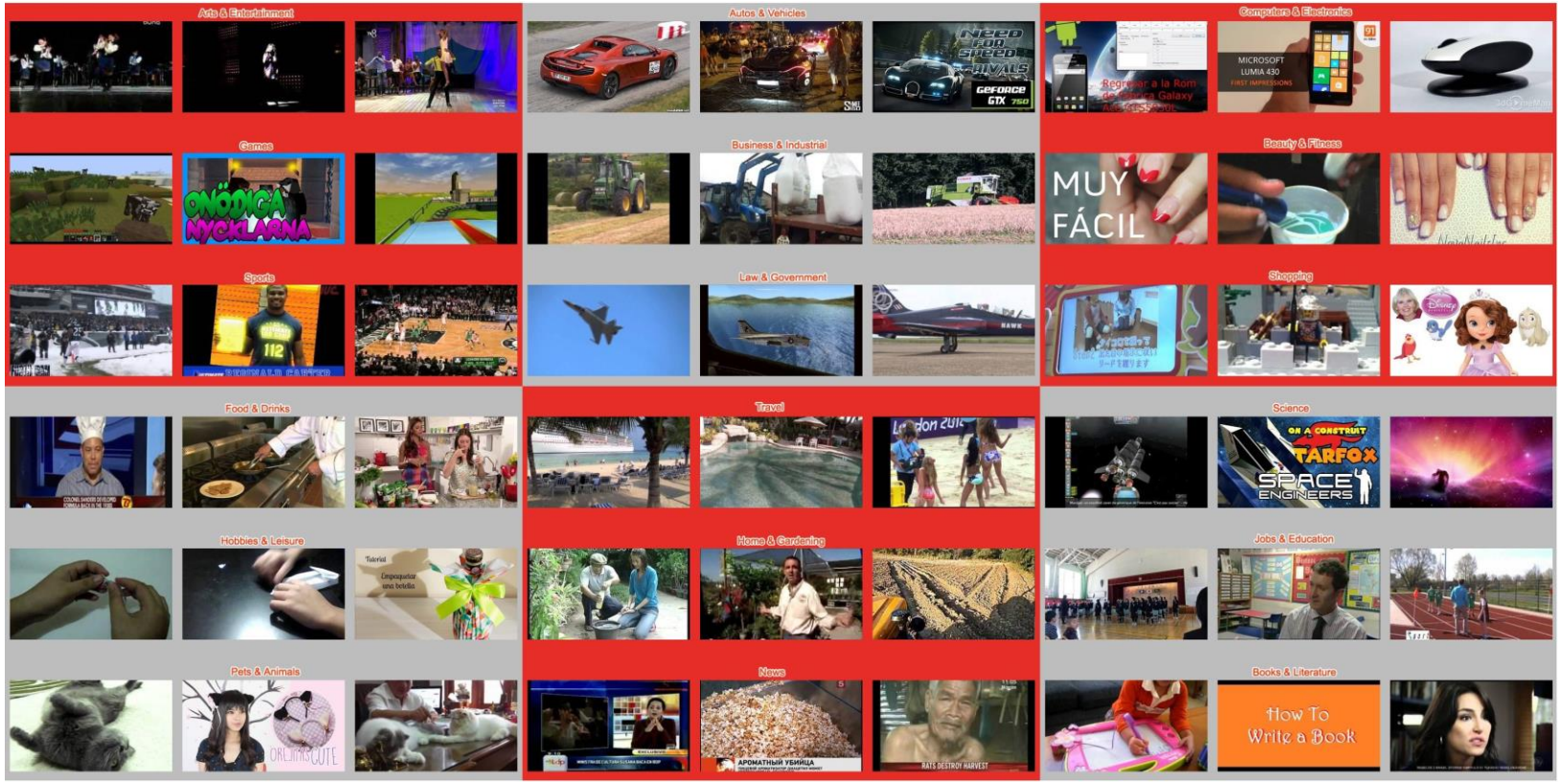
df_x = number of documents containing x

N = total number of documents

Deep Representation (Feature) Learning



Case Study: YouTube-8M dataset



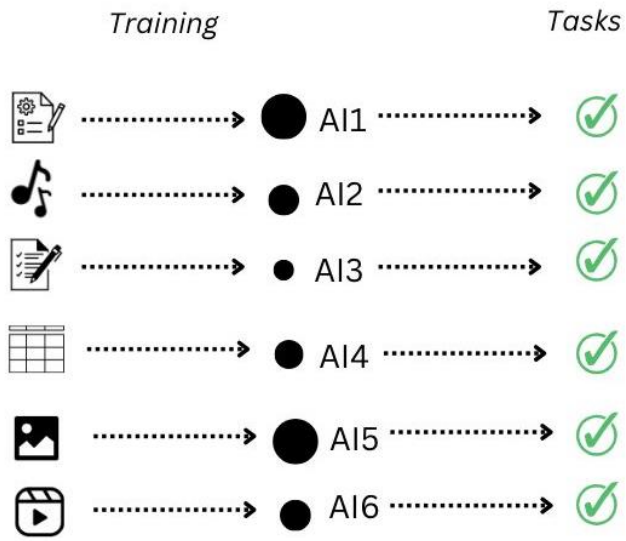
Only stores features instead of raw videos

https://research.google.com/youtube8m/

Recent Trend: Foundation Models

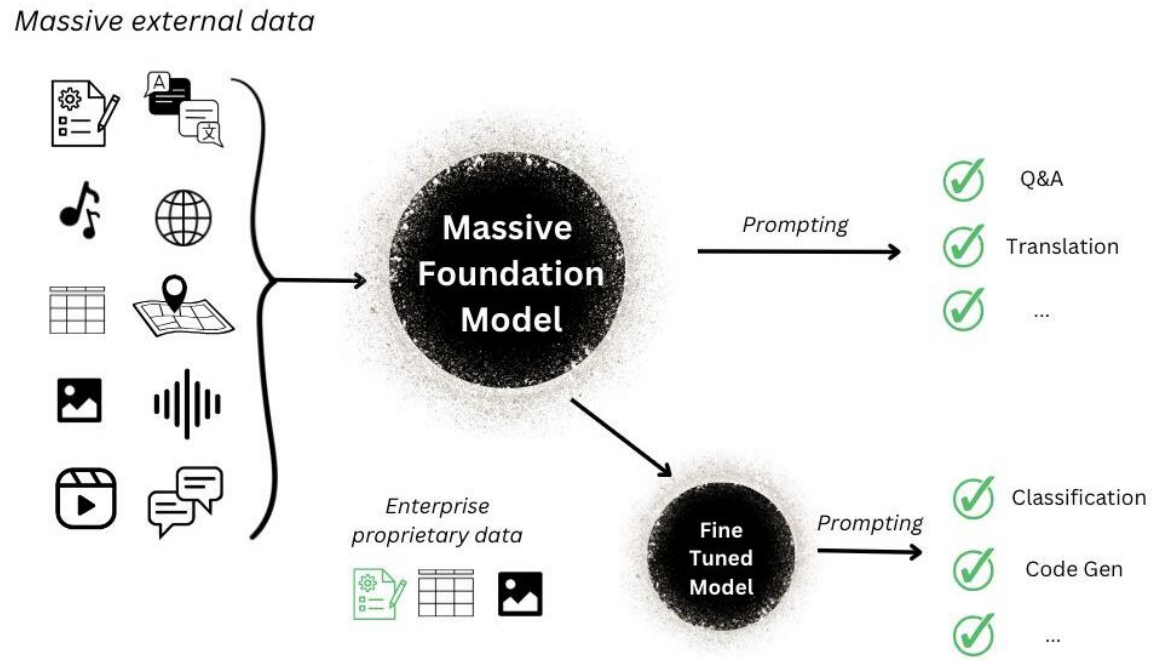
Foundation Models

Traditional ML



- Individual siloed models
- Require task-specific training
- Lots of human supervised training

Foundation Models



- Massive multi-tasking model
- Adaptable with little or no training
- Pre-trained unsupervised learning

Foundation Models

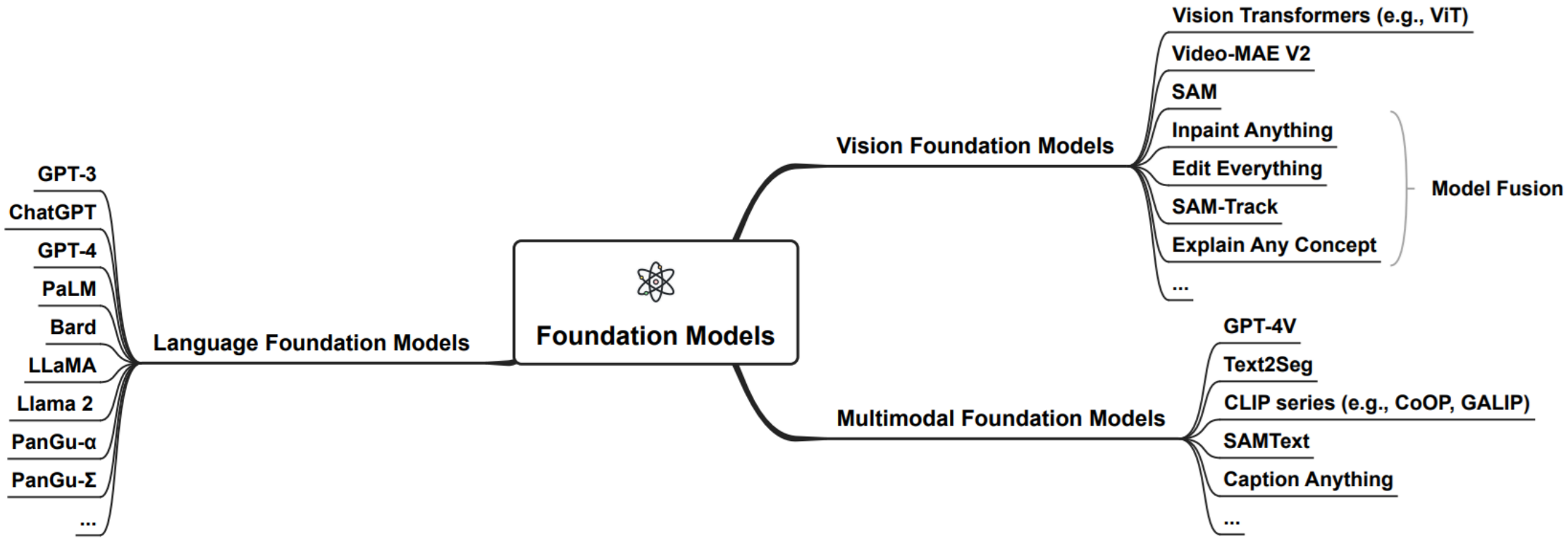


Image Encoder for Visual Features (DINOv3)

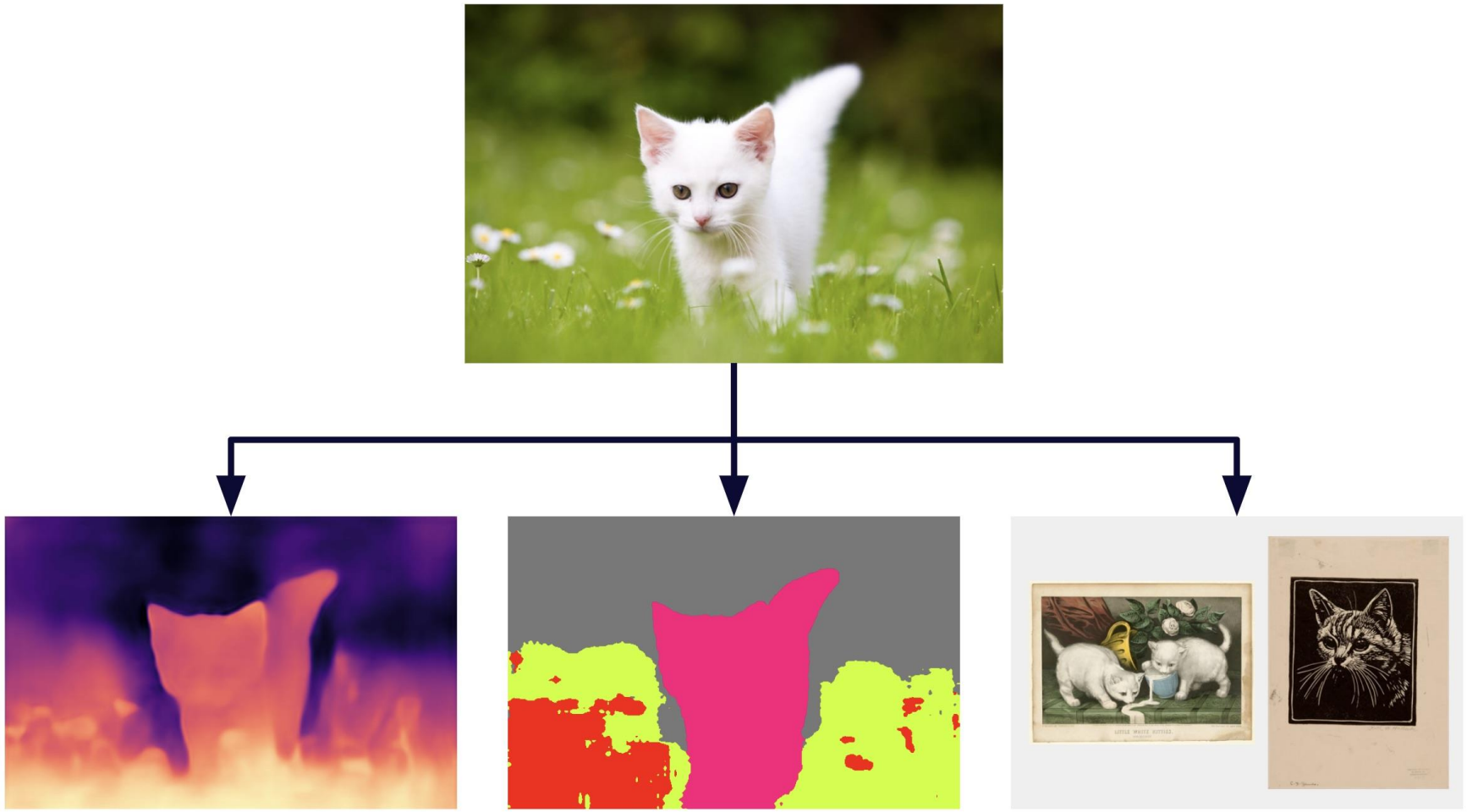
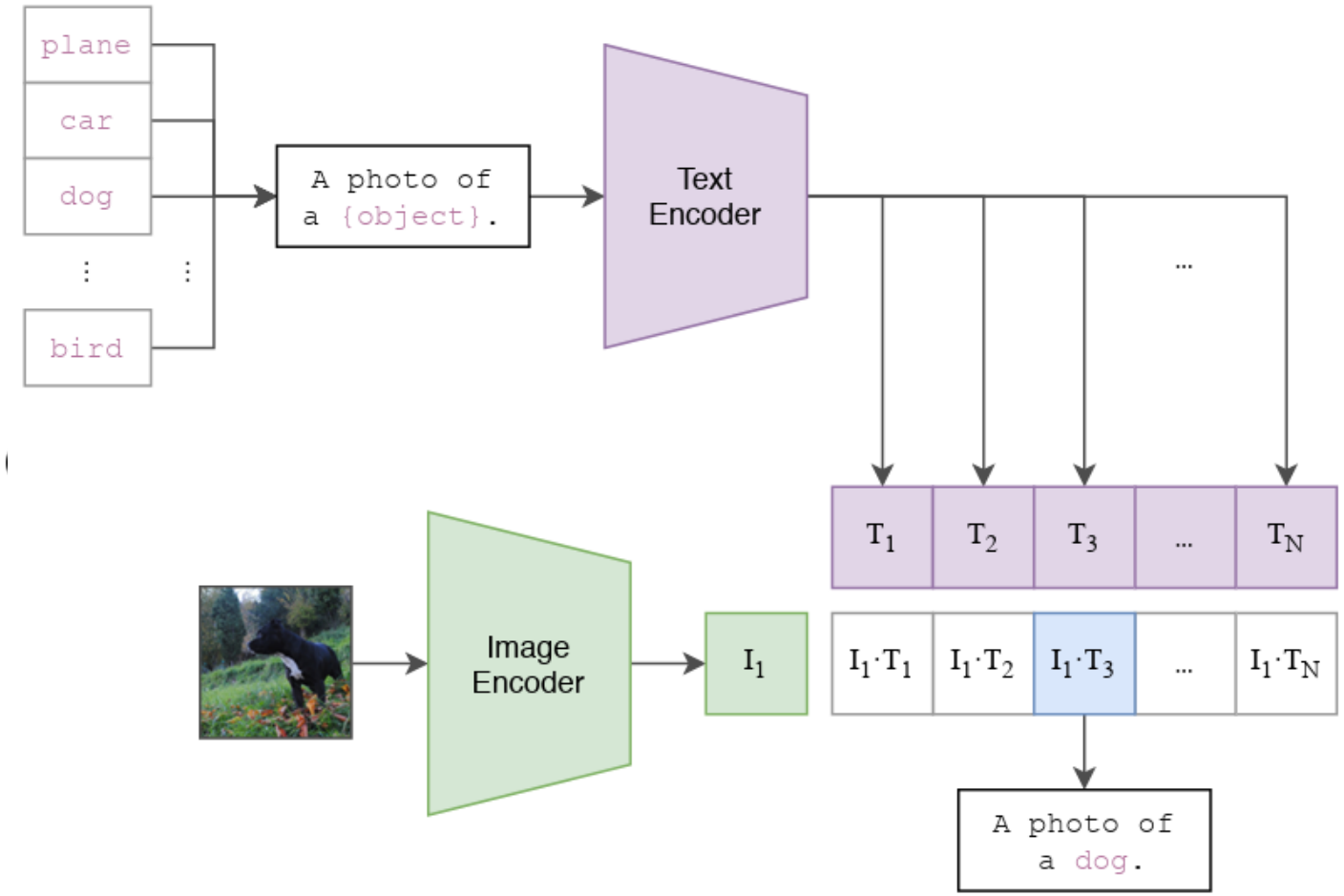
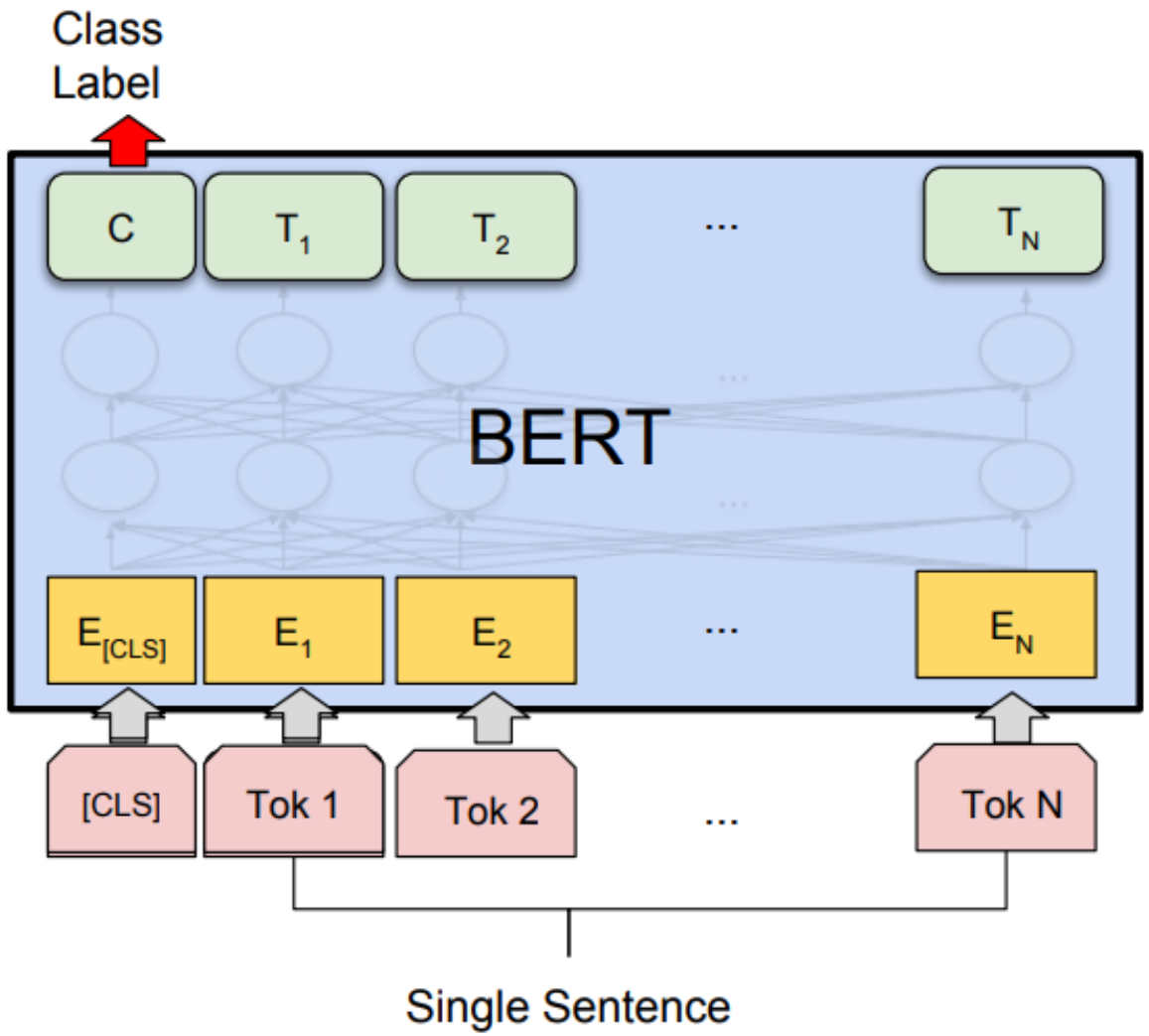


Image Encoder for Semantic Features (CLIP)



Text Encoder (BERT)



Outline: Data Pre-processing

