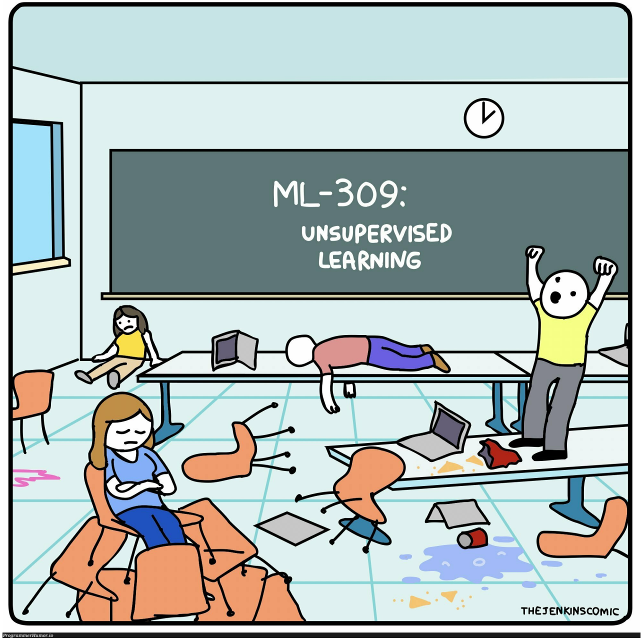
Unsupervised Learning

Clustering, Dimensionality Reduction, Data Preparation

kwekha.rostam.zhyar@nik.uni-obuda.hu

06

Unsupervised Learning



https://programmerhumor.io/programming-memes/unsupervised-learning/

Supervised vs. Unsupervised Learning

What is the difference?

- Supervised learning: The model learns from labeled data (x_i, y_i)
- Unsupervised learning: The model searches for structures and hidden patterns in unlabeled data $x_i \in X$

Examples:

- Supervised learning: Image classification (e.g., cat or dog?)
- Unsupervised learning: Automatic image grouping based on similarity without predefined labels

Formal Definition of Unsupervised Learning

Definition:

- Dataset: $D = \{x_n\}_{n=1}^{N}$, where $x_n \in \mathbb{R}^d$
- Data originates from an unknown distribution
- Goal: Learn useful properties from the unknown distribution

Hidden (latent) variables:

- Unsupervised learning often assumes that unobserved (hidden) variables explain the structure of the data

Clustering

Clustering

Definition:

- Discover hidden categories within the data
- Assign each data point a hidden label z_n

Example:

- Grouping documents by topic (politics, sports, economics)
- Automatically grouping images without knowing their content beforehand

Important: Clustering is different from classification: classification involves labeled data, while clustering groups data in an unsupervised manner.

K-Means

Task:

- Given a dataset $D = \{x_n\}_{n=1}^{N}$, where $x_n \in \mathbb{R}^d$
- The goal is to partition the data into K clusters
- The clusters are not predefined; they are discovered by the algorithm

Key Idea:

- Group elements based on their proximity
- Assign a prototype point to each cluster: μ_k, which represents the cluster center

Optimization Problem in K-Means

Formal Definition:

- Assign a hidden variable to each data point: which cluster does it belong to?
- Objective function:

$$\min_{\{z_n\},\{\mu_k\}} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{kn} d(x_n, \mu_k)$$

- Where:
  - $z_{kn} = 1$, if x_n belongs to the k-th cluster; otherwise, 0.
  - $d(x, \mu) = ||x - \mu||^2$ is the Euclidean distance.

Core Idea:

- Assign data based on minimal distance
- Continuously update cluster centers μ_k

Steps of the K-Means Algorithm

Expectation-Maximization (EM) Structure:

- 1. Initialization: Randomly select K cluster centers $\{\mu_k^{\text{old}}\}$
- 2. E-Step (Assignment):

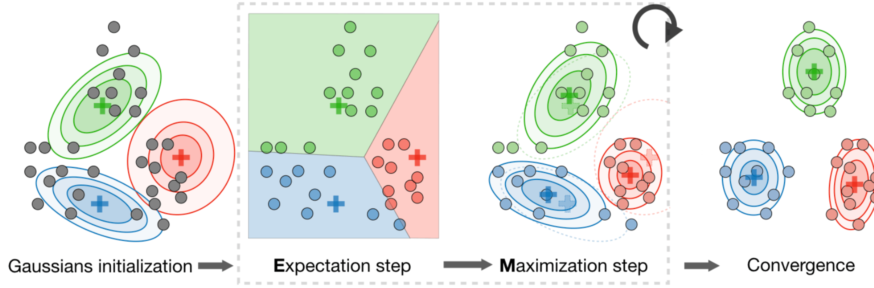
$$z_{kn}^{\text{new}} = \begin{cases} 1, & \text{if } k = \arg\min_j d(x_n, \mu_j^{\text{old}}) \\ 0, & \text{otherwise} \end{cases}$$

- 3. M-Step (Update):

$$\mu_k^{\text{new}} = \frac{\sum_{n=1}^{N} z_{kn}^{\text{new}} x_n}{\sum_{n=1}^{N} z_{kn}^{\text{new}}}$$

- 4. Convergence check: The algorithm runs until the new centers no longer change significantly.

EM Structure

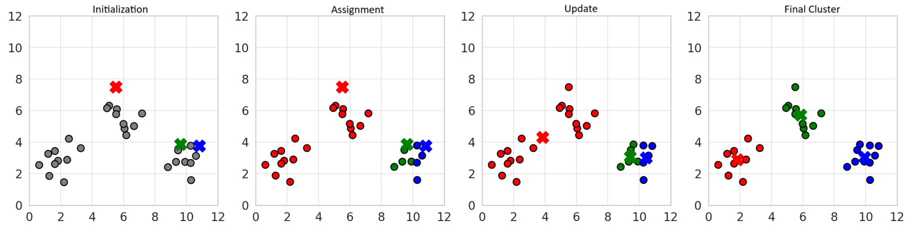


1. Figure: The steps of the EM algorithm structure: initialization, assignment (E), update (M), convergence check [1]

[1] https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-unsupervised-learning

K-Means Clustering Step by Step

Example: K=3 clusters, convergence step by step



The figure illustrates the steps of K-Means clustering:

- **Initialization**: Random centroids
- **Assignment**: Assign data points to the nearest cluster
- **Update**: Compute new cluster centroids
- After several iterations, final clusters are formed

How to Choose the Value of K?

The problem:

- Too small $K \rightarrow$ Too few clusters, poor grouping
- Too large $K \rightarrow$ Every data point could have its own cluster, making no sense

Determining K:

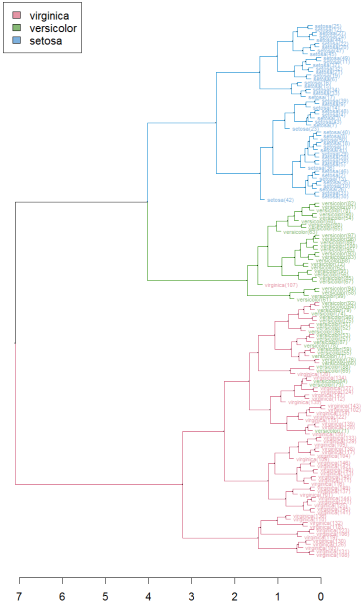
- Fine-tuning K until a predefined heuristic is met
- Hierarchical clustering (dendrogram): tree-structured representation

Hierarchical Clustering

Main Idea:

- Construct a nested hierarchy of clusters
- The result is a dendrogram showing the order of cluster merging
- Two main approaches:
  - The **agglomerative** approach builds from the bottom up—each sample starts as its own cluster and then merges
  - The **divisive** method works oppositely, starting from the top and gradually splitting into smaller clusters

Clustered Iris data set
(the labels give the true flower species)



https://cran.r-project.org/web/packages/dendextend/vignettes/Cluster_Analysis.html

Gaussian Mixture Model

K-Means vs GMM

- K-Means performs hard assignments—each point belongs to a single cluster
- GMM applies probabilistic clustering—each point has a probability of belonging to a cluster
- K-Means assumes spherical clusters, while GMM can model elliptical clusters

GMM:

- Assumes data comes from multiple Gaussian distributions
- Each cluster is modeled as a Gaussian distribution
- Uses the Expectation-Maximization (EM) algorithm for parameter optimization

Mathematical Model of GMM

Gaussian Distribution

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

GMM: The data comes from K Gaussian distributions:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

Explanation:

- π_k: Cluster weight (probability of belonging to a given cluster)
- μ_k: Cluster center (mean)
- Σ_k: Covariance matrix (variance and correlation)

Mathematical Model of GMM

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)$$

- $p(x)$ – The probability that a data point x comes from the entire mixture model.
- $\mathcal{N}(x \mid \mu_k, \Sigma_k)$ – The density function of the k-th Gaussian distribution, where:
  - μ_k: The mean of the distribution
  - Σ_k: Covariance matrix (shape, direction, variance)
- π_k – The weight of the k-th cluster (proportion of data belonging to it)

Σ_k determines the cluster:

- Size (larger variance = wider cluster),
- Shape (round or elongated),
- Orientation (can be rotated).

Expectation-Maximization Algorithm

- E-Step (Expectation): Compute probabilities of a point belonging to each cluster:

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i | \mu_j, \Sigma_j)}$$

- $\gamma_{ik} \in [0, 1]$, and $\sum_{k=1}^{K} \gamma_{ik} = 1$

Expectation-Maximization Algorithm

M-step: The parameters are updated based on the weighted average of γ_{ik}:

$$N_k = \sum_{i=1}^{N} \gamma_{ik}$$

$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} x_i$$

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} (x_i - \mu_k)(x_i - \mu_k)^T$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

Comparison of K-means and GMM

Similarities

- Both algorithms follow the **EM structure**:
  - **E-step:** assignment based on current parameters
  - **M-step:** update of cluster centroids
- Iterative convergence: each step reduces the loss function
- Assign data points to clusters

Differences

- **K-means:**
  - Hard assignment (x_i belongs to exactly one cluster)
  - Learns only cluster centroids (μ_k)
  - Assumes clusters are spherical
- **GMM:**
  - Soft assignment (probabilistic)
  - Learns cluster weights (π_k), centroids (μ_k), and covariances (Σ_k)
  - More flexible cluster shapes (e.g., ellipses)

DBSCAN – Density-Based Clustering

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- K-Means and GMM assume specific cluster shapes.
- DBSCAN does not require the number of clusters to be predefined.
- Handles clusters of different shapes and noise data.

Key concepts:

- **Core sample:** A point with at least minPoints neighbors within radius ϵ.
- **Border sample:** A neighbor of a core point, but not a core point itself.
- **Outlier:** A point that does not belong to any cluster.

DBSCAN Algorithm Steps

Algorithm steps:

1. Check the core point condition for each point: how many neighbors does it have within radius ϵ?
2. If a point is a core point, assign all reachable points to the same cluster.
3. If a point is a border point, assign it to the nearest cluster.
4. If a point is neither a core nor a border point, mark it as noise.

Visualization:
`https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/`

DBSCAN Advantages and Disadvantages

Advantages:

- Does not require a predefined number of clusters.
- Can handle arbitrarily shaped clusters, unlike K-Means, which assumes spherical clusters.
- Automatically detects noise data.

Disadvantages:

- Sensitive to parameters: Properly setting ϵ is crucial.
- Difficult to apply in high-dimensional spaces because distances become less informative.

Dimensionality Reduction

Dimensionality Reduction

Definition:

- Goal: Reduce the number of features while preserving the most important information
- A commonly used technique: Principal Component Analysis (PCA)

Why is it useful?

- Helps visualize data in lower dimensions
- Reduces noise and removes redundant data
- Improves computational efficiency on large datasets

Example:

- In an image database, retaining key colors and shapes while removing noisy details

Principal Component Analysis (PCA)

Goal: Reduce the dimensionality of data while preserving the maximum variance.

- PCA finds new axes (principal components) along which the variance of the data is maximized.
- The method projects the data onto these axes.
- Suitable for:
  - visualization (2D/3D),
  - noise reduction,
  - preprocessing before supervised learning.

PCA Algorithm Steps

1. **Normalize the data:** Scale each attribute so that its mean is 0 and standard deviation is 1.

$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

2. **Compute the covariance matrix:**

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} (x^{(i)})^T$$

3. **Compute principal components:** Eigenvectors and eigenvalues of Σ.
4. **Project:** Project data onto the axes corresponding to the largest eigenvalues.

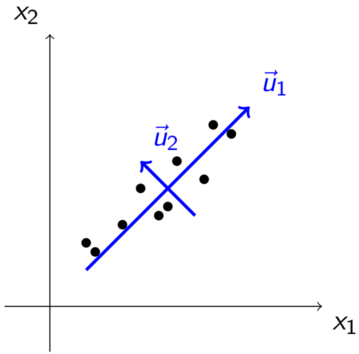
PCA: Motivation

Attributes are often correlated $\rightarrow$ redundant information is present.
Example: YouTube Content Creators

- Each data point represents a content creator.
- $x_1^{(i)}$: Technical skills (editing, audio, video quality).
- $x_2^{(i)}$: Enthusiasm/motivation (consistency, energy, authenticity).
- These characteristics are often correlated:
  - Motivated individuals tend to learn technical skills.
  - Good technical background enhances enjoyment and engagement.
- Goal: How can we describe these on a single scale?

PCA: Motivation

- The two features (technical knowledge and motivation) are highly correlated.
- We can assume that the data is distributed along a **diagonal axis**—this is the $\vec{u}_1$ direction.
- This direction may represent the combined characteristic of "content creator karma."
- Question: **How can we automatically find this axis?**



Step 1: Normalizing the Data

- For each j-th attribute:

$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

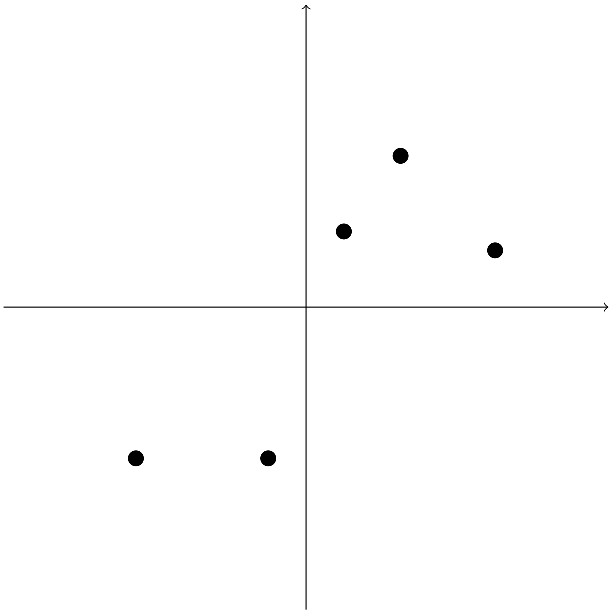
where

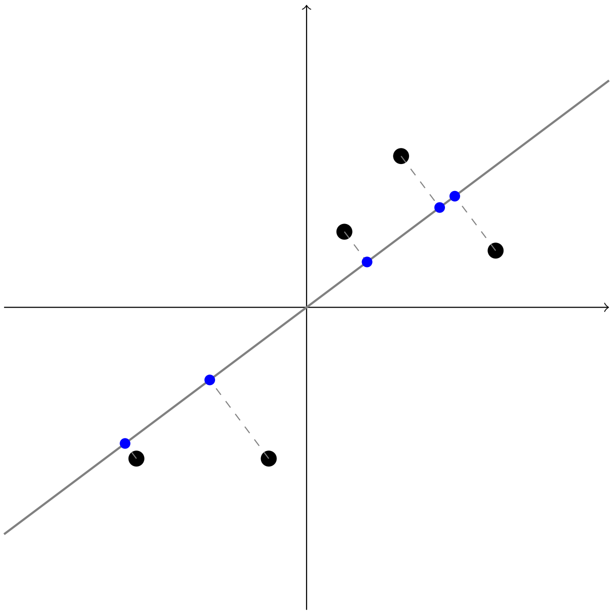
$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}, \quad \sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_j^{(i)} - \mu_j)^2$$

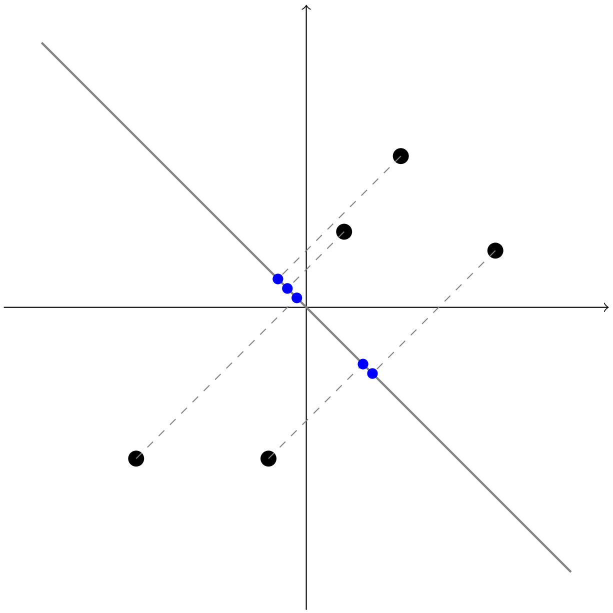
- Result: each attribute will have a mean of 0 and a standard deviation of 1.
- Why is this important?
  - Different measurement units can distort the analysis (e.g., speed *vs.* number of seats).
  - Scaling ensures that all features contribute equally.
- When can this step be skipped?
  - If we know that the data is already zero-centered (e.g., audio signals).
  - If all attributes are on the same scale (e.g., pixel values in an image).

Step 2: Determining the Principal Direction

- After normalizing the data, we seek the direction **u** that:
  - Is a **unit vector** ($\|\mathbf{u}\| = 1$),
  - and maximizes the **variance of the projected values**.
- Intuition:
  - The data contains a certain amount of variation (information).
  - The goal is to find the direction with the greatest variance—this represents the most significant "direction of change."
- This direction becomes the first principal component.





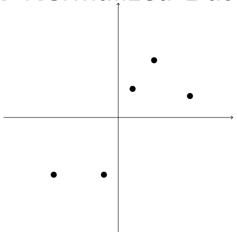


Selecting the Best Principal Component

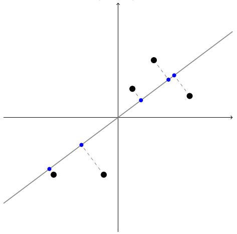
On a Sample Dataset:

- The dataset is already normalized.
- We examine two different axis directions (u) onto which we project the data.

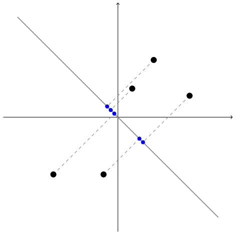
1. Normalized Data



2. Good Direction (u_1)



3. Poor Direction (u_2)



Summary:

- The first direction (u_1) maximizes the variance of the projected data.
- The second direction (u_2) does not align well with the data distribution.
- Goal: Automatically select the best u.

Principal Component Selection

Given an u unit vector and a data point x, the projection length is: $x^T u$
The goal is to maximize the variance of all projections:

$$\frac{1}{n}\sum_{i=1}^{n}(x^{(i)T}u)^2 =$$

$$\frac{1}{n}\sum_{i=1}^{n} u^T x^{(i)} x^{(i)T} u = u^T \left(\frac{1}{n}\sum_{i=1}^{n} x^{(i)} x^{(i)T}\right) u$$

The expression in parentheses is the covariance matrix:

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} x^{(i)} x^{(i)T}$$

This is an eigenvector-eigenvalue problem:

$$\Sigma u = \lambda u$$

Solution: u is the eigenvector corresponding to the largest eigenvalue of Σ.

Step 3: Covariance Matrix and Eigenvectors

- Compute the covariance matrix:

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} {x^{(i)}}^T$$

- This is a symmetric matrix, where:
  - Eigenvalues $\rightarrow$ Measure of variance
  - Eigenvectors $\rightarrow$ Principal component directions
- The first principal component (u_1) direction maximizes the variance along the data.

PCA Summary and Dimensionality Reduction

- If the goal is to find a 1-dimensional subspace for approximating the data, the optimal choice is:

$$u_1 = \text{Eigenvector corresponding to the largest eigenvalue}$$

- If we wish to project data onto a k-dimensional subspace $(k < d)$, we choose the first k eigenvectors:

$$u_1, u_2, \ldots, u_k$$

- The u_i vectors form a new, orthogonal basis.

Representing Data in the New Basis:

- The new k-dimensional representation of a data point $x^{(i)}$ is:

$$y^{(i)} = \begin{bmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_k^T x^{(i)} \end{bmatrix} \in \mathbb{R}^k$$

- The original $x^{(i)}$ was d-dimensional, now reduced to k-dimensions.
- The vectors $u_1, \ldots, u_k$ are called the **first k principal components** of the data.

Applications of PCA

1. **Dimensionality Reduction and Visualization**
   - If $k = 2$ or $k = 3$, the data can be visually represented.
2. **Preprocessing for Machine Learning Models**
   - Dimensionality reduction decreases computational costs.
   - Helps prevent overfitting: lower-dimensional inputs lead to simpler hypothesis classes.
3. **Noise Reduction and Pattern Recognition**
   - Face recognition: PCA projects 100x100 pixel faces into a lower-dimensional space (*eigenfaces* method).
   - Removes noise caused by small lighting variations and imaging differences.
   - Distance-based similarity measurement in the reduced-dimensional space: successful face comparison.

Anomaly Detection

What is anomaly detection?

- Algorithms that identify data points that significantly deviate from "normal" patterns.
- Typical applications: fraud detection, fault detection, network security.

Two popular solutions:

- **One-Class SVM:** defines a boundary around the data and considers points outside it as anomalies.
- **Isolation Forest:** isolates unusual points using random decision trees; efficient for large datasets.

Data Preparation

Data sets
in tutorials

Data sets in
the wild

https://nc233.com/tag/feature-engineering/

Data in Machine Learning

- Every ML project relies on a dataset.
- Possibility: open datasets (e.g., ImageNet: 14M+ labeled images, 20k+ categories).
- If no suitable dataset is available:
    - **Self-collection**: manually, via crowdsourcing, or automatically.
    - **Data from user behavior**.
    - **Transfer learning**: general model + fine-tuning on own data.

Selecting Suitable Data

- Is the data relevant to the task?
- Does it contain the required **output values**?
- Data size matters:
  - Is **more data collection** necessary?
  - Can some data be discarded to speed up processing?

How Much Data is Enough?

- Analyzing similar projects: estimation by analogy.
- **Learning curve** analysis: does performance still improve with more data?

Rule of thumb:

- Difficult tasks require millions of examples.
- For classification: at least hundreds or thousands of examples per class.
- At least 10x as many data points as input attributes.
- More data is required for nonlinear models than linear ones.
- Less data may be sufficient if regularization is used.

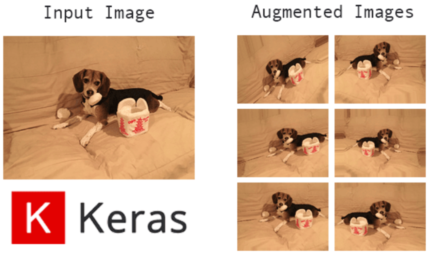
Data Cleaning: Potential Issues

- Are there errors in data entry?
- Missing fields? Need to determine how to handle them.
- Inconsistent textual data: `"Apple"` vs `"AAPL"` vs `"Apple Inc."`

Reliable data cleaning and preprocessing pipelines are the foundation of good models.

Data Augmentation

- Useful when data is scarce – especially for images, text, and speech.
- For images, for example:
    - Rotation, flipping.
    - Translation, cropping, scaling.
    - Adding noise, adjusting brightness.
- The label remains unchanged – making the model more robust.



https://pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/

Imbalanced Classes

- Example: among 10 million transactions, only 1000 are fraudulent.
- A "classifying all as valid" model achieves 99.99% accuracy – but is not useful.
- Solutions:
    - **Undersampling**: reducing the majority class.
    - **Oversampling**: duplicating the minority class.
    - **Weighted loss function**: higher penalty for misclassification.
    - **Boosting**: minority decisions may suffice.
    - **SMOTE, ADASYN**: generating synthetic examples.

Handling Outliers

- Outliers are distant data points (e.g., a \$316 price when others are around \$30).
- Linear models are sensitive to them.
- **Solutions:**
    - Logarithmic transformation: e.g., $20 \rightarrow 1.3$, $316 \rightarrow 2.5$
    - Decision trees, random forests are more robust to outliers.

What is Feature Engineering?

- The transformation or augmentation of attributes to make learning easier for the model.
- Goal: create **relevant, informative**, and **processable** features.
- Follows after correcting erroneous values.

Example:

- *Wait time* $\rightarrow$ discrete categories: 0–10, 10–30, 30–60, $>$60 minutes.
- *Date* $\rightarrow$ logical fields such as “weekend?” or “holiday?”

Typical Transformations and Encodings

- **Quantization (binning)**: continuous values $\rightarrow$ categories
- **Normalization**: standard deviation $= 1$ (particularly important for algorithms like k-NN)
- **One-hot encoding**: categories $\rightarrow$ binary fields
  - E.g., "weather = sunny/cloudy/rainy" $\rightarrow$ 3 separate attributes

Important: Some models (e.g., neural networks) can only handle numerical inputs.

Example of Quantization (Binning)

Problem: The "Wait Time" attribute is given in minutes:

$$\text{Wait Time (minutes)} = 4,\ 8,\ 12,\ 35,\ 50,\ 65$$

Goal: Transform into categories:

- `Short` (0–10 minutes)
- `Medium` (10–30 minutes)
- `Long` (30+ minutes)

Result:

$$4 \rightarrow \texttt{Short}, \quad 12 \rightarrow \texttt{Medium}, \quad 50 \rightarrow \texttt{Long}$$

Example of One-Hot Encoding

Attribute: Weather = `sunny`, `cloudy`, `rainy`
One-Hot Encoding:

Original Value	`sunny`	`cloudy`	`rainy`
sunny	1	0	0
cloudy	0	1	0
rainy	0	0	1

Note: Models can interpret the categories in numeric form this way.

Creating New Features with Expertise

Example: Real Estate Price Estimation

- Base feature: floor area $\rightarrow$ not enough.
- Additional features:
  - Number of rooms, bedrooms, bathrooms
  - Renovations, year built, condition
  - Heating, air conditioning, garden size, orientation

Environmental Factors:

- Postal code? School district? Average test scores?
- “Good neighborhood” is not always numerical – but can be treated as a feature.

The Importance of Good Features

- The quality of features is critical to the success of the model.
- A weak model can perform well with good features.
- Even a strong model cannot learn from poor representations.

Pedro Domingos:

> *"At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used."*

Thank you for your attention!

Sources:

- Osvaldo Simeone, A Brief Introduction to Machine Learning for Engineers.
- `https://scikit-learn.org/stable/modules/clustering.html`
- Andrew Ng, CS229 Lecture notes, Stanford University.
- Russell, Norvig, Artificial Intelligence: A Modern Approach, 4th Edition.