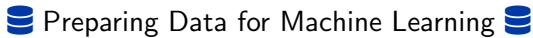
Data Preprocessing

Core Foundations for Machine Learning

Sarwan Ali

Department of Computer Science
Georgia State University



Today's Data Journey

Why Data Preprocessing Matters

⚠ **"Garbage In, Garbage Out"** ⚠

Real-world Data Issues:

- Inconsistent formats
- Missing values
- Outliers and noise
- Different scales
- Categorical variables
- Duplicate records

Impact on ML Models:

- Poor performance
- Biased predictions
- Convergence issues
- Unreliable results
- Computational inefficiency

Data scientists spend 80% of their time on data preprocessing!

Data Preprocessing Pipeline

Missing values, Outliers, Duplicates

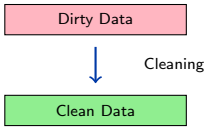
Scaling, Encoding, Feature Engineering

Data Cleaning Overview

Cleaning the Foundation

Common Data Quality Issues:

1. **Duplicate Records**
   - Exact duplicates
   - Near duplicates
2. **Inconsistent Data**
   - Different formats
   - Case sensitivity
   - Spelling variations
3. **Invalid Data**
   - Out-of-range values
   - Impossible combinations
4. **Outliers**



Handling Duplicates

Detection Strategies:
Exact Match:

- All features identical
- Simple to detect
- Use hash functions

Fuzzy Match:

- Similar but not identical
- Use similarity metrics
- Threshold-based decisions

Example Similarity Metrics:

$$\text{Jaccard} = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

$$\text{Cosine} = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}||\mathbf{b}|} \tag{2}$$

$$\text{Edit Distance} = \min \text{ operations} \tag{3}$$

Best Practice: Always investigate before removing duplicates!

Data Consistency

Common Inconsistencies:

Issue	Example	Solution
Case sensitivity	"USA", "usa", "Usa"	Standardize case
Date formats	"2023-01-15", "15/01/2023"	Unified format
Units	"5 km", "5000 m"	Convert to standard
Abbreviations	"Street", "St.", "St"	Expand or standardize
Encoding	UTF-8, ASCII issues	Consistent encoding

Validation Rules:

- Range checks: $18 \leq \text{age} \leq 120$
- Format validation: Email patterns, phone numbers
- Cross-field validation: End date $>$ Start date
- Domain constraints: Valid country codes

Types of Missing Data

? Understanding Missingness Patterns ?

1. Missing Completely At Random (MCAR)

- No pattern to missingness
- Independent of data values
- ✔ Safe to ignore

2. Missing At Random (MAR)

- Pattern depends on observed data
- Can be predicted from other variables
- ⚠ Handle carefully

3. Missing Not At Random (MNAR)

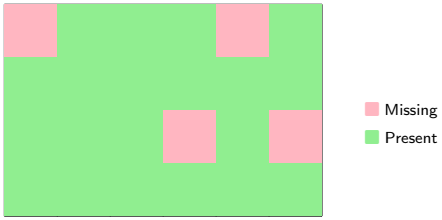
- Pattern depends on unobserved data
- Systematic reason for missingness
- ✖ Most problematic

Example:

- Income survey
- High earners don't respond
- Missingness $\propto$ actual income

Missing Value Detection

Visual Inspection Methods:



Missing Value Heatmap

Statistical Summary:

- Missing percentage per feature
- Correlation between missing patterns
- Little's MCAR test for randomness

Missing Value Strategies

1. **Deletion Methods:**
   - **Listwise deletion:** Remove entire rows
   - **Pairwise deletion:** Use available data
   - **When to use:** MCAR + sufficient data
2. **Imputation Methods:**
   - Simple: Mean, median, mode
   - Advanced: Regression, k-NN
   - Sophisticated: Multiple imputation

Simple Imputation Formulas:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4}$$

$$\text{median} = x_{(n+1)/2} \tag{5}$$

$$\text{mode} = \arg\max_x P(X = x) \tag{6}$$

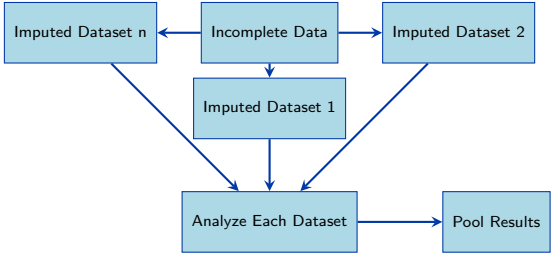
k-NN Imputation:

$$\hat{x}_i = \frac{1}{k}\sum_{j \in N_k(i)} x_j$$

where $N_k(i)$ are k nearest neighbors

Advanced Imputation Techniques

Multiple Imputation Process:



Matrix Factorization for Imputation:

$$X \approx UV^T \text{ where } U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{n \times k}$$

Minimize: $\|X_{observed} - (UV^T)_{observed}\|_F^2 + \lambda(\|U\|_F^2 + \|V\|_F^2)$

Why Feature Scaling?

📏 Bringing Features to the Same Scale 📏

Problem Example:

Age	Income ($)	Distance (km)
25	50,000	2.5
30	75,000	1.8
45	120,000	5.2

Issues with Different Scales:

- Distance-based algorithms dominated by large-scale features
- Gradient descent convergence problems
- Neural network training instability
- Feature importance misinterpretation

Income dominates Euclidean distance calculations!

Normalization vs Standardization

Min-Max Normalization:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Properties:

- Range: $[0, 1]$
- Preserves relationships
- Sensitive to outliers
- Good for bounded algorithms

Z-Score Standardization:

$$x_{std} = \frac{x - \mu}{\sigma}$$

Properties:

- Mean: 0, Std: 1
- Assumes normal distribution
- Less sensitive to outliers
- Good for algorithms assuming normality

Robust Scaling:

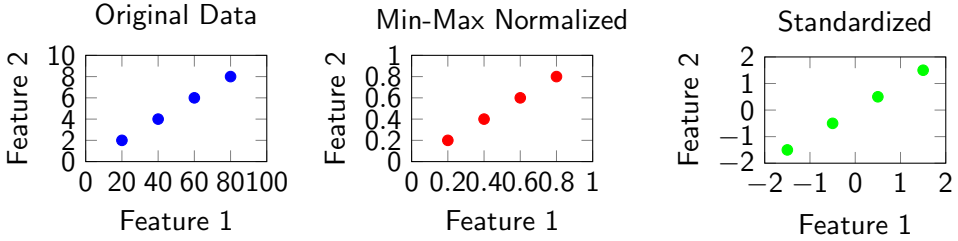
$$x_{robust} = \frac{x - \text{median}}{Q_3 - Q_1}$$

Unit Vector Scaling:

$$x_{unit} = \frac{x}{\|x\|_2}$$

Choice depends on data distribution and algorithm requirements!

Scaling Methods Comparison



When to Use Which Scaling?

Algorithm Type	Preferred Scaling	Reason
k-NN, k-Means	Min-Max or Standardization	Distance-based
SVM	Standardization	Assumes normal distribution
Neural Networks	Standardization	Gradient descent efficiency
Tree-based	None needed	Split-based decisions
PCA	Standardization	Variance-based
Logistic Regre.	Standardization	Coefficient interpretation

Important Notes:

- Apply same scaling to train/validation/test sets
- Fit scaler only on training data
- Store scaling parameters for new data
- Consider feature distributions when choosing method

Types of Categorical Variables

Handling Non-Numeric Data

Nominal Variables:

- No inherent order
- Examples: Color, Country, Gender
- Categories are mutually exclusive

Example:

Color
Red
Blue
Green

Ordinal Variables:

- Natural ordering exists
- Examples: Size, Rating, Education Level
- Relative ranking matters

Example:

Size	Order
Small	1
Medium	2
Large	3

Different encoding strategies needed for different types!

Label Encoding

Simple Integer Mapping:

Original	Label Encoded
Cat	0
Dog	1
Bird	2
Cat	0
Dog	1

Advantages:

- Simple and memory efficient
- Preserves all information
- Good for ordinal variables
- Works with tree-based algorithms

Disadvantages:

- Implies false ordering for nominal
- Can mislead distance-based algorithms
- Dog $-$ Cat $= 1$ has no meaning
- May create bias in linear models

Caution: Only use for ordinal variables or tree-based algorithms!

One-Hot Encoding

Binary Vector Representation:

Original	Cat	Dog	Bird
Cat	1	0	0
Dog	0	1	0
Bird	0	0	1
Cat	1	0	0
Dog	0	1	0

Mathematical Representation: For category c_i in variable X with k categories:

$$X_{one-hot} = [x_1, x_2, \ldots, x_k] \text{ where } x_j = \begin{cases} 1 & \text{if } X = c_j \\ 0 & \text{otherwise} \end{cases}$$

Advantages:

- No false ordering
- Works with all algorithms
- Interpretable coefficients
- Mathematically sound

Disadvantages:

- Increases dimensionality
- Sparse matrices
- Multicollinearity (dummy variable trap)
- Memory intensive for high cardinality

Advanced Encoding Techniques

1. **Target Encoding (Mean Encoding):**

$$\text{encoded}(c_i) = \frac{\sum_{j:x_j=c_i} y_j}{\sum_{j:x_j=c_i} 1}$$

- Replace category with target mean
- Risk of overfitting
- Use cross-validation or smoothing

2. **Binary Encoding:**

Category	Index	Binary	Encoded
Cat	0	00	[0, 0]
Dog	1	01	[0, 1]
Bird	2	10	[1, 0]
Fish	3	11	[1, 1]

3. **Hashing:**

$$\text{hash}(\text{category}) \mod n = \text{bucket index}$$

Useful for high cardinality, but may cause collisions.

Handling High Cardinality

Problem: Categories with many unique values

Strategies:

1. Frequency-Based Grouping:

- Keep top N categories
- Group rare categories as "Other"
- Based on occurrence threshold

2. Domain Knowledge:

- Logical groupings
- Hierarchical categories
- Business-driven consolidation

3. Embedding Techniques:

- Entity embeddings
- Learned representations
- Dimensionality reduction

Example - Cities:

- 1000+ unique cities
- Group by region/country
- Keep top 50, rest as "Other"

Rule of thumb: If cardinality ¿ 10-15, consider grouping strategies

Creating Better Features from Raw Data

What is Feature Engineering?

- Creating new features from existing ones
- Domain knowledge application
- Improving model performance
- Making patterns more obvious

Types of Feature Engineering:

- Mathematical transformations
- Interaction features
- Temporal features
- Aggregations

Example Transformations:

$$\text{BMI} = \frac{\text{Weight}}{\text{Height}^2} \tag{7}$$

$$\text{Age Group} = \lfloor \frac{\text{Age}}{10} \rfloor \tag{8}$$

$$\text{Interaction} = X_1 \times X_2 \tag{9}$$

$$\log(X) = \ln(X + 1) \tag{10}$$

Benefits:

- Better model interpretability
- Improved prediction accuracy
- Reduced training time

Mathematical Transformations

Common Transformations:

1. **Polynomial Features:**

$$X_{poly} = [x, x^2, x^3, \ldots, x^n]$$

2. **Logarithmic:** $X_{log} = \log(x + c)$ where c prevents $\log(0)$
3. **Square Root:** $X_{sqrt} = \sqrt{x}$
4. **Reciprocal:**

$$X_{recip} = \frac{1}{x + \epsilon}$$

5. **Binning/Discretization:**

$$X_{bin} = \begin{cases} 1 & \text{if } x \in [a_1, a_2) \\ 2 & \text{if } x \in [a_2, a_3) \\ \vdots & \vdots \\ n & \text{if } x \in [a_{n-1}, a_n] \end{cases}$$

6. **Trigonometric:**

$$X_{trig} = [\sin(x), \cos(x), \tan(x)]$$

Useful for cyclical features (time, angles)

Choose transformations based on data distribution and domain knowledge

Temporal Feature Engineering

Extracting Information from Timestamps:

Original	Feature	Example
2023-07-15 14:30:00	Year	2023
	Month	7
	Day	15
	Hour	14
	Day of Week	Saturday
	Quarter	Q3
	Is Weekend	True
	Season	Summer

Advanced Temporal Features:

- Time since last event: $t_{current} - t_{last}$
- Rolling statistics: Moving averages, standard deviations
- Lag features: Values from $t-1, t-2, \ldots$
- Cyclical encoding: $\sin(2\pi \cdot \frac{\text{hour}}{24})$, $\cos(2\pi \cdot \frac{\text{hour}}{24})$

Data Quality Metrics

Measuring Data Quality

1. **Completeness:** $C = \frac{\text{Non-null values}}{\text{Total values}} \times 100\%$
2. **Consistency:** $CS = \frac{\text{Consistent records}}{\text{Total records}} \times 100\%$
3. **Accuracy:** $A = \frac{\text{Correct values}}{\text{Total values}} \times 100\%$
4. **Uniqueness:** $U = \frac{\text{Unique records}}{\text{Total records}} \times 100\%$
5. **Validity:** $V = \frac{\text{Valid format values}}{\text{Total values}} \times 100\%$
6. **Timeliness:** $T = \frac{\text{Current records}}{\text{Total records}} \times 100\%$

Overall Data Quality Score:

$$DQ = w_1 \cdot C + w_2 \cdot CS + w_3 \cdot A + w_4 \cdot U + w_5 \cdot V + w_6 \cdot T$$

where $\sum w_i = 1$ and weights reflect business importance.

Outlier Detection Methods

Statistical Methods:

1. Z-Score Method: $z = \frac{x-\mu}{\sigma}$
Outlier if $|z| >$ threshold (typically 2-3)

2. IQR Method:

$$IQR = Q_3 - Q_1$$

$$\text{Lower bound} = Q_1 - 1.5 \times IQR$$

$$\text{Upper bound} = Q_3 + 1.5 \times IQR$$

3. Modified Z-Score:

$$M_i = \frac{0.6745(x_i - \tilde{x})}{MAD}$$

where MAD is median absolute deviation

Machine Learning Methods:

1. Isolation Forest:

- Tree-based anomaly detection
- Isolates outliers faster
- Works well in high dimensions

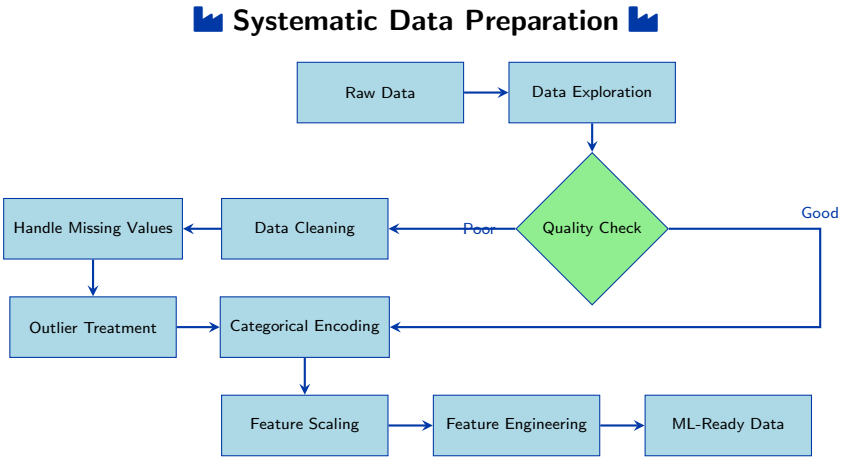
2. Local Outlier Factor (LOF):

$$LOF_k(x) = \frac{\sum_{y \in N_k(x)} \frac{lrd_k(y)}{lrd_k(x)}}{|N_k(x)|}$$

3. One-Class SVM:

- Learns normal data boundary
- Kernel-based approach
- Good for complex patterns

Building a Preprocessing Pipeline



Pipeline Implementation Best Practices

Key Principles:

1. **Reproducibility:**
   - Set random seeds
   - Document all steps
   - Version control transformations
   - Save preprocessing parameters
2. **Data Leakage Prevention:**
   - Fit only on training data
   - Transform test data using training parameters
   - No future information in features
   - Separate validation properly
3. **Modular Design:**
   - Separate functions for each step
   - Reusable components
   - Easy to modify and debug
   - Pipeline orchestration tools
4. **Performance Optimization:**
   - Vectorized operations
   - Memory-efficient processing
   - Parallel processing where possible
   - Caching intermediate results

Critical: Never fit preprocessing on test data!

Common Preprocessing Mistakes

⚠ Pitfalls to Avoid ⚠

1. **Data Leakage:** Using future information or test statistics
2. **Inconsistent Preprocessing:** Different steps for train/test
3. **Over-Engineering:** Creating too many irrelevant features
4. **Ignoring Domain Knowledge:** Purely algorithmic approach
5. **Not Handling Rare Categories:** Issues with unseen categories
6. **Scaling Before Splitting:** Computing statistics on entire dataset
7. **Ignoring Missing Patterns:** Not understanding why data is missing
8. **One-Size-Fits-All:** Same preprocessing for different algorithms

Remember: Good preprocessing is algorithm and problem specific!

Key Takeaways

Data Preprocessing Essentials

Core Steps:

- Data cleaning and quality assessment
- Missing value handling strategies
- Appropriate scaling techniques
- Categorical variable encoding
- Feature engineering opportunities

Critical Success Factors:

- Domain knowledge integration
- Algorithm-specific preprocessing
- Avoiding data leakage
- Systematic pipeline approach

Decision Framework:

- Analyze data distribution
- Understand missing patterns
- Choose appropriate techniques
- Validate preprocessing impact
- Monitor for concept drift

Quality Metrics:

- Completeness, consistency
- Accuracy and validity
- Model performance improvement
- Interpretability preservation

Great preprocessing is the foundation of successful machine learning!

Questions?

sali85@student.gsu.edu

Next: Types of Machine Learning: Supervised, unsupervised, and reinforcement learning overview