Data Preprocessing Basics

Definition: The process of cleaning, transforming, and organizing raw data into a format suitable for machine learning algorithms.

Why it matters:

- Real-world data is often messy, incomplete, or inconsistent
- Most ML algorithms expect clean, structured input
- Poor preprocessing leads to poor model performance
- Can consume 60-80% of a data scientist's time

Common steps:

- Data cleaning (removing duplicates, fixing errors)
- Data transformation (scaling, encoding)
- Feature selection/engineering
- Data splitting

Handling Missing Data

Types of missing data:

- **MCAR** (Missing Completely at Random): Missing values are independent of observed/unobserved data
- **MAR** (Missing at Random): Missing depends on observed data but not unobserved
- **MNAR** (Missing Not at Random): Missing depends on the unobserved values themselves

Strategies:

Deletion methods:

- Listwise deletion: Remove entire rows with any missing values
- Pairwise deletion: Use available data for each analysis
- *Pros*: Simple, preserves data distribution if MCAR
- *Cons*: Reduces sample size, potential bias

Imputation methods:

- Mean/median/mode imputation: Replace with central tendency
- Forward fill/backward fill: Use previous/next valid observation
- Interpolation: Estimate based on surrounding values
- K-nearest neighbors: Use similar observations
- Multiple imputation: Create multiple datasets with different imputations
- Model-based: Use algorithms to predict missing values

Best practices: Always analyze missingness patterns before choosing strategy

Label Encoding & Normalization

Label Encoding

Purpose: Convert categorical variables into numerical format for ML algorithms

Methods:

- **Ordinal encoding**: Assign integers (0,1,2...) - use when categories have natural order
- **One-hot encoding**: Create binary columns for each category - use for nominal data
- **Binary encoding**: Convert to binary representation - memory efficient for high cardinality
- **Target encoding**: Replace categories with target variable statistics

Normalization/Standardization

Purpose: Scale numerical features to similar ranges

Min-Max Scaling (Normalization):

- Formula: (x - min) / (max - min)
- Result: Values between 0 and 1
- Sensitive to outliers

Z-score Standardization:

- Formula: (x - mean) / standard deviation
- Result: Mean=0, std=1
- Less sensitive to outliers

Robust Scaling:

- Uses median and interquartile range
- Very robust to outliers

When to use: Apply when features have different scales/units, required for algorithms like SVM, neural networks, k-means

Train/Test Split

Purpose: Create separate datasets for training and evaluating model performance

Basic split:

- Training set: Used to train the model (typically 70-80%)
- Test set: Used for final evaluation (typically 20-30%)
- **Critical**: Test set should never be used during model development

Train/Validation/Test split:

- Training: Fit model parameters
- Validation: Tune hyperparameters, model selection

- Test: Final unbiased performance estimate
- Common ratios: 60/20/20 or 70/15/15 or 70/20/10

Cross-validation:

- K-fold: Split training data into k subsets, train on k-1, validate on 1
- Provides more robust performance estimates
- Useful when data is limited

Stratified splitting:

- Ensures each split maintains the same proportion of target classes
- Essential for imbalanced datasets
- Helps prevent bias in performance estimates

Key considerations:

- Random seed for reproducibility
- Time-based splits for temporal data
- Avoid data leakage between sets
- Ensure splits are representative of the full dataset

Implementation tip: Always perform train/test split BEFORE any preprocessing to avoid data leakage.