Data Preprocessing

Essential Steps Before Machine Learning And Data Analysis

By: Abdelrhman Khalil

Agenda

- Understanding the data
- Data cleaning (missing values, duplicates, outliers)
- Data transformation (encoding, scaling)
- Feature engineering & selection
- Train/Test split & validation
- Final checks, saving, and best practices
- Practical visual examples

What is Data Preprocessing?

- Preparing raw data so models can learn from it
- Cleaning, transforming, and organizing data
- A critical step: 'Garbage in → Garbage out'

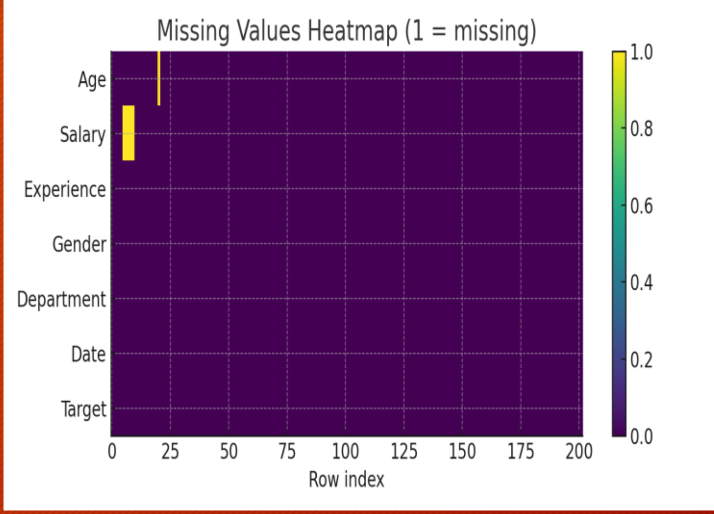
Why Data Preprocessing Matters

- Improves model performance and reliability
- Reduces errors, speeds up training
- Helps models generalize to new/unseen data

Step 1: Data Understanding

- Check number of rows and columns
- Identify feature types: numeric, categorical, date, text
- Locate the target variable and missing values

Missing Values - Detection



- Look for NaN or empty values
- Use summary counts and visual checks (heatmap of missing values)
- Find patterns: Random or systematic missingness?

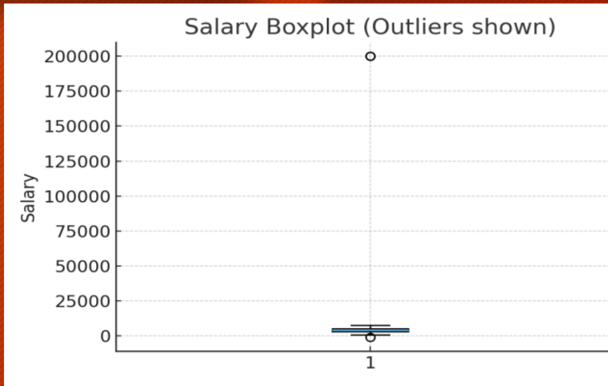
Missing Values - Handling

- Remove rows/columns with many missing values
- Simple imputation: mean / median / mode
- Advanced: model-based imputation (KNN, regression)
- Time-series: forward-fill or backward-fill

Duplicates

- Detect exact duplicate rows and remove them
- Check unique keys (IDs) to avoid accidental removal
- Duplicates can be valid (e.g., repeated purchases) — check context

Outliers - Detection



- Use boxplots, scatter plots, or statistical tests (Z-score, IQR)
- Visual checks often reveal measurement errors or true extreme values

Outliers - Handling

- Remove if clearly bad data (e.g., entry mistake)
- Cap/Winsorize values to reduce effect
- Transform values (log, sqrt) to reduce skew
- Keep if they are meaningful (fraud, rare events)

Encoding Categorical Data

- Label Encoding: convert categories to integers (use for ordinal data)
- One-Hot Encoding: binary columns for each category (use for nominal data)
- Target Encoding: replace category with average target (use with caution)

Feature Scaling

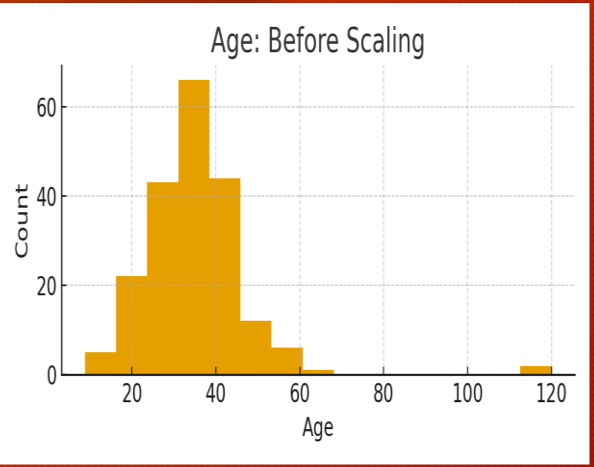
- **Standardization (Z-score):**

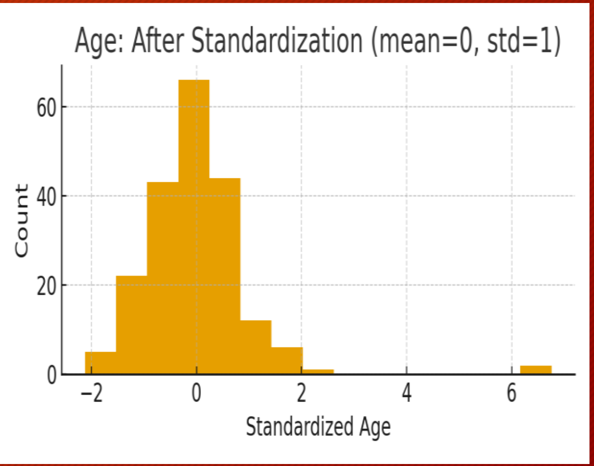
  mean=0, std=1 — good for many models

- **Normalization (Min-Max):**

  scale between 0 and 1 — useful for neural networks

- **Robust Scaler:** uses median and IQR — resistant to outliers





Other Transformations

- Log or sqrt transforms to reduce skewness
- **Binning (discretization):** convert continuous → categorical
- **Datetime features:** extract year, month, weekday, hour
- **Text features:** tokenization, TF-IDF, embeddings

  (for text data)

Feature Engineering

- **Create new features:**

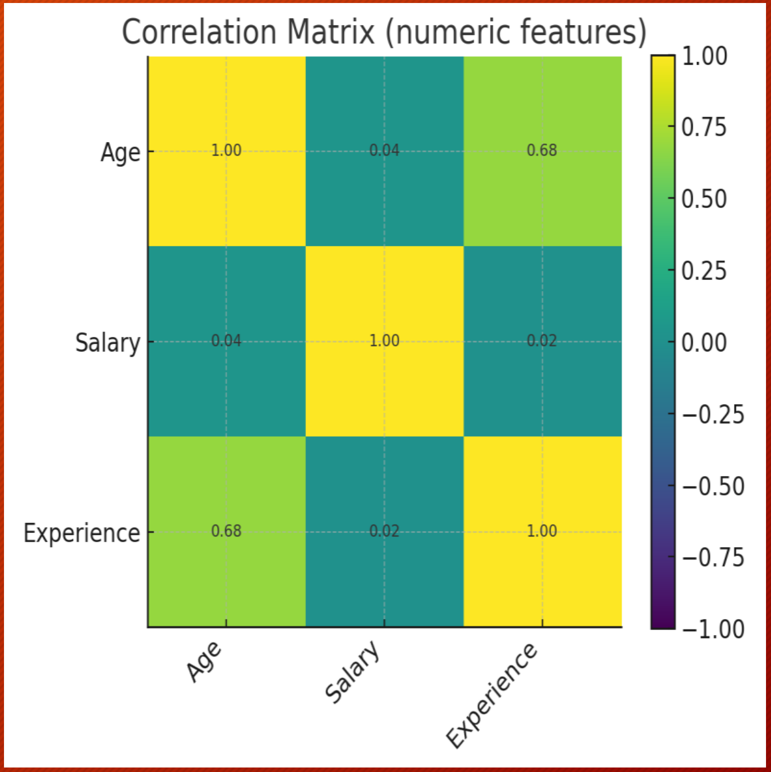
  e.g., Speed = Distance ÷ Time

- **Combine features:**

  ratios, differences, aggregations

- **Domain knowledge:**

  often suggests best features



Feature Selection

- **Filter methods:** correlation, chi-square, mutual information
- **Wrapper methods:** recursive feature elimination (RFE)
- **Embedded methods:** feature importance from tree models
- **Dimensionality reduction:** PCA (careful with interpretability)

Train/Test Split & Validation

- **Hold-out:** train (70-80%) vs test (20-30%)
- Use stratified split if classes are imbalanced
- Validation set or cross-validation for hyperparameter tuning (k-fold)

Final Checks before Modeling

- No missing values remain in features used by the model
- All features are numeric or properly encoded
- Check class balance, and handle imbalance if necessary (resampling)
- Document the preprocessing steps and save transformations (scalers, encoders)

Saving & Reproducibility

- Save raw data and processed data separately
- Save encoders/scalers (pickle or joblib) for future use
- Use version control for code and data where possible (Git)
- Write README with the preprocessing steps

Tools & Libraries

- **Python libraries:** pandas, numpy, scikit-learn, matplotlib
- **Visualization:** matplotlib (or seaborn for advanced users)
- **Pipelines:** scikit-learn Pipeline and ColumnTransformer
- **Others:** Excel/Google Sheets for small quick checks; Jupyter Notebooks for exploration

Practice & Resources

- **Datasets:** Kaggle, UCI Machine Learning Repository
- **Practice projects:** Titanic dataset, House Prices, Iris
- **Learn by doing:** build a small pipeline and document each step

Common Mistakes & Tips

- Imputing before splitting (risk of data leakage)
- Scaling the entire dataset before train/test split
- Removing outliers blindly without checking context
- Using one-hot on very high-cardinality columns without strategy

Conclusion

Steps:

- Understand
- Clean
- Transform
- Transform
- Split
- Check
- Save

By: Abdelrhman Khalil