Classwork 3: Data Preprocessing

Categorical Encoding, Missing Values, and Transformations

Prof. Asc. Endri Raco, Ph.D.

November 2025

Overview

Objective: Master data preprocessing techniques essential for building robust predictive models, including handling categorical variables, missing values, outliers, and variable transformations.

Dataset: `donor_transactions.csv` (same as Classwork 2)

Submission: R Markdown file (.Rmd) + knitted PDF

Due Date: November 27, 2025, 23:59

Weight: 15% of final grade

Part 1: Creating Dummy Variables (20 points)

Task 1.1: Single Categorical Variable (8 points)

You have a `segment` variable with three categories: Gold, Silver, Bronze.

```
# Load required libraries
library(tidyverse)
library(fastDummies)

# Load your basetable from Classwork 2
basetable <- read_csv("basetable_classwork2.csv")

# Examine the segment variable
table(basetable$segment)
```

Tasks:

a) Create dummy variables for `segment` using the `fastDummies` package
b) Ensure you drop one category to avoid multicollinearity
c) Remove the original `segment` column
d) Show the first 10 rows with donor_id and the new dummy variables

Questions to answer:

- Which category did you choose as reference? Why?
- How many dummy variables were created?
- What does it mean when all dummy variables equal 0?

Deliverable: Code + output table + written explanations

Task 1.2: Multiple Categorical Variables (7 points)

Create dummy variables for multiple categorical columns simultaneously.

```
# Identify all categorical variables in your basetable
# Likely candidates: segment, country, gender, etc.

# Create dummies for all categorical variables at once
categorical_vars <- c("segment", "country")  # Add more if you have them

basetable_with_dummies <- dummy_cols(
  basetable,
  select_columns = categorical_vars,
  remove_first_dummy = TRUE,
  remove_selected_columns = TRUE
)
```

Tasks:

a) Create dummy variables for at least 2 categorical variables
b) Document which reference category was chosen for each variable
c) Count total number of dummy columns created
d) Verify no multicollinearity by checking correlation between dummies

Deliverable:

- Summary table showing original categories and resulting dummies
- Explanation of reference category selection
- Verification that no perfect correlation exists

Task 1.3: Manual Dummy Creation (5 points)

Create dummy variables manually to understand the process.

```
# Manual approach for segment variable
basetable$segment_Gold <- as.integer(basetable$segment == "Gold")
basetable$segment_Silver <- as.integer(basetable$segment == "Silver")
# Bronze is reference (both equal 0)

# Verify your manual dummies match the package output
```

Questions:

a) Why use `as.integer()` instead of just TRUE/FALSE?
b) How would you manually create dummies for a variable with 5 categories?
c) What happens if a donor's segment is NA?

Deliverable: Working code + answers to conceptual questions

Part 2: Handling Missing Values (25 points)

Task 2.1: Missing Value Assessment (7 points)

First, understand the extent and patterns of missing data.

```r
# Count missing values per column
missing_summary <- data.frame(
  variable = names(basetable),
  n_missing = sapply(basetable, function(x) sum(is.na(x))),
  pct_missing = sapply(basetable, function(x)
    round(100 * sum(is.na(x)) / length(x), 2))
)

# Sort by percentage missing
missing_summary <- missing_summary[order(-missing_summary$pct_missing), ]
```

Tasks:

a) Create the missing value summary table
b) Identify variables with >10% missing values
c) Create a visualization (bar chart) of missing percentages
d) Discuss patterns: Are missing values random or systematic?

Deliverable:

- Missing value summary table
- Visualization of missing data
- Written analysis of missingness patterns

Task 2.2: Imputation Strategy Selection (8 points)

Develop appropriate imputation strategies for different variable types.

```r
# Strategy 1: Mean imputation for roughly normal variables
# Example: age
if (sum(is.na(basetable$age)) > 0) {
  mean_age <- mean(basetable$age, na.rm = TRUE)
  basetable$age_imputed <- basetable$age
  basetable$age_imputed[is.na(basetable$age_imputed)] <- mean_age
}

# Strategy 2: Median imputation for skewed variables
# Example: max_donation

# Strategy 3: Zero imputation for count variables
# Example: donations_last_year
```

Tasks:

a) For EACH variable with missing values, justify your imputation strategy:

   - Mean: Use for approximately normal distributions
   - Median: Use for skewed distributions with outliers
   - Zero: Use when missing means "absence" (e.g., no donations)
   - Mode: Use for categorical variables

b) Implement your chosen strategies for at least 5 variables

c) Create "before and after" histograms for 3 numeric variables

Deliverable:

- Imputation strategy table with justifications
- Code implementing all strategies
- Before/after visualizations
- Discussion of how imputation affects distributions

Task 2.3: Missing Value Indicators (5 points)

Create indicator variables that flag when values were missing.

```r
# Create missing indicators BEFORE imputing
basetable$missing_email <- as.integer(is.na(basetable$email))
basetable$missing_phone <- as.integer(is.na(basetable$phone))
basetable$missing_age <- as.integer(is.na(basetable$age))

# Analyze: Are donors with missing email less likely to donate?
aggregate(target ~ missing_email, data = basetable, mean)
```

Tasks:

a) Create missing indicators for at least 3 variables
b) Calculate the donation rate for each missing indicator
c) Identify which missing indicators are most predictive
d) Explain why "missingness" itself might be informative

Deliverable:

- Code creating indicators
- Donation rate comparison table
- Top 3 most predictive missing indicators
- Business interpretation of findings

Task 2.4: Advanced Imputation (5 points)

Implement a more sophisticated imputation method.

```r
# Option 1: Group-based imputation
# Impute age based on segment
basetable <- basetable %>%
  group_by(segment) %>%
  mutate(age_imputed_group = ifelse(
    is.na(age),
    median(age, na.rm = TRUE),
    age
  )) %>%
  ungroup()

# Option 2: Predictive imputation using simple regression
# Use other variables to predict missing values
```

Tasks:

a) Implement group-based imputation for at least 1 variable

b) Compare simple median imputation vs. group-based imputation
c) Calculate improvement in variance preservation

Deliverable:

- Code for advanced imputation
- Comparison table of methods
- Discussion of when to use each approach

Part 3: Handling Outliers (25 points)

Task 3.1: Outlier Detection (8 points)

Identify outliers using multiple methods.

```r
# Method 1: Visual inspection
par(mfrow = c(2, 2))
boxplot(basetable$mean_donation, main = "Mean Donation")
boxplot(basetable$max_donation, main = "Max Donation")
boxplot(basetable$age, main = "Age")
boxplot(basetable$days_since_last, main = "Days Since Last")

# Method 2: Statistical thresholds
# Define outliers as values beyond 3 SD
detect_outliers_sd <- function(x, n_sd = 3) {
  mean_x <- mean(x, na.rm = TRUE)
  sd_x <- sd(x, na.rm = TRUE)
  lower <- mean_x - n_sd * sd_x
  upper <- mean_x + n_sd * sd_x

  outliers <- x < lower | x > upper
  return(outliers)
}

# Method 3: IQR method
detect_outliers_iqr <- function(x) {
  q1 <- quantile(x, 0.25, na.rm = TRUE)
  q3 <- quantile(x, 0.75, na.rm = TRUE)
  iqr <- q3 - q1

  lower <- q1 - 1.5 * iqr
  upper <- q3 + 1.5 * iqr

  outliers <- x < lower | x > upper
  return(outliers)
}
```

Tasks:

a) Create boxplots for at least 5 numeric variables
b) Apply both SD method and IQR method to detect outliers
c) Create a summary table comparing methods
d) Identify which variables have the most outliers

Deliverable:

- Boxplot visualizations
- Outlier detection summary table
- Comparison of SD vs. IQR methods
- Discussion of which method is more appropriate for your data

---

Task 3.2: Winsorization (8 points)

Apply winsorization to cap extreme values.

```r
# Install DescTools if needed
# install.packages("DescTools")
library(DescTools)

# Winsorize at 5th and 95th percentiles
basetable$mean_donation_wins <- Winsorize(
  basetable$mean_donation,
  probs = c(0.05, 0.95),
  na.rm = TRUE
)

# Compare distributions
par(mfrow = c(1, 2))
hist(basetable$mean_donation, main = "Original",
     xlab = "Mean Donation", breaks = 30)
hist(basetable$mean_donation_wins, main = "Winsorized",
     xlab = "Mean Donation", breaks = 30)
```

Tasks:

a) Winsorize at least 3 numeric variables
b) Use both 5%/95% and 1%/99% winsorization levels
c) Create before/after histograms for each variable
d) Calculate summary statistics (mean, SD, min, max) before and after

Deliverable:

- Winsorization code for multiple variables
- Before/after visualizations
- Summary statistics comparison table
- Recommendation on which percentile levels to use

Task 3.3: Standard Deviation Method (9 points)

Implement the mean ± 3 SD capping method.

```r
# Function to cap outliers at mean ± n SD
cap_outliers_sd <- function(x, n_sd = 3) {
  mean_x <- mean(x, na.rm = TRUE)
  sd_x <- sd(x, na.rm = TRUE)

  lower_limit <- mean_x - n_sd * sd_x
  upper_limit <- mean_x + n_sd * sd_x

  # Cap values
```

```r
  x_capped <- pmin(pmax(x, lower_limit), upper_limit)

  return(x_capped)
}

# Apply to variables
basetable$mean_donation_capped <- cap_outliers_sd(
  basetable$mean_donation,
  n_sd = 3
)
```

Tasks:

a) Implement the SD capping function
b) Apply to at least 3 variables with outliers
c) Experiment with 2 SD, 3 SD, and 4 SD thresholds
d) Compare SD method vs. Winsorization on the same variable
e) Analyze how many values were capped at each threshold

Deliverable:

- Capping function implementation
- Comparison of different SD thresholds
- Winsorization vs. SD method comparison
- Recommendation table: which method for which variable
- Discussion of trade-offs

Part 4: Variable Transformations (20 points)

Task 4.1: Log Transformation (7 points)

Apply logarithmic transformation to right-skewed variables.

```r
# Identify skewed variables
skewness_check <- sapply(
  basetable[, sapply(basetable, is.numeric)],
  function(x) {
    library(e1071)
    skewness(x, na.rm = TRUE)
  }
)

# Variables with skewness > 1 are candidates for log transform
print(sort(skewness_check, decreasing = TRUE))

# Apply log transformation
# Handle zeros with log1p (log(x + 1))
basetable$log_mean_donation <- log1p(basetable$mean_donation)
basetable$log_max_donation <- log1p(basetable$max_donation)
```

Tasks:

a) Calculate skewness for all numeric variables
b) Identify variables with absolute skewness > 1
c) Apply log transformation to at least 3 skewed variables

d) Create before/after histograms showing improvement
e) Verify that transformed variables have lower skewness

Deliverable:

- Skewness summary table
- Log transformation code
- Before/after distribution comparisons (histograms + QQ plots)
- New skewness values showing improvement

Task 4.2: Other Transformations (6 points)

Explore alternative transformation methods.

```
# Square root transformation (milder than log)
basetable$sqrt_mean_donation <- sqrt(basetable$mean_donation)

# Inverse transformation
basetable$inv_recency <- 1 / (basetable$days_since_last + 1)

# Box-Cox transformation (finds optimal power)
library(MASS)
bc <- boxcox(mean_donation ~ 1, data = basetable,
             lambda = seq(-2, 2, 0.1))
optimal_lambda <- bc$x[which.max(bc$y)]
```

Tasks:

a) Apply square root transformation to 2 variables
b) Create inverse transformation for recency (so recent = high value)
c) Implement Box-Cox transformation for 1 variable
d) Compare all transformation methods on the same variable

Deliverable:

- Code for all transformation types
- Comparison table: original vs. log vs. sqrt vs. Box-Cox
- Visualization showing distribution improvements
- Recommendations for which transformation to use when

Task 4.3: Transformation Impact Analysis (7 points)

Analyze how transformations affect model performance.

```
# Simple logistic regression comparison
# Model 1: Original variables
model_original <- glm(
  target ~ mean_donation + days_since_last,
  data = basetable,
  family = "binomial"
)

# Model 2: Transformed variables
model_transformed <- glm(
  target ~ log_mean_donation + inv_recency,
```

```r
  data = basetable,
  family = "binomial"
)

# Compare AIC
AIC(model_original)
AIC(model_transformed)
```

Tasks:

a) Fit logistic regression with original variables
b) Fit logistic regression with transformed variables
c) Compare model fit statistics (AIC, deviance)
d) Compare coefficient interpretability
e) Calculate predicted probabilities for both models

Deliverable:

- Model comparison table (AIC, deviance, pseudo-R^2)
- Coefficient comparison and interpretation
- Discussion: Is the improvement worth the interpretation complexity?

Part 5: Creating Interaction Features (10 points)

Task 5.1: Meaningful Interactions (6 points)

Create interaction terms that make business sense.

```r
# Frequency × Recency interaction
basetable$freq_recency <-
  basetable$donation_count / (basetable$days_since_last + 1)

# Frequency × Monetary interaction
basetable$freq_monetary <-
  basetable$donation_count * basetable$mean_donation

# Three-way RFM interaction
basetable$rfm_score <-
  (1 / (basetable$days_since_last + 1)) *
  basetable$donation_count *
  basetable$mean_donation
```

Tasks:

a) Create at least 5 meaningful interaction features
b) Explain the business logic behind each interaction
c) Examine the distribution of interaction features
d) Identify top 10 donors by RFM score

Deliverable:

- Interaction creation code
- Business justification for each interaction
- Distribution summary statistics
- Top/bottom 10 donors comparison

Task 5.2: Interaction Impact (4 points)

Test whether interactions improve model performance.

```r
# Model without interactions
model_no_int <- glm(
  target ~ donation_count + days_since_last + mean_donation,
  data = basetable,
  family = "binomial"
)

# Model with interactions
model_with_int <- glm(
  target ~ donation_count + days_since_last + mean_donation +
           freq_recency + rfm_score,
  data = basetable,
  family = "binomial"
)

# Compare models
anova(model_no_int, model_with_int, test = "Chisq")
```

Tasks:

a) Fit models with and without interactions
b) Perform likelihood ratio test
c) Compare predictive accuracy on holdout set
d) Identify which interactions are most important

Deliverable:

- Model comparison statistics
- Likelihood ratio test results
- Discussion of which interactions add value

Grading Rubric

Component	Points	Criteria
Part 1: Dummy Variables	20	Correct implementation, proper multicollinearity handling, clear documentation
Part 2: Missing Values	25	Appropriate strategy selection, thorough analysis, missing indicators
Part 3: Outliers	25	Multiple detection methods, proper capping techniques, method comparison
Part 4: Transformations	20	Appropriate transformations, impact analysis, clear visualizations
Part 5: Interactions	10	Business-motivated interactions, performance evaluation

Component	Points	Criteria
Code Quality	**10**	**Clean, well-commented, reproducible**
Report Quality	**10**	**Clear explanations, professional presentation**
Total	**120**	**(100 + 20 quality points)**

Submission Guidelines

What to Submit

1. **R Markdown file** (.Rmd) with all code, analysis, and explanations
2. **Knitted PDF** showing all output, visualizations, and tables
3. **Processed dataset** (CSV) with all new features created

File Naming Convention

`Classwork3_FirstnameLastname.Rmd`

`Classwork3_FirstnameLastname.pdf`

`basetable_preprocessed_FirstnameLastname.csv`

Submission Platform

Upload to course portal under "Assignments > Classwork 3"

Tips for Success

1. **Document your decisions -** Explain WHY you chose each method
2. **Compare methods -** Don't just apply one technique, compare alternatives
3. **Visualize everything -** Before/after plots reveal transformation impact
4. **Think about business -** Do your preprocessing choices make sense?
5. **Check for errors -** Look for Inf, NaN, NA after transformations
6. **Save intermediate results -** Keep original variables for comparison
7. **Use functions -** Write reusable code for repetitive tasks
8. **Test on subsets first -** Verify methods work before applying to all data

---

Frequently Asked Questions

Q: Should I apply all transformations to all variables?

A: No! Only transform variables that need it. Check distribution first.

Q: What if my variable has no missing values?

A: Great! Document that and move to next variable. No imputation needed.

Q: How do I know if winsorization or SD method is better?

A: Check your data distribution. Normal data → SD method. Skewed data → winsorization.

Q: Can I create more than 5 interactions?

A: Yes, but be selective. Focus on interactions with business justification.

Q: What if log transformation creates Inf values?

A: Use `log1p()` instead of `log()`, or add a small constant before logging.

Q: Should I remove outliers or cap them?

A: Generally cap them (winsorize). Only remove if they're data errors.

Q: How many dummy variables should segment create?

A: If segment has 3 categories, create 2 dummies (drop 1 for reference).

Academic Integrity

- You may discuss concepts with classmates
- You must write your own code and explanations
- Copying code from others = automatic zero
- Using AI for debugging = OK
- Using AI to write entire solutions = NOT OK

The goal is learning, not just completion.

Good Luck!

Clean data is the foundation of good models.

Take time to understand each preprocessing step.