BIN371_m2

2025-08-10

---

1) Overview

This document completes Milestone 2 by: - Describing datasets and final columns (Data Description) - Stating inclusion/exclusion rules (Data Selection) - Cleaning, imputing, and standardising data (Data Cleaning Process) - Running feature selection tests (Attribute/Feature Selection) - Transformations, one-hot matrices, and 70/30 splits (Transformations & Aggregation)

All outputs go to `outputs_m2/`.

2) Load Data

```r
# Adjust paths if your CSVs live elsewhere
paths <- list(
  Anthropometry     = "anthropometry_national_zaf.csv",
  Literacy          = "literacy_national_zaf.csv",
  MaternalMortality = "maternal-mortality_national_zaf.csv",
  ARI_Symptoms      = "symptoms-of-acute-respiratory-infection-ari_national_zaf.csv"
)

read_base <- function(p) read.csv(p, stringsAsFactors = FALSE, check.names = FALSE)
df_list_raw <- lapply(paths, read_base)
names(df_list_raw) <- names(paths)

# Output directory
out_dir <- "outputs_m2"
if (!dir.exists(out_dir)) dir.create(out_dir, recursive = TRUE)
```

3) Standardize Columns & Create `Value_num`

```r
# Standardize names + parse Value -> Value_num (handles "12%", "1,234", etc.)
clean_df <- function(df){
  # unify column names
  names(df) <- gsub("\\s+", "_", names(df))
  names(df) <- gsub("[^A-Za-z0-9_]", "", names(df))
  # numeric extraction for Value
  if ("Value" %in% names(df)) {
    df$Value_num <- readr::parse_number(as.character(df$Value))
  }
  df
}

df_list <- lapply(df_list_raw, clean_df)

# Quick sanity: which datasets had non-numeric Value originally?
for (nm in names(df_list)) {
  df <- df_list[[nm]]
  if ("Value" %in% names(df)) {
    bad <- suppressWarnings(sum(is.na(as.numeric(df$Value)) & !is.na(df$Value)))
    cat(nm, "- non-numeric Value entries initially:", bad, "\n")
  }
}
```

```
## Anthropometry - non-numeric Value entries initially: 1
## Literacy - non-numeric Value entries initially: 1
## MaternalMortality - non-numeric Value entries initially: 1
## ARI_Symptoms - non-numeric Value entries initially: 1
```

4) Data Description (Shapes & Columns)

```r
dims <- tibble(
  Dataset = names(df_list),
  Rows    = sapply(df_list, nrow),
  Columns = sapply(df_list, ncol)
)

# Show first 8 column names for each dataset
col_preview <- purrr::imap_dfr(df_list, ~tibble(
  Dataset = .y,
  Columns = paste(head(names(.x), 8), collapse = ", ")
))

knitr::kable(dims, caption = "Dataset Dimensions (Before Selection)")
```

Table 1: Dataset Dimensions (Before Selection)

Dataset	Rows	Columns
Anthropometry	38	30
Literacy	21	30
MaternalMortality	22	30
ARI_Symptoms	27	30

```r
knitr::kable(col_preview, caption = "Column Preview (First 8 per Dataset)")
```

Table 2: Column Preview (First 8 per Dataset)

Dataset	Columns
Anthropometry	ISO3, DataId, Indicator, Value, Precision, DHS_CountryCode, CountryName, SurveyYear
Literacy	ISO3, DataId, Indicator, Value, Precision, DHS_CountryCode, CountryName, SurveyYear
MaternalMortality	ISO3, DataId, Indicator, Value, Precision, DHS_CountryCode, CountryName, SurveyYear
ARI_Symptoms	ISO3, DataId, Indicator, Value, Precision, DHS_CountryCode, CountryName, SurveyYear

5) Data Selection (Inclusion/Exclusion Rules)

Keep (if present): `Indicator`, `Value`, `Value_num`, `Sex`, `Age`, `Region`, `Period`, `Subgroup`.
Drop: free-text/comments, near-constant columns, and fields with **>40% missing** (unless essential).

```r
keep_cols <- c("Indicator","Value","Value_num","Sex","Age","Region","Period","Subgroup")

prune_by_cols <- function(df) {
  cols <- intersect(keep_cols, names(df))
  df[, cols, drop = FALSE]
}
df_sel <- lapply(df_list, prune_by_cols)

# Missing rate per column
col_missing_rate <- function(df) {
  tibble(
    Column     = names(df),
    Missing    = sapply(df, function(x) sum(is.na(x))),
    PctMissing = round(100 * sapply(df, function(x) sum(is.na(x))) / max(1, nrow(df)), 2)
  ) %>% arrange(desc(PctMissing))
}

missing_reports <- purrr::imap(df_sel, ~col_missing_rate(.x) %>% mutate(Dataset = .y))
missing_overview <- dplyr::bind_rows(missing_reports)

# Drop columns with > 40% missing unless essential
essentials <- c("Indicator","Value_num")
drop_by_missing <- function(df, threshold = 40){
  rates <- 100 * sapply(df, function(x) sum(is.na(x))) / max(1, nrow(df))
  keep <- names(df)[rates <= threshold | names(df) %in% essentials]
  df[, keep, drop = FALSE]
}
df_sel <- lapply(df_sel, drop_by_missing, threshold = 40)

knitr::kable(head(missing_overview, 30), caption = "Missingness Overview (Top 30 Rows)")
```

Table 3: Missingness Overview (Top 30 Rows)

Column	Missing	PctMissing	Dataset
Value_num	1	2.63	Anthropometry
Indicator	0	0.00	Anthropometry
Value	0	0.00	Anthropometry
Value_num	1	4.76	Literacy
Indicator	0	0.00	Literacy
Value	0	0.00	Literacy
Value_num	1	4.55	MaternalMortality
Indicator	0	0.00	MaternalMortality
Value	0	0.00	MaternalMortality
Value_num	1	3.70	ARI_Symptoms
Indicator	0	0.00	ARI_Symptoms
Value	0	0.00	ARI_Symptoms

6) Data Cleaning Process

- Remove exact duplicates
- Impute `Value_num` by median within `(Indicator × Sex)` → then `Indicator` → global fallback
- Optional winsorization of `Value_num` at 1st/99th percentiles
- Cast categorical fields to factors; build `Value_scaled`

```
# 6.1 Remove exact duplicates
df_clean <- lapply(df_sel, function(df) df[!duplicated(df), , drop = FALSE])

# 6.2 Impute Value_num
impute_group_median <- function(df) {
  if (!"Value_num" %in% names(df)) return(df)
  # by Indicator x Sex
  if (all(c("Indicator","Sex") %in% names(df))) {
    df <- df %>%
      group_by(Indicator, Sex) %>%
      mutate(Value_num = ifelse(is.na(Value_num),
                                median(Value_num, na.rm = TRUE),
                                Value_num)) %>%
      ungroup()
  }
  # by Indicator
  if ("Indicator" %in% names(df)) {
    df <- df %>%
      group_by(Indicator) %>%
      mutate(Value_num = ifelse(is.na(Value_num),
                                median(Value_num, na.rm = TRUE),
                                Value_num)) %>%
      ungroup()
  }
  # global fallback
  if (any(is.na(df$Value_num))) {
    gmed <- median(df$Value_num, na.rm = TRUE)
    df$Value_num[is.na(df$Value_num)] <- gmed
  }
  df
}
df_clean <- lapply(df_clean, impute_group_median)

# 6.3 Winsorize (optional)
winsorize <- function(x, p = c(0.01, 0.99)){
  qs <- quantile(x, probs = p, na.rm = TRUE)
  x <- pmax(x, qs[1]); x <- pmin(x, qs[2]); x
}
df_clean <- lapply(df_clean, function(df){
  if ("Value_num" %in% names(df) && sum(!is.na(df$Value_num)) > 10) {
    df$Value_winz <- winsorize(df$Value_num)
  }
  df
})

# 6.4 Cast factors & scale
prep_factors_scale <- function(df){
  for (v in c("Sex","Age","Region","Subgroup","Indicator","Period")){
    if (v %in% names(df)) df[[v]] <- as.factor(df[[v]])
  }
  if ("Value_winz" %in% names(df)) {
    df$Value_scaled <- as.numeric(scale(df$Value_winz))
  } else if ("Value_num" %in% names(df)) {
    df$Value_scaled <- as.numeric(scale(df$Value_num))
  }
  df
}
df_clean <- lapply(df_clean, prep_factors_scale)

# Save interim cleaned CSVs
purrr::iwalk(df_clean, ~write.csv(.x, file.path(out_dir, paste0("clean_", .y, ".csv")), row.names = FALS
```

7) Attribute / Feature Selection

- **Spearman** for numeric predictors vs `Value_scaled` (robust if none exist)
- **Kruskal–Wallis** for categorical predictors vs `Value_scaled`
- Optionally create `Period_num` as a numeric proxy to have at least one numeric predictor

```
# Helper: numeric predictors (excluding target variants)
num_preds <- function(df){
  nums <- names(df)[sapply(df, is.numeric)]
  setdiff(nums, c("Value_num","Value_winz","Value_scaled"))
}

# OPTIONAL numeric 'Period_num' to enable at least one numeric predictor
df_clean <- lapply(df_clean, function(df){
```

```r
  if ("Period" %in% names(df) && !("Period_num" %in% names(df))) {
    df$Period_num <- as.numeric(df$Period)  # encoded levels; interpret cautiously
  }
  df
})

# Spearman (robust to empty results)
spearman_report <- function(df, name){
  out <- tibble()
  if ("Value_scaled" %in% names(df)) {
    preds <- num_preds(df)
    if (length(preds) > 0) {
      for (nm in preds) {
        x <- df[[nm]]
        if (length(unique(na.omit(x))) > 5) {
          ct <- suppressWarnings(cor.test(df$Value_scaled, x, method = "spearman"))
          out <- bind_rows(out, tibble(Predictor = nm, Rho = unname(ct$estimate), P = ct$p.value))
        }
      }
    }
  }
  if (nrow(out) == 0) out <- tibble(Predictor = NA_character_, Rho = NA_real_, P = NA_real_)
  out %>% mutate(Dataset = name)
}
spearman_all <- purrr::imap_dfr(df_clean, spearman_report)

if ("P" %in% names(spearman_all) && any(!is.na(spearman_all$P))) {
  knitr::kable(dplyr::arrange(spearman_all, P) %>% head(15),
               caption = "Top Numeric Predictors by Spearman (smallest p-values)")
} else {
  knitr::kable(spearman_all,
               caption = "Spearman results (none available - no eligible numeric predictors)")
}
```

Table 4: Spearman results (none available — no eligible numeric predictors)

Predictor	Rho	P	Dataset
NA	NA	NA	Anthropometry
NA	NA	NA	Literacy
NA	NA	NA	MaternalMortality
NA	NA	NA	ARI_Symptoms

```r
# Kruskal-Wallis (categorical vs Value_scaled)
kw_report <- function(df, name){
  out <- tibble()
  if ("Value_scaled" %in% names(df)) {
    cats <- names(df)[sapply(df, is.factor)]
    for (nm in cats) {
      if (length(unique(na.omit(df[[nm]]))) >= 2) {
        kt <- suppressWarnings(kruskal.test(Value_scaled ~ df[[nm]], data = df))
        out <- bind_rows(out, tibble(Variable = nm, KW_ChiSq = unname(kt$statistic), P = kt$p.value))
      }
    }
  }
  if (nrow(out) == 0) out <- tibble(Variable = NA_character_, KW_ChiSq = NA_real_, P = NA_real_)
  out %>% mutate(Dataset = name)
}
kw_all <- purrr::imap_dfr(df_clean, kw_report)

if ("P" %in% names(kw_all) && any(!is.na(kw_all$P))) {
  knitr::kable(dplyr::arrange(kw_all, P) %>% head(15),
               caption = "Top Categorical Predictors by Kruskal-Wallis (smallest p-values)")
} else {
  knitr::kable(kw_all, caption = "Kruskal-Wallis results (none available)")
}
```

Table 5: Top Categorical Predictors by Kruskal–Wallis (smallest p-values)

Variable	KW_ChiSq	P	Dataset
Indicator	22.90079	0.0017738	ARI_Symptoms
Indicator	20.02415	0.0450105	MaternalMortality
Indicator	36.83802	0.2957453	Anthropometry
Indicator	20.00000	0.4579297	Literacy

8) Transformations & Split (One-Hot + 70/30)

```
# One-hot encode via model.matrix; Value_scaled is the numeric response proxy
one_hot_mats <- lapply(df_clean, function(df){
  if (!"Value_scaled" %in% names(df)) return(NULL)
  mm <- model.matrix(Value_scaled ~ . - Value - Value_num - Value_winz, data = df)
  list(
    X = as.matrix(mm),
    y = as.numeric(df$Value_scaled)
  )
})

dir.create(out_dir, showWarnings = FALSE)
set.seed(42)

# NOTE: iwalk passes (value, name), so function must be (mat_list, name)
split_and_save <- function(mat_list, name){
  if (is.null(mat_list)) return(invisible(NULL))
  n <- length(mat_list$y)
  if (n < 2) return(invisible(NULL))  # nothing to split
  idx <- sample(seq_len(n), size = max(1, floor(0.7 * n)))
  train <- list(X = mat_list$X[idx, , drop = FALSE], y = mat_list$y[idx])
  test  <- list(X = mat_list$X[-idx, , drop = FALSE], y = mat_list$y[-idx])
  saveRDS(train, file.path(out_dir, paste0("train_", name, ".rds")))
  saveRDS(test,  file.path(out_dir, paste0("test_",  name, ".rds")))
  invisible(TRUE)
```

Table 6: Missingness After Cleaning (Total NA counts)

Dataset	MissingTot	Rows
Anthropometry	0	38

Dataset	MissingTot	Rows
Literacy	0	21

Dataset	MissingTot	Rows
MaternalMortality	0	22

Dataset	MissingTot	Rows
ARI_Symptoms	0	27

```
}

purrr::iwalk(one_hot_mats, split_and_save)
```

9) Before/After Quality Snapshot

```
# Missingness after cleaning
clean_missing_reports <- purrr::imap(df_clean, ~tibble(
  Dataset    = .y,
  MissingTot = sum(sapply(.x, function(c) sum(is.na(c)))),
  Rows       = nrow(.x)
))
knitr::kable(clean_missing_reports, caption = "Missingness After Cleaning (Total NA counts)")
```

```
# Outliers (IQR) count on Value_num (pre-winsor) if present
iqr_flags <- function(x){
  q1 <- quantile(x, 0.25, na.rm=TRUE); q3 <- quantile(x, 0.75, na.rm=TRUE); i <- q3-q1
  (x < (q1 - 1.5*i)) | (x > (q3 + 1.5*i))
}
outlier_report <- purrr::imap_dfr(df_clean, function(df, nm){
  if (!"Value_num" %in% names(df)) return(tibble(Dataset = nm, Outliers_Value_num = NA_integer_))
  tibble(Dataset = nm, Outliers_Value_num = sum(iqr_flags(df$Value_num), na.rm = TRUE))
})
knitr::kable(outlier_report, caption = "Outliers (IQR) Count on Value_num (Pre-Winsor)")
```

Table 7: Outliers (IQR) Count on Value_num (Pre-Winsor)

Dataset	Outliers_Value_num
Anthropometry	8
Literacy	4
MaternalMortality	4
ARI_Symptoms	0

10) Submission Notes

- Include this knitted report.
- Include `outputs_m2/` contents:
  - `clean_*.csv` (cleaned datasets)
  - `train_*.rds` and `test_*.rds` (design matrices for Milestone 3)

- In your write-up: explain rules, justify thresholds, and interpret the top factors (from KW / Spearman if any).

```
cat("\nMilestone 2 complete.\nOutputs written to: ", normalizePath(out_dir), "\n")
```

```
##
## Milestone 2 complete.
## Outputs written to:  C:\Users\Suhil Jugroop\OneDrive\Documents\GitHub\BIN371-Project\outputs_m2
```