BIN371_m3

2025-08-18

1) Objective

We evaluate predictive models on the **prepared train/test splits** for four datasets:

- Anthropometry
- ARI Symptoms
- Literacy
- Maternal Mortality

Target is `Value_scaled` (numeric) → **regression**.
We fit **Linear Regression (LM)**, **Decision Tree (rpart)**, and **Random Forest (RF)** on each dataset's **train** set, evaluate on the **test** set using **RMSE, MAE, R²**, and choose the **best model by lowest RMSE**. Diagnostics include residual plots and RF feature importance.

2) Inputs & Paths

Provide the absolute paths to your **Training** and **Test** folders (Windows paths may use forward slashes).

```
# --- where your train/test RDS live ---
train_dir <- "C:/Users/Suhil Jugroop/OneDrive/Documents/GitHub/BIN371-Project/Milestone 3/Training-data
test_dir  <- "C:/Users/Suhil Jugroop/OneDrive/Documents/GitHub/BIN371-Project/Milestone 3/Test-data_m2"

# --- resolve the directory of this Rmd (works in Knit and in the editor) ---
rmd_path <- tryCatch(
  rstudioapi::getSourceEditorContext()$path,
  error = function(e) knitr::current_input()
)
rmd_dir <- if (nzchar(rmd_path)) normalizePath(dirname(rmd_path)) else normalizePath(getwd())

# --- output folder next to the Rmd ---
out_dir <- file.path(rmd_dir, "outputs_m3")
if (!dir.exists(out_dir)) dir.create(out_dir, recursive = TRUE)

# --- dataset names and RDS paths ---
datasets <- c("Anthropometry","ARI_Symptoms","Literacy","MaternalMortality")

train_paths <- setNames(file.path(train_dir, paste0("train_", datasets, ".rds")), datasets)
test_paths  <- setNames(file.path(test_dir,  paste0("test_",  datasets, ".rds")),  datasets)

# sanity check
out_dir
```

```
## [1] "C:\\Users\\Suhil Jugroop\\OneDrive\\Documents\\GitHub\\BIN371-Project\\Milestone 3/outputs_m3"
```

```r
train_paths
```

```
##
##     "C:/Users/Suhil Jugroop/OneDrive/Documents/GitHub/BIN371-Project/Milestone 3/Training-data_m2/tra
##
##      "C:/Users/Suhil Jugroop/OneDrive/Documents/GitHub/BIN371-Project/Milestone 3/Training-data_m2/t
##
##          "C:/Users/Suhil Jugroop/OneDrive/Documents/GitHub/BIN371-Project/Milestone 3/Training-data_
##
## "C:/Users/Suhil Jugroop/OneDrive/Documents/GitHub/BIN371-Project/Milestone 3/Training-data_m2/train_
```

```r
test_paths
```

```
##
##     "C:/Users/Suhil Jugroop/OneDrive/Documents/GitHub/BIN371-Project/Milestone 3/Test-data_m2/test_A
##
##      "C:/Users/Suhil Jugroop/OneDrive/Documents/GitHub/BIN371-Project/Milestone 3/Test-data_m2/test_
##
##          "C:/Users/Suhil Jugroop/OneDrive/Documents/GitHub/BIN371-Project/Milestone 3/Test-data_m2/t
##                                                                                                      I
## "C:/Users/Suhil Jugroop/OneDrive/Documents/GitHub/BIN371-Project/Milestone 3/Test-data_m2/test_Mater
```

3) Helper Functions (Metrics, Fitters, Predictors)

```r
# --- metrics ---
rmse <- function(truth, pred) sqrt(mean((truth - pred)^2, na.rm = TRUE))
mae  <- function(truth, pred) mean(abs(truth - pred), na.rm = TRUE)
r2   <- function(truth, pred) {
  ss_res <- sum((truth - pred)^2, na.rm = TRUE)
  ss_tot <- sum((truth - mean(truth, na.rm = TRUE))^2, na.rm = TRUE)
  1 - ss_res/ss_tot
}

# --- coerce train/test list(X, y) to data.frame with Value_scaled ---
as_df_xy <- function(obj) {
  stopifnot(is.list(obj), all(c("X","y") %in% names(obj)))
  df <- as.data.frame(obj$X)
  df$Value_scaled <- as.numeric(obj$y)
  df
}

# --- model fitting for regression ---
fit_models <- function(df_train) {
  # defensive: drop columns with Inf/NaN
  bad <- vapply(df_train, function(v) any(is.infinite(v) | is.nan(v)), logical(1))
  if (any(bad)) df_train <- df_train[, !bad, drop = FALSE]

  form <- as.formula("Value_scaled ~ .")
  mod_lm <- tryCatch(lm(form, data = df_train), error = function(e) NULL)

  ctrl   <- rpart.control(minsplit = 10, cp = 0.001, maxdepth = 10, xval = 5)
  mod_dt <- tryCatch(rpart(form, data = df_train, method = "anova", control = ctrl), error = function(e

  mod_rf <- tryCatch(randomForest(form, data = df_train, ntree = 500), error = function(e) NULL)

  list(lm = mod_lm, dt = mod_dt, rf = mod_rf)
}

safe_predict <- function(model, newdata) {
  if (is.null(model)) return(rep(NA_real_, nrow(newdata)))
  as.numeric(tryCatch(predict(model, newdata = newdata), error = function(e) rep(NA_real_, nrow(newdata
}

evaluate_models <- function(models, df_test) {
  truth <- df_test$Value_scaled
  tibble(
    Model = c("Linear Regression","Decision Tree","Random Forest"),
    RMSE  = c(
      rmse(truth, safe_predict(models$lm, df_test)),
      rmse(truth, safe_predict(models$dt, df_test)),
      rmse(truth, safe_predict(models$rf, df_test))
    ),
    MAE   = c(
      mae(truth, safe_predict(models$lm, df_test)),
      mae(truth, safe_predict(models$dt, df_test)),
      mae(truth, safe_predict(models$rf, df_test))
    ),
    R2    = c(
      r2(truth, safe_predict(models$lm, df_test)),
      r2(truth, safe_predict(models$dt, df_test)),
      r2(truth, safe_predict(models$rf, df_test))
    )
  ) %>% arrange(RMSE)
}
```

4) Load Train/Test Splits

note this will produce a ton of empty data, don't worry about it, that's just how these things go when converted to binary, there is a way to hide it in the code apparently but I'm lazy

```
train_objs <- lapply(train_paths, readRDS)
test_objs  <- lapply(test_paths,  readRDS)

train_df <- lapply(train_objs, as_df_xy)
test_df  <- lapply(test_objs,  as_df_xy)

# quick preview
purrr::imap(train_df, ~{cat("\nTRAIN:", .y, "\n"); print(glimpse(.x))})
purrr::imap(test_df,  ~{cat("\nTEST :", .y, "\n"); print(glimpse(.x))})
```

5) Train Models Per Dataset

```r
models <- lapply(train_df, fit_models)

# status report
purrr::imap(models, ~{
  cat("\n", .y, " models:\n", sep = "")
  cat("  LM:", if (is.null(.x$lm)) "NULL" else "OK", "\n")
  cat("  DT:", if (is.null(.x$dt)) "NULL" else "OK", "\n")
  cat("  RF:", if (is.null(.x$rf)) "NULL" else "OK", "\n")
})
```

```
##
## Anthropometry models:
##   LM: OK
##   DT: OK
##   RF: NULL
##
## ARI_Symptoms models:
##   LM: OK
##   DT: OK
##   RF: NULL
##
## Literacy models:
##   LM: OK
##   DT: OK
##   RF: NULL
##
## MaternalMortality models:
##   LM: OK
##   DT: OK
##   RF: NULL
```

```
## $Anthropometry
## NULL
##
## $ARI_Symptoms
## NULL
##
## $Literacy
## NULL
##
## $MaternalMortality
## NULL
```

6) Evaluate & Choose Best Model (by RMSE)

```r
results <- purrr::imap_dfr(models, ~{
  ds <- .y
  ev <- evaluate_models(.x, test_df[[ds]])
  ev$Dataset <- ds
  ev
})

results_ranked <- results %>%
  group_by(Dataset) %>%
  arrange(RMSE, .by_group = TRUE) %>%
  mutate(Rank = row_number()) %>%
  ungroup()

knitr::kable(results_ranked, digits = 4, caption = "Model Comparison per Dataset (sorted by RMSE)")
```

Table 1: Model Comparison per Dataset (sorted by RMSE)

Model	RMSE	MAE	R2	Dataset	Rank
Linear Regression	0.3856	0.1906	0.8006	ARI_Symptoms	1
Decision Tree	0.8916	0.7969	-0.0659	ARI_Symptoms	2
Random Forest	NaN	NaN	1.0000	ARI_Symptoms	3
Decision Tree	0.8612	0.7427	-0.0470	Anthropometry	1
Linear Regression	0.9238	0.3872	-0.2048	Anthropometry	2
Random Forest	NaN	NaN	1.0000	Anthropometry	3
Decision Tree	0.4938	0.4903	-0.1139	Literacy	1
Linear Regression	0.5561	0.3072	-0.4123	Literacy	2
Random Forest	NaN	NaN	1.0000	Literacy	3
Decision Tree	0.7319	0.6432	-0.0030	MaternalMortality	1
Linear Regression	1.4526	0.8965	-2.9506	MaternalMortality	2
Random Forest	NaN	NaN	1.0000	MaternalMortality	3

```r
best_models <- results_ranked %>%
  filter(Rank == 1) %>%
  select(Dataset, Best_Model = Model, RMSE, MAE, R2)

knitr::kable(best_models, digits = 4, caption = "Chosen Model per Dataset (lowest RMSE)")
```

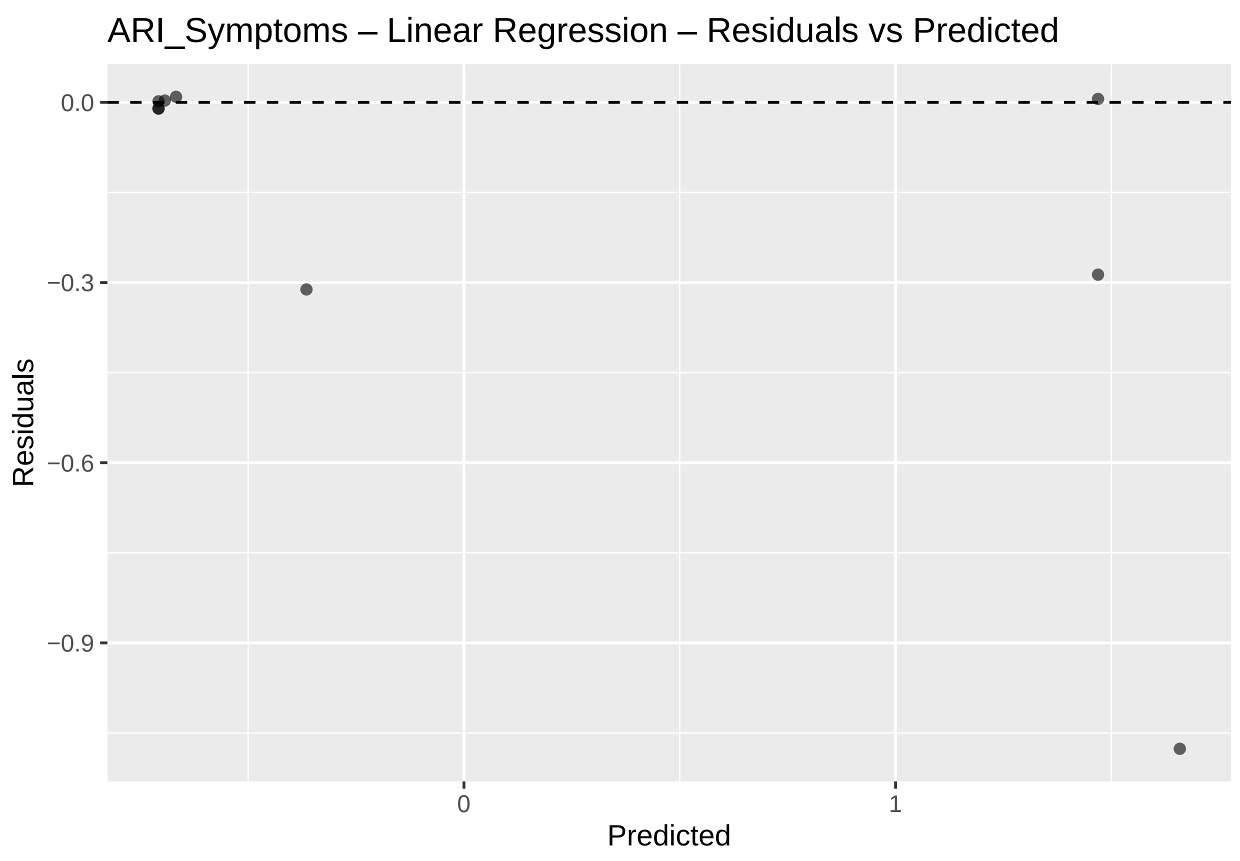
Table 2: Chosen Model per Dataset (lowest RMSE)

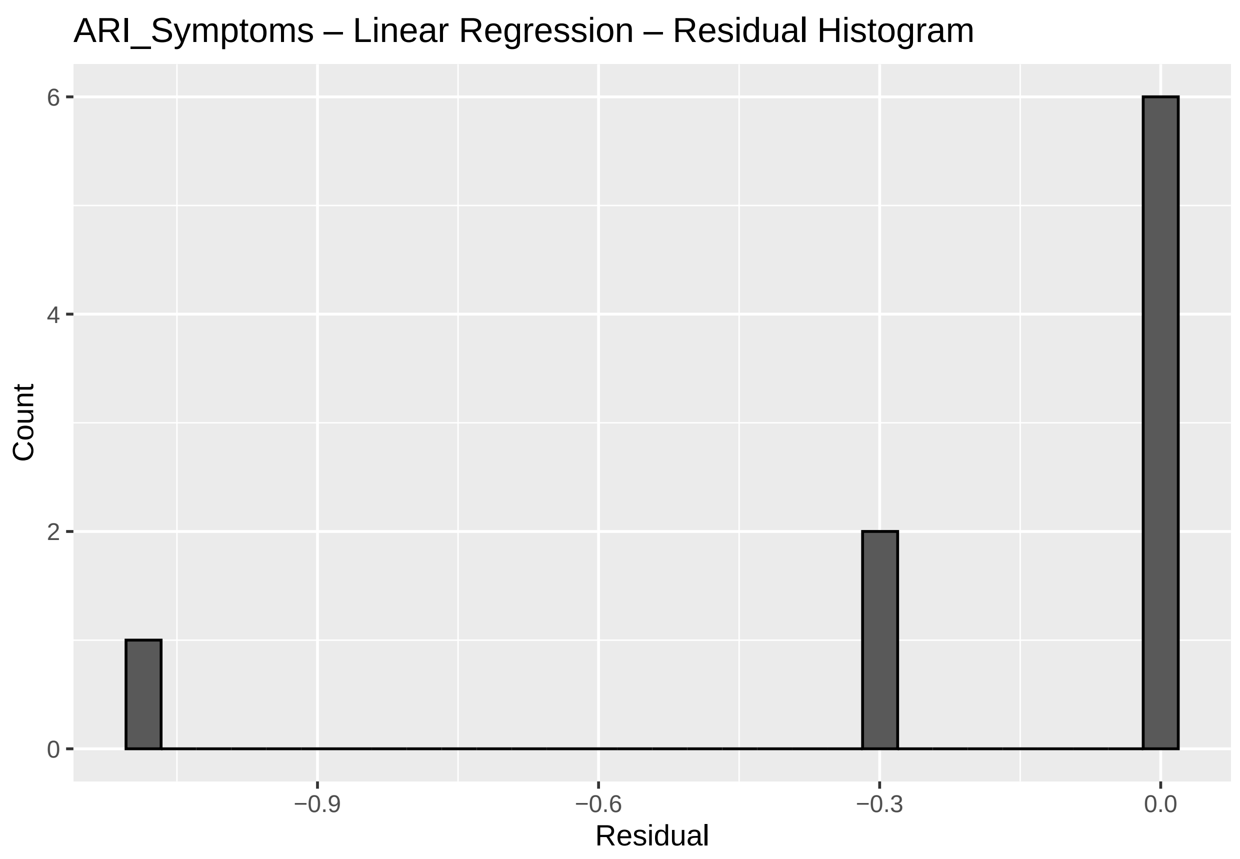
Dataset	Best_Model	RMSE	MAE	R2
ARI_Symptoms	Linear Regression	0.3856	0.1906	0.8006
Anthropometry	Decision Tree	0.8612	0.7427	-0.0470
Literacy	Decision Tree	0.4938	0.4903	-0.1139
MaternalMortality	Decision Tree	0.7319	0.6432	-0.0030

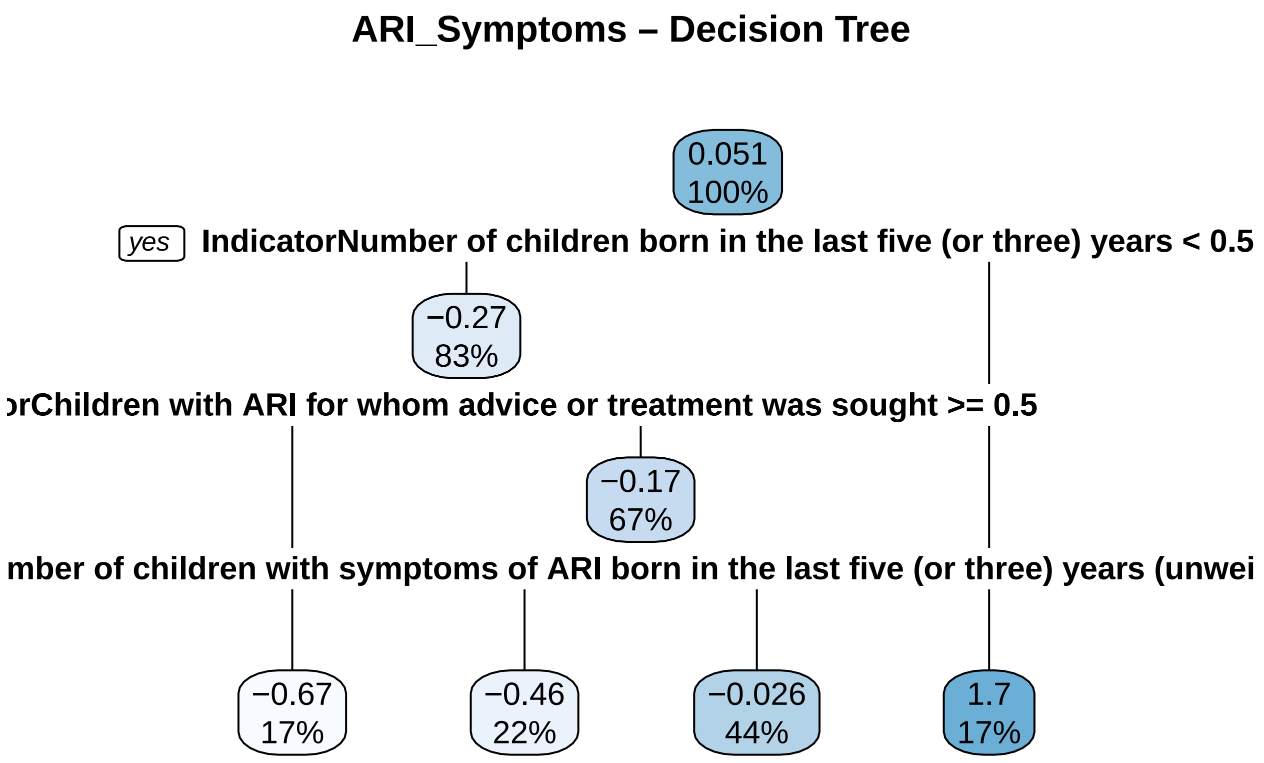
7) Diagnostics (Residuals, RF Importance, Tree Plot)

```r
plot_residuals <- function(model, df_test, title){
  pred <- safe_predict(model, df_test)
  res  <- df_test$Value_scaled - pred
  p1 <- ggplot(data.frame(Pred = pred, Resid = res), aes(Pred, Resid)) +
    geom_point(alpha = 0.6) + geom_hline(yintercept = 0, linetype = "dashed") +
    labs(title = paste0(title, " - Residuals vs Predicted"), x = "Predicted", y = "Residuals")
  p2 <- ggplot(data.frame(Resid = res), aes(Resid)) +
    geom_histogram(bins = 30, color = "black") +
    labs(title = paste0(title, " - Residual Histogram"), x = "Residual", y = "Count")
  list(p1 = p1, p2 = p2)
}

plot_rf_importance <- function(rf_model, title, top_n = 15){
  if (is.null(rf_model) || is.null(rf_model$importance)) return(NULL)
  imp <- as.data.frame(rf_model$importance)
  imp$Feature <- rownames(imp)
  imp <- imp %>% arrange(desc(IncNodePurity)) %>% head(top_n)
  ggplot(imp, aes(x = reorder(Feature, IncNodePurity), y = IncNodePurity)) +
    geom_col() + coord_flip() +
    labs(title = paste0(title, " - RF Variable Importance (Top ", top_n, ")"), x = "Feature", y = "IncN
}

purrr::pwalk(list(best_models$Dataset, best_models$Best_Model), function(ds, mlabel){
  cat("\n## Diagnostics:", ds, "(", mlabel, ")\n")
  mdl <- switch(mlabel,
                "Linear Regression" = models[[ds]]$lm,
                "Decision Tree"     = models[[ds]]$dt,
                "Random Forest"     = models[[ds]]$rf)
  testd <- test_df[[ds]]

  # Residuals
  rp <- plot_residuals(mdl, testd, paste0(ds, " - ", mlabel))
  print(rp$p1); print(rp$p2)

  # Tree plot if DT exists
  if (!is.null(models[[ds]]$dt)) {
    rpart.plot(models[[ds]]$dt, main = paste0(ds, " - Decision Tree"))
  }

  # RF importance (always useful to show)
  if (!is.null(models[[ds]]$rf)) {
    vip <- plot_rf_importance(models[[ds]]$rf, ds, top_n = 15)
    if (!is.null(vip)) print(vip)
  }
})
```

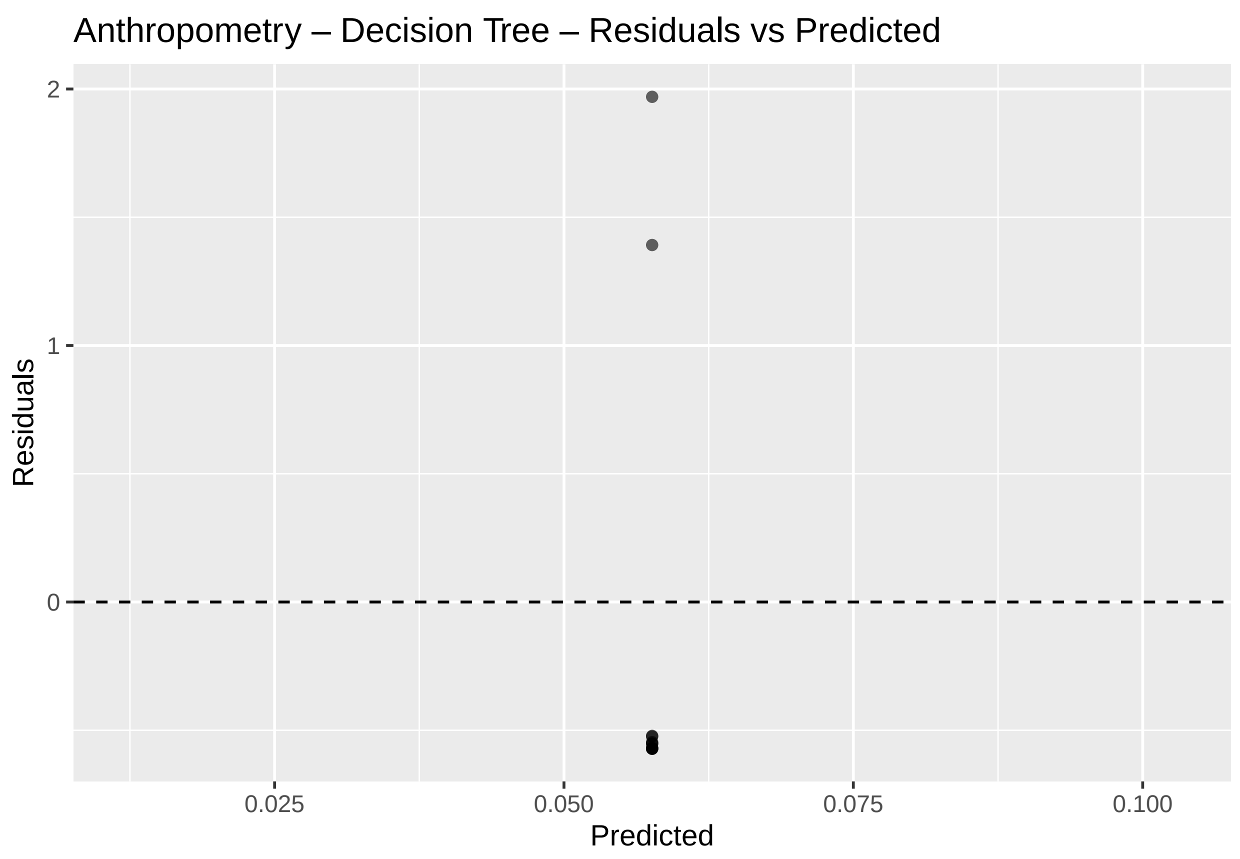
```
##
## ## Diagnostics: ARI_Symptoms ( Linear Regression )
```

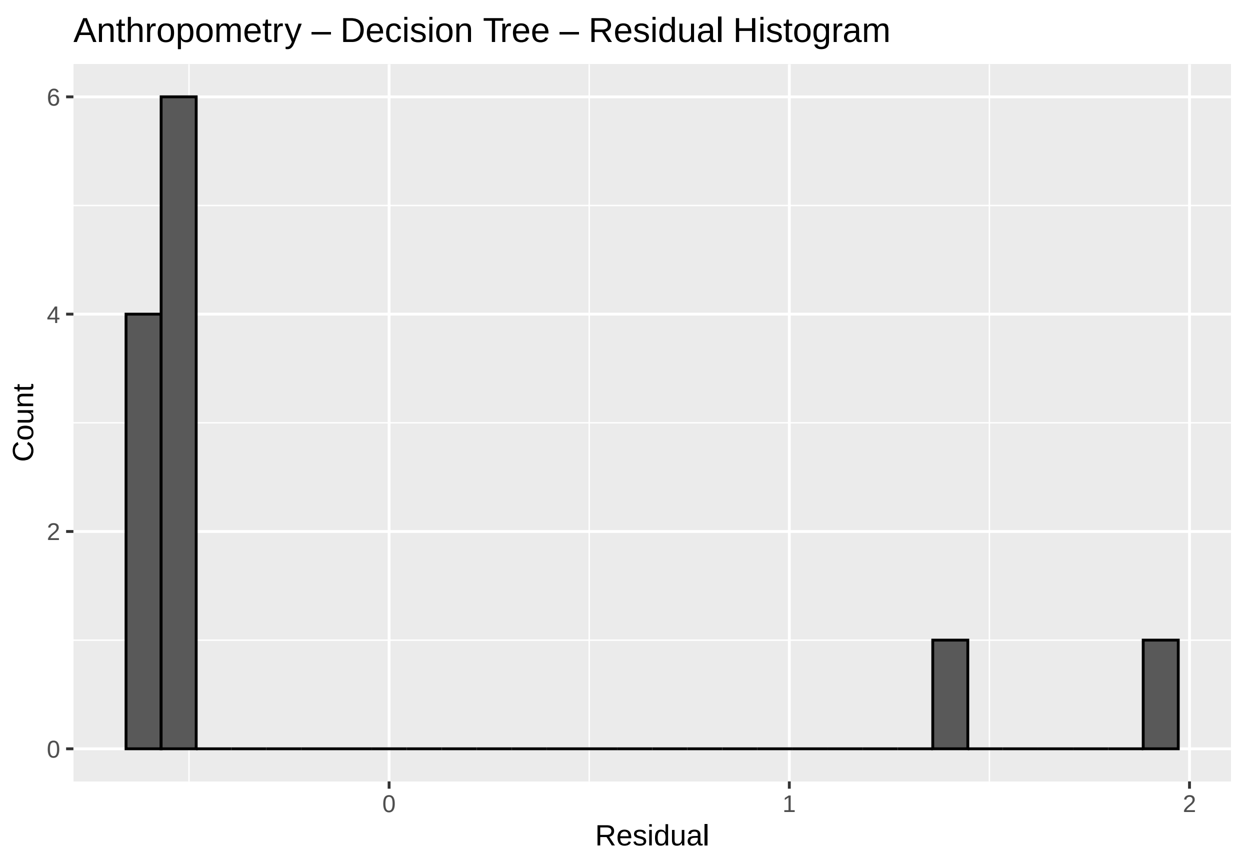

ARI_Symptoms – Linear Regression – Residuals vs Predicted
Residuals
0.0
−0.3
−0.6
−0.9
0
1
Predicted

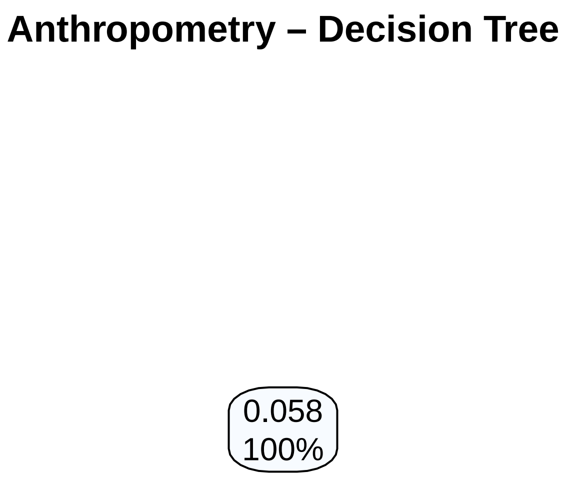

ARI_Symptoms – Linear Regression – Residual Histogram
Count
6
4
2
0
−0.9
−0.6
−0.3
0.0
Residual


ARI_Symptoms – Decision Tree
0.051
100%
yes
IndicatorNumber of children born in the last five (or three) years < 0.5
−0.27
83%
orChildren with ARI for whom advice or treatment was sought >= 0.5
−0.17
67%
mber of children with symptoms of ARI born in the last five (or three) years (unwei
−0.67
17%
−0.46
22%
−0.026
44%
1.7
17%

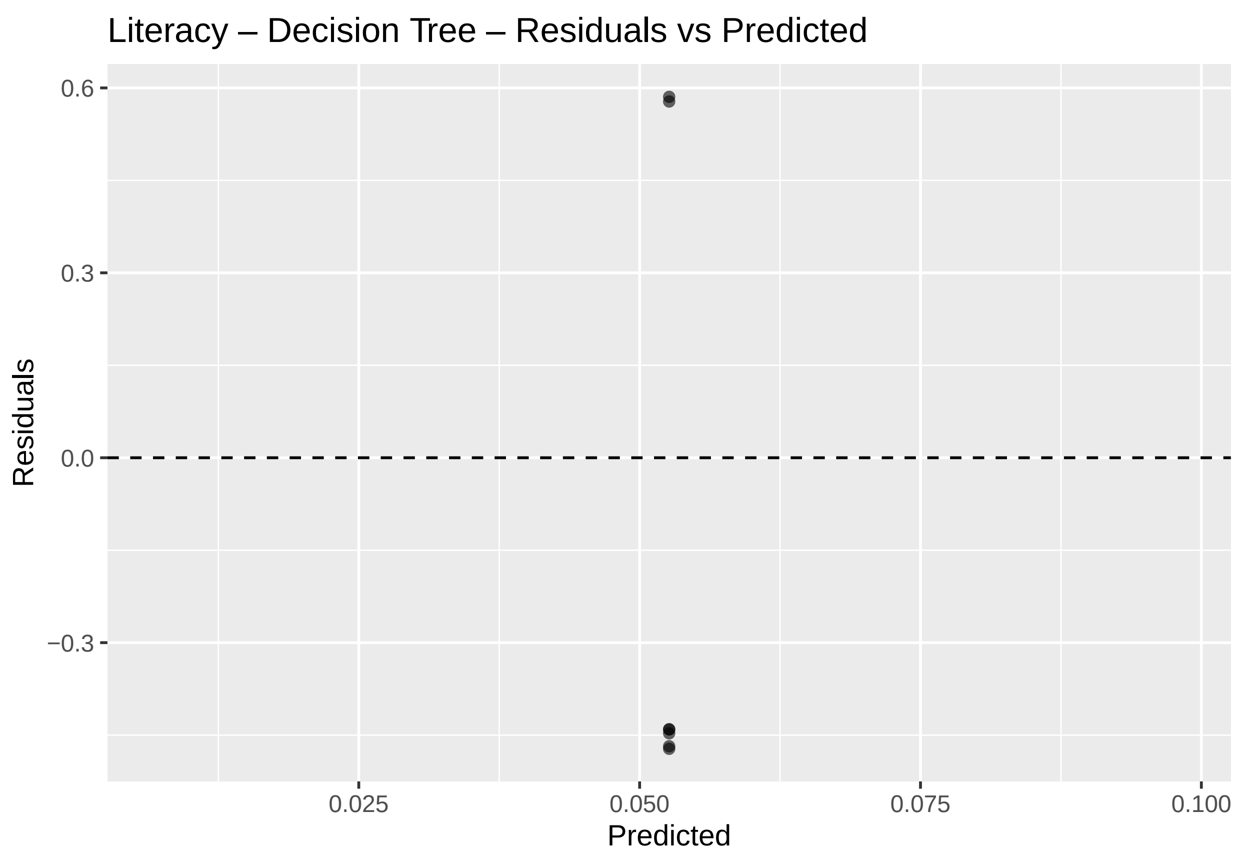
```
##
## ## Diagnostics: Anthropometry ( Decision Tree )
```

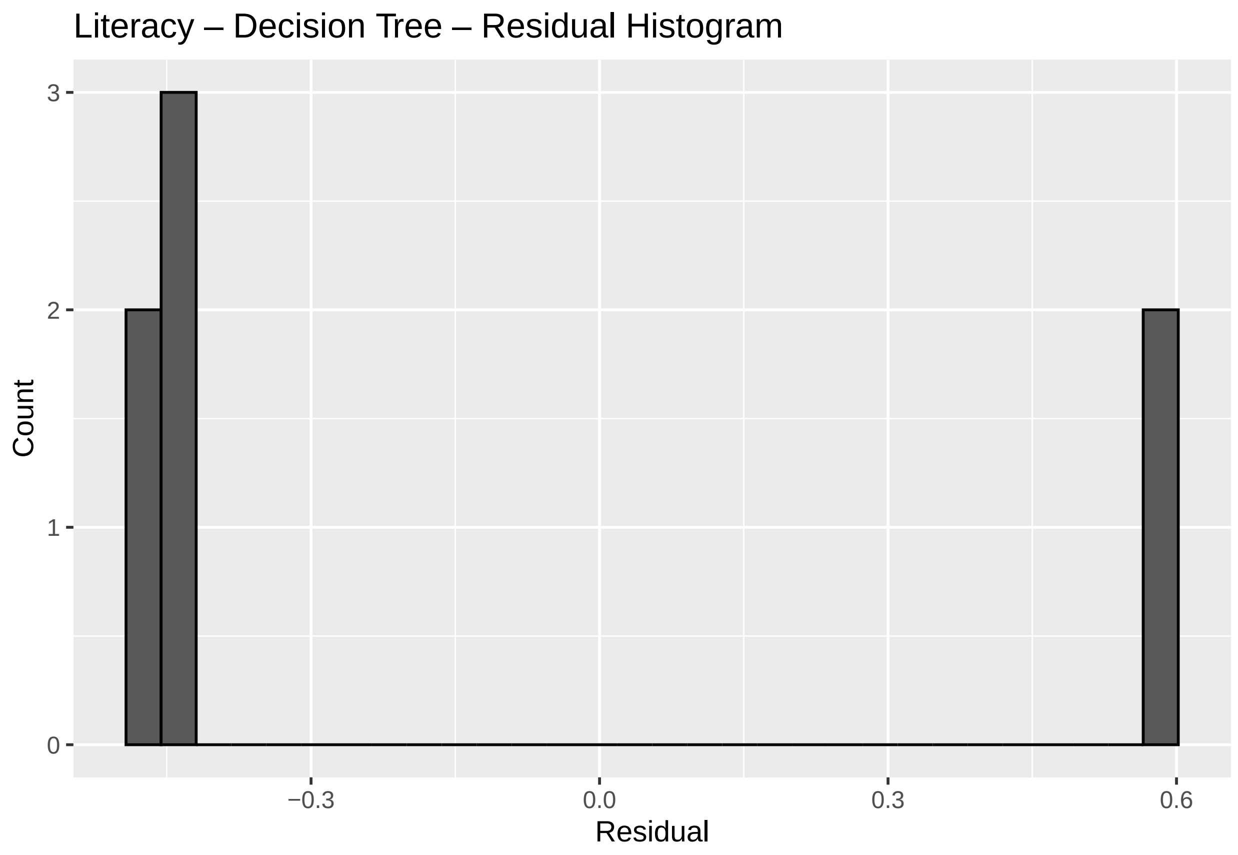

Anthropometry – Decision Tree – Residuals vs Predicted
Residuals
Predicted
2
1
0
0.025
0.050
0.075
0.100


Anthropometry – Decision Tree – Residual Histogram
Count
6
4
2
0
0
1
2
Residual

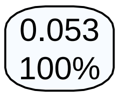


```
##
## ## Diagnostics: Literacy ( Decision Tree )
```

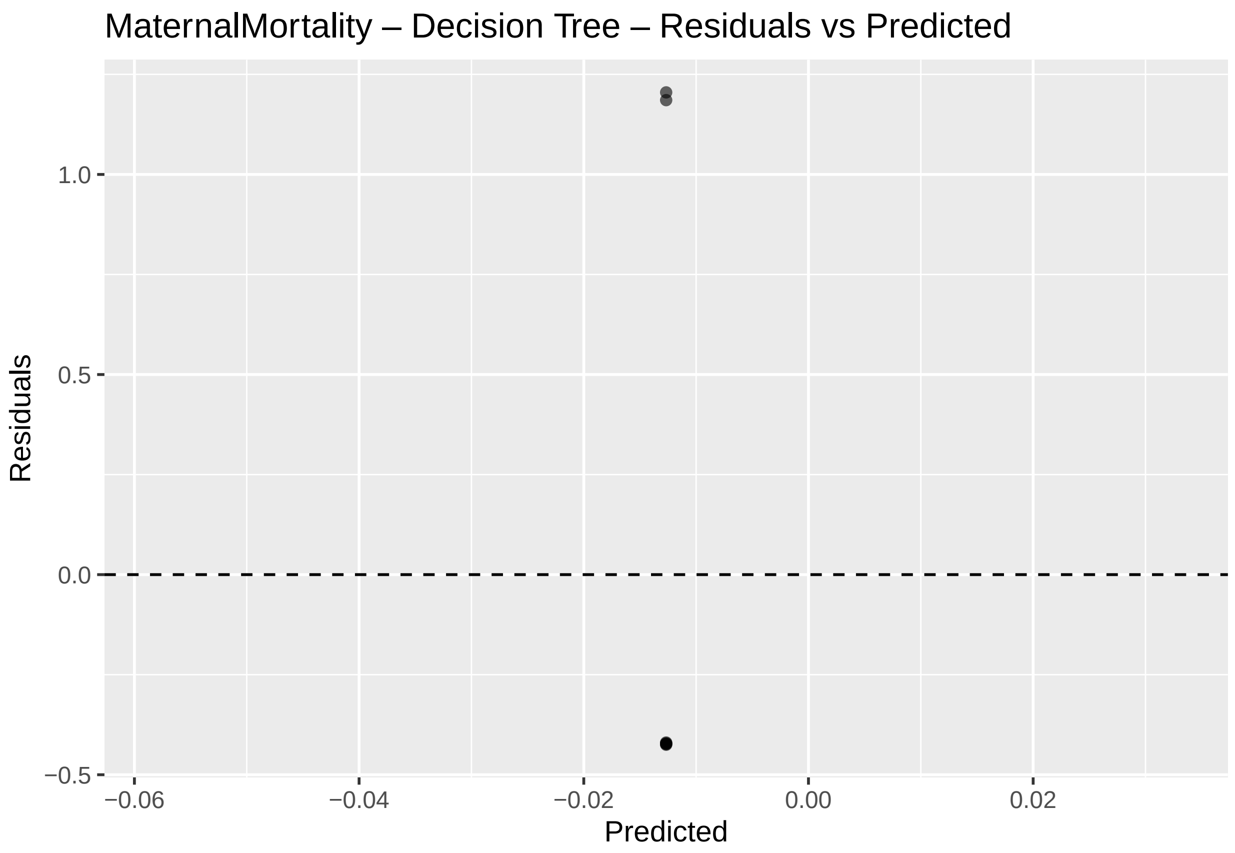

Literacy – Decision Tree – Residuals vs Predicted
Residuals
0.6
0.3
0.0
−0.3
0.025
0.050
0.075
0.100
Predicted

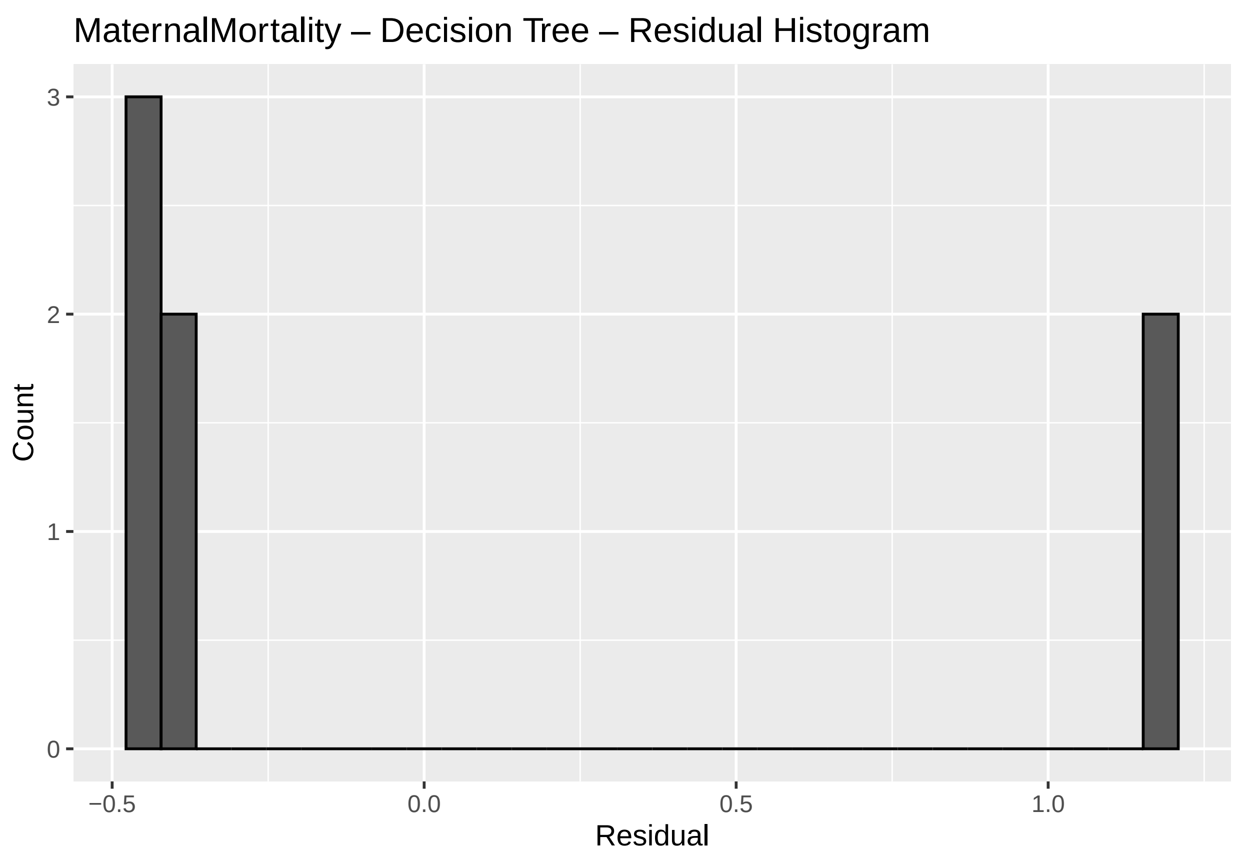

Literacy – Decision Tree – Residual Histogram
Count
3
2
1
0
−0.3
0.0
0.3
0.6
Residual

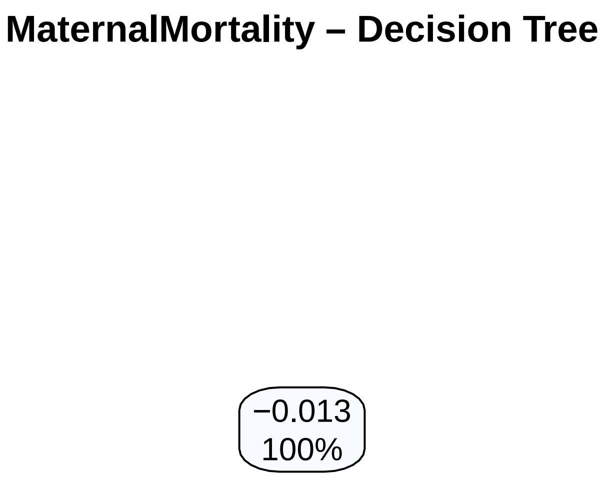
Literacy – Decision Tree



```
##
## ## Diagnostics: MaternalMortality ( Decision Tree )
```


MaternalMortality – Decision Tree – Residuals vs Predicted
Residuals
1.0
0.5
0.0
−0.5
−0.06
−0.04
−0.02
0.00
0.02
Predicted


MaternalMortality – Decision Tree – Residual Histogram
Count
3
2
1
0
−0.5
0.0
0.5
1.0
Residual



8) Save Winning Models

```
save_winner <- function(ds, model_label) {
  mdl <- switch(model_label,
                "Linear Regression" = models[[ds]]$lm,
                "Decision Tree"     = models[[ds]]$dt,
                "Random Forest"     = models[[ds]]$rf)
  if (!is.null(mdl)) {
    fn <- file.path(out_dir, paste0("best_model_", gsub("[^A-Za-z0-9_]", "", ds), "_", gsub(" ", "", moc
    saveRDS(mdl, fn)
    cat("Saved:", fn, "\n")
  }
}
purrr::pwalk(list(best_models$Dataset, best_models$Best_Model), save_winner)
```

```
## Saved: C:\Users\Suhil Jugroop\OneDrive\Documents\GitHub\BIN371-Project\Milestone 3/outputs_m3/best_mc
## Saved: C:\Users\Suhil Jugroop\OneDrive\Documents\GitHub\BIN371-Project\Milestone 3/outputs_m3/best_mc
## Saved: C:\Users\Suhil Jugroop\OneDrive\Documents\GitHub\BIN371-Project\Milestone 3/outputs_m3/best_mc
## Saved: C:\Users\Suhil Jugroop\OneDrive\Documents\GitHub\BIN371-Project\Milestone 3/outputs_m3/best_mc
```

9) Rubric Coverage (place in report)

- **Modelling Technique (10%)**: Regression task; techniques compared: LM, DT, RF; assumptions noted (LM linearity/homoscedasticity; RF nonparametric).
- **Test Design (10%)**: Predefined train/test splits; evaluation with RMSE/MAE/R^2; fixed random seed.
- **Model Building (10%)**: Training code and parameters (`rpart.control`, `ntree=500` for RF).
- **Assess the Model (10%)**: Comparison table + best model table; diagnostics (residuals, tree, RF importance).
- **Visual Storytelling (20%)**: Use the generated plots; optionally port to Power BI/Shiny.
- **Reporting & Documentation (10%)**: Knit this Rmd to HTML/PDF and narrate results.
- **Oral Presentation (20%)**: Summarize best model per dataset, key features, recommendations.
- **Project Files & Code (10%)**: Include this Rmd, saved models in `outputs_m3/`, and figures.

```
cat("\nMilestone 3 complete.\nBest models saved to: ", normalizePath(out_dir), "\n")
```

```
##
## Milestone 3 complete.
## Best models saved to:  C:\Users\Suhil Jugroop\OneDrive\Documents\GitHub\BIN371-Project\Milestone 3\ou
```