Modelling – Dataset A (Dean)

Group K

2025-10-03

Table 1: First 10 rows of Dataset A (Health)

Dataset	SurveyYear	CharacteristicId	CharacteristicCategory	CharacteristicLabel	IndicatorId	IndicatorType	Value	DenominatorW
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_DOC	I	28.5	2871
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_DOC	I	30.0	4122
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_DOC	I	27.3	2010
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_NRS	I	66.6	2871
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_NRS	I	65.0	4122
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_NRS	I	68.4	2010
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_TBA	I	0.1	2871
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_OTH	I	0.6	2871
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_OTH	I	0.7	4122
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_MIS	S	1.2	2871

```r
health_i <- dataA %>%
  filter(
    Dataset == "Access_To_Healthcare_01",
    CharacteristicCategory == "Total",
    CharacteristicLabel == "Total",
    IndicatorType == "I"
  ) %>%
  mutate(
    SurveyYear   = as.factor(SurveyYear),
    IndicatorId  = as.factor(IndicatorId),
    Value        = as.numeric(Value),
    DenominatorWeighted   = as.numeric(DenominatorWeighted),
    DenominatorUnweighted = as.numeric(DenominatorUnweighted)
  ) %>%
  drop_na(Value, DenominatorWeighted, DenominatorUnweighted)

print_tbl(head(health_i, 10), "Filtered Health (I) - head(10)")
```

Table 2: Filtered Health (I) – head(10)

Dataset	SurveyYear	CharacteristicId	CharacteristicCategory	CharacteristicLabel	IndicatorId	IndicatorType	Value	Denominator
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_DOC	I	28.5	2871
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_DOC	I	30.0	4122
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_DOC	I	27.3	2010
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_NRS	I	66.6	2871
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_NRS	I	65.0	4122
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_NRS	I	68.4	2010
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_TBA	I	0.1	2871
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_OTH	I	0.6	2871
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_OTH	I	0.7	4122
Access_To_Healthcare_01	1998	1000	Total	Total	RH_ANCP_W_NON	I	2.9	2871

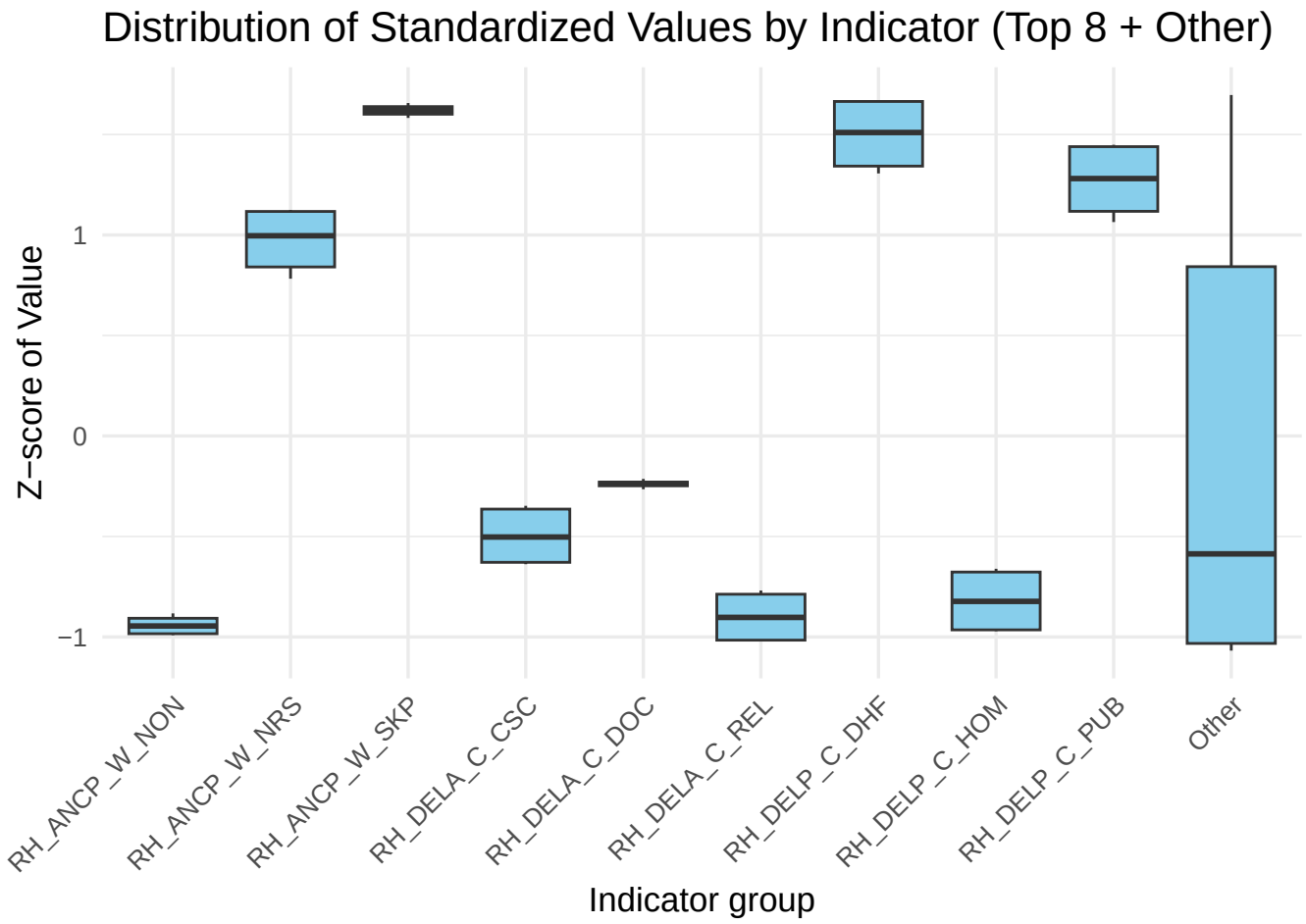
```r
med_val <- median(health_i$Value, na.rm = TRUE)
model_df <- health_i %>%
  mutate(
    HighValue = factor(if_else(Value >= med_val, "Yes", "No"), levels = c("No","Yes")),
    DW_log = log1p(DenominatorWeighted),
    DU_log = log1p(DenominatorUnweighted),
    IndicatorId = forcats::fct_lump_n(IndicatorId, n = 5, other_level = "Other")
  )

print_tbl(as.data.frame(table(model_df$HighValue)),
          "Class balance: HighValue (No/Yes)")
```

Table 3: Class balance: HighValue (No/Yes)

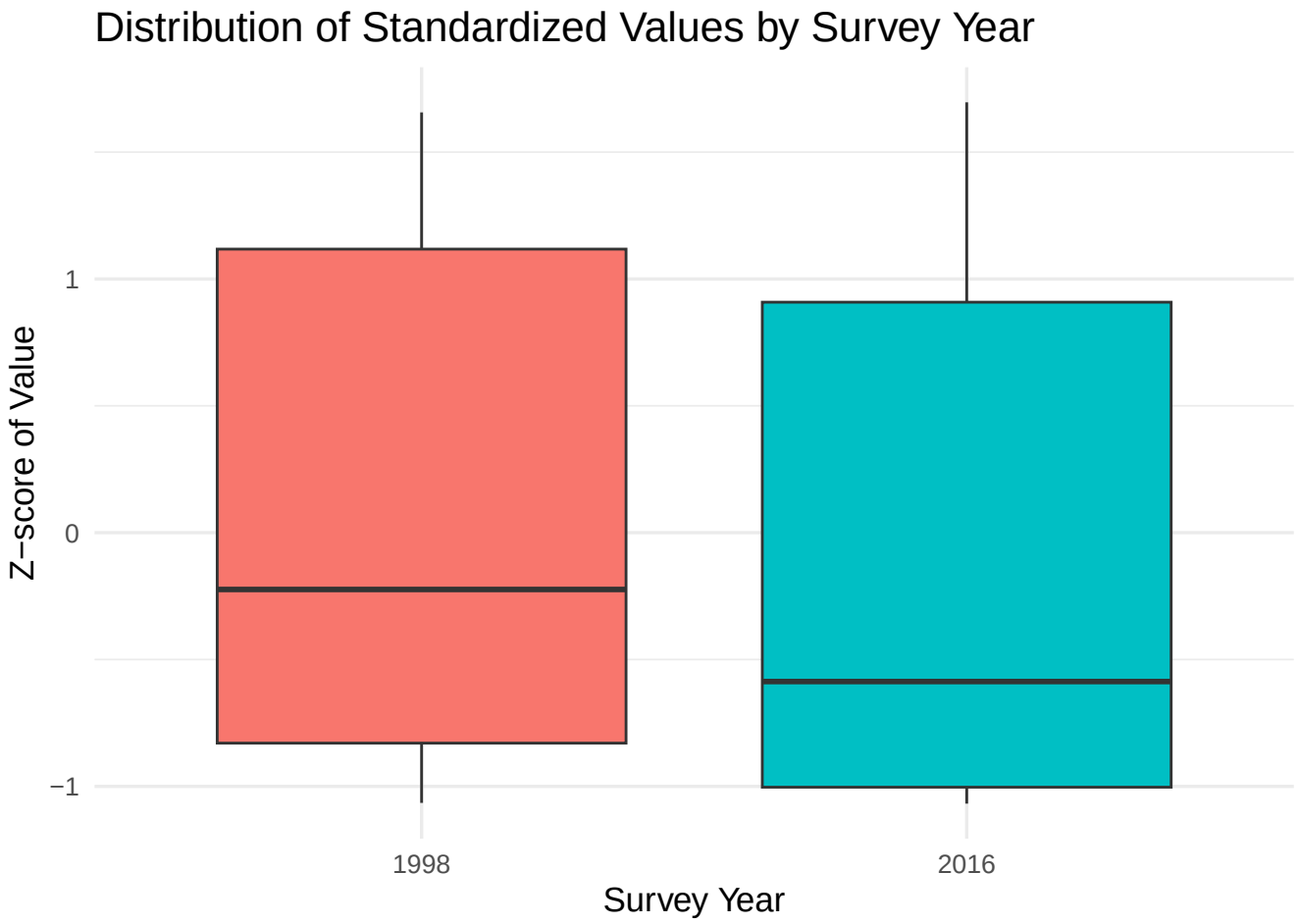
Var1	Freq
No	82
Yes	82

```r
# Build plotting data in the same spirit as Group C
plot_df <- health_i %>%
  mutate(
    # Standardized value (z-score)
    Value_zscore = as.numeric(scale(Value)),
    # Use lumped IndicatorId to avoid too many categories on plots
    IndicatorGroup = forcats::fct_lump_n(IndicatorId, n = 8, other_level = "Other"),
    # Use SurveyYear as the "population-like" grouping (since Group A lacks PopulationGroup)
    YearGroup = SurveyYear
  ) %>%
  drop_na(Value_zscore, DenominatorWeighted)
```

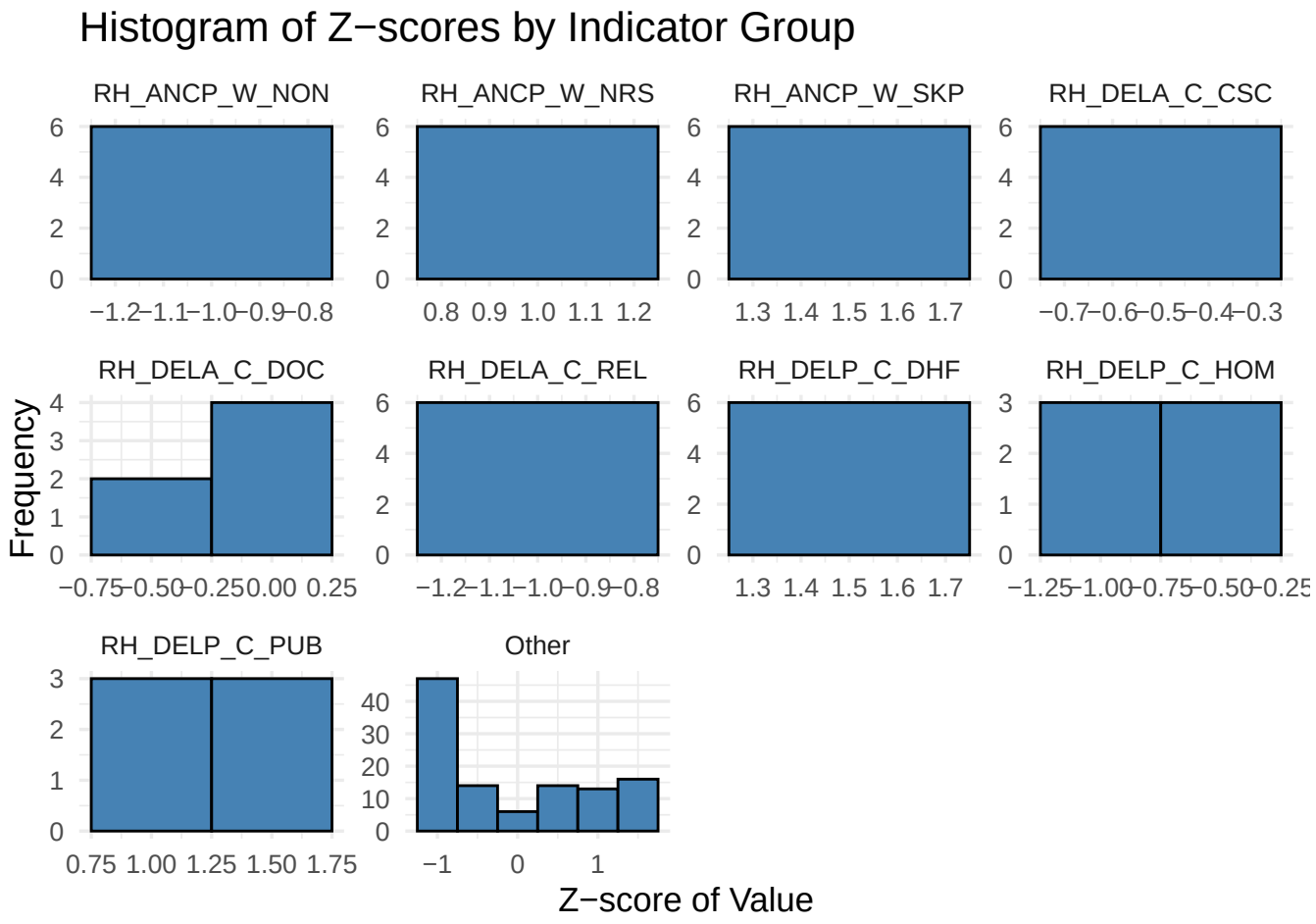
```r
ggplot(plot_df, aes(x = IndicatorGroup, y = Value_zscore)) +
  geom_boxplot(fill = "skyblue", outlier.color = "red") +
  theme_minimal(base_size = 13) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Distribution of Standardized Values by Indicator (Top 8 + Other)",
       x = "Indicator group", y = "Z-score of Value")
```



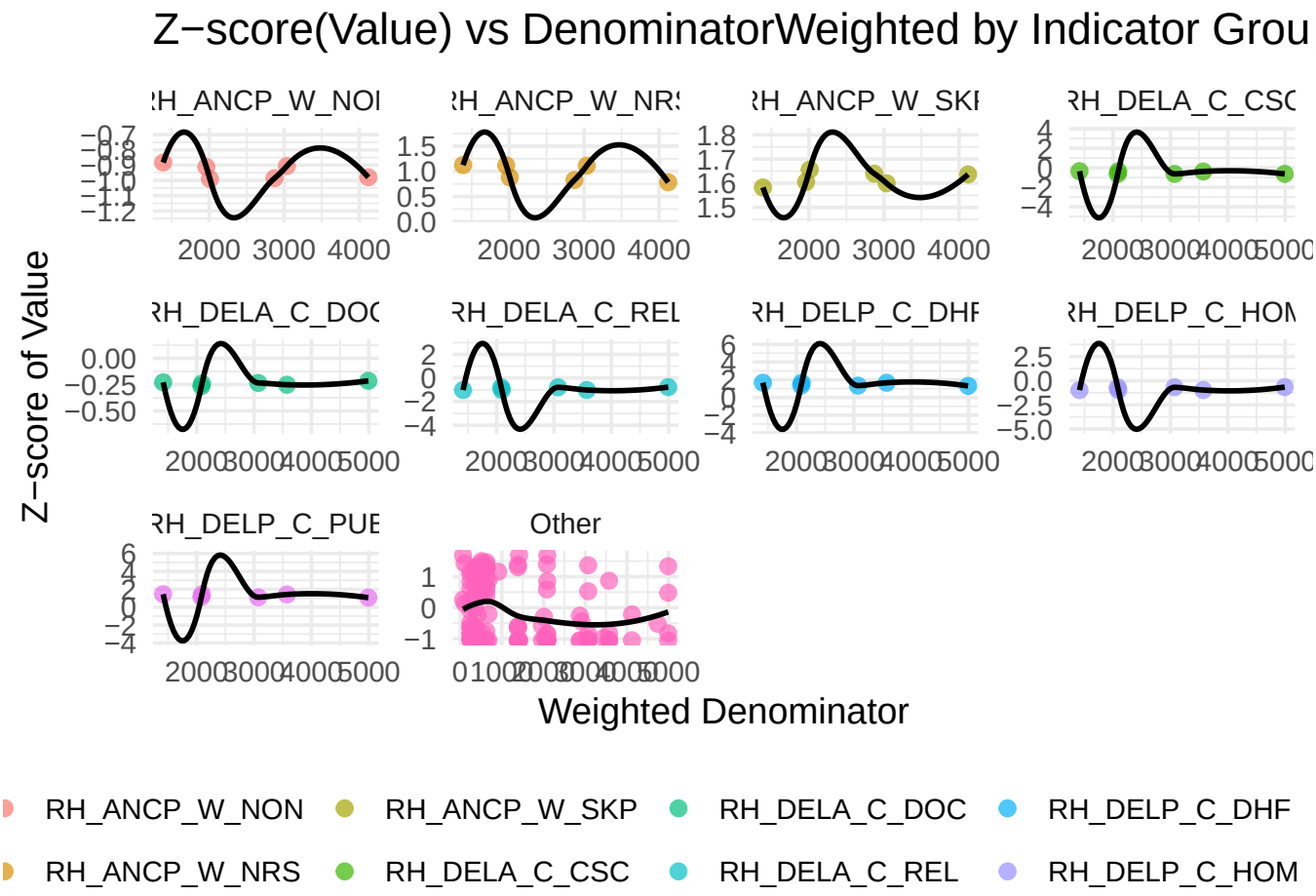

```r
ggplot(plot_df, aes(x = YearGroup, y = Value_zscore, fill = YearGroup)) +
  geom_boxplot(show.legend = FALSE) +
  theme_minimal(base_size = 13) +
  labs(title = "Distribution of Standardized Values by Survey Year",
       x = "Survey Year", y = "Z-score of Value")
```




```
ggplot(plot_df, aes(x = Value_zscore)) +
  geom_histogram(binwidth = 0.5, fill = "steelblue", color = "black") +
  facet_wrap(~ IndicatorGroup, scales = "free") +
  theme_minimal(base_size = 13) +
  labs(title = "Histogram of Z-scores by Indicator Group",
       x = "Z-score of Value", y = "Frequency")
```




```r
ggplot(plot_df, aes(x = DenominatorWeighted, y = Value_zscore, color = IndicatorGroup)) +
  geom_point(alpha = 0.7, size = 2.6) +
  geom_smooth(method = "loess", se = FALSE, color = "black") +
  facet_wrap(~ IndicatorGroup, scales = "free") +
  theme_minimal(base_size = 14) +
  labs(title = "Z-score(Value) vs DenominatorWeighted by Indicator Group",
       x = "Weighted Denominator", y = "Z-score of Value",
       color = "Indicator") +
  theme(legend.position = "bottom")
```




```r
set.seed(42)
idx <- caret::createDataPartition(model_df$HighValue, p = 0.7, list = FALSE)
train_df <- model_df[idx, ]  %>% mutate(
  SurveyYear  = forcats::fct_drop(SurveyYear),
  IndicatorId = forcats::fct_drop(IndicatorId)
)
test_df  <- model_df[-idx, ] %>% mutate(
  SurveyYear  = factor(SurveyYear,  levels = levels(train_df$SurveyYear)),
  IndicatorId = factor(IndicatorId, levels = levels(train_df$IndicatorId))
)

print_tbl(tibble(Split = c("Train","Test"),
                 Rows = c(nrow(train_df), nrow(test_df))),
          "Train/Test split sizes")
```

Table 4: Train/Test split sizes

Split	Rows
Train	116
Test	48

```
factor_preds   <- c("SurveyYear","IndicatorId")
valid_factors  <- factor_preds[sapply(train_df[factor_preds], function(x) nlevels(x) >= 2)]
num_preds      <- c("DW_log")    # avoid collinearity with DU_log
all_preds      <- c(valid_factors, num_preds)
logit_formula  <- reformulate(termlabels = all_preds, response = "HighValue")

logit_fit <- glm(logit_formula, data = train_df, family = binomial(),
                 control = list(maxit = 100))

logit_prob <- predict(logit_fit, newdata = test_df, type = "response")
logit_pred <- factor(if_else(logit_prob >= 0.5, "Yes", "No"), levels = c("No","Yes"))

cm_logit <- caret::confusionMatrix(logit_pred, test_df$HighValue, positive = "Yes")

roc_logit <- pROC::roc(response = test_df$HighValue,
                       predictor = as.numeric(logit_prob),
                       levels = c("No","Yes"), quiet = TRUE)
auc_logit <- pROC::auc(roc_logit)

png("../outputs/visuals/roc_logit.png", width = 900, height = 650)
plot.roc(roc_logit, main = sprintf("ROC - Logistic Regression (AUC = %.3f)", as.numeric(auc_log
dev.off()
```

pdf 2

```
print_tbl(as.data.frame(t(cm_logit$overall)) |> mutate(across(everything(), as.numeric)),
          "Logistic Regression - Overall Metrics (Test)", digits = 3)
```

Table 5: Logistic Regression – Overall Metrics (Test)

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.667	0.333	0.516	0.796	0.5	0.015	0.803

```
print_tbl(as.data.frame(t(cm_logit$byClass))  |> mutate(across(everything(), as.numeric)),
          "Logistic Regression - Class Metrics (Test)", digits = 3)
```

Table 6: Logistic Regression – Class Metrics (Test)

Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
0.708	0.625	0.654	0.682	0.654	0.708	0.68	0.5	0.354	0.542	0.667

```r
logit_metrics <- tibble(
  Metric = c("Accuracy","Kappa","Sensitivity","Specificity","AUC"),
  Value  = c(as.numeric(cm_logit$overall["Accuracy"]),
             as.numeric(cm_logit$overall["Kappa"]),
             as.numeric(cm_logit$byClass["Sensitivity"]),
             as.numeric(cm_logit$byClass["Specificity"]),
             as.numeric(auc_logit))
)
readr::write_csv(logit_metrics, "../outputs/summary_tables/logit_metrics.csv")

tree_formula <- reformulate(termlabels = all_preds, response = "HighValue")
tree_fit <- rpart::rpart(
  tree_formula, data = train_df, method = "class",
  control = rpart.control(cp = 0.01, minsplit = 10, maxdepth = 6)
)

png("../outputs/visuals/tree_plot.png", width = 1000, height = 700)
rpart.plot::rpart.plot(tree_fit, cex = 0.75, type = 2, extra = 104, under = TRUE,
                       main = "Decision Tree - HighValue")
dev.off()
```

pdf 2

```r
tree_prob <- predict(tree_fit, newdata = test_df, type = "prob")[, "Yes"]
tree_pred <- factor(if_else(tree_prob >= 0.5, "Yes", "No"), levels = c("No","Yes"))

cm_tree <- caret::confusionMatrix(tree_pred, test_df$HighValue, positive = "Yes")

roc_tree <- pROC::roc(response = test_df$HighValue,
                      predictor = as.numeric(tree_prob),
                      levels = c("No","Yes"), quiet = TRUE)
auc_tree <- pROC::auc(roc_tree)

png("../outputs/visuals/roc_tree.png", width = 900, height = 650)
plot.roc(roc_tree, main = sprintf("ROC - Decision Tree (AUC = %.3f)", as.numeric(auc_tree)))
dev.off()
```

pdf 2

```r
print_tbl(as.data.frame(t(cm_tree$overall)) |> mutate(across(everything(), as.numeric)),
          "Decision Tree - Overall Metrics (Test)", digits = 3)
```

Table 7: Decision Tree – Overall Metrics (Test)

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.708	0.417	0.559	0.83	0.5	0.003	0.423

```
print_tbl(as.data.frame(t(cm_tree$byClass)) |> mutate(across(everything(), as.numeric)),
          "Decision Tree - Class Metrics (Test)", digits = 3)
```

Table 8: Decision Tree – Class Metrics (Test)

Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
0.792	0.625	0.679	0.75	0.679	0.792	0.731	0.5	0.396	0.583	0.708

```r
varimp <- as.data.frame(varImp(tree_fit))
varimp$Feature <- rownames(varimp)
varimp <- varimp %>% arrange(desc(Overall))
print_tbl(head(varimp, 10), "Decision Tree - Top 10 Feature Importance")
```

Table 9: Decision Tree – Top 10 Feature Importance

	Overall	Feature
IndicatorId	15.976	IndicatorId
DW_log	11.800	DW_log
SurveyYear	0.704	SurveyYear

```r
readr::write_csv(
  tibble(Metric=c("Accuracy","Kappa","Sensitivity","Specificity","AUC"),
         Value=c(as.numeric(cm_tree$overall["Accuracy"]),
                 as.numeric(cm_tree$overall["Kappa"]),
                 as.numeric(cm_tree$byClass["Sensitivity"]),
                 as.numeric(cm_tree$byClass["Specificity"]),
                 as.numeric(auc_tree))),
  "../outputs/summary_tables/tree_metrics.csv"
)
readr::write_csv(varimp, "../outputs/summary_tables/tree_variable_importance.csv")
```

```r
perf_tbl <- tibble(
  Model = c("Logistic Regression", "Decision Tree"),
  Accuracy = c(cm_logit$overall["Accuracy"], cm_tree$overall["Accuracy"]) %>% as.numeric(),
  Kappa    = c(cm_logit$overall["Kappa"],    cm_tree$overall["Kappa"]) %>% as.numeric(),
  Sensitivity = c(cm_logit$byClass["Sensitivity"], cm_tree$byClass["Sensitivity"]) %>% as.numer
  Specificity = c(cm_logit$byClass["Specificity"], cm_tree$byClass["Specificity"]) %>% as.numer
  AUC = c(as.numeric(auc_logit), as.numeric(auc_tree))
)
print_tbl(perf_tbl, "Model Performance Comparison (Test Set)", digits = 3)
```

Table 10: Model Performance Comparison (Test Set)

Model	Accuracy	Kappa	Sensitivity	Specificity	AUC
Logistic Regression	0.667	0.333	0.708	0.625	0.832
Decision Tree	0.708	0.417	0.792	0.625	0.786

```r
readr::write_csv(perf_tbl, "../outputs/summary_tables/model_performance_comparison.csv")
```

```r
# pick the best row using dplyr-qualified calls to avoid masking
best <- perf_tbl %>%
```

```
  dplyr::arrange(dplyr::desc(AUC), dplyr::desc(Accuracy)) %>%
  dplyr::slice_head(n = 1)

cat("**Summary.** We trained two models on Dataset A's health indicators (binary target: HighVa
    "\\(\\geq\\) median(Value)). The table above shows test-set metrics. ",
    "Based on AUC (primary) and Accuracy (secondary), the better model in this run is ",
    sprintf("**%s** (AUC = %.3f, Accuracy = %.3f). ",
            best$Model, best$AUC, best$Accuracy),
    "We also saved ROC curves, a tree plot, and CSV metrics in `../outputs/`.",
    sep = "")
```

Summary. We trained two models on Dataset A's health indicators (binary target: HighValue $\geq$ median(Value)). The table above shows test-set metrics. Based on AUC (primary) and Accuracy (secondary), the better model in this run is **Logistic Regression** (AUC = 0.832, Accuracy = 0.667). We also saved ROC curves, a tree plot, and CSV metrics in `../outputs/`.