Supplementary File 1

Practical Application of the Dementia Risk Algorithm

We provide an accessible Excel tool (Supplementary Data 1) to allow clinicians and researchers to estimate individual 10-year dementia risk using four routinely available variables: age, cognition, glucose, and cardiovascular risk score. The algorithm applies the regression equation derived from our LASSO model and automatically recalibrates the baseline intercept when applied to individuals under 70 years of age, where dementia prevalence is much lower than in the training cohort (aged ≥70). For those aged 70 and above, the model is used without recalibration.

To use the tool, users enter predictor values for each individual. The calculator then outputs the absolute 10-year probability of dementia and a classification ("At risk" if ≥0.277; "Lower risk" otherwise). For younger adults (40–69 years), the tool requires age-band–specific prevalence estimates from the target population. By default, the tool uses global prevalence figures, but these can be replaced with national or local estimates. This ensures that predicted risks remain consistent with epidemiological reality while preserving the relative contributions of predictors within the model.

The algorithm is designed for transparency and ease of use: prevalence entries are validated, the intercept adjustment is performed automatically, and outputs are presented in both probability and binary classification formats. The approach mirrors best practices in cardiovascular risk prediction, offering a clinically interpretable tool that can be readily adapted to diverse populations.

Supplementary File 2

R Code

```
# ============================
# MAS Dementia Prediction — Final Reproducible Script
# Trains 4 models on 70% of MAS baseline data, validates on 30%,
# evaluates with AUC, Youden & FIXED adjusted thresholds, calibration, DCA,
# and writes publication-ready outputs to Desktop.
# ============================

# --- Packages ---
required_packages <- c(
  "readxl", "pROC", "caret", "glmnet", "randomForest", "xgboost",
  "ResourceSelection", "rmda", "dplyr", "ggplot2", "PRROC"
)
invisible(lapply(required_packages, function(pkg) {
  if (!suppressWarnings(require(pkg, character.only = TRUE))) {
    install.packages(pkg, dependencies = TRUE)
    library(pkg, character.only = TRUE)
  }
}))

# --- Paths & seed ---
input_path  <- "C:/Users/olegm/OneDrive/Desktop/Study4Imputed.xlsx"
output_path <- "C:/Users/olegm/OneDrive/Desktop/"
stopifnot(dir.exists(output_path))
```

```
set.seed(123)

# --- Load data ---
data <- readxl::read_excel(input_path)
data$Dementia <- factor(data$Dementia, levels = c(0, 1))
if ("Timetoevent" %in% names(data)) data$Timetoevent <- NULL

# --- Train/Test split (stratified 70/30) ---
idx <- caret::createDataPartition(data$Dementia, p = 0.7, list = FALSE)
train_data <- data[idx, , drop = FALSE]
test_data  <- data[-idx, , drop = FALSE]

# --- Adjusted (fixed) thresholds used for the paper (override Youden) ---
# If you prefer to revert to Youden for any model, set that entry to NA.
fixed_thresholds <- list(
  Logit = 0.346,   # Logistic
  Lasso = 0.277,   # LASSO
  RF    = 0.340,   # Random Forest
  XGB   = 0.027    # XGBoost
)

# --- Helpers ---
to_num <- function(x) {
  if (is.null(x)) return(numeric(0))
  if (is.factor(x)) x <- as.character(x)
```

```
  if (is.data.frame(x)) x <- x[[1]]
  if (is.list(x)) x <- unlist(x, recursive = TRUE, use.names = FALSE)
  as.numeric(x)
}
fmt_p <- function(p) {
  if (is.na(p)) return("NA")
  if (p < 0.001) return("< 0.001")
  formatC(p, digits = 3, format = "f")
}

# --- Core evaluator: saves ROC, metrics .txt, DCA input, one-row CSV; returns a row ---
evaluate_model <- function(model_name, probs, actual_factor, tag,
                           output_path, force_threshold = NA_real_) {
  probs  <- to_num(probs)
  actual <- factor(actual_factor, levels = c(0, 1))
  actual_num <- as.numeric(as.character(actual))

  # Safety
  stopifnot(length(probs) == length(actual))
  if (anyNA(probs) || anyNA(actual)) stop("Missing values in probs/actual.")

  # ROC/AUC (explicit direction & levels)
  roc_obj <- pROC::roc(response = actual, predictor = probs,
                       levels = c("0", "1"), direction = ">")
  auc_val <- as.numeric(pROC::auc(roc_obj))
```

```
# Threshold: Youden by default; override if force_threshold provided
youden_thr <- as.numeric(pROC::coords(roc_obj, "best",
                                      best.method = "youden",
                                      ret = "threshold"))
threshold <- ifelse(is.na(force_threshold), youden_thr, force_threshold)

# Classify & confusion metrics
predicted <- factor(ifelse(probs > threshold, 1, 0), levels = c(0, 1))
cm <- caret::confusionMatrix(predicted, actual, positive = "1")
sensitivity <- as.numeric(cm$byClass["Sensitivity"])
specificity <- as.numeric(cm$byClass["Specificity"])
ppv <- as.numeric(cm$byClass["Pos Pred Value"])
npv <- as.numeric(cm$byClass["Neg Pred Value"])

# Calibration (Brier, intercept, slope)
probs_clamped <- pmin(pmax(probs, 1e-6), 1 - 1e-6)
brier <- mean((actual_num - probs)^2)
cal_fit <- glm(actual_num ~ qlogis(probs_clamped), family = binomial)
cal_intercept <- unname(coef(cal_fit)[1])
cal_slope     <- unname(coef(cal_fit)[2])

# PR-AUC (useful with skew)
pr_obj <- PRROC::pr.curve(scores.class0 = probs[actual_num == 1],
                          scores.class1 = probs[actual_num == 0],
```

```
                        curve = FALSE)
pr_auc <- unname(pr_obj$auc.integral)


# Hosmer–Lemeshow (for logistic-type models only)
hl_stat <- "NA"
if (model_name %in% c("Logistic", "LASSO")) {
  hoslem <- ResourceSelection::hoslem.test(actual_num, probs, g = 10)
  hl_stat <- sprintf("X2 = %.2f, p = %s",
                     as.numeric(hoslem$statistic), fmt_p(hoslem$p.value))
}


# Save metrics .txt (journal-friendly)
writeLines(
  c(
    paste(model_name, "AUC:", round(auc_val, 3)),
    paste("PR-AUC:", round(pr_auc, 3)),
    paste("Brier:", round(brier, 4)),
    paste("Cal. Intercept:", round(cal_intercept, 3)),
    paste("Cal. Slope:", round(cal_slope, 3)),
    paste("Sensitivity:", round(sensitivity, 3)),
    paste("Specificity:", round(specificity, 3)),
    paste("PPV:", round(ppv, 3)),
    paste("NPV:", round(npv, 3)),
    paste("Threshold (used):", round(threshold, 3),
          ifelse(is.na(force_threshold), "(Youden)", "(Adjusted)")),
```

```
    paste("Hosmer–Lemeshow:", hl_stat)
  ),
  file.path(output_path, paste0(tag, "_Metrics.txt"))
)

# Save ROC plot
png(file.path(output_path, paste0(tag, "_ROC.png")), width = 800, height = 600)
plot(roc_obj, main = paste("ROC Curve:", model_name), col = "blue", lwd = 2)
dev.off()

# Save DCA input
write.csv(
  data.frame(probs = probs, Dementia = actual_num),
  file.path(output_path, paste0(tag, "_DCA_Input.csv")),
  row.names = FALSE
)

# One-row CSV for comparison table
out_row <- data.frame(
  Model = model_name,
  AUC = auc_val,
  PR_AUC = pr_auc,
  Brier = brier,
  Cal_Intercept = cal_intercept,
  Cal_Slope = cal_slope,
```

```
    Sensitivity = sensitivity,
    Specificity = specificity,
    PPV = ppv,
    NPV = npv,
    Threshold_Used = threshold,
    HL_Test = hl_stat,
    stringsAsFactors = FALSE
  )
  write.csv(out_row,
            file.path(output_path, paste0(tag, "_Metrics_Row.csv")),
            row.names = FALSE)
  out_row
}

# ============================
# Train models
# ============================

comparison_rows <- list()

# 1) Logistic regression
logit_model <- glm(Dementia ~ ., data = train_data, family = binomial)
logit_probs <- predict(logit_model, newdata = test_data, type = "response")
comparison_rows[["Logistic"]] <-
  evaluate_model("Logistic", logit_probs, test_data$Dementia, "Logit",
```

```
            output_path, force_threshold = fixed_thresholds$Logit)
saveRDS(logit_model, file.path(output_path, "Logit_Model.rds"))

# 2) LASSO (glmnet) — CV on train, predict test
X_train <- model.matrix(Dementia ~ . - 1, data = train_data)
X_test  <- model.matrix(Dementia ~ . - 1, data = test_data)
y_train_num <- as.numeric(as.character(train_data$Dementia))
cv_fit <- glmnet::cv.glmnet(X_train, y_train_num, alpha = 1, family = "binomial")
lasso_probs <- as.numeric(predict(cv_fit, newx = X_test, s = "lambda.min", type =
"response"))
comparison_rows[["LASSO"]] <-
  evaluate_model("LASSO", lasso_probs, test_data$Dementia, "Lasso",
            output_path, force_threshold = fixed_thresholds$Lasso)
saveRDS(cv_fit, file.path(output_path, "Lasso_Model.rds"))

# Save non-zero LASSO coefficients (for reporting)
coefs <- coef(cv_fit, s = "lambda.min")
coef_df <- data.frame(Feature = rownames(coefs), Coefficient = as.numeric(coefs),
row.names = NULL)
nz_coefs <- subset(coef_df, Coefficient != 0)
write.csv(nz_coefs, file.path(output_path, "Lasso_Coefficients.csv"), row.names = FALSE)

# 3) Random Forest
set.seed(123)
rf_model <- randomForest::randomForest(Dementia ~ ., data = train_data,
```

```
                                ntree = 1000, importance = TRUE)
rf_probs <- predict(rf_model, newdata = test_data, type = "prob")[, 2]
comparison_rows[["Random Forest"]] <-
  evaluate_model("Random Forest", rf_probs, test_data$Dementia, "RF",
                 output_path, force_threshold = fixed_thresholds$RF)
saveRDS(rf_model, file.path(output_path, "RF_Model.rds"))

# RF variable importance
png(file.path(output_path, "RF_Importance.png"), width = 800, height = 600)
varImpPlot(rf_model, main = "Variable Importance: Random Forest")
dev.off()

# 4) XGBoost
y_train_xgb <- as.numeric(as.character(train_data$Dementia))
dtrain <- xgboost::xgb.DMatrix(data = X_train, label = y_train_xgb)
dtest  <- xgboost::xgb.DMatrix(data = X_test)
set.seed(123)
xgb_model <- xgboost::xgboost(
  data = dtrain, objective = "binary:logistic",
  eval_metric = "auc", nrounds = 300, verbose = 0
)
xgb_probs <- predict(xgb_model, newdata = dtest)
comparison_rows[["XGBoost"]] <-
  evaluate_model("XGBoost", xgb_probs, test_data$Dementia, "XGB",
                 output_path, force_threshold = fixed_thresholds$XGB)
```

```
saveRDS(xgb_model, file.path(output_path, "XGB_Model.rds"))

# ============================
# Comparison table, Calibration, DCA
# ============================

# Model comparison CSV
comparison_df <- dplyr::bind_rows(comparison_rows)
write.csv(comparison_df, file.path(output_path, "Model_Comparison.csv"), row.names =
FALSE)

# Calibration plot (Logistic, deciles)
cal_data <- data.frame(
  predicted = logit_probs,
  actual = as.numeric(as.character(test_data$Dementia))
)
qs <- unique(quantile(cal_data$predicted, probs = seq(0, 1, 0.1)))
if (length(qs) < 3L) qs <- seq(0, 1, length.out = 11)
cal_data$bin <- cut(cal_data$predicted, breaks = qs, include.lowest = TRUE)
cal_plot_data <- cal_data %>%
  group_by(bin) %>%
  summarise(Predicted = mean(predicted), Observed = mean(actual), .groups = "drop")
png(file.path(output_path, "Logit_Calibration_Plot.png"), width = 800, height = 600)
ggplot(cal_plot_data, aes(x = Predicted, y = Observed)) +
  geom_point(size = 3, color = "blue") +
```

```
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  xlim(0, 1) + ylim(0, 1) +
  labs(title = "Calibration Plot: Logistic Regression",
       x = "Predicted Probability", y = "Observed Proportion") +
  theme_minimal()
dev.off()

# Decision-curve analysis (load per-model DCA inputs saved above)
dca_logit <- read.csv(file.path(output_path, "Logit_DCA_Input.csv"))
dca_lasso <- read.csv(file.path(output_path, "Lasso_DCA_Input.csv"))
dca_rf    <- read.csv(file.path(output_path, "RF_DCA_Input.csv"))
dca_xgb   <- read.csv(file.path(output_path, "XGB_DCA_Input.csv"))

thresh_seq <- seq(0.01, 0.99, by = 0.01)
dc_logit <- rmda::decision_curve(Dementia ~ probs, data = dca_logit, family = binomial,
thresholds = thresh_seq)
dc_lasso <- rmda::decision_curve(Dementia ~ probs, data = dca_lasso, family = binomial,
thresholds = thresh_seq)
dc_rf    <- rmda::decision_curve(Dementia ~ probs, data = dca_rf,    family = binomial,
thresholds = thresh_seq)
dc_xgb   <- rmda::decision_curve(Dementia ~ probs, data = dca_xgb,   family = binomial,
thresholds = thresh_seq)

# Save Net Benefit tables
```

```
write.csv(as.data.frame(dc_logit$derived.data), file.path(output_path,
"Logit_DCA_NetBenefit.csv"), row.names = FALSE)
write.csv(as.data.frame(dc_lasso$derived.data), file.path(output_path,
"Lasso_DCA_NetBenefit.csv"), row.names = FALSE)
write.csv(as.data.frame(dc_rf$derived.data),    file.path(output_path,
"RF_DCA_NetBenefit.csv"),    row.names = FALSE)
write.csv(as.data.frame(dc_xgb$derived.data),   file.path(output_path,
"XGB_DCA_NetBenefit.csv"),   row.names = FALSE)

# Combined DCA plot
png(file.path(output_path, "Decision_Curve_Comparison.png"), width = 800, height = 600)
rmda::plot_decision_curve(
  list(dc_logit, dc_lasso, dc_rf, dc_xgb),
  curve.names = c("Logistic", "LASSO", "Random Forest", "XGBoost"),
  cost.benefit.axis = FALSE,
  confidence.intervals = FALSE,
  standardize = TRUE,
  col = c("blue", "green", "orange", "red"),
  lty = 1, lwd = 2
)
dev.off()
```

Supplementary File 1

Practical Application of the Dementia Risk Algorithm

We provide an accessible Excel tool (Supplementary Data 1) to allow clinicians and researchers to estimate individual 10-year dementia risk using four routinely available variables: age, cognition, glucose, and cardiovascular risk score. The algorithm applies the regression equation derived from our LASSO model and automatically recalibrates the baseline intercept when applied to individuals under 70 years of age, where dementia prevalence is much lower than in the training cohort (aged ≥70). For those aged 70 and above, the model is used without recalibration.

To use the tool, users enter predictor values for each individual. The calculator then outputs the absolute 10-year probability of dementia and a classification ("At risk" if ≥0.277; "Lower risk" otherwise). For younger adults (40–69 years), the tool requires age-band–specific prevalence estimates from the target population. By default, the tool uses global prevalence figures, but these can be replaced with national or local estimates. This ensures that predicted risks remain consistent with epidemiological reality while preserving the relative contributions of predictors within the model.

The algorithm is designed for transparency and ease of use: prevalence entries are validated, the intercept adjustment is performed automatically, and outputs are presented in both probability and binary classification formats. The approach mirrors best practices in cardiovascular risk prediction, offering a clinically interpretable tool that can be readily adapted to diverse populations.