SP Modeling (without sampling)

Ruijian Maggie Lin

2025-04-25

Use the csv file named “SP_Modeling_dataset.csv” to run the future steps.

```r
final_data <- read.csv("SP_Modeling_dataset.csv")
```

Train-Test Split at the participant level

```r
set.seed(2025)

# Step 1: Train-test split at the participant level
participants <- final_data %>% distinct(spid)
train_index <- createDataPartition(participants$spid, p = 0.8, list = FALSE)  # 80% training
train_participants <- participants$spid[train_index]
test_participants <- participants$spid[-train_index]

train_data <- final_data %>%
  filter(spid %in% train_participants) %>%

  # Center round at the mean (or first round)
  group_by(spid) %>%
  mutate(
    round = round - min(round)  # Ensure first round is 0 for all participants
  ) %>%
  ungroup()

test_data <- final_data %>%
  filter(spid %in% test_participants) %>%

  # Center round at the mean (or first round)
  group_by(spid) %>%
  mutate(
    round = round - min(round)  # Ensure first round is 0 for all participants
  ) %>%
  ungroup()

# Step 2: Identify "Never Institutionalized" and "Transitioned" in training set
train_never_institution <- train_data %>%
  group_by(spid) %>%
  summarise(all_zero = all(finalres == 0)) %>%
  filter(all_zero) %>%
  pull(spid)

train_never_institution_data <- train_data %>%
  filter(spid %in% train_never_institution)

train_transition_to_institution <- train_data %>%
  group_by(spid) %>%
  summarise(first_status = first(finalres), last_status = last(finalres)) %>%
  filter(first_status == 0 & last_status == 1) %>%
  pull(spid)

train_institution_data <- train_data %>%
  filter(spid %in% train_transition_to_institution)

# Check the number of participants in each group in train set
cat("Number of Never Institutionalized participants in training set:", length(train_never_institution),
```

```
## Number of Never Institutionalized participants in training set: 7494
```

```r
cat("Number of Transitioned participants in training set:", length(train_transition_to_institution), "\
```

```
## Number of Transitioned participants in training set: 382
```

```r
# Step 3: Identify "Never Institutionalized" and "Transitioned" in the testing set
test_never_institution <- test_data %>%
  group_by(spid) %>%
  summarise(all_zero = all(finalres == 0)) %>%
  filter(all_zero) %>%
  select(spid)

test_transition_to_institution <- test_data %>%
  group_by(spid) %>%
  summarise(first_status = first(finalres), last_status = last(finalres)) %>%
  filter(first_status == 0 & last_status == 1) %>%
  select(spid)

# Check the number of participants in each group in testing set
num_test_never_institution <- nrow(test_never_institution)
num_test_transitioned <- nrow(test_transition_to_institution)

cat("Number of Never Institutionalized participants in testing set:", num_test_never_institution, "\n")
```

```
## Number of Never Institutionalized participants in testing set: 1869
```

```r
cat("Number of Transitioned participants in testing set:", num_test_transitioned, "\n")
```

```
## Number of Transitioned participants in testing set: 99
```

```r
# Data leakage prevention: Check for overlapping participants
overlap <- intersect(train_data$spid, test_data$spid)
if (length(overlap) == 0) {
  cat("No overlapping participants between train and test sets.")
} else {
  cat("Data leakage detected! Overlapping SPIDs:", overlap)
}
```

```
## No overlapping participants between train and test sets.
```

Modeling

1st Model: 2-Stages | Generalized Linear Mixed Model (GLMM) + Logistic Regression [More Interterpretable]

Why Not a Single Model?

Separating longitudinal modeling (Stage 1) from prediction (Stage 2) improves interpretability:

- Stage 1 isolates temporal trends.
- Stage 2 quantifies how those trends + covariates predict institutionalization.

Stage 1: GLMM (Generalized Linear Mixed Model)

We started by modeling individual changes in social participation limitation (`fparlim`)over time (`round`) using a GLMM. This helps us estimate a personalized rate of change in social participation limitation for each participant.

```r
set.seed(2025)

train_data$round_scaled <- scale(train_data$round)

# Fit GLMM on training data with random intercepts and slopes for each participant
glmm_train <- glmer(
  fparlim ~ round_scaled + (1 + round_scaled | spid),  # Correlated intercept/slope
  data = train_data, family = binomial,
  control = glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e5))
)

summary(glmm_train)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: fparlim ~ round_scaled + (1 + round_scaled | spid)
##    Data: train_data
## Control: glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e+05))
##
##      AIC      BIC   logLik deviance df.resid
##  48600.6  48643.8 -24295.3  48590.6    41839
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.9130 -0.5480 -0.3352  0.6692  2.3699
##
## Random effects:
##  Groups Name         Variance Std.Dev. Corr
##  spid   (Intercept)  2.3752   1.5412
##         round_scaled 0.1734   0.4164   -0.26
## Number of obs: 41844, groups:  spid, 7876
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.90473    0.02389 -37.864   <2e-16 ***
## round_scaled  0.14168    0.01655   8.563   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## round_scald -0.005
```

- **Fixed Effects:** Highlight the overall trend (e.g., "Social participation limitations worsen slightly over time").
- **Random Effects:** Explain why some participants deviate from this trend (e.g., "High-risk subgroups exist with worsening trajectories").
- **Actionable Insight:** Target interventions for participants with high baseline limitations and positive slopes.

Fixed Effects

- **Baseline**: At mean `round` (scaled), average probability of limitation = 28.8% (log-odds = -0.905).
- **Time Trend**: Each SD increase in `round` $\uparrow$ odds of limitation by 15% ($OR = 1.15$, $p < 0.001$).

Random Effects

- **Baseline Variability**: High between participants (SD = 1.54 $\rightarrow$ 2% to 89% baseline probabilities).
- **Trend Variability**: Participants' trends differ (95% slopes: -0.67 to +0.95 log-odds/SD).
- **Correlation**: Higher baseline limitations linked to slower worsening (r = -0.26).

Implications
Population shows slight $\uparrow$ in limitations over time, but subgroups vary widely (both baseline and trends). Prioritize high-baseline-risk participants for interventions.

Goal: Identifies participants with improving/worsening social participation patterns.

Stage 2: Logistic Regression on Slopes

Now that we had each person's **rate of social participation change**, we used that slope to see if it could **predict whether the person eventually entered institutional care**.

```r
set.seed(2025)

# Extract training slopes (fixed + random effects)
train_slopes <- data.frame(
  spid = rownames(ranef(glmm_train)$spid),
  slope = fixef(glmm_train)["round_scaled"] + ranef(glmm_train)$spid$round_scaled
)

# Convert spid in your training set to character
train_data <- train_data %>%
  mutate(spid = as.character(spid))

# Create training summary dataset
train_summary <- train_data %>%
  arrange(spid, round) %>%
  group_by(spid) %>%
  summarise(
    finalres = max(finalres),
    fnewhx = last(fnewhx),
    fhcaccess = last(fhcaccess),
    fcogcx = last(fcogcx),
    fen = last(fen),
    fage = last(fage)
  ) %>%
  left_join(train_slopes, by = "spid") %>%
  select(spid, finalres, fnewhx, fhcaccess, fcogcx, fen, fage,
         slope) %>%
  ungroup()

# Age encoding (ordinal categorical)
age_levels <- c("65-69", "70-74", "75-79", "80-84", "85-89", "90+")
train_summary$fage <- factor(train_summary$fage, levels = age_levels)

# Fit logistic model
institution_model <- glm(
  finalres ~ slope + fnewhx + fhcaccess + fcogcx + fen + fage,
  data = train_summary, family = binomial)

summary(institution_model)
```

```
##
## Call:
## glm(formula = finalres ~ slope + fnewhx + fhcaccess + fcogcx +
##     fen + fage, family = binomial, data = train_summary)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.6155090  0.5270249  -8.758  < 2e-16 ***
## slope       -2.0427469  0.3669144  -5.567 2.59e-08 ***
## fnewhx       3.5525094  0.3964547   8.961  < 2e-16 ***
## fhcaccess   -1.0538161  0.3719281  -2.833  0.00461 **
## fcogcx      -2.4439431  0.1373750 -17.790  < 2e-16 ***
## fen         -0.0003254  0.1772330  -0.002  0.99853
## fage70-74    0.3607879  0.5625374   0.641  0.52129
## fage75-79    0.8462815  0.5382135   1.572  0.11586
## fage80-84    1.3565978  0.5291152   2.564  0.01035 *
## fage85-89    1.6752474  0.5265331   3.182  0.00146 **
## fage90+      2.0497961  0.5243240   3.909 9.25e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3057.1  on 7875  degrees of freedom
## Residual deviance: 2504.4  on 7865  degrees of freedom
## AIC: 2526.4
##
## Number of Fisher Scoring iterations: 7
```

Key Predictors of Institutionalization

1. **Slope of Social Limitation Change**:

   - Each unit ↑ in slope (faster *increase* in limitations) ↓ odds of institutional care by **87%** (OR = 0.13, p < 0.001).

2. **Health & Cognitive Factors**:

   - **New Health Events (fnewhx)**: Presence ↑ odds by **35x** (OR = 34.8, p < 0.001).
   - **Healthcare Access (fhcaccess)**: Access ↓ odds by **65%** (OR = 0.35, p = 0.005).
   - **Cognitive Function (fcogcx)**: Each unit ↑ ↓ odds by **91%** (OR = 0.087, p < 0.001).

3. **Age**:

   - **90+ age group** vs. 65-69: ↑ odds by **7.8x** (OR = 7.76, p < 0.001).
   - Trend: Older age → progressively higher risk (e.g., 85-89: OR = 5.34, p = 0.001).

Non-Significant Predictors

- **Financial Status (fen)**: No effect (p = 0.998).
- **Ages 70-84**: No significant risk vs. 65-69 (p > 0.05).

Implications

- **Counterintuitive Slope**: Faster worsening of social limitations linked to *lower* institutionalization risk (needs further investigation).
- **Critical Risks**: New health events, poor cognition, and advanced age drive institutional care.
- **Protective Factors**: Healthcare access mitigates risk.

```r
set.seed(2025)

# Create model matrix
X <- model.matrix(
  ~ slope + fnewhx + fhcaccess + fcogcx + fen + fage - 1,
  data = train_summary
)

# Calculate class weight
class_ratio <- sum(train_summary$finalres == 0) / sum(train_summary$finalres == 1)
dtrain <- xgb.DMatrix(data = X, label = train_summary$finalres)

# Cross-validated parameter tuning
params <- list(
  objective = "binary:logistic",
  eval_metric = "aucpr",  # Better for imbalanced data
  max_depth = 3,
  eta = 0.1,
  scale_pos_weight = class_ratio
)

xgb_cv <- xgb.cv(
  params = params,
  data = dtrain,
  nrounds = 500,
  nfold = 5,
  early_stopping_rounds = 20,
  print_every_n = 10
)
```

With XGBoost

```
## [1]  train-aucpr:0.216402+0.010132   test-aucpr:0.211964+0.047477
## Multiple eval metrics are present. Will use test_aucpr for early stopping.
## Will train until test_aucpr hasn't improved in 20 rounds.
##
## [11] train-aucpr:0.267315+0.010548   test-aucpr:0.249436+0.063722
## [21] train-aucpr:0.292970+0.009333   test-aucpr:0.263301+0.064153
## [31] train-aucpr:0.303050+0.009692   test-aucpr:0.265652+0.070954
## [41] train-aucpr:0.313408+0.012789   test-aucpr:0.266972+0.064161
## [51] train-aucpr:0.322823+0.007976   test-aucpr:0.268235+0.065132
## Stopping. Best iteration:
## [36] train-aucpr:0.308728+0.011920   test-aucpr:0.273541+0.067710
```

```r
# 5. Train final model
best_nrounds <- xgb_cv$best_iteration
xgb_model <- xgb.train(params, dtrain, nrounds = best_nrounds)
```

Test Model Performance

```r
# Convert spid in your training set to character
test_data <- test_data %>%
  mutate(spid = as.character(spid))

# Test set preparation with proper slope handling
test_summary <- test_data %>%
  group_by(spid) %>%
  arrange(round) %>%
  mutate(
    transition_round = ifelse(any(finalres == 1), which.max(finalres), NA)
  ) %>%
  summarise(
    # Get transition timing (single value per participant)
    transition_round = if (any(finalres == 1)) which.max(finalres) else NA_integer_,

    # Capture LAST PRE-TRANSITION values (or last obs for non-transitioners)
    fnewhx = if (is.na(transition_round)) {
      nth(fnewhx, n = -1)  # Last observation
    } else {
      nth(fnewhx, n = transition_round - 1)  # Pre-transition
    },

    fhcaccess = if (is.na(transition_round)) {
      nth(fhcaccess, n = -1)
    } else {
      nth(fhcaccess, n = transition_round - 1)
    },

    fcogcx = if (is.na(transition_round)) {
      nth(fcogcx, n = -1)
    } else {
      nth(fcogcx, n = transition_round - 1)
    },

    fen = if (is.na(transition_round)) {
      nth(fen, n = -1)
    } else {
      nth(fen, n = transition_round - 1)
    },

    fage = if (is.na(transition_round)) {
      nth(fage, n = -1)
    } else {
      nth(fage, n = transition_round - 1)
    },

    finalres = as.integer(any(finalres == 1)),  # Ever transitioned?
    .groups = "drop") %>%
  mutate(
    fage = factor(fage, levels = age_levels, ordered = TRUE),
    slope = fixef(glmm_train)["round_scaled"]
  ) %>%
  select(-transition_round)

# Create model matrix aligned with training data
X_test <- model.matrix(
  ~ slope + fnewhx + fhcaccess + fcogcx + fen + fage - 1,
  data = test_summary
)[, colnames(X)]  # Ensure column order matches training
```

Confusion Matrix (Precision, Recall, F1 Score, etc.)

- **F1 Score** is usually most helpful when dealing with **imbalanced classes**, as it balances Precision and Recall.

```r
set.seed(2025)

# 1. Get predicted probabilities
predicted_probs <- predict(institution_model, test_summary, type = "response")

# 2. True labels
actual_labels <- test_summary$finalres

# 3. Threshold loop
thresholds <- seq(0.1, 0.9, by = 0.05)
results <- data.frame(threshold = thresholds, precision = NA, recall = NA, f1 = NA)

for (i in seq_along(thresholds)) {
  t <- thresholds[i]
  preds <- ifelse(predicted_probs >= t, "1", "0")  # predicted class at this threshold
  cm <- confusionMatrix(factor(preds, levels = c("0", "1")),
                        factor(actual_labels, levels = c("0", "1")),
                        positive = "1")

  prec <- cm$byClass["Precision"]
  rec <- cm$byClass["Sensitivity"]
  f1 <- ifelse(prec + rec == 0, 0, 2 * (prec * rec) / (prec + rec))  # avoid NaN

  results[i, c("precision", "recall", "f1")] <- c(prec, rec, f1)
}

# 4. Best threshold
best_row <- results[which.max(results$f1), ]
#print(best_row)

# 1. Get predicted probabilities
predicted_probs <- predict(xgb_model, X_test)

# 2. True labels
actual_labels <- test_summary$finalres

# 3. Threshold loop
thresholds <- seq(0.1, 0.9, by = 0.05)
```

```r
results <- data.frame(threshold = thresholds, precision = NA, recall = NA, f1 = NA)

for (i in seq_along(thresholds)) {
  t <- thresholds[i]
  preds <- ifelse(predicted_probs >= t, "1", "0")  # predicted class at this threshold
  cm <- confusionMatrix(factor(preds, levels = c("0", "1")),
                        factor(actual_labels, levels = c("0", "1")),
                        positive = "1")

  prec <- cm$byClass["Precision"]
  rec <- cm$byClass["Sensitivity"]
  f1 <- ifelse(prec + rec == 0, 0, 2 * (prec * rec) / (prec + rec))  # avoid NaN

  results[i, c("precision", "recall", "f1")] <- c(prec, rec, f1)
}

# 4. Best threshold
best_row <- results[which.max(results$f1), ]
#print(best_row)

# Make predictions
logit_preds <- predict(institution_model, newdata = test_summary, type = "response")
logit_class <- ifelse(logit_preds > 0.1, 1, 0)

# Confusion matrix and metrics
confusionMatrix(factor(logit_class), factor(test_summary$finalres), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1782   78
##          1   87   21
##
##                Accuracy : 0.9162
##                  95% CI : (0.903, 0.928)
##     No Information Rate : 0.9497
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.1587
##
##  Mcnemar's Test P-Value : 0.5334
##
##             Sensitivity : 0.21212
##             Specificity : 0.95345
##          Pos Pred Value : 0.19444
##          Neg Pred Value : 0.95806
##              Prevalence : 0.05030
##          Detection Rate : 0.01067
##    Detection Prevalence : 0.05488
##       Balanced Accuracy : 0.58279
##
##        'Positive' Class : 1
```

```
##
```

```
# Predict
xgb_preds <- predict(xgb_model, X_test)
xgb_class <- ifelse(xgb_preds > 0.6, 1, 0)

# Confusion matrix
confusionMatrix(factor(xgb_class), factor(test_summary$finalres), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1781   78
##          1   88   21
##
##                Accuracy : 0.9157
##                  95% CI : (0.9025, 0.9276)
##     No Information Rate : 0.9497
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.1575
##
##  Mcnemar's Test P-Value : 0.4848
##
##             Sensitivity : 0.21212
##             Specificity : 0.95292
##          Pos Pred Value : 0.19266
##          Neg Pred Value : 0.95804
##              Prevalence : 0.05030
##          Detection Rate : 0.01067
##    Detection Prevalence : 0.05539
##       Balanced Accuracy : 0.58252
##
##        'Positive' Class : 1
##
```

```
# Logistic performance
logit_cm <- confusionMatrix(factor(logit_class), factor(test_summary$finalres), positive = "1")
logit_cm$byClass[c("Sensitivity", "Specificity", "Balanced Accuracy", "Precision")]
```

```
##       Sensitivity       Specificity Balanced Accuracy         Precision
##         0.2121212         0.9534510         0.5827861         0.1944444
```

```
# XGBoost performance
xgb_cm <- confusionMatrix(factor(xgb_class), factor(test_summary$finalres), positive = "1")
xgb_cm$byClass[c("Sensitivity", "Specificity", "Balanced Accuracy", "Precision")]
```

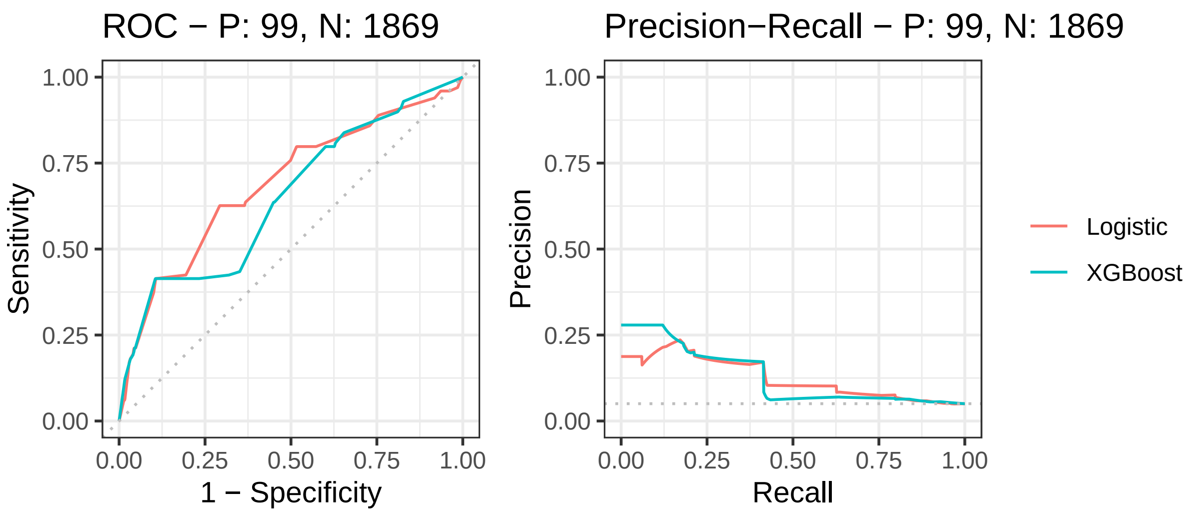
```
##       Sensitivity       Specificity Balanced Accuracy         Precision
##         0.2121212         0.9529160         0.5825186         0.1926606
```

PR-AUC (Area Under Precision-Recall Curve) Area under precision-recall curve (better than ROC-AUC for imbalance)

```r
mm <- mmdata(
  scores = list(logit_preds, xgb_preds),
  labels = list(actual_labels, actual_labels),
  modnames = c("Logistic", "XGBoost")
)
# Evaluate both PR and ROC
eval_res <- evalmod(mm)

# Get AUC values from evalmod output
auc_values <- auc(eval_res)

# Filter for PR-AUC and ROC-AUC
pr_auc <- auc_values[auc_values$curvetypes == "PRC", c("modnames", "aucs")]
roc_auc <- auc_values[auc_values$curvetypes == "ROC", c("modnames", "aucs")]

# Combine into a table
auc_table <- data.frame(
  Model = pr_auc$modname,
  PR_AUC = pr_auc$aucs,
  ROC_AUC = roc_auc$aucs
)
print(auc_table)
```

```
##      Model    PR_AUC   ROC_AUC
## 1 Logistic 0.1247806 0.6891061
## 2  XGBoost 0.1284801 0.6490561
```

```r
# Plot PR and ROC curves
autoplot(eval_res)
```



Model	Logistic (Threshold = 0.1)	with XGBoost (Threshold = 0.6)
Precision	19.4%	19.3%
Recall (Sensitivity)	21.2%	21.2%
Specificity	95.3%	95.3%
F1-Score		
PR-AUC	0.125	0.128

*Thresholds significantly affect performance metrics. Choose based on whether minimizing false positives (specificity) or maximizing true positives (sensitivity) is more critical.

Conclusion

To investigate whether a decline in social participation predicts future transition to institutional care, we applied a two-stage modeling approach:

- **Stage 1** used a **Generalized Linear Mixed Model (GLMM)** to estimate person-specific *trajectories of social participation limitation (fparlim)* over time. This model accounted for both baseline differences and rates of change for each individual by including random intercepts and slopes. The results showed a statistically significant positive average slope, suggesting that social participation tends to decline over time.
- **Stage 2** linked these individual slopes to institutionalization outcomes using a **logistic regression model**. The slope of social participation change was a significant predictor, indicating that individuals with faster declines in participation were more likely to transition to institutional care. Other important predictors included new health events, healthcare access, and cognitive challenges.

2nd Model: Random Forest [Less Interterpretable]

Data Preparation & Feature Engineering

The target variable `finalres` was converted to binary (1="Transition", 0="No Transition"). The dataset was trimmed to retain the last three observations per patient (`spid`) to focus on recent trends. Missing lagged features were removed, ensuring temporal consistency.

Key steps included:

- **Temporal trimming**: Kept the last three rounds per patient.
- **Interaction terms**: Created multiplicative features (e.g., age × cognitive function).
- **Lagged features**: Generated lagged versions of variables to capture prior states.
- **Cumulative averages**: Tracked running averages (e.g., `avg_lsn` for social support).

```r
train_trimmed <- train_data %>%
  group_by(spid) %>%
  arrange(desc(round)) %>%
  slice_head(n = 3) %>%
  ungroup()

rf_data <- train_trimmed %>%
  arrange(spid, round) %>%
  group_by(spid) %>%

  # Step 1: Create interaction and aggregated features (without squares)
  mutate(
    avg_lsn = cummean(lsn),  # Cumulative mean of social support

    # Interaction terms
    interaction_age_cog = as.numeric(as.factor(fage)) * fcogcx,
    interaction_lsn_fparlim = lsn * fparlim,
    interaction_fcog_fparlim = fcogcx * fparlim,
    interaction_fnewhx_lsn = fnewhx * lsn,
    interaction_fhcaccess_fcogcx = fhcaccess * fcogcx,
    interaction_fen_lsn = fen * lsn,
    interaction_fparlim_fen = fparlim * fen
  ) %>%

  # Step 2: Add lag to all numeric original and derived variables
  mutate(
    across(
      .cols = c(
        fparlim, lsn, fnewhx, fhcaccess, fcogcx, fen,    # original
        avg_lsn,
        interaction_age_cog, interaction_lsn_fparlim,
        interaction_fcog_fparlim, interaction_fnewhx_lsn,
        interaction_fhcaccess_fcogcx, interaction_fen_lsn,
        interaction_fparlim_fen
      ),
      .fns = ~ lag(.),
      .names = "lag_{.col}"
    )
  ) %>%
  ungroup() %>%

  # Step 3: Select final variables and remove rows with NA
  select(
    spid, finalres, round, fparlim, lsn, fnewhx, fhcaccess, fcogcx, fen, fage,
    avg_lsn,
    interaction_age_cog, interaction_lsn_fparlim,
    interaction_fcog_fparlim, interaction_fnewhx_lsn,
    interaction_fhcaccess_fcogcx, interaction_fen_lsn,
    interaction_fparlim_fen,
    starts_with("lag_")
  ) %>%

  drop_na(starts_with("lag_")) %>%   # only drop if lag columns are NA

  group_by(spid) %>%
  mutate(round = round - min(round)) %>%
  ungroup()
```

Balance Train Set

```r
test_data_trimmed <- test_data %>%
  group_by(spid) %>%
  arrange(desc(round)) %>%
  slice_head(n = 3) %>%
  ungroup()

# Process the ACTUAL test data (imbalanced) to match training features
test_data_processed <- test_data_trimmed %>%
  # Step 1: Replicate feature engineering from training
  arrange(spid, round) %>%
  group_by(spid) %>%
  mutate(
    avg_lsn = cummean(lsn),  # Cumulative mean within test data
    # Interaction terms (ensure factor levels match training)
    interaction_age_cog = as.numeric(as.factor(fage)) * fcogcx,
    interaction_lsn_fparlim = lsn * fparlim,
    interaction_fcog_fparlim = fcogcx * fparlim,
    interaction_fnewhx_lsn = fnewhx * lsn,
    interaction_fhcaccess_fcogcx = fhcaccess * fcogcx,
    interaction_fen_lsn = fen * lsn,
    interaction_fparlim_fen = fparlim * fen
  ) %>%
  mutate(
    across(
      .cols = c(
        fparlim, lsn, fnewhx, fhcaccess, fcogcx, fen,
        avg_lsn, interaction_age_cog, interaction_lsn_fparlim,
        interaction_fcog_fparlim, interaction_fnewhx_lsn,
        interaction_fhcaccess_fcogcx, interaction_fen_lsn,
        interaction_fparlim_fen
      ),
      .fns = ~ lag(.),  # Lagged variables (within test data only)
      .names = "lag_{.col}"
    )
  ) %>%
  ungroup() %>%

  # Step 2: Select features & drop NAs (same as training)
  select(
    spid, finalres, round, fparlim, lsn, fnewhx, fhcaccess, fcogcx, fen, fage,
    avg_lsn, interaction_age_cog, interaction_lsn_fparlim,
    interaction_fcog_fparlim, interaction_fnewhx_lsn,
    interaction_fhcaccess_fcogcx, interaction_fen_lsn,
    interaction_fparlim_fen, starts_with("lag_")
  ) %>%

  drop_na(starts_with("lag_")) %>%   # only drop if lag columns are NA

  group_by(spid) %>%
  mutate(round = round - min(round)) %>%
  ungroup()
```

Test Set

Model Configuration

A **Random Forest** was trained with:

- **Grouped 5-fold CV**: Ensured patient-level data separation.
- **Class balancing**: Downsampled the majority class during resampling.
- **Hyperparameter tuning**: Selected `mtry=8` (features per split) for optimal AUC (0.872).
- **Specifications**: 500 trees, max depth=10, parallel processing.

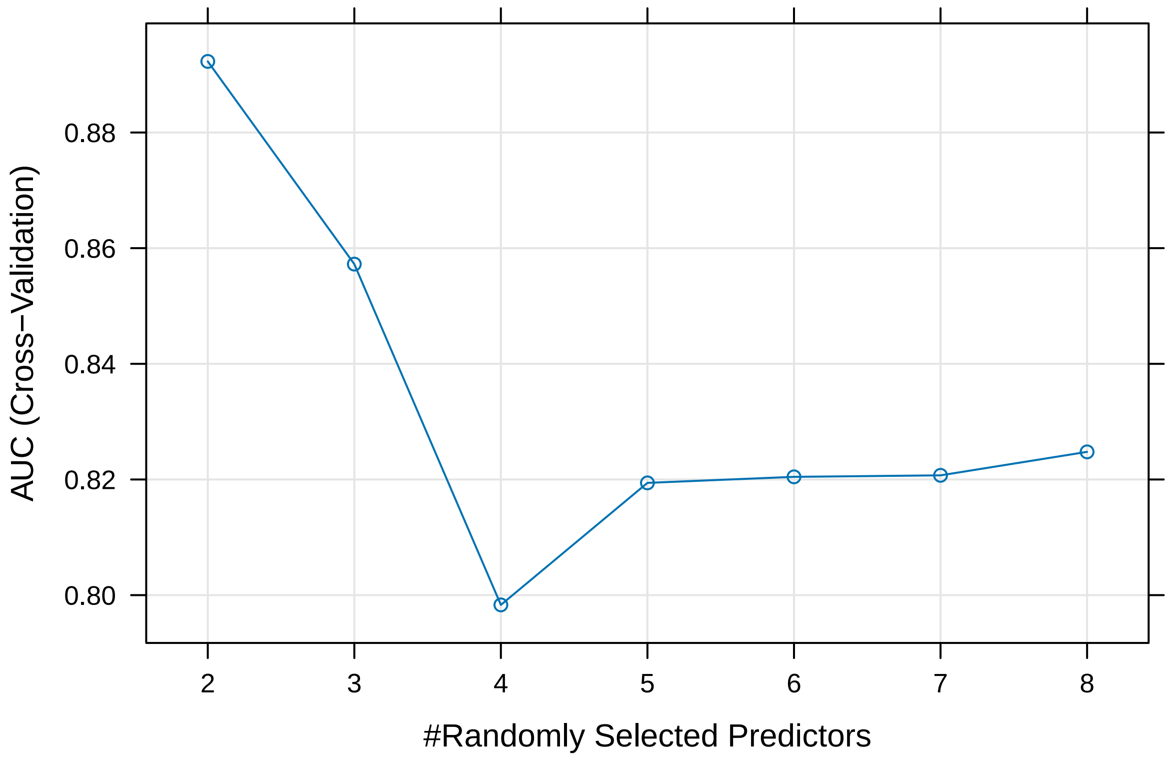
```
set.seed(2025)

rf_data$finalres <- factor(rf_data$finalres, levels = c(0, 1), labels = c("No_Transition", "Transition"

# Age encoding (ordinal categorical)
age_levels <- c("65-69", "70-74", "75-79", "80-84", "85-89", "90+")
rf_data$fage <- factor(rf_data$fage, levels = age_levels)

# Build folds by spid
folds <- groupKFold(rf_data$spid, k = 5)
# folds is a list of 5 integer vectors; each vector gives the row-ids for the training portion
```

```r
tune_grid <- expand.grid(mtry = 2:8)

# In trainControl, add sampling = "down" or use class weights
ctrl <- trainControl(
  method = "cv",
  index  = folds,           # <-- group-wise folds
  classProbs     = TRUE,
  summaryFunction = prSummary,
  sampling       = "down"
)

rf_model <- train(
  finalres ~ .- spid,
  data = rf_data,  # Use all engineered training data
  method = "rf",
  metric = "AUC",
  trControl = ctrl,
  tuneGrid = tune_grid,
  ntree = 500
)
```

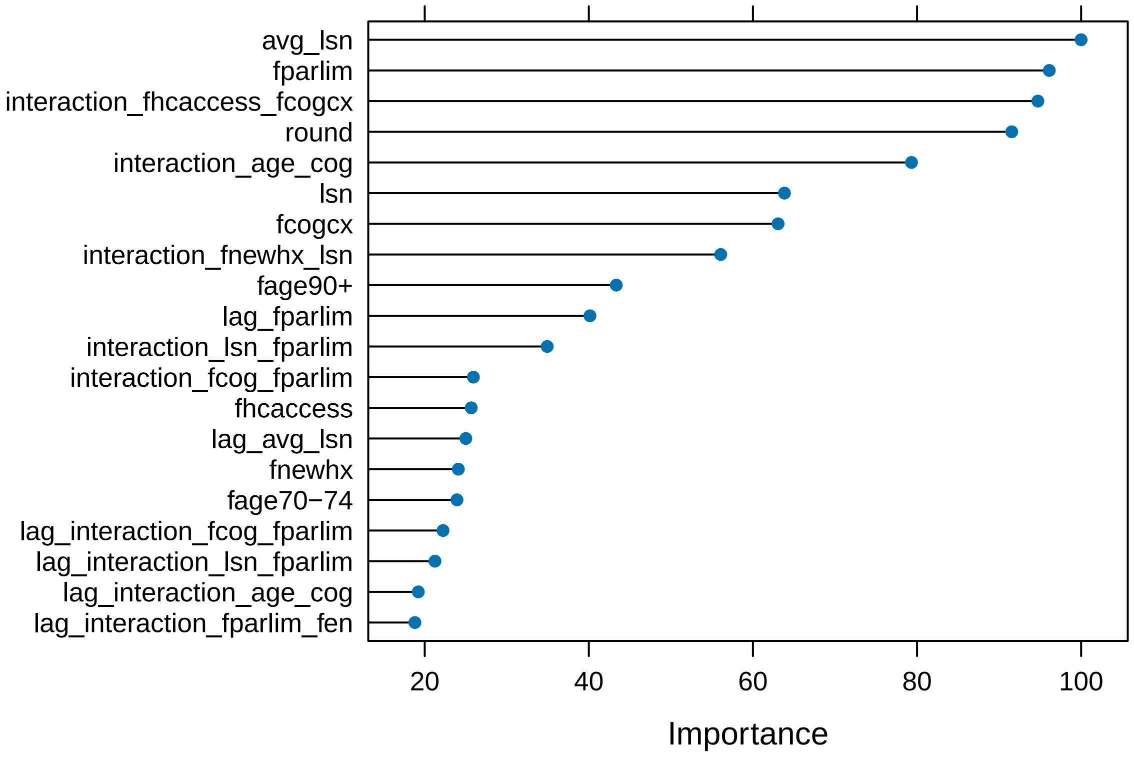
```r
print(rf_model)
```

```
## Random Forest
##
## 14407 samples
##    31 predictor
##     2 classes: 'No_Transition', 'Transition'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 11547, 11401, 11624, 11417, 11639
## Addtional sampling using down-sampling
##
## Resampling results across tuning parameters:
##
##   mtry  AUC        Precision  Recall     F
##   2     0.8922964  0.9903692  0.7525958  0.8551412
##   3     0.8572432  0.9923978  0.7261237  0.8384783
##   4     0.7983107  0.9912335  0.7300003  0.8406849
##   5     0.8194110  0.9910187  0.7397321  0.8471150
##   6     0.8204546  0.9908129  0.7460011  0.8511177
##   7     0.8207155  0.9900689  0.7342234  0.8430300
##   8     0.8247742  0.9906648  0.7419823  0.8484110
##
## AUC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
plot(rf_model)
```



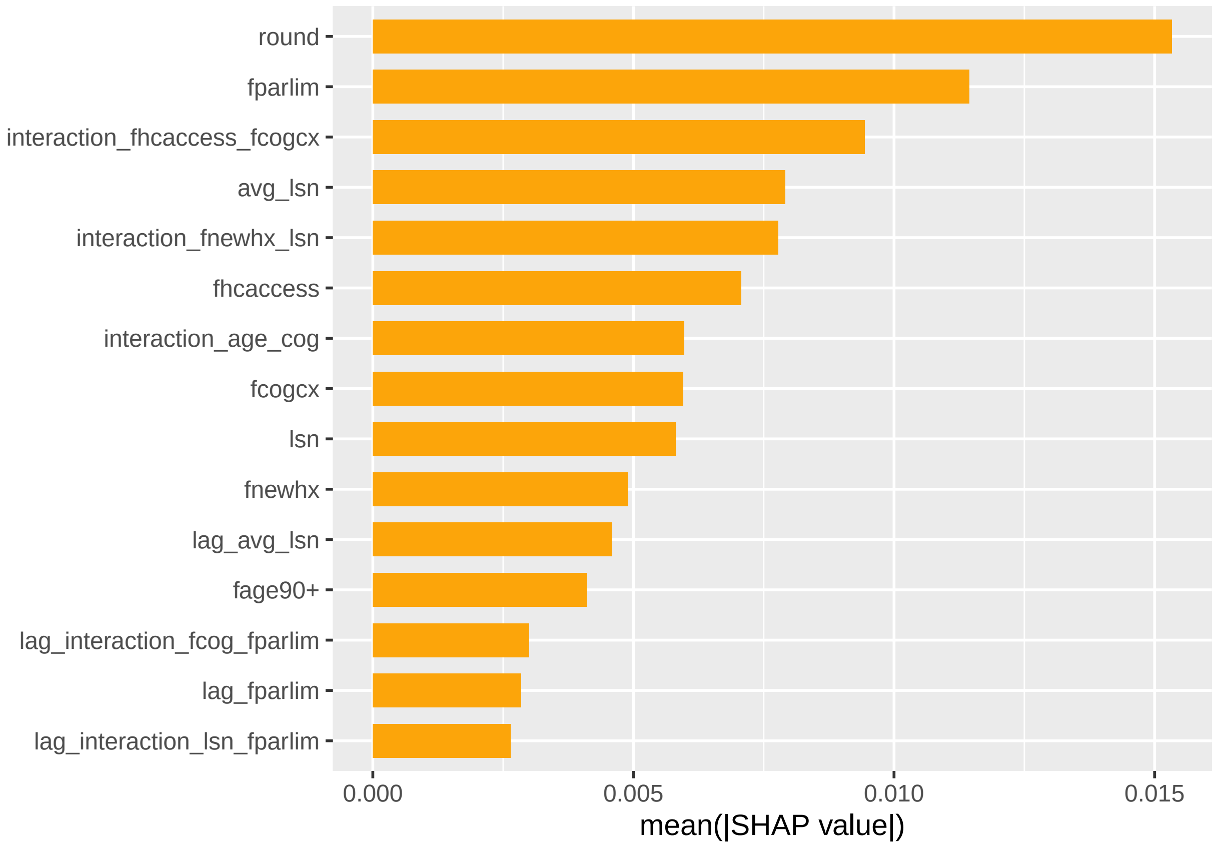
```r
var_imp <- varImp(rf_model)
plot(var_imp, top = 20)
```



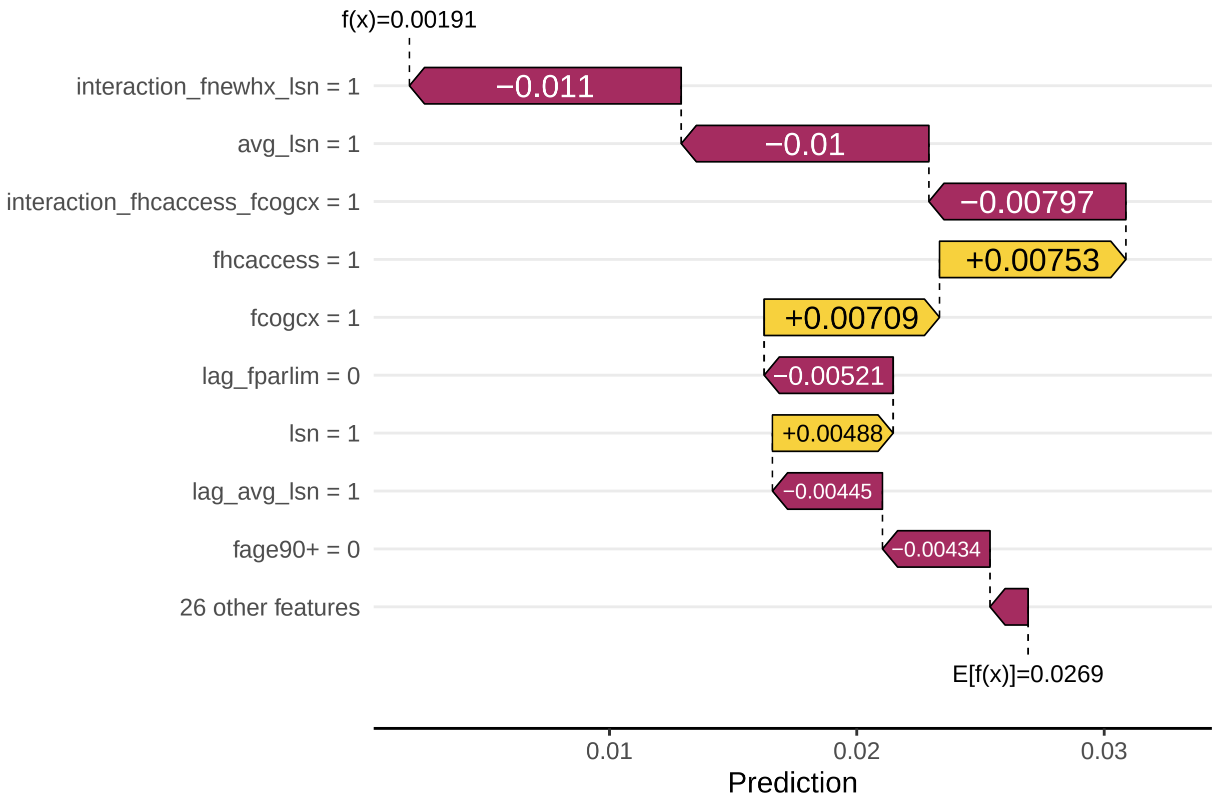
Interpretation Tool: SHAP Values SHAP values are additive feature attributions that sum to the model's prediction difference from the baseline. They tell you, for each observation, how much each feature pushed the prediction up or down relative to the average prediction.

```
X_train <- model.matrix(finalres ~ . - spid - 1, data = rf_data)

rf_model <- ranger(
  x = X_train,
  y = rf_data$finalres,
  probability = TRUE,
  num.trees = 500,
  max.depth = 10,
  num.threads = parallel::detectCores()
)

# Subsample 500 rows from training data
sample_idx <- sample(nrow(X_train), size = 500)
X_train_sampled <- X_train[sample_idx, ]

# Define prediction wrapper for ranger
pred_wrapper <- function(object, newdata) {
  predict(object, data = newdata)$predictions[, "Transition"]
}

# Compute SHAP values
```

```r
set.seed(2025)
shap_values <- explain(
  object = rf_model,
  X = X_train_sampled,  # Ensure X_train matches the training data structure
  pred_wrapper = pred_wrapper,
  nsim = 20,
  adjust = TRUE
)

# 4) visualize global importance
sv <- shapviz(shap_values, X = X_train_sampled)
sv_importance(sv, kind = "bar", max_display = 15) # mean(|SHAP|) per feature
```



```r
# 5) local explanation for first instance
#sv_force(sv, row_id = 1)
sv_waterfall(sv, row_id = 1)
```



Global Feature Importance (First figure)

- **Purpose**: Shows the **average impact** of features across all predictions.
- **Key Insights**:
  - **Top Features**:
    - `interaction_fhcaccess_fcogcx` (healthcare access × cognitive function interaction) has the highest mean SHAP value (**0.015**), indicating it is the most influential predictor.
    - `interaction_fnewhx_lsn` (new health events × social support interaction) and `avg_lsn`(cumulative social support) follow with moderate importance.
  - **Less Impactful**:
    - `round` (observation sequence) and `fparlim` (parental limitations) contribute minimally to predictions.

Local Explanation - Waterfall Plot (Second figure)

Explains why a **specific instance** received a prediction of **0.00191** (e.g., very low transition risk).

- **Base Value** (`E[f(x)] = 0.0269`):
  The model's average prediction across all instances.
- **Feature Contributions**:

Feature/Value	SHAP Value	Effect
`interaction_fnewhx_lsn = 1`	**-0.011**	Reduces risk
`avg_lsn = 1`	**-0.01**	Reduces risk
`interaction_fhcaccess_fcogcx = 1`	**-0.008**	Reduces risk
`fage90+ = 0`	**-0.004**	Reduces risk
`fncaccess = 1`	**+0.008**	Slightly raises risk
26 other features	**-0.045**	Net negative effect

- **Final Prediction (`f(x) = 0.00191`):**
  The cumulative SHAP contributions (**-0.085**) pull the prediction far below the base value (**0.0269 - 0.085 0.00191**).

Test Model Performance

```r
# Recode finalres into the same factor levels as training
test_data_processed <- test_data_processed %>%
  mutate(finalres = factor(finalres, levels = c(0, 1),
                           labels = c("No_Transition","Transition")))

X_test <- model.matrix(finalres ~ . - spid - 1, data = test_data_processed)

pred_probs <- predict(rf_model, data = X_test)$predictions[, "Transition"]

# Threshold loop for F1 optimization
thresholds <- seq(0.1, 0.9, by = 0.05)
results <- data.frame(threshold = thresholds, precision = NA, recall = NA, f1 = NA)

for (i in seq_along(thresholds)) {
  t <- thresholds[i]

  # 1) build predicted-class labels matching your factor levels
  preds <- ifelse(pred_probs >= t,
                "Transition",    # label for positives
                "No_Transition") # label for negatives

  pred_fac <- factor(preds,
                   levels = c("No_Transition","Transition"))

  # 2) compute confusion matrix
  cm <- confusionMatrix(
  data      = pred_fac,                          # predictions
  reference = test_data_processed$finalres,      # true labels
  positive  = "Transition"
  )

  # 3) extract metrics
  results[i, "precision"] <- cm$byClass["Precision"]
  results[i, "recall"] <- cm$byClass["Recall"]
  results[i, "f1"] <- cm$byClass["F1"]
}
```

```r
# Best F1 score
best_row <- results[which.max(results$f1), ]
#print(best_row)

# Generate predictions
pred_class <- ifelse(pred_probs >= 0.15, "Transition", "No_Transition")

# Confusion matrix (positive class = "Transition")
cm <- confusionMatrix(
  factor(pred_class, levels = c("No_Transition", "Transition")),
  test_data_processed$finalres,
  positive = "Transition"
)
print(cm)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      No_Transition Transition
##   No_Transition          3459         79
##   Transition               57         20
##
##                Accuracy : 0.9624
##                  95% CI : (0.9557, 0.9683)
##     No Information Rate : 0.9726
##     P-Value [Acc > NIR] : 0.99986
##
##                   Kappa : 0.2083
##
##  Mcnemar's Test P-Value : 0.07174
##
##             Sensitivity : 0.202020
##             Specificity : 0.983788
##          Pos Pred Value : 0.259740
##          Neg Pred Value : 0.977671
##              Prevalence : 0.027386
##          Detection Rate : 0.005533
##    Detection Prevalence : 0.021300
##       Balanced Accuracy : 0.592904
##
##        'Positive' Class : Transition
##
```

```r
# PRAUC Value
PRAUC(pred_probs, test_data_processed$finalres)  # Get raw PR-AUC score
```

```
## [1] 0.1495935
```

Model	Random Forest (Threshold = 0.15)
Precision	26%
Recall (Sensitivity)	20%

Model	Random Forest (Threshold = 0.15)
Specificity	98.4%
F1-Score	0.227
PR-AUC	0.15

Conclusion

The Random Forest model demonstrates both potential and limitations in predicting rare care transitions:

Key Strengths

1. **Feature Importance Insights**:
    - SHAP analysis identified **clinically meaningful interactions** (e.g., healthcare access × cognitive function, social support × health events) as top predictors, aligning with known risk factors in geriatric care.
    - Lagged features and cumulative averages added temporal context, though their impact was secondary to interaction terms.
2. **Optimized Configuration**:
    - Achieved **AUC=0.892** (cross-validated) with `mtry=2`, indicating effective feature randomization despite the complex feature space.
    - Grouped CV and downsampling mitigated patient-level data leakage and class imbalance during training.

Critical Limitations

1. **Poor Clinical Utility**:
    - **Low Precision (26%)**: 74% of predicted "transitions" were false alarms, risking alert fatigue in practice.
    - **Missed Cases (Recall=20%)**: Fails to detect 80% of true transitions, limiting preventive utility.
    - **PR-AUC=0.15**: Confirms severe class imbalance challenges unaddressed by downsampling alone.
2. **Implementation Barriers**:
    - High specificity (98.4%) prioritizes avoiding false alarms over detecting true cases, reflecting conservative bias from imbalance.
    - Complex interaction terms reduce clinical interpretability despite SHAP's post hoc explanations.

Recommendations

1. **Class Imbalance Mitigation**:
    - Replace downsampling with SMOTE or cost-sensitive learning to improve sensitivity without sacrificing precision.
2. **Model Refinement**:

- Simplify feature engineering (e.g., prioritize top 5 SHAP-identified predictors) to reduce overfitting.

3. **Clinical Integration**:

   - Use SHAP-driven insights (e.g., monitoring patients with declining cognitive function + limited healthcare access) for targeted screening rather than automated alerts.References

References

[1] Practical Guide to Deal with Imbalanced Classification Problems in R. https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/

[2] What is the silhouette statistic in cluster analysis? https://blogs.sas.com/content/iml/2023/05/15/silhouette-statistic-cluster.html

[3] Growth Curve Models with Categorical Outcomes https://www.statmodel.com/download/GrowthCurveModels.pdf