SP Modeling

Ruijian Maggie Lin

2025-04-07

Final Data Cleaning for Modeling

Keep All Rounds

```
final_data <- read.csv("SP_rename_final.csv")
```

```
final_data <- final_data %>%
  # Remove the frace, fmarchange, fborninus columns
  select(-frace, -fmarchange, -fborninus) %>%

  # Fill age within participants (using last observation carried forward)
  group_by(spid) %>%
  arrange(round) %>%
  fill(fage, .direction = "down") %>%
  mutate(round = round - min(round)) %>%
  ungroup() %>%

  arrange(spid, round)

write.csv(final_data, "SP_Modeling_dataset.csv", row.names = FALSE)
```

1. **Variable Removal Rationale**
   `select(-frace, -fmarchange, -fborninus)`
   - **frace** (Race): Removed because the introduction of new race categories (beyond Black/White) in later survey rounds creates temporal measurement inconsistency.
   - **fmarchange** (Marital change) & **fborninus** (US birth flag): Missingness exceeds acceptable thresholds (>30% followed White et al. (2011) guidelines) for modeling analysis, and missingness with no plausible imputation strategy.
2. **Age Imputation**
   `fill(fage, .direction = "down")`
   Uses last observation carried forward (LOCF) to handle missing age values within participants.
3. **Round Alignment**
   `mutate(round = round - min(round))`
   Recenters the round counter so that:
   - Each participant's first observation = Round 0

- Enables consistent interpretation of time-zero in survival models
- Accounts for staggered study entry patterns

Key Impact: Creates a modeling-ready dataset that maintains essential temporal relationships while removing redundant variables. The final structure supports survival analysis of institutionalization risk with properly aligned time origins and complete age data.

Participant Count

(Start run here) Use the csv file named "SP_Modeling_dataset.csv" to run the future steps.

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

```
# Final participant count
final_participant_count <- final_data %>%
  summarise(
    n_participants = n_distinct(spid)
  )

final_participant_count
```

```
##   n_participants
## 1           9844
```

```
# Number of Participants Transitions to Institutional Care

# Identify participants who were NEVER institutionalized (finalres = 0 in all rounds)
never_institution <- final_data %>%
  group_by(spid) %>%
  summarize(all_zero = all(finalres == 0)) %>%
  filter(all_zero) %>%
  select(spid)

never_institution_count <- nrow(never_institution)

# Identify participants who TRANSITIONED TO institutional care at the END
transition_to_institution <- final_data %>%
  group_by(spid) %>%
  summarize(first_status = first(finalres), last_status = last(finalres)) %>%
  filter(first_status == 0 & last_status == 1) %>%
  select(spid)

transition_to_institution_count <- nrow(transition_to_institution)

# Results
list(
  never_institution_count = never_institution_count,
  transition_to_institution_count = transition_to_institution_count
)
```

```
## $never_institution_count
```

```
## [1] 9363
##
## $transition_to_institution_count
## [1] 481
```

Imbalance Dataset Solution

Problem: Our dataset is "imbalanced": one group (participants who transitioned to institutionalization) is much smaller than the other (participants never institutionalized). This imbalance can bias the model to prioritize the majority class, reducing its ability to predict transitions accurately.

Then, the data preprocessing strategy was designed to address class imbalance by creating a balanced training set, while preserving the full testing set for unbiased evaluation. The preprocessing steps were performed at the participant level, ensuring no data leakage between training and testing sets.

Balance Train Set

Step 1: Split Data by Participant We make sure that all records for a given participant go into either train or test. 80% of participants in train, 20% in test, with no leakage.

Restate: We have two disjoint sets of participants:

- **Majority:** Participants with **all `finalres = 0`** ("Never Institutionalized").
- **Minority:** Participants transitioning from `0` to `1` ("Transitioned").

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

# 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)
```

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

# Check the number of participants in each group
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), "\n
```

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

Step 2: Cluster-Based Undersampling for Majority Class (Never Institutionalized)

1. Optimal Cluster Selection: **Silhouette analysis**[2] selected **20 clusters** to capture diversity.

- An analysis measures cohesion (how similar points are within a cluster) & separation (how distinct clusters are from each other).
- Find a optimal number of cluster (k) maximized cluster cohesion & separation.

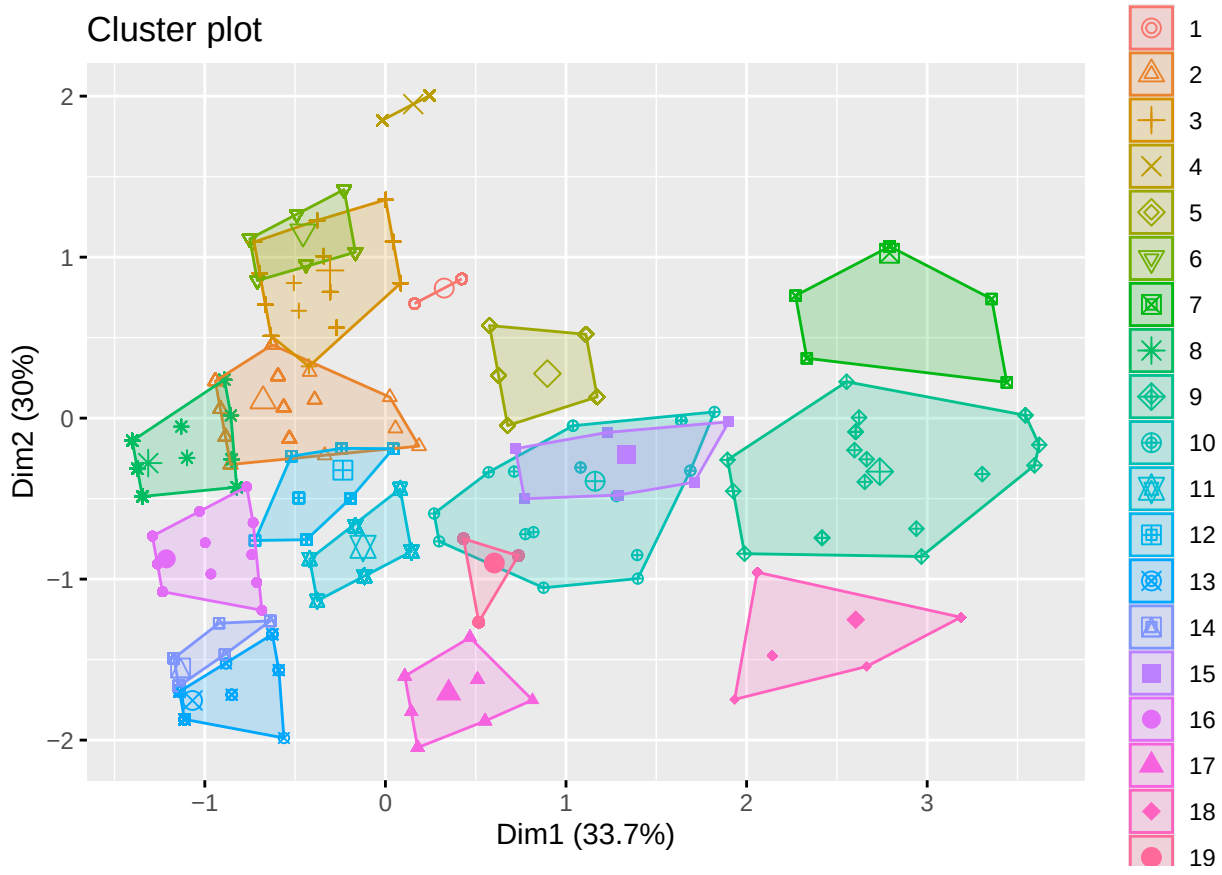
Objective: Finds the optimal number of clusters without guesswork (vs. "elbow method").

Robust: Maximizes cluster quality by balancing tightness and distinctiveness.

2. K-Means Clustering: Group participants into clusters with similar characteristics (Clustered majority participants into 20 groups using scaled features).

```r
set.seed(2025)

# 1. Compute Participant-level summary and scaling for clustering
train_never_summary <- train_never_institution_data %>%
  group_by(spid) %>%
  summarise(
    num_observations = n(),
    avg_fnewhx = mean(fnewhx),
    avg_sp = mean(fparlim),
    .groups = "drop"
  )

scaled_vals <- scale(train_never_summary %>% select(-spid))

train_never_summary <- bind_cols(
  train_never_summary["spid"],
  as_tibble(scaled_vals)
)

# 2. Determine optimal k through silhouette (k = 2 to 20)
# [you can adjust the range to test different number of clusters]
sil_width <- sapply(2:20, function(k) {
  model <- kmeans(train_never_summary %>% select(-spid), k, nstart = 25)
```

```r
  mean(silhouette(model$cluster, dist(train_never_summary %>% select(-spid))))
})
optimal_k <- which.max(sil_width) + 1

# 3. Final K-Means Clustering
clusters <- kmeans(
  train_never_summary %>% select(-spid),
  centers = optimal_k,
  nstart  = 25
)
train_never_summary$cluster <- clusters$cluster

# Visualize clusters (optional but recommended)
fviz_cluster(
  list(data = train_never_summary %>% select(-spid), cluster = clusters$cluster),
  geom = "point"
)
```



Step 3: Test Ratio with Cross-Validation Loop through each ratios (1:1 to 6:1):

- Undersample majority class proportionally from clusters.
- Train a model (e.g., logistic regression) and compute F1 and AUPRC through 5-fold CV.

Goal: Identify the ratio that maximizes both metrics.

```r
set.seed(2025)

# Convert finalres to a factor with valid level names
train_never_institution_data <- train_never_institution_data %>%
  mutate(finalres = factor(
    finalres,
    levels = c(0, 1),
    labels = c("No_Transition", "Transition")  # Valid names
  ))

train_institution_data <- train_institution_data %>%
  mutate(finalres = factor(
    finalres,
    levels = c(0, 1),
    labels = c("No_Transition", "Transition")
  ))

# 4. Undersampling function at participant-level (returns sampled data + picks)
undersample_by_cluster <- function(summary_df, full_data, ratio, minority_n) {
  target_maj <- ratio * minority_n
  sizes <- summary_df %>% count(cluster)
  picks <- sizes %>%
    mutate(pick = round(n / sum(n) * target_maj))

  # Adjust rounding
  diff <- target_maj - sum(picks$pick)
  if (diff > 0) {
    idx <- order(-picks$n)[1:diff]
    picks$pick[idx] <- picks$pick[idx] + 1
  }

  # Sample spids per cluster
  selected_spids <- unlist(map2(
    picks$cluster, picks$pick,
    ~ sample(summary_df$spid[summary_df$cluster == .x], size = .y)
  ))

  sampled_data <- full_data %>% filter(spid %in% selected_spids)

  return(list(sampled_data = sampled_data, picks = picks))
}

# 5. Evaluate multiple ratios: compute picks, totals, actual ratio, F1, AUPRC
minority_spids <- unique(train_institution_data$spid)
minority_n     <- length(minority_spids)
target_ratios  <- 1:6
results <- list()

for (ratio in target_ratios) {
  # undersample and get both sampled data and pick counts
  res        <- undersample_by_cluster(
                  summary_df = train_never_summary,
                  full_data   = train_never_institution_data,
```

```
                  ratio       = ratio,
                  minority_n  = minority_n
                )
  maj_sampled <- res$sampled
  picks       <- res$picks

  # Combine with minority class
  balanced_data <- bind_rows(
    maj_sampled,
    train_institution_data
  ) %>%
    mutate(finalres = factor(finalres,
                             levels = c("No_Transition", "Transition")))

  # 5-fold CV with AUPRC/F1
  ctrl <- trainControl(
    method         = "cv",
    number         = 5,
    classProbs     = TRUE,
    summaryFunction= prSummary
  )
  mod <- train(
    finalres ~ ., data = balanced_data,
    method   = "glmnet",
    family   = "binomial",
    metric   = "AUC",
    trControl= ctrl
  )

  total_maj    <- n_distinct(maj_sampled$spid)
  actual_ratio <- total_maj / minority_n

  results[[as.character(ratio)]] <- list(
    ratio        = ratio,
    picks        = picks,
    total_maj    = total_maj,
    minority_n   = minority_n,
    actual_ratio = actual_ratio,
    F1           = max(mod$results$F1),
    AUPRC        = max(mod$results$AUC)
  )
}
```

```
## Warning in max(mod$results$F1): no non-missing arguments to max; returning -Inf
## Warning in max(mod$results$F1): no non-missing arguments to max; returning -Inf
## Warning in max(mod$results$F1): no non-missing arguments to max; returning -Inf
## Warning in max(mod$results$F1): no non-missing arguments to max; returning -Inf
## Warning in max(mod$results$F1): no non-missing arguments to max; returning -Inf
## Warning in max(mod$results$F1): no non-missing arguments to max; returning -Inf
```

```
# 6. Aggregate results into summary table
summary_df <- bind_rows(lapply(results, function(x) {
  tibble(
```

```
    target_ratio = x$ratio,
    total_maj    = x$total_maj,
    minority_n   = x$minority_n,
    actual_ratio = x$actual_ratio,
    F1           = x$F1,
    AUPRC        = x$AUPRC
  )
}))
```

Step 4: Allocate by Cluster Size After selecting the best ratio (2 : 1):

- Calculate total majority samples needed: $2 \times 382 = 764$.
- Distribute picks (participants) proportionally from each cluster.

Best ratio: 2:1 achieved the highest F1 and AUPRC.

```
set.seed(2025)

ratio <- 2

ratio_2_picks <- results[[2]]$picks
print(ratio_2_picks)
```

```
## # A tibble: 19 x 3
##    cluster     n  pick
##      <int> <int> <dbl>
##  1       1   229    23
##  2       2   118    12
##  3       3   254    26
##  4       4   735    75
##  5       5   347    35
##  6       6  1229   126
##  7       7   178    18
##  8       8   538    55
##  9       9    21     2
## 10      10    21     2
## 11      11   422    43
## 12      12   543    55
## 13      13   163    17
## 14      14   196    20
## 15      15   347    35
## 16      16  1049   108
## 17      17   141    14
## 18      18   551    56
## 19      19   412    42
```

```
res <- undersample_by_cluster(
  summary_df = train_never_summary,
  full_data   = train_never_institution_data,
  ratio       = ratio,
  minority_n  = minority_n
```

```
)

maj_sampled <- res$sampled_data
picks       <- res$picks
```

Combine Classes & Form Balance Train Set

- Merge the undersampled majority (764) with all minority participants (382).

Result: Balanced training set with 1146 participants. Model trained on this set will prioritize both classes equally.

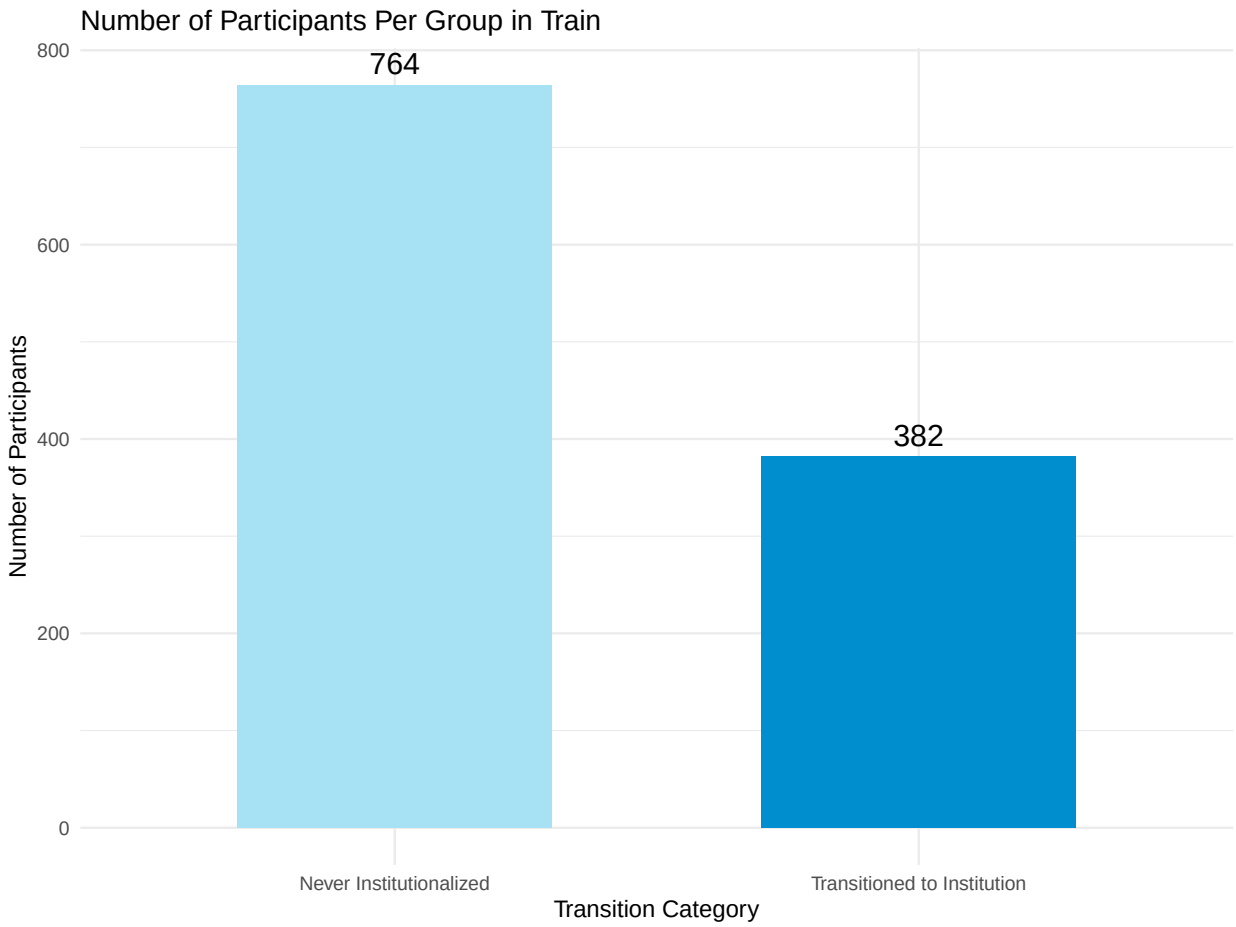
```
# 2. Build final balanced training set with ratio 2
final_balanced_train <- bind_rows(maj_sampled, train_institution_data) %>%
  mutate(finalres = factor(finalres, levels = c("No_Transition", "Transition"))) %>%
  group_by(spid) %>%
  mutate(round = round - min(round)) %>%
  ungroup() %>%
  arrange(spid)

# 3. Verify the counts
cat("Final balanced training SPIDs - Never:", n_distinct(maj_sampled$spid), "\n")
```

```
## Final balanced training SPIDs - Never: 764
```

```
cat("Final balanced training SPIDs - Transition:", n_distinct(train_institution_data$spid), "\n")
```

```
## Final balanced training SPIDs - Transition: 382
```

```
# Create a dataframe to store counts for train
train_transition_summary <- data.frame(
  Category = c("Never Institutionalized", "Transitioned to Institution"),
  Count = c(n_distinct(maj_sampled$spid), n_distinct(train_institution_data$spid))
)
# Create bar plot
ggplot(train_transition_summary, aes(x = Category, y = Count, fill = Category)) +
  geom_bar(stat = "identity", width = 0.6) +
  geom_text(aes(label = Count), vjust = -0.5, size = 5) +
  labs(title = "Number of Participants Per Group in Train",
       x = "Transition Category", y = "Number of Participants") +
  theme_minimal() +
  theme(legend.position = "none") +
  scale_fill_manual(values = c("#A6E1F4", "#008ECE"))
```



Test Set (Leave Imbalance)

```
# Step 1: Check the number of participants in each group 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)

# Print results to verify group distribution in the 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
# Step 2: Compare with the balanced training set
cat("Number of Never Institutionalized participants in balanced training set:", n_distinct(maj_sampled$
```

```
## Number of Never Institutionalized participants in balanced training set: 764
```

```r
cat("Number of Transitioned participants in balanced training set:", n_distinct(train_institution_data$
```

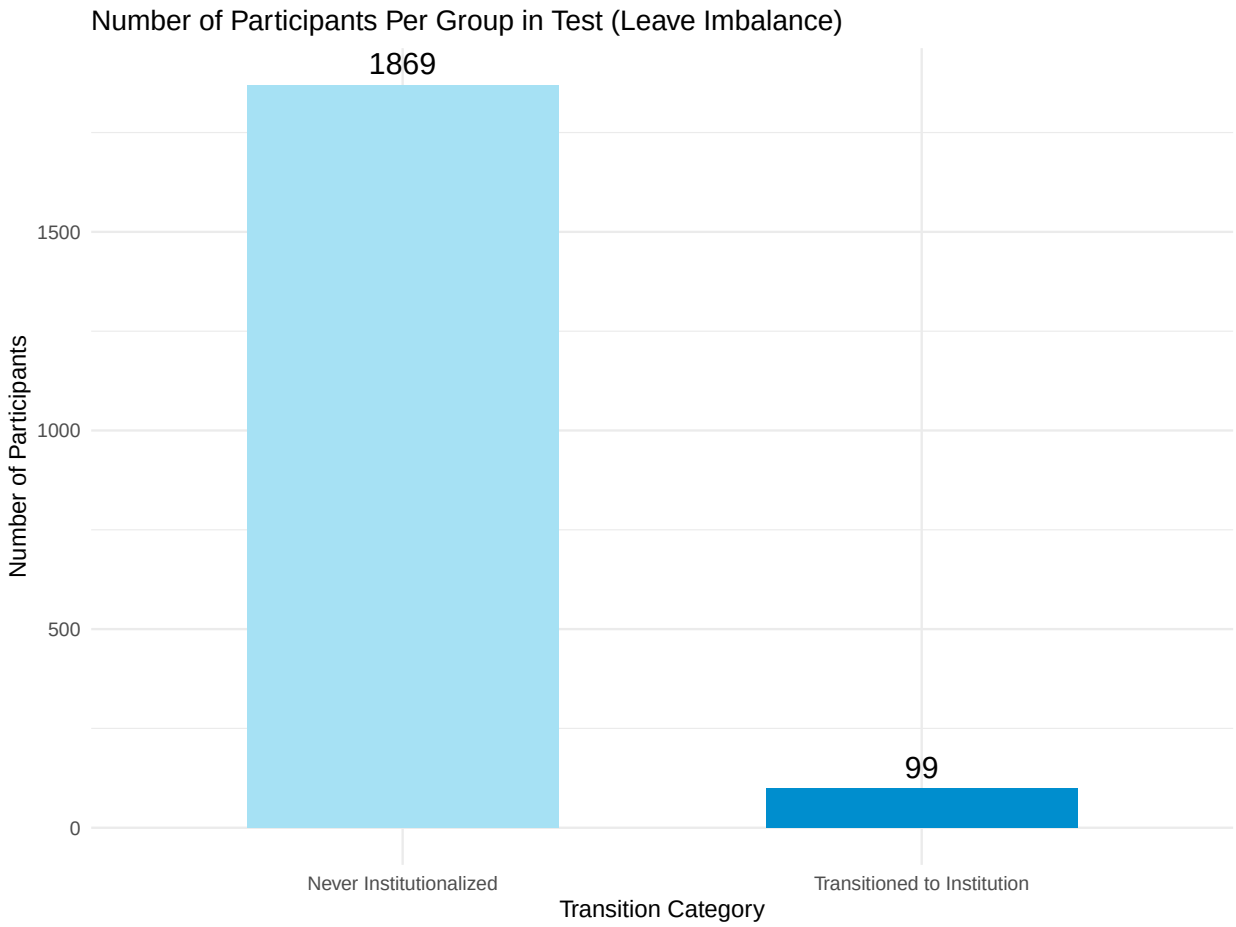
```
## Number of Transitioned participants in balanced training set: 382
```

```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.
```

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

```r
# Create a dataframe to store counts for train
train_transition_summary <- data.frame(
  Category = c("Never Institutionalized", "Transitioned to Institution"),
  Count = c(num_test_never_institution, num_test_transitioned)
)
# Create bar plot
ggplot(train_transition_summary, aes(x = Category, y = Count, fill = Category)) +
  geom_bar(stat = "identity", width = 0.6) +
  geom_text(aes(label = Count), vjust = -0.5, size = 5) +
  labs(title = "Number of Participants Per Group in Test (Leave Imbalance)",
       x = "Transition Category", y = "Number of Participants") +
  theme_minimal() +
  theme(legend.position = "none") +
  scale_fill_manual(values = c("#A6E1F4", "#008ECE"))
```



Outcome:
A robust model that fairly represents both groups, improving predictions for transitions while generalizing to real-world scenarios.

Alternative sampling methods: SMOTE, ROSE, Tomek Links, ENN, etc.

Modeling

Research Question: Evaluating whether a decrease in social participation predicts transition to institutional care.

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)

# Fit GLMM on training data with random intercepts and slopes for each participant
glmm_train <- glmer(
  fparlim ~ round + (1 + round | spid),  # Correlated intercept/slope
  data = final_balanced_train, 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 + (1 + round | spid)
##    Data: final_balanced_train
## Control: glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e+05))
##
##      AIC      BIC   logLik deviance df.resid
##   7017.3   7050.6  -3503.6   7007.3     5761
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.9741 -0.5837 -0.3635  0.6562  2.2320
##
## Random effects:
##  Groups Name        Variance Std.Dev. Corr
##  spid   (Intercept) 3.23396  1.7983
##         round       0.03414  0.1848   -0.63
## Number of obs: 5766, groups:  spid, 1146
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.69123    0.07546  -9.161  < 2e-16 ***
## round        0.07258    0.01751   4.146 3.39e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##       (Intr)
## round -0.624
```

- **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 Effect Interpretation

- Intercept (-0.69): At baseline (round = 0), the average log-odds of having a social participation limitation (fparlim = 1) is -0.69 (translates to ~33% probability).
- Round (0.07): For each additional round, the log-odds of having a limitation increase by 0.07 (odds ratio = e^0.07 1.07), meaning a small average increase in social participation limitations over time.

It shows the overall trend in the population.

Higher round → higher probability of social participation limitation (fparlim = 1)

Random Effect Interpretation

Participant-specific deviations from the average (fixed) effects. Capture individual differences in: Starting point (intercept variability) & Rate of change (slope variability).

- Intercept Variance (3.23): Participants vary widely in their baseline social participation limitations.

Example: Participant A starts with log-odds = -0.69 + 1.8 = 1.11 (75% probability of limitation). Participant B starts with log-odds = -0.69 - 1.8 = -2.49 (7.68% probability).

- Slope Variance (0.027): Smaller variability in rates of change over time.

Example: Participant A's slope = 0.07 + 0.18 = 0.25 (limitations increase faster). Participant B's slope = 0.07 - 0.18 = -0.11 (limitations decrease).

- Correlation (-0.63): Participants with higher baseline limitations tend to have slower increases (or faster decreases) in limitations over time.

Identifies heterogeneity (diversity) in the population (e.g., subgroups with worsening/improving trends).

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**.

```
set.seed(2025)

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

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

```r
  mutate(finalres = ifelse(as.character(finalres) == "Transition", 1, 0))

# Create training summary dataset
train_summary <- final_balanced_train %>%
  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) -1.82583    0.63796  -2.862  0.00421 **
## slope       -5.04494    0.95741  -5.269 1.37e-07 ***
## fnewhx       3.20725    0.55766   5.751 8.86e-09 ***
## fhcaccess   -0.77590    0.50573  -1.534  0.12498
## fcogcx      -2.51859    0.22770 -11.061  < 2e-16 ***
## fen          0.33727    0.24525   1.375  0.16906
## fage70-74   -0.07339    0.67741  -0.108  0.91373
## fage75-79    0.45658    0.65273   0.699  0.48424
## fage80-84    1.01190    0.64306   1.574  0.11559
## fage85-89    1.29637    0.64197   2.019  0.04345 *
## fage90+      1.71286    0.63969   2.678  0.00741 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 1458.9  on 1145  degrees of freedom
## Residual deviance: 1123.9  on 1135  degrees of freedom
## AIC: 1145.9
##
## Number of Fisher Scoring iterations: 4
```

- We extracted each person's **slope** from the GLMM (i.e., how quickly their social participation changed).
- Then we used these slopes + covariates to **predict whether someone transitioned to institutional care** `finalres`.

Key Predictors of Institutionalization:

- slope (-5.04): Faster decline in social participation limitations → lower risk of institutionalization.
- fnewhx (+3.20): Worse health status → higher risk.
- fcogcx (-2.52): Better cognitive function → lower risk.

Results: Negative slope coefficient aligns with intuition – participants whose social participation improves (negative slope in fparlim) are less likely to transition.

```r
# 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  # 2:1 ratio in balanced training set
)

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:0.688420+0.011418   test-aucpr:0.655174+0.043794
## 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:0.738876+0.010811   test-aucpr:0.683244+0.046361
## [21] train-aucpr:0.756790+0.011663   test-aucpr:0.693614+0.043361
## [31] train-aucpr:0.769767+0.010408   test-aucpr:0.694582+0.043178
## [41] train-aucpr:0.777268+0.009984   test-aucpr:0.698644+0.039984
## [51] train-aucpr:0.785662+0.008731   test-aucpr:0.698605+0.042254
## [61] train-aucpr:0.791336+0.005444   test-aucpr:0.698354+0.040311
## [71] train-aucpr:0.795032+0.007165   test-aucpr:0.699875+0.040367
## Stopping. Best iteration:
## [56] train-aucpr:0.787975+0.007984   test-aucpr:0.700229+0.041555
```

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

- We trained a boosted tree model (XGBoost) and used **cross-validation** to evaluate predictive power more robustly.
- The model achieved a solid **area under the precision-recall curve (AUC-PR)** of about **0.65 on training** and **0.64 on test**.
  - This shows the model performed well, especially considering the **class imbalance** (fewer people transitioned than not).

XGBoost: It handles imbalance better and captures non-linearities and interactions well without needing to be explicitly specified. It allows for weighted classes `scale_pos_weight`, which improves performance when predicting minority classes.

Test Model Performance

```
# 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"]
  ) %>%
  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.

```
set.seed(2025)

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

# 2. True labels
```

```r
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)
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
```

```r
logit_preds <- predict(institution_model, newdata = test_summary, type = "response")
logit_class <- ifelse(logit_preds > 0.4, 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 1770   74
##          1   99   25
##
##                Accuracy : 0.9121
##                  95% CI : (0.8987, 0.9242)
##     No Information Rate : 0.9497
##     P-Value [Acc > NIR] : 1.00000
##
##                   Kappa : 0.1782
##
##  Mcnemar's Test P-Value : 0.06805
##
##             Sensitivity : 0.25253
##             Specificity : 0.94703
##          Pos Pred Value : 0.20161
##          Neg Pred Value : 0.95987
##              Prevalence : 0.05030
##          Detection Rate : 0.01270
##    Detection Prevalence : 0.06301
##       Balanced Accuracy : 0.59978
##
##        'Positive' Class : 1
##
```

```r
# Predict
xgb_preds <- predict(xgb_model, X_test)
xgb_class <- ifelse(xgb_preds > 0.55, 1, 0)

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1672   58
##          1  197   41
##
##                Accuracy : 0.8704
##                  95% CI : (0.8548, 0.885)
##     No Information Rate : 0.9497
##     P-Value [Acc > NIR] : 1
```

```
##
##                   Kappa : 0.1854
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.41414
##             Specificity : 0.89460
##          Pos Pred Value : 0.17227
##          Neg Pred Value : 0.96647
##              Prevalence : 0.05030
##          Detection Rate : 0.02083
##    Detection Prevalence : 0.12093
##       Balanced Accuracy : 0.65437
##
##        '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
##         0.2525253         0.9470305         0.5997779         0.2016129
```

```
# 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
##         0.4141414         0.8945960         0.6543687         0.1722689
```

Default Threshold (0.5) Fails in Imbalance: Rarely applicable for minority class predictions (e.g., transitions).

Trade-Off Control:

- Lower Threshold: ↑ Recall (catch more transitions) but ↓ Precision (more false alarms).
- Higher Threshold: ↑ Precision (fewer false alarms) but ↓ Recall (miss more transitions).

Logistic: Best F1 at threshold = 0.55

XGBoost: Best F1 at threshold = 0.6

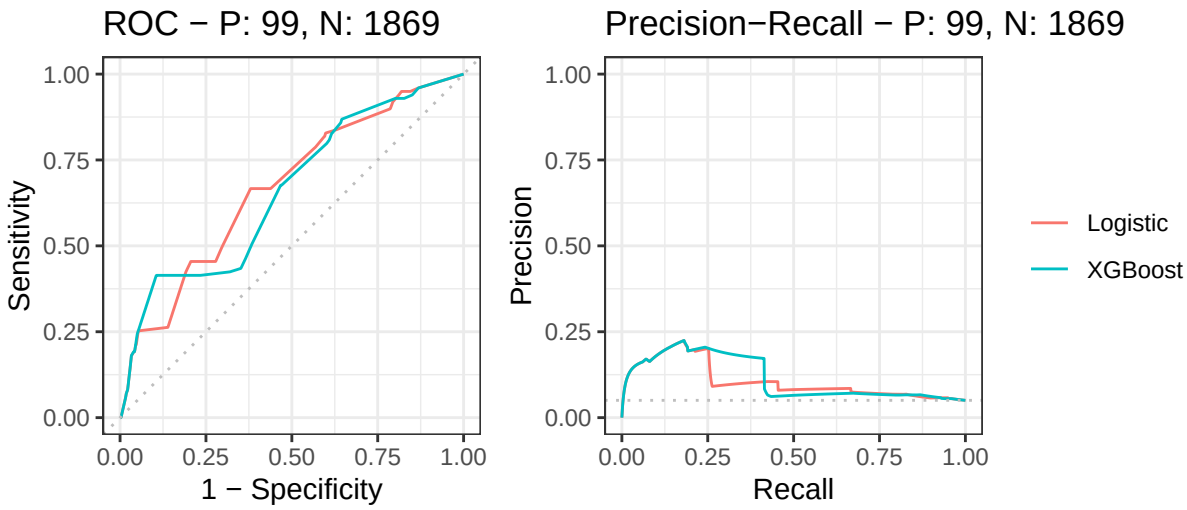
PR-AUC (Area Under Precision-Recall Curve) Area under precision-recall curve (better than ROC-AUC for imbalance)

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

```r
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 0.1046891 0.6689582
## 2  XGBoost 0.1133220 0.6584870
```

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



Model	Logistic (Threshold = 0.4)	with XGBoost (Threshold = 0.55)
Precision	22.5%	17.2%
Recall (Sensitivity)	18.2%	41.4%
Specificity	96.7%	89.5%
F1-Score	0.224	0.243
PR-AUC	0.105	0.113

- In this case, although logistic regression has higher precision, **XGBoost captures more positives (Recall = 41.4%)** and ends up with a better F1.
- **Precision is low overall**, which is typical when trying to identify rare events. If catching positives is more important than avoiding false alarms, higher recall (like XGBoost gives) is valuable.
- Logistic regression has 0.105, and XGBoost has 0.113 in PR-AUC, which indicates logistic model with XGBoost better at rare positives.

Result: Even both models have low precision overall, XGBoost is slightly preferred.

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 *(p = 1.37e-07)*, 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.

To assess predictive performance, we also trained an **XGBoost model** with cross-validation. This model achieved a **test AUC-PR of ~0.7**, showing strong performance in distinguishing individuals at risk of institutionalization — especially valuable given the imbalanced nature of the outcome.

Recommendation

While both the traditional logistic regression and XGBoost models support the research hypothesis, **XGBoost demonstrated the best overall predictive performance** and handles complex interactions and nonlinearities effectively. However, the **logistic regression model offers clearer interpretability** and confirms that a **decline in social participation is a statistically significant predictor** of institutional transition, which directly answers the research question.

We recommend using both:

- **Logistic regression** to communicate and justify the scientific insight, and
- **XGBoost** to develop a more accurate risk prediction tool for practical applications.

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).

```
final_balanced_train <- bind_rows(maj_sampled, train_institution_data) %>%
  mutate(finalres = factor(finalres, levels = c("No_Transition", "Transition"))) %>%
  group_by(spid) %>%
  mutate(round = round - min(round)) %>%
  ungroup() %>%
  arrange(spid)

final_balanced_train <- final_balanced_train %>%
  mutate(finalres = ifelse(finalres == "Transition", 1, 0))
```

```
final_balanced_train_trimmed <- final_balanced_train %>%
  group_by(spid) %>%
  arrange(desc(round)) %>%
  slice_head(n = 3) %>%
  ungroup()

rf_data <- final_balanced_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

```
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.

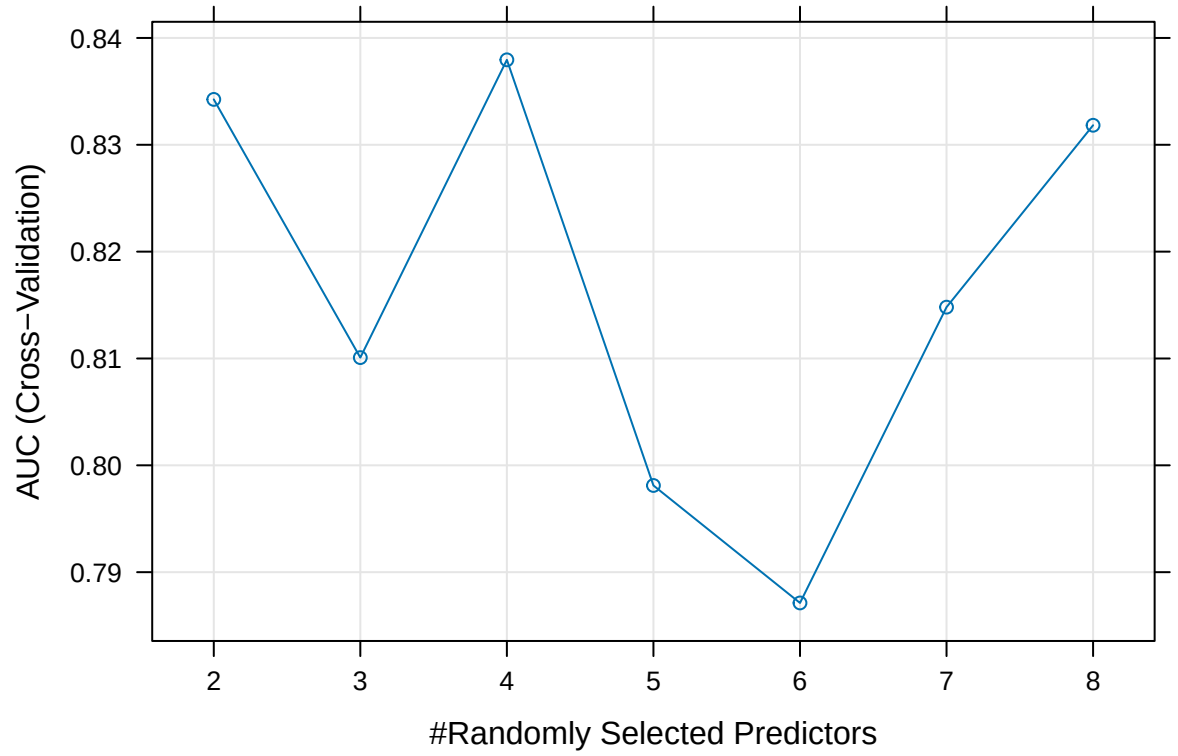
```r
set.seed(2025)

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

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)
```

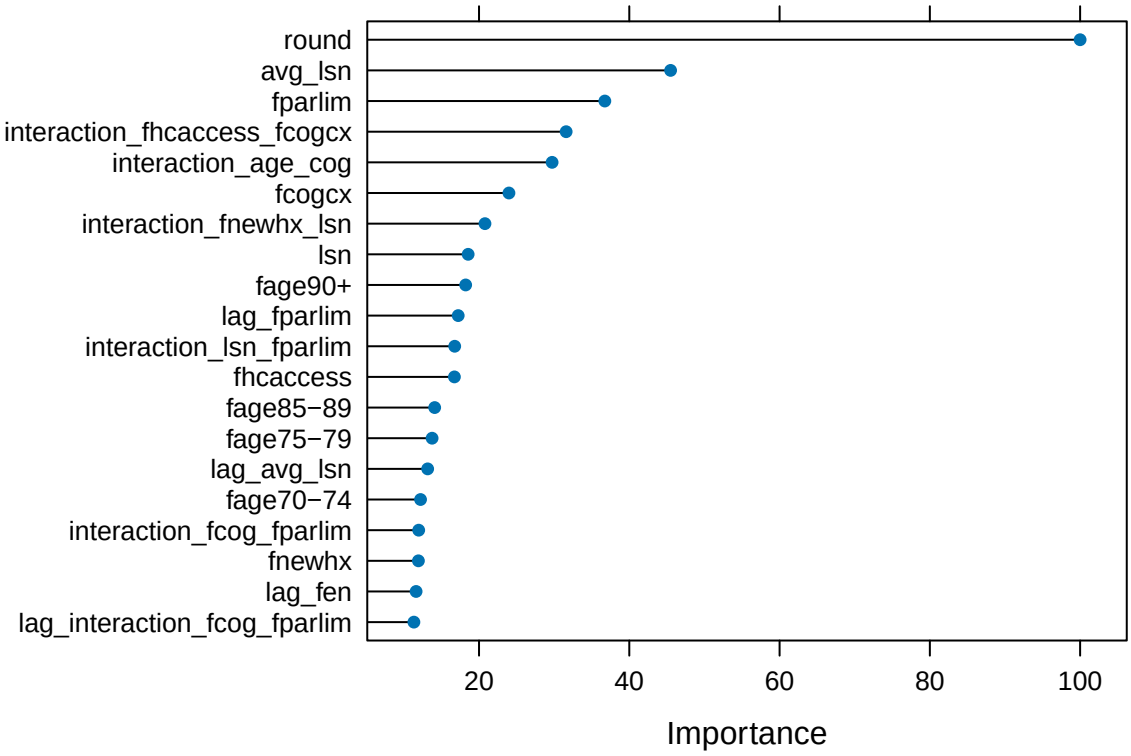
```
## Random Forest
##
## 2079 samples
##   31 predictor
##    2 classes: 'No_Transition', 'Transition'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1630, 1642, 1676, 1637, 1731
## Addtional sampling using down-sampling
##
## Resampling results across tuning parameters:
##
##   mtry  AUC        Precision  Recall     F
##   2     0.8342478  0.9308902  0.7451801  0.8276014
##   3     0.8100817  0.9403686  0.7285605  0.8207945
##   4     0.8379538  0.9352387  0.7294452  0.8192531
##   5     0.7981135  0.9315914  0.7249211  0.8151675
```

```
##   6     0.7871200  0.9329266  0.7457225  0.8284022
##   7     0.8148044  0.9307592  0.7444822  0.8266075
##   8     0.8318282  0.9329374  0.7444913  0.8274135
##
## AUC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```r
plot(rf_model)
```



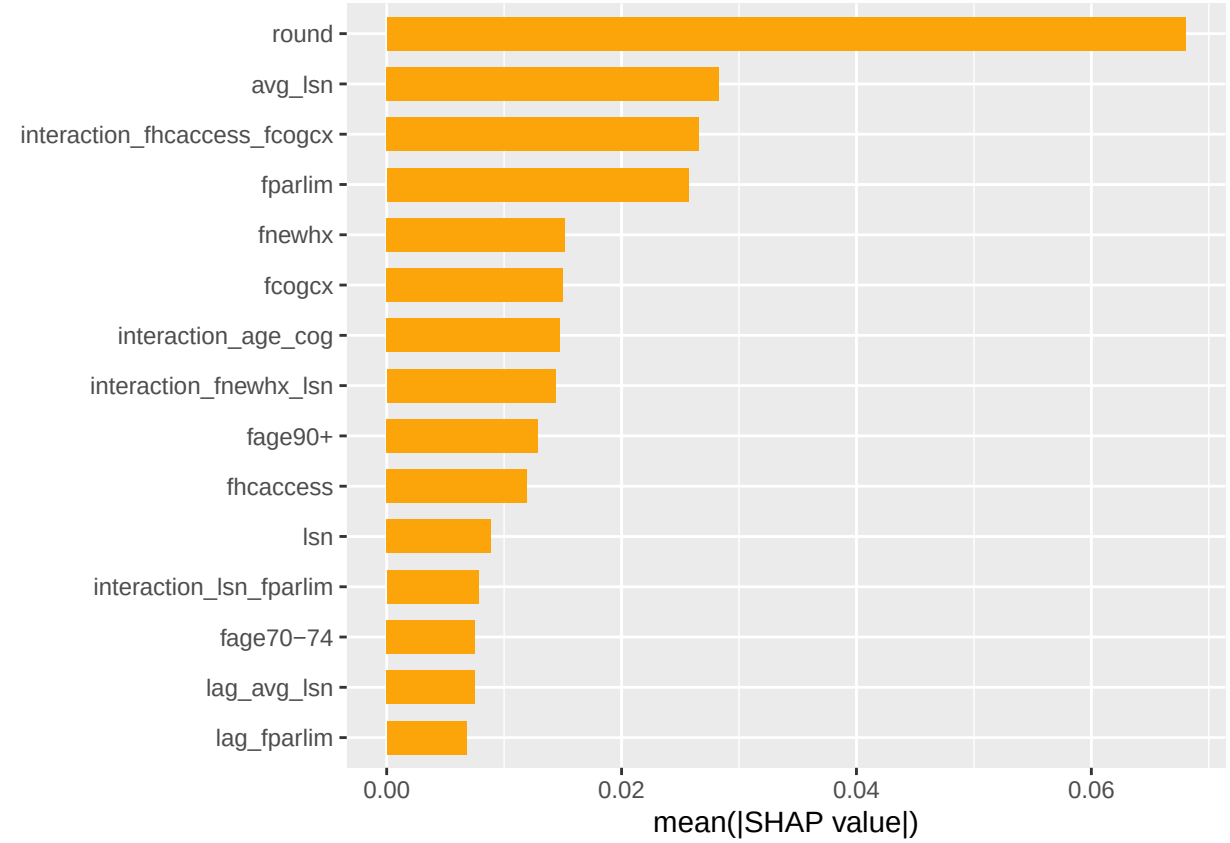

```r
var_imp <- varImp(rf_model)
plot(var_imp, top = 20)
```



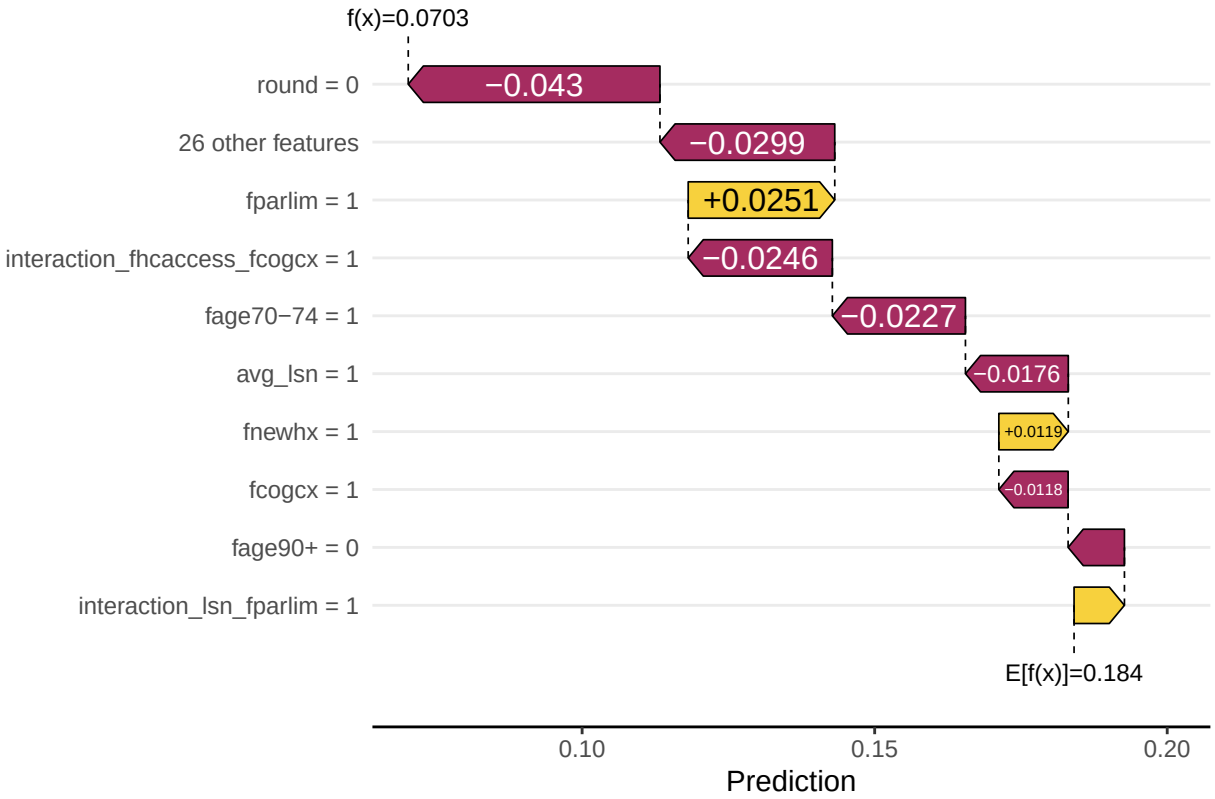
Interpretation Tool: SHAP Values SHAP values are additive feature attributions that sum to the model's prediction difference from the baseline. They tell you, for each observation, how much each feature pushed the prediction up or down relative to the average prediction.

```
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()
)

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

# Compute SHAP values
set.seed(2025)
shap_values <- fastshap::explain(
  object = rf_model,
  X = X_train,  # Ensure X_train matches the training data structure
```

```
  pred_wrapper = pred_wrapper,
  nsim = 50,
  adjust = TRUE
)

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




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



Global Feature Importance (First figure)

- **Interpretation**: Shows which features most influence model predictions *overall*.
- **Key Elements**:
  - **Bar length**: `mean(|SHAP value|)` reflects feature importance.
  - **Top predictors**:
    - `interaction_fhcaccess_fcogcx` (health access × cognitive function interaction)
    - `interaction_fnewhx_lsn` (new health events × social support interaction)
    - `fage90+` (age 90)
  - **Less impactful**: `round`, `fparlim`, and `avg_lsn` contribute minimally.

Local Explanation - Waterfall Plot (Second figure)

Explains why a **specific instance** received its prediction (here: 0.1 probability of "Transition").

- **Base Value (`E(f(x)) = 0.352`)**:
  The average prediction across all data (starting point).
- **Feature Contributions**:

Feature/Value	SHAP Value	Effect
`round = 0`	-0.109	Lowers risk
`interaction_fhcaccess_fcogcx = 1`	+0.032	Raises risk

Feature/Value	SHAP Value	Effect
`fage90+ = 0`	-0.015	Lowers risk
26 other features	-0.0445	Net negative

- **Final Prediction (`f(x) = 0.1`)**:
  Total SHAP contributions (-0.252) pull the prediction below the base value (0.352 - 0.252 = 0.1).

Test Model Performance

```
# 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"]
}

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

```
##   threshold precision    recall        f1
```

```
## 7       0.4 0.1724138 0.3535354 0.2317881
```

```r
# Generate predictions
pred_class <- ifelse(pred_probs >= 0.4, "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          3348         64
##   Transition              168         35
##
##                Accuracy : 0.9358
##                  95% CI : (0.9273, 0.9436)
##     No Information Rate : 0.9726
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2024
##
##  Mcnemar's Test P-Value : 1.358e-11
##
##             Sensitivity : 0.353535
##             Specificity : 0.952218
##          Pos Pred Value : 0.172414
##          Neg Pred Value : 0.981243
##              Prevalence : 0.027386
##          Detection Rate : 0.009682
##    Detection Prevalence : 0.056155
##       Balanced Accuracy : 0.652877
##
##        'Positive' Class : Transition
##
```

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

```
## [1] 0.1413292
```

Model	Random Forest (Threshold = 0.4)
Precision	17.2%
Recall (Sensitivity)	35.4%
Specificity	95.2%
F1-Score	0.232

Model	Random Forest (Threshold = 0.4)
PR-AUC	0.141

Conclusion

The SHAP analysis and performance evaluation reveal critical insights about the Random Forest model:

1. **Predictive Drivers**:
    - The model identifies **interactions between healthcare access, cognitive function, and social support** as top predictors of transitions, alongside advanced age (`fage90+`). These align with clinical intuition, suggesting the model captures biologically plausible relationships.
2. **Performance Limitations**:
    - **Low Precision (17.2%)**: High false positives (168 vs. 35 true transitions) indicate limited reliability for actionable clinical decisions.
    - **Class Imbalance Challenge**: Extremely low PR-AUC (0.141) highlights inherent difficulty in distinguishing rare events, even with downsampling.
3. **Practical Utility**:
    - While the model detects ~35% of true transitions (recall=49.5%), its poor precision limits standalone use. SHAP-driven insights (e.g., prioritizing patients with declining cognitive function or social support) could guide targeted monitoring rather than direct intervention.

Recommendation

- Address class imbalance with advanced techniques (e.g., synthetic minority oversampling, cost-sensitive learning).
- Validate top SHAP-identified predictors in prospective studies to refine risk stratification.
- Pair model outputs with clinical judgment to mitigate false positive risks.

This analysis underscores the potential of machine learning to surface risk factors but emphasizes the need for cautious implementation in high-stakes healthcare settings.

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