SPI 586a: Refined Final Proposal (Updated version of proposal submitted in Week 7)

Fatima Mumtaz

2024-05-05

Install and load relevant packages

```
# install.packages("haven")

library(haven)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(ggplot2)
library(tidymodels)
```

```
## ── Attaching packages ─────────────────────────────────────── tidymodels 1.1.1 ──
```

```
## ✔ broom        1.0.5     ✔ rsample      1.2.1
## ✔ dials        1.2.0     ✔ tibble       3.2.1
## ✔ infer        1.0.6     ✔ tidyr        1.3.1
## ✔ modeldata    1.3.0     ✔ tune         1.1.2
## ✔ parsnip      1.1.1     ✔ workflows    1.1.3
## ✔ purrr        1.0.2     ✔ workflowsets 1.0.1
## ✔ recipes      1.0.9     ✔ yardstick    1.3.0
```

```
## ── Conflicts ────────────────────────────────────────── tidymodels_conflicts() ──
## ✖ purrr::discard()         masks scales::discard()
## ✖ tidyr::expand()          masks Matrix::expand()
## ✖ dplyr::filter()          masks stats::filter()
## ✖ dplyr::lag()             masks stats::lag()
## ✖ purrr::lift()            masks caret::lift()
## ✖ tidyr::pack()            masks Matrix::pack()
## ✖ yardstick::precision()   masks caret::precision()
## ✖ yardstick::recall()      masks caret::recall()
## ✖ yardstick::sensitivity() masks caret::sensitivity()
## ✖ yardstick::specificity() masks caret::specificity()
## ✖ recipes::step()          masks stats::step()
## ✖ tidyr::unpack()          masks Matrix::unpack()
## ✖ recipes::update()        masks Matrix::update(), stats::update()
## • Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ readr     2.1.5
## ✔ lubridate 1.9.3     ✔ stringr   1.5.1
```

```
## ── Conflicts ──────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ readr::col_factor() masks scales::col_factor()
## ✖ purrr::discard()    masks scales::discard()
## ✖ tidyr::expand()     masks Matrix::expand()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ stringr::fixed()    masks recipes::fixed()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ purrr::lift()       masks caret::lift()
## ✖ tidyr::pack()       masks Matrix::pack()
## ✖ readr::spec()       masks yardstick::spec()
## ✖ tidyr::unpack()     masks Matrix::unpack()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflic
ts to become errors
```

Reading the dataset

```
### Read the .dta file
nfhs5_data <- read_dta("IAKR7EFL.DTA")

### Data processing

## Relevant variables in the dataset, to be extracted for analysis
# anemia_level_child = hw57
# size_of_child_at_birth = m18
# sex_of_child = b4
# caste = s116
# state = v101
# residence_type = v102
# religion = v130
# wealth_index = v190
# anemia_level_mother = v457
# educ_level_mother = v107
# smoke_cigarette_mother = v463a
# total_children_ever_born_to_mother = v201
# chew_tobacco_mother = v463c
# number_of_living_siblings = v218


## Make subset of data with only relevant variables
relevant_variables <- c("hw57", "m18", "s234", "b4", "s116", "v101", "v102",
                        "v130", "v190", "v457", "v107",
                        "v463a", "v201", "v463c", "v218", "v113", "v116")
nfhs5_subset <- nfhs5_data %>%
  select(all_of(relevant_variables))

## Counting no of rows and columns
# The dataset has 232920 rows, each row representing data for a child
nrow(nfhs5_subset)
```

```
## [1] 232920
```

```
# The dataset has 17 variables that have been extracted from the origial dataset, out
of which 3 variables will be used to calculate the predicted variable (or child hunge
r labels) and the remaining 12 will be used as input features to the ML model
ncol(nfhs5_subset)
```

```
## [1] 17
```

Data processing

```r
# Create a copy of the subset that will comprise of cleaned data to be used for ML mo
dels
nfhs5_subset_clean <- nfhs5_subset

# Variables "anemia level of child / hw57", "size_of_child_at_birth / m18" and "Pregn
ancy end in miscarriage, abortion, or still birth / S234" will be used to create labe
l (predicted variable) for child hunger.
# The rows with values "missing", "not applicable" and "don't know" for variables "hw
57" and "m18" were removed
nfhs5_subset_clean <- subset(nfhs5_subset_clean, !(hw57 %in% c(9, NA)))
nfhs5_subset_clean <- subset(nfhs5_subset_clean, !(m18 %in% c(8, 9, NA)))
# "s234" was recoded to indicate if the child died during birth (1 if death due to mi
scarriage, abortion or still birth) or not (0 if not applicable).
nfhs5_subset_clean <- nfhs5_subset_clean %>%
  mutate(
    death_during_birth = ifelse(s234 %in% c(1,2,3), 1, 0),
         )
# "s234" was then deleted as the column is no longer needed (death_during_birth - sub
stitute column)
nfhs5_subset_clean <- subset(nfhs5_subset_clean, select = -s234)

# Create label for "hunger", using the formula: If child is either undernourished, st
unted/wasted or died during birth, mark child as a case of child hunger.
# undernourished: if anemia level of child (hw57) is severe (1) or moderate (2), the
child is marked as undernourished
# stunted/wasted: if size of child at birth (m18) is smaller than average (4) or very
small (5), the child is marked as stunted/wasted
nfhs5_subset_clean <- nfhs5_subset_clean %>%
  mutate(
    undernourished = ifelse(hw57 %in% c(1,2), 1, 0),
    stunted_wasted = ifelse(m18 %in% c(4,5), 1, 0),
    hunger = ifelse(undernourished==1 | stunted_wasted==1 | death_during_birth==1, 1,
0)
         )

# For the input features, remove any rows with values "missing", "not applicable" and
"don't know"
nfhs5_subset_clean <- subset(nfhs5_subset_clean, !(s116 %in% c(8, NA)))
nfhs5_subset_clean <- nfhs5_subset_clean %>%
  mutate(
    v130 = ifelse(v130 %in% c(1), 1, ifelse(v130 %in% c(2), 2, 3)),
         )
nfhs5_subset_clean <- subset(nfhs5_subset_clean, !(v130 %in% c(NA)))
nfhs5_subset_clean <- subset(nfhs5_subset_clean, !(v457 %in% c(NA)))
nfhs5_subset_clean <- subset(nfhs5_subset_clean, !(v107 %in% c(99, NA)))
nfhs5_subset_clean <- subset(nfhs5_subset_clean, !(v463a %in% c(9, NA)))
nfhs5_subset_clean <- subset(nfhs5_subset_clean, !(v463c %in% c(9, NA)))
```

```
nfhs5_subset_clean <- subset(nfhs5_subset_clean, !(v113 %in% c(99)))
nfhs5_subset_clean <- subset(nfhs5_subset_clean, !(v116 %in% c(99, NA)))

# As we will run classification model, the predicted variable is converted to a facto
r variable.
nfhs5_subset_clean$hunger <- factor(nfhs5_subset_clean$hunger)

# Making variable names readable
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(anemia_level_child = hw57)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(size_of_child_at_birth = m18)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(sex_of_child = b4)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(caste = s116)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(state = v101)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(residence_type = v102)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(religion = v130)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(wealth_index = v190)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(anemia_level_mother = v457)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(educ_level_mother = v107)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(smoke_cigarette_mother = v463a)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(total_children_ever_born_to_mothe
r = v201)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(chew_tobacco_mother = v463c)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(number_of_living_siblings = v218)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(drinking_water_source = v113)
nfhs5_subset_clean <- nfhs5_subset_clean %>% rename(toilet_facility_type = v116)
```

Create train and test splits

```
# Set seed
set.seed(306)
# Create train-test split
split <- initial_split(nfhs5_subset_clean, prop = 2/3) # Set aside 1/3 for testing
train <- training(split)
test <- testing(split)
```

Model 1 - LASSO

```
### Install and load relevant packages

# install.packages("rsample")
library(rsample)

### Fit a GLM with lasso penalty to find the best predictors for child hunger

# Specify the LASSO logistic regression model
spec_lasso <- logistic_reg(mixture = 1, penalty = 0.01) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

# Specify the LASSO recipe
recipe_lasso <- recipe(hunger ~ sex_of_child+ caste+ state+ residence_type+
                         religion+ wealth_index+ anemia_level_mother+
                         educ_level_mother+ smoke_cigarette_mother+
                         total_children_ever_born_to_mother+ chew_tobacco_mother+
                         number_of_living_siblings + drinking_water_source +
                         toilet_facility_type,
                       data = train)

# Specify the LASSO workflow
wf_lasso <- workflow() %>%
  add_model(spec_lasso) %>%
  add_recipe(recipe_lasso)

# Fit the model workflow in train data set
my_lasso <- fit(wf_lasso, data = train)

# Predict the hunger variable on test data set
lasso_predicted <- predict(my_lasso, new_data=test)

### Test model accuracy and find the most significant variables

# Compare the predicted value of hunger against the true labels in the test set
test$accuracy = ifelse(lasso_predicted$.pred_class == test$hunger, 1, 0)

# Report the percentage of accurate predictions made by the model
mean(test$accuracy)
```

```
## [1] 0.5609569
```

```
# With an increasing level of penalty, check for the variables that are found to be s
ignificant, and the ones that are dropped
coeff_lasso_comp <- tidy(my_lasso, penalty = 0) %>%
  bind_rows(tidy(my_lasso, penalty = 0.001)) %>%
  bind_rows(tidy(my_lasso, penalty = 0.01)) %>%
  bind_rows(tidy(my_lasso, penalty = 0.02)) %>%
  bind_rows(tidy(my_lasso, penalty = 0.05))

# As the penalty increases, more number of coefficients are dropped to 0, leaving beh
ind the most significant input features with non-zero coefficient
coeff_lasso_comp %>%
  pivot_wider(term,names_from = penalty, names_prefix = "penalty=",
              values_from = estimate)
```

```
## Warning: Specifying the `id_cols` argument by position was deprecated in tidyr 1.
3.0.
## ℹ Please explicitly name `id_cols`, like `id_cols = term`.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## # A tibble: 15 × 6
##    term                 `penalty=0` `penalty=0.001` `penalty=0.01` `penalty=0.02`
##    <chr>                      <dbl>           <dbl>          <dbl>          <dbl>
##  1 (Intercept)              1.40            1.40         0.951            0.661
##  2 sex_of_child             0               0            0                0
##  3 caste                   -0.0198         -0.0198       0                0
##  4 state                   -0.00606        -0.00606     -0.00160          0
##  5 residence_type           0.0221          0.0221       0                0
##  6 religion                -0.198          -0.198       -0.154           -0.103
##  7 wealth_index            -0.0216         -0.0216      -0.00305          0
##  8 anemia_level_mother     -0.255          -0.255       -0.221           -0.178
##  9 educ_level_mother       -0.0182         -0.0182       0                0
## 10 smoke_cigarette_mo…      0               0            0                0
## 11 total_children_eve…      0               0            0                0
## 12 chew_tobacco_mother     -0.131          -0.131        0                0
## 13 number_of_living_s…     -0.0672         -0.0672      -0.0257           0
## 14 drinking_water_sou…      0               0            0                0
## 15 toilet_facility_ty…      0.00111         0.00111      0.0000591        0
## # ℹ 1 more variable: `penalty=0.05` <dbl>
```

Model 2 - CART

```r
### Install and load relevant packages

# install.packages("relaimpo")
# install.packages("parsnip")
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
##
## Attaching package: 'rpart'
```

```
## The following object is masked from 'package:dials':
##
##     prune
```

```r
library(relaimpo)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: boot
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
## Loading required package: survey
```

```
## Loading required package: grid
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:boot':
##
##     aml
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
##
## Attaching package: 'survey'
```

```
## The following object is masked from 'package:graphics':
##
##     dotchart
```

```
## Loading required package: mitools
```

```
## This is the global version of package relaimpo.
```

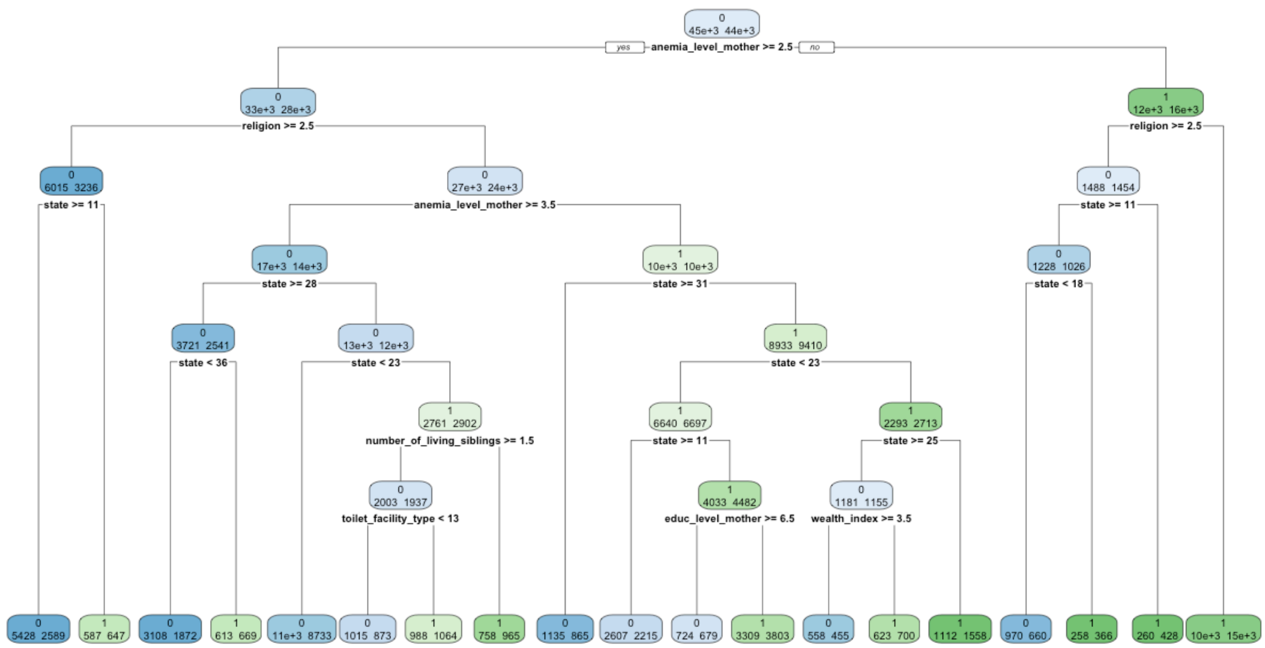
```
## If you are a non-US user, a version with the interesting additional metric pmvd is
available
```

```
## from Ulrike Groempings web site at prof.beuth-hochschule.de/groemping.
```

```
library(parsnip)

### Specify the model

# Use rpart to generate decision tree
class_tree_spec <- decision_tree() %>%
  set_engine("rpart") %>%
  set_mode("classification")

# Create decision tree workflow
class_tree_wf <- workflow() %>%
  add_model(class_tree_spec %>%
              set_args(cost_complexity = 0.001)) %>%
  add_formula(hunger ~ sex_of_child+ caste+ state+ residence_type+
                         religion+ wealth_index+ anemia_level_mother+
                         educ_level_mother+ smoke_cigarette_mother+
                         total_children_ever_born_to_mother+ chew_tobacco_mother+
                         number_of_living_siblings+ drinking_water_source +
                         toilet_facility_type)

# Fit the workflow of the model on train data set
class_tree_fit <- fit(class_tree_wf, data=train)

# Plot the tree
class_tree_fit %>%
  extract_fit_engine() %>%
  rpart.plot(extra=1, main="Classification Tree for Child Hunger")
```

```
## Warning: Cannot retrieve the data used to build the model (so cannot determine rou
ndint and is.binary for the variables).
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```

Classification Tree for Child Hunger




```
### Test model accuracy and find the most significant variables

# Extract the fitted model from the workflow
fitted_tree_model <- pull_workflow_fit(class_tree_fit)
```

```
## Warning: `pull_workflow_fit()` was deprecated in workflows 0.2.3.
## ℹ Please use `extract_fit_parsnip()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# Find the variables that are found to be most significant in the model
var_importance <- fitted_tree_model$fit$variable.importance
var_importance
```

```
##                  anemia_level_mother                                state
##                           601.254686                           341.093125
##                             religion                         wealth_index
##                           282.429308                            33.753524
##                  toilet_facility_type                               caste
##                            20.041144                            19.548272
##            number_of_living_siblings total_children_ever_born_to_mother
##                            14.439535                            14.269375
##                  chew_tobacco_mother                     educ_level_mother
##                            10.642248                             8.655100
##                drinking_water_source                       residence_type
##                             7.467404                             3.562586
##               smoke_cigarette_mother
##                             1.354096
```

```
# Predict hunger using the model on test data set
cart_predictions <- predict(class_tree_fit, new_data=test)

# Check for model accuracy on the test data set
mycart <- mean(cart_predictions$.pred_class == test$hunger)
mycart
```

```
## [1] 0.57196
```

Model 3: Random Forest

```
### Install and load relevant packages

# install.packages("vip")
library(ranger)
library(dplyr)
library(tidyverse)
library(parsnip)
library(vip)
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

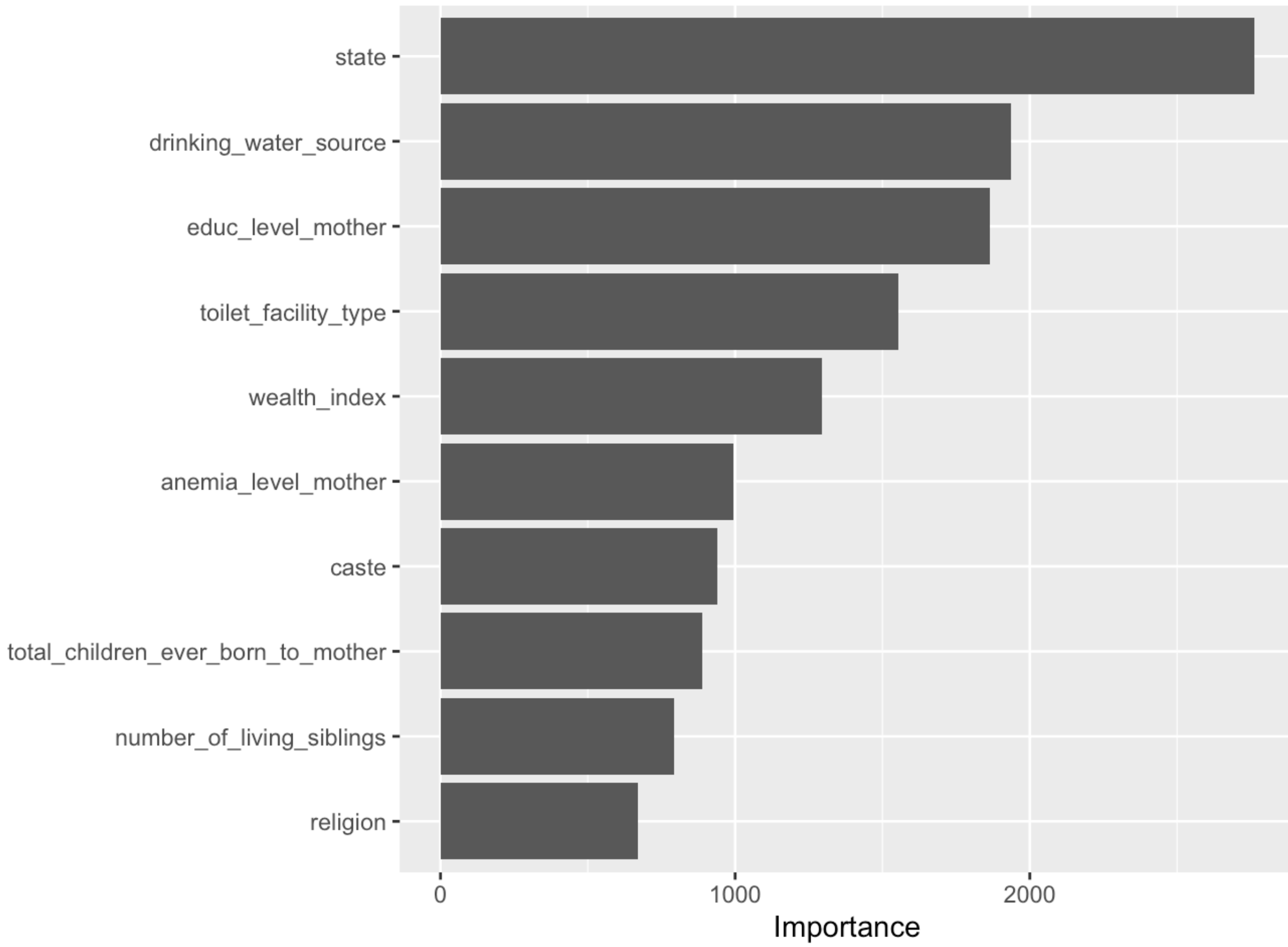
```
### Specify the model

# Use ranger engine to set up random forest specification
randomforest_spec <- rand_forest(trees=200) %>% set_engine("ranger", importance = "im
purity") %>%
  set_mode("classification")

# Setup recipe for the random forest
randomforest_rec <- recipe(hunger ~ sex_of_child+ caste+ state+ residence_type+
                         religion+ wealth_index+ anemia_level_mother+
                         educ_level_mother+ smoke_cigarette_mother+
                         total_children_ever_born_to_mother+ chew_tobacco_mother+
                         number_of_living_siblings + drinking_water_source +
                         toilet_facility_type, data=train)

# Setup workflow of random forest
randomforest_wf <- workflow() %>% add_model(randomforest_spec) %>%
  add_recipe(randomforest_rec)

# Fit the model
randomforest_fit <- fit(randomforest_wf, data=train)
randomforest_fit
```

```
## ══ Workflow [trained] ══════════════════════════════════════════════════════════
## Preprocessor: Recipe
## Model: rand_forest()
##
## ── Preprocessor ────────────────────────────────────────────────────────────────
## 0 Recipe Steps
##
## ── Model ───────────────────────────────────────────────────────────────────────
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, num.trees = ~200,      importance = ~"impurity", num.threads = 1, verbose = FALSE,      seed = sample.int(10^5, 1), probability = TRUE)
##
## Type:                             Probability estimation
## Number of trees:                  200
## Sample size:                      88702
## Number of independent variables:  14
## Mtry:                             3
## Target node size:                 10
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.2419148
```

```
### Test model accuracy and find the most significant variables

# Generate the predictions using test data set
randomforest_predicted <- predict(randomforest_fit, new_data=test)

# Calculate accuracy of predictions on test data set
myrandomforest <- mean(randomforest_predicted$.pred_class == test$hunger)
myrandomforest
```

```
## [1] 0.5776871
```

```
# Get variable importance measures
importance <- vip::vip(randomforest_fit)

# Print variable importance
print(importance)
```



Overall Finding from the 3 Models:

Model | Predictive Accuracy | Top 5 most influencial variables identified

LASSO | 56.09% | (1) Anemia level of mother, (2) Religion, (3) Number of living siblings, (4) State, (5) Wealth index

CART | 57.19% | (1) Religion, (2) Anemia level of mother, (3) State, (4) Wealth index,

(5) Toilet facility type

Random Forest | 57.81% | (1) State, (2) Drinking water source, (3) Education level of mother, (4) Toilet facility type, (5) Wealth index

It is observed that the predictive accuracy is similar across models.

The common input features that were present in the top 5 most important variables across models include: State, Wealth index.

Both CART and Random Forest models have the following common variables: State, Wealth index, toilet facility type

While religion and anemia level of mother are found to be highly predictive of child hunger in LASSO and CART model, the two variables are not the most predictive for random forest model.

Future work:

(1) Provide the data set with more number of input features, which may be more predictive of child hunger

(2) Improve the robustness of measurement method for the construct (child hunger)

(3) Tune parameters of the model to reduce overfitting and improve model performance