samer

**Introduction**

Economic well-being is a critical factor for governments, policymakers, and NGOs to design effective interventions, particularly in regions like Africa where traditional household surveys are expensive and infrequent. This study leverages readily available geospatial data to predict a wealth index, a standardized measure ranging from 0 to 1, derived from household survey responses. The wealth index reflects average wealth across clusters of respondents based on factors like asset ownership. Utilizing datasets with continent-wide coverage—such as human settlement layers, land cover, nighttime light emissions, and distances to capitals and shorelines—this analysis aims to develop a scalable predictive model. The dataset, “wealth.csv,” includes variables like population density, land use fractions, and urban/rural classifications, offering insights into economic patterns. Through descriptive analysis, unsupervised and supervised learning, and advanced techniques like tuning and stacking, this study explores relationships between these variables and wealth, proposing new features and evaluating model performance with metrics like Mean Absolute Percentage Error (MAPE) and detection prevalence

2025-05-23

## R Markdown

library(tidymodels)

## Warning: le package 'tidymodels' a été compilé avec la version R 4.4.2

## ── Attaching packages ────────────────────────────────────── tidymodels 1.2.0 ──

## ✔ broom 1.0.7 ✔ recipes 1.1.0  
## ✔ dials 1.4.0 ✔ rsample 1.2.1  
## ✔ dplyr 1.1.4 ✔ tibble 3.2.1  
## ✔ ggplot2 3.5.1 ✔ tidyr 1.3.1  
## ✔ infer 1.0.7 ✔ tune 1.2.1  
## ✔ modeldata 1.4.0 ✔ workflows 1.1.4  
## ✔ parsnip 1.2.1 ✔ workflowsets 1.1.0  
## ✔ purrr 1.0.2 ✔ yardstick 1.3.1

## Warning: le package 'dials' a été compilé avec la version R 4.4.3

## Warning: le package 'infer' a été compilé avec la version R 4.4.2

## Warning: le package 'modeldata' a été compilé avec la version R 4.4.2

## Warning: le package 'parsnip' a été compilé avec la version R 4.4.2

## Warning: le package 'recipes' a été compilé avec la version R 4.4.2

## Warning: le package 'rsample' a été compilé avec la version R 4.4.2

## Warning: le package 'tune' a été compilé avec la version R 4.4.2

## Warning: le package 'workflows' a été compilé avec la version R 4.4.2

## Warning: le package 'workflowsets' a été compilé avec la version R 4.4.2

## Warning: le package 'yardstick' a été compilé avec la version R 4.4.2

## ── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
## ✖ purrr::discard() masks scales::discard()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ recipes::step() masks stats::step()  
## • Learn how to get started at https://www.tidymodels.org/start/

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()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ stringr::fixed() masks recipes::fixed()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ readr::spec() masks yardstick::spec()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(finetune)

## Warning: le package 'finetune' a été compilé avec la version R 4.4.3

library(kknn)

## Warning: le package 'kknn' a été compilé avec la version R 4.4.3

library(kernlab)

##   
## Attachement du package : 'kernlab'  
##   
## L'objet suivant est masqué depuis 'package:purrr':  
##   
## cross  
##   
## L'objet suivant est masqué depuis 'package:ggplot2':  
##   
## alpha  
##   
## L'objet suivant est masqué depuis 'package:dials':  
##   
## buffer  
##   
## L'objet suivant est masqué depuis 'package:scales':  
##   
## alpha

library(ranger)

## Warning: le package 'ranger' a été compilé avec la version R 4.4.3

library(lightgbm)

## Warning: le package 'lightgbm' a été compilé avec la version R 4.4.3

library(bonsai)

## Warning: le package 'bonsai' a été compilé avec la version R 4.4.3

library(themis)

## Warning: le package 'themis' a été compilé avec la version R 4.4.3

library(ggplot2)  
library(baguette)

## Warning: le package 'baguette' a été compilé avec la version R 4.4.3

library(dials)  
library(pROC)

## Warning: le package 'pROC' a été compilé avec la version R 4.4.3

## Type 'citation("pROC")' for a citation.  
##   
## Attachement du package : 'pROC'  
##   
## Les objets suivants sont masqués depuis 'package:stats':  
##   
## cov, smooth, var

library(tidyverse)  
library(tidymodels)  
library(tidyclust)

## Warning: le package 'tidyclust' a été compilé avec la version R 4.4.3

##   
## Attachement du package : 'tidyclust'

## Les objets suivants sont masqués depuis 'package:parsnip':  
##   
## knit\_engine\_docs, list\_md\_problems

library(corrplot)

## Warning: le package 'corrplot' a été compilé avec la version R 4.4.3

## corrplot 0.95 loaded

library(Rtsne)

## Warning: le package 'Rtsne' a été compilé avec la version R 4.4.3

library(embed)

## Warning: le package 'embed' a été compilé avec la version R 4.4.3

library(tidytext)

## Warning: le package 'tidytext' a été compilé avec la version R 4.4.3

##### Q1. Read the file “wealth.csv” and inspect the dataset.

df=read.csv(file.choose())  
glimpse(df)

## Rows: 21,454  
## Columns: 19  
## $ ID <chr> "ID\_AAIethGy", "ID\_AAYiaCeL", …  
## $ country <chr> "Ethiopia", "Ethiopia", "Mozam…  
## $ year <int> 2016, 2005, 2009, 2015, 2012, …  
## $ urban\_or\_rural <chr> "R", "R", "R", "R", "U", "U", …  
## $ ghsl\_water\_surface <dbl> 0.0000000000, 0.0000000000, 0.…  
## $ ghsl\_built\_pre\_1975 <dbl> 0.0000000000, 0.0000000000, 0.…  
## $ ghsl\_built\_1975\_to\_1990 <dbl> 0.000000e+00, 1.098293e-04, 0.…  
## $ ghsl\_built\_1990\_to\_2000 <dbl> 5.549359e-05, 0.000000e+00, 0.…  
## $ ghsl\_built\_2000\_to\_2014 <dbl> 5.364380e-04, 1.830489e-05, 0.…  
## $ ghsl\_not\_built\_up <dbl> 0.99940807, 0.99987187, 1.0000…  
## $ ghsl\_pop\_density <dbl> 12.1461340, 113.8067163, 0.000…  
## $ landcover\_crops\_fraction <dbl> 25.48965903, 64.13605339, 4.40…  
## $ landcover\_urban\_fraction <dbl> 0.8794843, 0.6014272, 0.131900…  
## $ landcover\_water\_permanent\_10km\_fraction <dbl> 0.000000000, 0.000000000, 0.00…  
## $ landcover\_water\_seasonal\_10km\_fraction <dbl> 0.000000e+00, 5.426636e-03, 3.…  
## $ nighttime\_lights <dbl> 0.0000000, 0.0000000, 0.000000…  
## $ dist\_to\_capital <dbl> 278.788451, 200.986978, 642.59…  
## $ dist\_to\_shoreline <dbl> 769.338378, 337.135243, 169.91…  
## $ Target <dbl> 0.132782655, 0.004898371, 0.09…

This dataset describes 21,454 locations, mostly in African countries, with details on whether they’re urban or rural, land use (crops, buildings, water), population density, nighttime lights (indicating infrastructure), and distances to capitals or shorelines. The Target variable likely measures something like poverty or wealth, with higher values possibly showing better or worse conditions. Most areas are rural with low development and infrastructure.

##### Q2. perform some descriptive analysis on the different variables of the dataset and study their relationship

to the target variable.

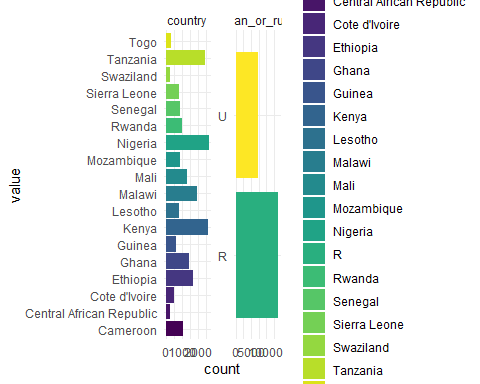
descriptive analysis

summary(df)

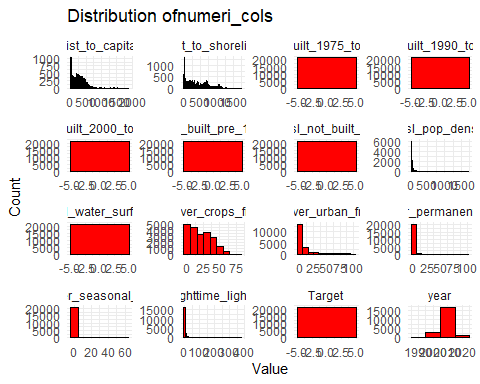
## ID country year urban\_or\_rural   
## Length:21454 Length:21454 Min. :1994 Length:21454   
## Class :character Class :character 1st Qu.:2008 Class :character   
## Mode :character Mode :character Median :2011 Mode :character   
## Mean :2010   
## 3rd Qu.:2014   
## Max. :2016   
## ghsl\_water\_surface ghsl\_built\_pre\_1975 ghsl\_built\_1975\_to\_1990  
## Min. :0.00000 Min. :0.0000000 Min. :0.0000000   
## 1st Qu.:0.00000 1st Qu.:0.0000000 1st Qu.:0.0000000   
## Median :0.00000 Median :0.0001975 Median :0.0007092   
## Mean :0.02826 Mean :0.0382224 Mean :0.0286437   
## 3rd Qu.:0.00000 3rd Qu.:0.0079866 3rd Qu.:0.0098682   
## Max. :0.96996 Max. :0.8771158 Max. :0.6850103   
## ghsl\_built\_1990\_to\_2000 ghsl\_built\_2000\_to\_2014 ghsl\_not\_built\_up   
## Min. :0.0000000 Min. :0.0000000 Min. :0.0008593   
## 1st Qu.:0.0000428 1st Qu.:0.0001241 1st Qu.:0.8978674   
## Median :0.0010009 Median :0.0018706 Median :0.9919189   
## Mean :0.0126889 Mean :0.0183863 Mean :0.8737999   
## 3rd Qu.:0.0081277 3rd Qu.:0.0149363 3rd Qu.:0.9995324   
## Max. :0.5155339 Max. :0.6491589 Max. :1.0000000   
## ghsl\_pop\_density landcover\_crops\_fraction landcover\_urban\_fraction  
## Min. : 0.000 Min. : 0.000 Min. : 0.0000   
## 1st Qu.: 3.849 1st Qu.: 5.611 1st Qu.: 0.7988   
## Median : 17.633 Median :18.509 Median : 2.7702   
## Mean : 95.757 Mean :21.034 Mean :13.9991   
## 3rd Qu.: 63.226 3rd Qu.:33.590 3rd Qu.:12.6215   
## Max. :1741.256 Max. :80.065 Max. :98.7841   
## landcover\_water\_permanent\_10km\_fraction landcover\_water\_seasonal\_10km\_fraction  
## Min. : 0.00000 Min. : 0.00000   
## 1st Qu.: 0.00000 1st Qu.: 0.00132   
## Median : 0.00052 Median : 0.02915   
## Mean : 1.48685 Mean : 0.71489   
## 3rd Qu.: 0.15236 3rd Qu.: 0.38197   
## Max. :99.16402 Max. :56.20164   
## nighttime\_lights dist\_to\_capital dist\_to\_shoreline Target   
## Min. : 0.0000 Min. : 0.1053 Min. : 0.1121 Min. :0.0000   
## 1st Qu.: 0.0000 1st Qu.: 115.8909 1st Qu.: 126.3795 1st Qu.:0.1958   
## Median : 0.1373 Median : 256.7365 Median : 327.2720 Median :0.2936   
## Mean : 8.5065 Mean : 289.7223 Mean : 402.6085 Mean :0.3507   
## 3rd Qu.: 4.8302 3rd Qu.: 401.1531 3rd Qu.: 643.9107 3rd Qu.:0.4990   
## Max. :382.9328 Max. :1897.3516 Max. :1769.5239 Max. :1.0000

This dataset of 21,454 locations, primarily in African countries (1994–2016), describes urban/rural settings, land use, and socio-economic factors. Most locations are rural with low urban development (high ghsl\_not\_built\_up, mean 0.87) and limited infrastructure (low nighttime\_lights, median 0.14). Land is often used for crops (mean 21%), with minimal water presence. Distances to capitals (mean 290 km) and shorelines (mean 403 km) vary widely. The Target variable (0–1, mean 0.35) likely indicates a socio-economic outcome like wealth or poverty, with higher values possibly reflecting better conditions.

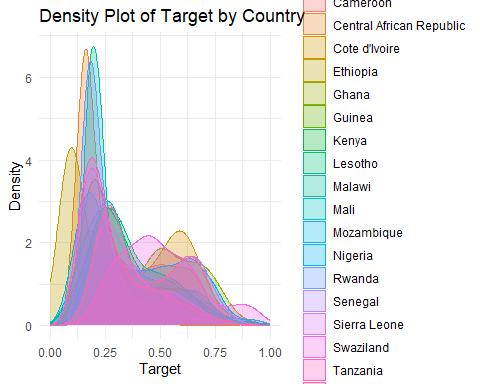
catego\_cols <- df %>%  
 select(where(is.character), where(is.factor)) %>%  
 mutate(across(everything(), as.character))   
  
catego\_cols %>%  
 select(-ID) %>%   
 pivot\_longer(cols = everything()) %>%  
 ggplot(aes(x = value, fill = value)) +  
 facet\_wrap(~name, scales = "free", ncol = 3) +  
 geom\_bar() +   
 coord\_flip() +  
 theme\_minimal() +  
 scale\_fill\_viridis\_d()

 Country Distribution: Tanzania has the highest count (around 2,000), followed by Nigeria and Kenya (both around 1,500–2,000). Cameroon and the Central African Republic have the lowest counts (around 500–1,000). Urban vs. Rural: Most locations are rural (“R”), with a count of about 10,000, while urban (“U”) locations total around 5,000. Country-Specific Trends: Tanzania, Togo, and Swaziland lean more urban, while Malawi, Mozambique, and Nigeria have a stronger rural presence. This suggests a predominantly rural dataset with varying country representation.

numeri\_cols <- df %>%  
 select(where(is.numeric))  
  
numeri\_cols %>%  
 pivot\_longer(cols = everything()) %>%  
 ggplot(aes(x = value)) +   
 geom\_histogram(binwidth = 10, fill = "red", color = "black") +   
 theme\_minimal() +  
 facet\_wrap(~ name, scales = "free") +   
 labs(title = "Distribution ofnumeri\_cols", x = "Value", y = "Count")

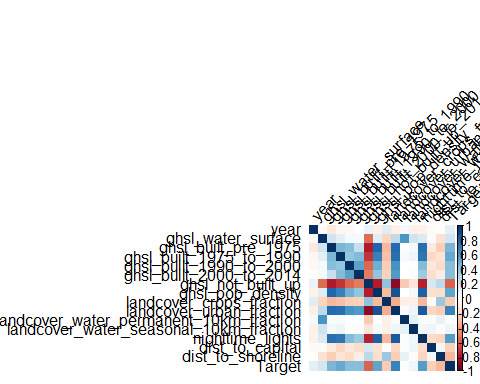
 Distances: dist\_to\_capital and dist\_to\_shoreline are right-skewed, with most locations within 500 km of capitals and shorelines, though some are as far as 2,000 km. Built-up Areas: ghsl\_built\_pre\_1975, ghsl\_built\_1975\_to\_1990, ghsl\_built\_1990\_to\_2000, and ghsl\_built\_2000\_to\_2014 are heavily skewed toward 0, indicating most areas have little urban development across all periods. Land Use: ghsl\_not\_built\_up is near 1 for most locations, showing predominantly non-urban land. landcover\_crops\_fraction peaks around 0–25%, while landcover\_urban\_fraction is mostly under 25%. Water and Lights: ghsl\_water\_surface, landcover\_water\_permanent\_10km, and landcover\_water\_seasonal\_10km are near 0, indicating minimal water presence. nighttime\_lights is mostly 0, suggesting limited infrastructure. Population and Target: ghsl\_pop\_density is right-skewed, with most values under 100. Target is centered around 0.3, ranging from 0 to 1. Year: Data spans 1994–2016, with a peak around 2010. Overall, the dataset reflects mostly rural, underdeveloped areas with low urban growth, limited infrastructure, and varying socio-economic conditions.

df %>%  
 ggplot(aes(x = Target, color = country, fill = country)) +  
 geom\_density(alpha = 0.3) +   
 theme\_minimal() +  
 labs(title = "Density Plot of Target by Country",  
 x = "Target",  
 y = "Density")

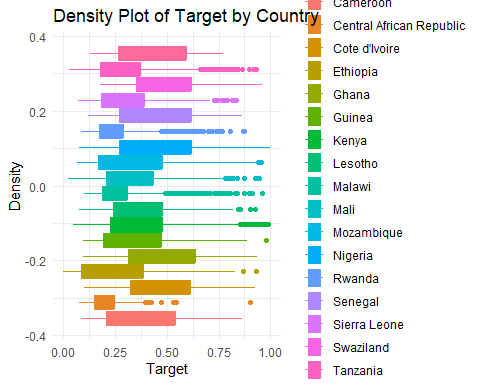
 Most countries have a peak Target value around 0.25–0.5, indicating a common range for this variable, Ethiopia, Kenya, and Nigeria show higher densities at lower Target values (around 0.25), suggesting a larger proportion of lower values. Cameroon, Central African Republic, and Togo have flatter distributions with less concentration, indicating more variability in Target values. Few countries (e.g., Swaziland, Togo) show noticeable density at higher Target values (above 0.75), suggesting rare higher outcomes. Overall, the Target variable tends to cluster around 0.25–0.5 across most countries, with some variation in spread and peak locations.

EDA multi var

cor\_matrix <- cor(numeri\_cols)  
corrplot(cor\_matrix, method = "color", type = "full", tl.col = "black", tl.srt = 45)

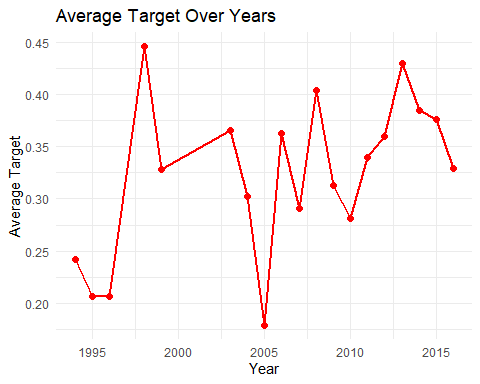
 Urban areas (landcover\_urban\_fraction) and lights (nighttime\_lights) are strongly linked (0.8). Built-up areas over time (ghsl\_built\_\*) are highly related (0.6–0.9) but opposite to undeveloped land (ghsl\_not\_built\_up, -0.6 to -0.9). Crops (landcover\_crops\_fraction) and urban areas are negatively linked (-0.6). Target is moderately tied to urban settings and lights (0.4–0.5) but less so with undeveloped land (-0.4). Distances and water variables show weak connections. Urban development and Target (possibly wealth) align, while rural areas do not.

df %>%  
 ggplot(aes(x = Target, color = country, fill = country)) +  
 theme\_minimal() +  
 geom\_boxplot() +   
  
 labs(title = "Density Plot of Target by Country",  
 x = "Target",  
 y = "Density")

 This density plot shows how the Target variable (0 to 1) varies across 18 African countries. Most countries have Target values clustered between 0.25 and 0.5, . Ethiopia, Kenya, and Nigeria have more data points around 0.25, suggesting lower values, while Swaziland and Togo show some higher values (above 0.75). Cameroon and the Central African Republic have more spread-out values. Overall, Target values are generally moderate across most countries.

library(dplyr)  
library(ggplot2)  
  
mean\_df <- df %>%  
 group\_by(year) %>%  
 summarise(mean\_tar = mean(Target, na.rm = TRUE))   
mean\_df %>%  
 ggplot(aes(x = as.numeric(year), y = mean\_tar)) +  
 geom\_line(color = "red", size = 1) +   
 geom\_point(color = "red", size = 2) +   
 theme\_minimal() +  
 labs(title = "Average Target Over Years",  
 x = "Year",  
 y = "Average Target")

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

 This line graph shows the average Target value (possibly a socio-economic indicator like wealth) from 1994 to 2016. It starts at 0.20, peaks at 0.45 around 1997, dips to 0.20 in 2007, and fluctuates between 0.25 and 0.45 until 2016, ending around 0.35. The trend is highly variable with no clear upward or downward pattern.

##### Q3. Reproduce all the plots (logic) of the data scientist (Unuspervised Learning) in the paper exam (see

f ile attached) and explain the logic behind it.

unsupervised learn:

pca

pca\_cols<-df %>% select(-c(Target,ID,urban\_or\_rural))  
pca\_recipe <- recipe(~., data = pca\_cols) %>%  
 update\_role(country,new\_role = 'id') %>%   
 step\_normalize(all\_predictors()) %>%  
 step\_pca(all\_predictors())  
pca\_prep <- prep(pca\_recipe)  
pca\_df<-juice(pca\_prep)  
pca\_prep

##

## ── Recipe ──────────────────────────────────────────────────────────────────────

##

## ── Inputs

## Number of variables by role

## predictor: 15  
## id: 1

##

## ── Training information

## Training data contained 21454 data points and no incomplete rows.

##

## ── Operations

## • Centering and scaling for: year and ghsl\_water\_surface, ... | Trained

## • PCA extraction with: year and ghsl\_water\_surface, ... | Trained

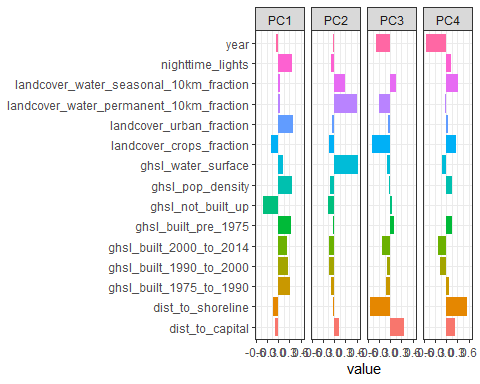
tidy(pca\_prep, 1)

## # A tibble: 30 × 4  
## terms statistic value id   
## <chr> <chr> <dbl> <chr>   
## 1 year mean 2010. normalize\_1hO7f  
## 2 ghsl\_water\_surface mean 0.0283 normalize\_1hO7f  
## 3 ghsl\_built\_pre\_1975 mean 0.0382 normalize\_1hO7f  
## 4 ghsl\_built\_1975\_to\_1990 mean 0.0286 normalize\_1hO7f  
## 5 ghsl\_built\_1990\_to\_2000 mean 0.0127 normalize\_1hO7f  
## 6 ghsl\_built\_2000\_to\_2014 mean 0.0184 normalize\_1hO7f  
## 7 ghsl\_not\_built\_up mean 0.874 normalize\_1hO7f  
## 8 ghsl\_pop\_density mean 95.8 normalize\_1hO7f  
## 9 landcover\_crops\_fraction mean 21.0 normalize\_1hO7f  
## 10 landcover\_urban\_fraction mean 14.0 normalize\_1hO7f  
## # ℹ 20 more rows

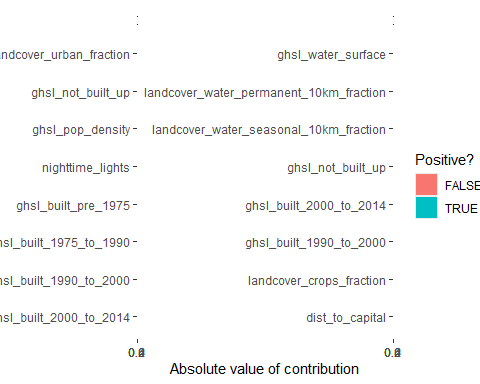
tidied\_pca <- tidy(pca\_prep, 2)  
tidied\_pca

## # A tibble: 225 × 4  
## terms value component id   
## <chr> <dbl> <chr> <chr>   
## 1 year -0.0424 PC1 pca\_SmQbs  
## 2 ghsl\_water\_surface 0.133 PC1 pca\_SmQbs  
## 3 ghsl\_built\_pre\_1975 0.336 PC1 pca\_SmQbs  
## 4 ghsl\_built\_1975\_to\_1990 0.322 PC1 pca\_SmQbs  
## 5 ghsl\_built\_1990\_to\_2000 0.263 PC1 pca\_SmQbs  
## 6 ghsl\_built\_2000\_to\_2014 0.233 PC1 pca\_SmQbs  
## 7 ghsl\_not\_built\_up -0.393 PC1 pca\_SmQbs  
## 8 ghsl\_pop\_density 0.364 PC1 pca\_SmQbs  
## 9 landcover\_crops\_fraction -0.194 PC1 pca\_SmQbs  
## 10 landcover\_urban\_fraction 0.395 PC1 pca\_SmQbs  
## # ℹ 215 more rows

tidied\_pca %>%  
filter(  
component == "PC1" |  
component == "PC2" |  
component == "PC3" |  
component == "PC4"  
) %>%  
mutate(component = fct\_inorder(component)) %>%  
ggplot(aes(value, terms, fill = terms)) +  
geom\_col(show.legend = FALSE) +  
facet\_wrap(~component, nrow = 1) +  
labs(y = NULL) +  
theme\_bw()



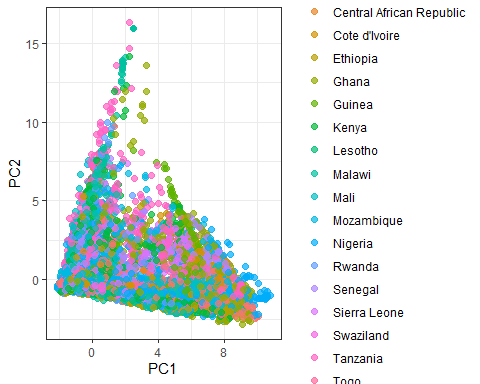
tidied\_pca %>%  
filter(component %in% paste0("PC", 1:2)) %>%  
group\_by(component) %>%  
top\_n(8, abs(value)) %>%  
ungroup() %>%  
mutate(terms = reorder\_within(terms, abs(value), component)) %>%  
ggplot(aes(abs(value), terms, fill = value > 0)) +  
geom\_col() +  
facet\_wrap(~component, scales = "free\_y") +  
scale\_y\_reordered() +  
labs(  
x = "Absolute value of contribution",  
y = NULL, fill = "Positive?"  
)



library(ggrepel)

## Warning: le package 'ggrepel' a été compilé avec la version R 4.4.3

juice(pca\_prep) |>  
 ggplot(aes(PC1, PC2)) +  
 geom\_point(aes(color = country), alpha = 0.7, size = 2) +  
 ggrepel::geom\_text\_repel(  
 aes(label = ""),  
 max.overlaps = 40,  
 size = 3  
 ) +  
 labs(color = "Country") +  
 theme\_bw()

 PC1: Dominated by nighttime\_lights and landcover\_urban\_fraction (positive), and ghsl\_not\_built\_up (negative), suggesting it captures urban development vs. undeveloped land. PC2: Strongly influenced by ghsl\_water\_surface and landcover\_water\_permanent\_10km\_fraction (positive), indicating a water-related factor. PC3: Highlights year and dist\_to\_capital (positive), possibly reflecting temporal or distance-to-capital trends. PC4: Driven by dist\_to\_shoreline (positive) and ghsl\_built\_pre\_1975 (negative), suggesting a coastal vs. older built-up contrast. Each component reflects a different aspect of the data, from urbanization to water presence and geographic factors.

contributions of variables to PC1 and PC2, with colors indicating positive (blue) or negative (red) influence. For PC1, landcover\_urban\_fraction, ghsl\_pop\_density, and nighttime\_lights contribute positively (around 0.6), while ghsl\_not\_built\_up contributes negatively (around 0.6), highlighting an urban vs. rural contrast. For PC2, ghsl\_water\_surface and landcover\_water\_permanent\_10km\_fraction contribute positively (around 0.6), while ghsl\_not\_built\_up and landcover\_crops\_fraction contribute negatively (around 0.4), reflecting a water vs. land use pattern.

scatter plot displays locations from 18 African countries on the first two principal components (PC1 and PC2). PC1 (x-axis) likely represents urban development (higher values indicate more urbanization), and PC2 (y-axis) reflects water presence (higher values indicate more water). Most points cluster near PC1 values of 0–4 and PC2 values of 0–5, indicating predominantly rural areas with low water presence. Some outliers (e.g., Togo, Swaziland) have higher PC1 values (up to 8), showing more urbanized areas, and a few (e.g., Kenya, Guinea) have higher PC2 values (up to 15), indicating greater water presence. Overall, most locations are rural with limited water.

umap

library(uwot)

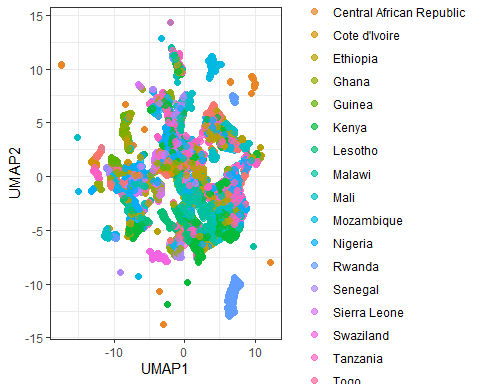
## Warning: le package 'uwot' a été compilé avec la version R 4.4.3

## Le chargement a nécessité le package : Matrix

##   
## Attachement du package : 'Matrix'

## Les objets suivants sont masqués depuis 'package:tidyr':  
##   
## expand, pack, unpack

umap\_recip <- recipe(~ ., data = pca\_cols) %>%  
 update\_role(country, new\_role = "id") %>%  
 step\_normalize(all\_predictors())  
  
umap\_prep <- prep(umap\_recip)  
  
prepr\_df <- juice(umap\_prep)  
  
numeric\_data <- prepr\_df %>%  
 select(-country)  
  
set.seed(42)  
result\_umap <- umap(numeric\_data, n\_components = 2, n\_neighbors = 15, min\_dist = 0.1)  
  
umap\_df <- data.frame(  
 UMAP1 = result\_umap[, 1],  
 UMAP2 = result\_umap[, 2],  
 country = prepr\_df$country  
)  
  
res\_umap <- umap\_df %>%  
 ggplot(aes(UMAP1, UMAP2)) +  
 geom\_point(aes(color = country), alpha = 0.7, size = 2) +  
 ggrepel::geom\_text\_repel(  
 aes(label = ""),   
 max.overlaps = 40,  
 size = 3  
 ) +  
 labs(color = "Country") +  
 theme\_bw()  
  
print(res\_umap)

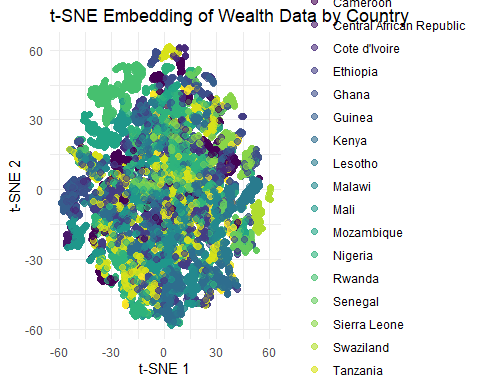
 This UMAP plot visualizes data points from 18 African countries across two dimensions (UMAP1 and UMAP2). The plot shows clusters of points, with each country represented by a unique color. Most data points cluster around UMAP1 0 to 10 and UMAP2 -5 to 5, indicating a central group of similar locations, likely rural areas. Outliers, such as Togo (pink) and Central African Republic (orange), appear at the edges (e.g., UMAP1 10–15 or UMAP2 10–15), suggesting distinct characteristics, possibly urban or unique environmental factors. Overall, the data shows a mix of overlapping and distinct country patterns.

tsne

library(dplyr)  
library(ggplot2)  
library(Rtsne)  
  
numeri\_cols <- df %>%  
 select(where(is.numeric)) %>%  
 select(-Target)   
  
  
numeri\_cols <- numeri\_cols[!duplicated(numeri\_cols), ]  
  
numeri\_cols <- scale(numeri\_cols)  
  
org <- which(complete.cases(df %>% select(where(is.numeric)) %>% select(-Target)) &   
 !duplicated(df %>% select(where(is.numeric)) %>% select(-Target)))  
country\_clean <- df$country[org]  
  
set.seed(42)   
tsne\_result <- Rtsne(numeri\_cols, dims = 2, perplexity = 20, verbose = TRUE)

## Performing PCA  
## Read the 21295 x 15 data matrix successfully!  
## OpenMP is working. 1 threads.  
## Using no\_dims = 2, perplexity = 20.000000, and theta = 0.500000  
## Computing input similarities...  
## Building tree...  
## - point 10000 of 21295  
## - point 20000 of 21295  
## Done in 3.71 seconds (sparsity = 0.003826)!  
## Learning embedding...  
## Iteration 50: error is 111.879429 (50 iterations in 4.37 seconds)  
## Iteration 100: error is 108.599984 (50 iterations in 4.50 seconds)  
## Iteration 150: error is 93.115285 (50 iterations in 3.68 seconds)  
## Iteration 200: error is 89.825135 (50 iterations in 3.63 seconds)  
## Iteration 250: error is 88.288950 (50 iterations in 3.62 seconds)  
## Iteration 300: error is 3.811293 (50 iterations in 3.52 seconds)  
## Iteration 350: error is 3.327559 (50 iterations in 3.53 seconds)  
## Iteration 400: error is 3.009524 (50 iterations in 3.76 seconds)  
## Iteration 450: error is 2.775555 (50 iterations in 4.00 seconds)  
## Iteration 500: error is 2.592848 (50 iterations in 3.96 seconds)  
## Iteration 550: error is 2.445422 (50 iterations in 4.31 seconds)  
## Iteration 600: error is 2.323824 (50 iterations in 3.80 seconds)  
## Iteration 650: error is 2.221315 (50 iterations in 4.33 seconds)  
## Iteration 700: error is 2.132933 (50 iterations in 4.29 seconds)  
## Iteration 750: error is 2.056004 (50 iterations in 4.36 seconds)  
## Iteration 800: error is 1.988203 (50 iterations in 4.04 seconds)  
## Iteration 850: error is 1.928568 (50 iterations in 4.12 seconds)  
## Iteration 900: error is 1.875208 (50 iterations in 4.04 seconds)  
## Iteration 950: error is 1.828073 (50 iterations in 3.94 seconds)  
## Iteration 1000: error is 1.787089 (50 iterations in 3.99 seconds)  
## Fitting performed in 79.80 seconds.

tsne\_df <- data.frame(  
 TSNE1 = tsne\_result$Y[, 1],  
 TSNE2 = tsne\_result$Y[, 2],  
 country = country\_clean   
)  
  
ggplot(tsne\_df, aes(x = TSNE1, y = TSNE2, color = country)) +  
 geom\_point(alpha = 0.6, size = 2) +  
 theme\_minimal() +  
 labs(title = "t-SNE Embedding of Wealth Data by Country",  
 x = "t-SNE 1",  
 y = "t-SNE 2",  
 color = "Country") +  
 scale\_color\_viridis\_d()

 This t-SNE plot visualizes wealth data for 18 African countries across two dimensions (t-SNE 1 and t-SNE 2). The data points form clusters, with most countries overlapping around t-SNE 1 -30 to 30 and t-SNE 2 -30 to 30, suggesting similar wealth patterns. Outliers like Togo (yellow) and Swaziland (light yellow) appear at the edges (e.g., t-SNE 2 30–60), indicating distinct wealth profiles. The dense central cluster reflects a mix of countries with moderate wealth similarity, while peripheral points highlight unique cases

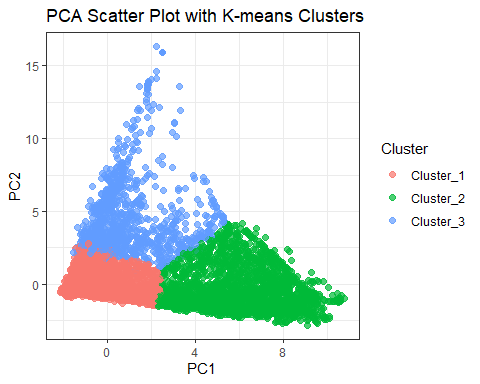
kmeans clustring

km\_cols<-df %>% select(-c(ID,country,Target,urban\_or\_rural))  
km\_clus <- k\_means(num\_clusters = 3) %>%  
 set\_engine("stats") %>%  
 set\_mode("partition")  
  
kmeans\_recipe <- recipe(~., data = km\_cols) %>%  
 step\_normalize(all\_predictors())  
  
kmeans\_wf <- workflow() %>%  
 add\_model(km\_clus) %>%  
 add\_recipe(kmeans\_recipe)  
  
kmeans\_fit <- fit(kmeans\_wf, data = km\_cols)  
kmeans\_fit %>% sse\_ratio()

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 sse\_ratio standard 0.608

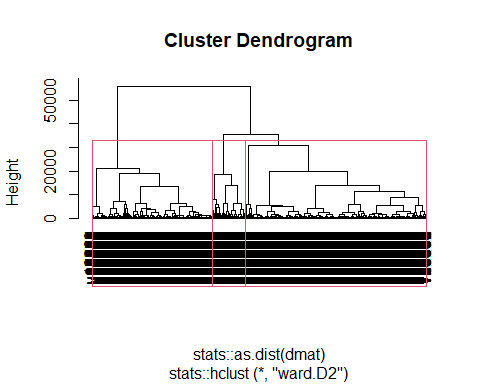
#silhouette\_avg(kmeans\_fit, new\_data = km\_cols)  
  
cluster\_assignments <- augment(kmeans\_fit, km\_cols) %>%  
 select(.pred\_cluster)

join\_data <- pca\_df %>%  
 mutate(.cluster = cluster\_assignments$.pred\_cluster )  
join\_data %>%  
 ggplot(aes(PC1, PC2)) +  
 geom\_point(aes(color = .cluster), alpha = 0.7, size = 2) +  
 ggrepel::geom\_text\_repel(  
 aes(label = ""),   
 max.overlaps = 40,  
 size = 3  
 ) +  
 labs(title = "PCA Scatter Plot with K-means Clusters",  
 x = "PC1",  
 y = "PC2",  
 color = "Cluster") +  
 theme\_bw()

 This PCA scatter plot uses k-means clustering to group data points into three clusters based on PC1 (urban development) and PC2 (water presence). Cluster\_1 (red) has low PC1 (0–2) and low PC2 (0–2), indicating rural areas with little water. Cluster\_2 (green) has slightly higher PC1 (0–4) and PC2 (0–15), suggesting rural areas with more water. Cluster\_3 (blue) has higher PC1 (4–8) and low PC2 (0–5), reflecting more urbanized areas with less water. Most points are rural, with Cluster\_3 showing the most urban development.

heirarchical cluster

kmean\_hc <- hier\_clust(linkage\_method = "ward.D2") %>%  
 set\_mode("partition") %>%  
 set\_engine("stats")  
 hier\_clust\_wf<-workflow() %>%   
 add\_recipe(kmeans\_recipe) %>%   
 add\_model(kmean\_hc)  
 df\_fit <- kmean\_hc %>% fit(formula = ~.,data = km\_cols)  
 # df\_fit %>% sse\_ratio()  
 df\_fit %>% extract\_fit\_engine() %>% plot(h =-1)  
 rect.hclust(df\_fit %>% extract\_fit\_engine(),k = 3)

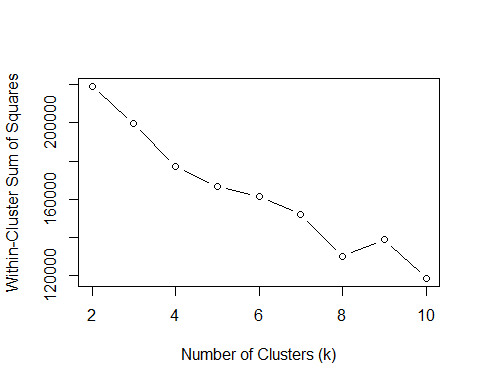
 This cluster dendrogram groups the dataset’s locations based on similarity, using the as.dist(dmat) distance metric. The height (y-axis) represents the dissimilarity between clusters, ranging from 0 to over 4000. At a height of around 2000, the dendrogram splits into three main clusters, indicating three distinct groups of locations. Below this height, smaller sub-clusters form, showing finer groupings. The plot suggests a hierarchical structure, with most locations being relatively similar (low height) but with some distinct groups emerging at higher dissimilarity levels.

##### Q4. Perform your own Unsupervised Learning analysis on this dataset.

my own Unsupervised Learning analysis

First lets use elbow

set.seed(42)  
wss <- map\_dbl(2:10, function(k) {  
 model <- k\_means(num\_clusters = k) %>%  
 set\_engine("stats") %>%  
 set\_mode("partition")  
 wf <- workflow() %>%  
 add\_model(model) %>%  
 add\_recipe(kmeans\_recipe)  
 fit <- fit(wf, data = km\_cols)  
 sum(fit$fit$fit$fit$withinss)  
})  
plot(2:10, wss, type = "b", xlab = "Number of Clusters (k)", ylab = "Within-Cluster Sum of Squares")

 we see that 2 cluster is better

slelct 2 clusterS

library(tidyverse)  
library(tidymodels)  
library(tidyclust)  
library(uwot)  
library(Rtsne)  
library(ggplot2)  
  
  
km\_cols <- df %>% select(-c(ID, country, Target, urban\_or\_rural))  
  
kmeans\_model <- k\_means(num\_clusters = 2) %>%  
 set\_engine("stats") %>%  
 set\_mode("partition")  
  
kmeans\_recipe <- recipe(~., data = km\_cols) %>%  
 step\_normalize(all\_predictors())  
  
kmeans\_wf <- workflow() %>%  
 add\_model(kmeans\_model) %>%  
 add\_recipe(kmeans\_recipe)  
  
kmeans\_fit <- fit(kmeans\_wf, data = km\_cols)

#kmeans\_silhouette <- silhouette\_avg(kmeans\_fit, new\_data = km\_cols)  
#kmeans\_sil\_score <- as.numeric(kmeans\_silhouette)   
#cat("Silhouette Score for K-means (2 clusters):", kmeans\_sil\_score, "\n")

Hierarchical Clustering with average Linkage

hier\_model <- hier\_clust(num\_clusters = 2, linkage\_method = "average") %>%  
 set\_mode("partition") %>%  
 set\_engine("stats")  
  
hier\_wf <- workflow() %>%  
 add\_model(hier\_model) %>%  
 add\_recipe(kmeans\_recipe)  
  
hier\_fit <- fit(hier\_wf, data = km\_cols)

pca\_recipe <- recipe(~., data = km\_cols) %>%  
 step\_normalize(all\_predictors()) %>%  
 step\_pca(all\_predictors(), num\_comp = 3)  
  
pca\_prep <- prep(pca\_recipe)  
pca\_df <- juice(pca\_prep)

str(kmeans\_fit %>% extract\_cluster\_assignment())

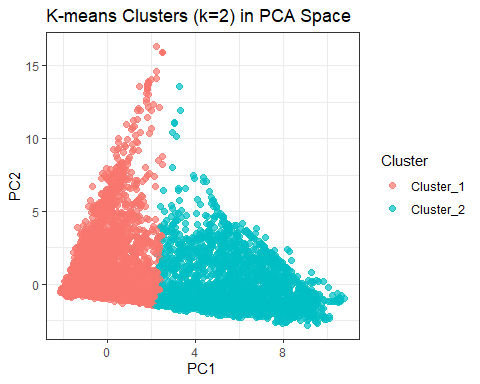
## tibble [21,454 × 1] (S3: tbl\_df/tbl/data.frame)  
## $ .cluster: Factor w/ 2 levels "Cluster\_1","Cluster\_2": 1 1 1 1 1 1 1 1 1 1 ...

str(hier\_fit %>% extract\_cluster\_assignment())

## tibble [21,454 × 1] (S3: tbl\_df/tbl/data.frame)  
## $ .cluster: Factor w/ 2 levels "Cluster\_1","Cluster\_2": 1 1 1 1 1 1 1 1 1 1 ...

#####Q5. Present any results from Q4.

pca\_df <- pca\_df %>%  
 mutate(kmeans\_cluster = kmeans\_fit %>% extract\_cluster\_assignment() %>% pull(.cluster),  
 hier\_cluster = hier\_fit %>% extract\_cluster\_assignment() %>% pull(.cluster))  
  
pca\_kmeans\_plot <- ggplot(pca\_df, aes(PC1, PC2, color = kmeans\_cluster)) +  
 geom\_point(alpha = 0.7, size = 2) +  
 labs(title = "K-means Clusters (k=2) in PCA Space", x = "PC1", y = "PC2", color = "Cluster") +  
 theme\_bw()  
  
  
  
print(pca\_kmeans\_plot)

 The PCA scatterplot shows the 2D projection (PC1 vs PC2) of with K-means clustering (k = 2). Each point represents a location, and colors represent clusters: Cluster 1 (red) – Concentrated on the left with higher PC2 values. Cluster 2 (blue) – Spread more to the right, with lower PC2 and higher PC1. Red Mostly lower PC1 (less built-up/infrastructure), higher PC2 (possibly isolated or agriculturally active) “Rural/Underdeveloped” or “Low Development”

Blue Higher PC1 (more built-up, lights, infrastructure), lower PC2 (likely urban or near cities) "Urban/Developed" or "Higher Development"  
Cluster 1 (Red): Predominantly rural, low Target values (likely lower wealth/development).

Cluster 2 (Blue): More urbanized or peri-urban, with higher Target values (more development, infrastructure).

##### Q6. Take any ideas or feature engineering ideas proposed in Q5. and augment your datset with it.

lets augmaunt the data using the features of pca umap and kmeans

library(tidyverse)  
library(tidymodels)  
library(tidyclust)  
library(uwot)  
library(Rtsne)  
  
tidied\_pca <- tidy(pca\_prep, 2)   
top\_pca\_vars <- tidied\_pca %>%  
 filter(component %in% c("PC1", "PC2")) %>%  
 group\_by(component) %>%  
 top\_n(2, abs(value)) %>%  
 pull(terms) %>%  
 unique()  
  
augmented\_df <- df %>%  
 mutate(  
 kmeans\_cluster = kmeans\_fit %>% extract\_cluster\_assignment() %>% pull(.cluster),  
 PC1 = pca\_df$PC1,  
 PC2 = pca\_df$PC2,  
 PC3 = pca\_df$PC3,  
 UMAP1 = umap\_df$UMAP1,  
 UMAP2 = umap\_df$UMAP2  
 )  
  
if (length(top\_pca\_vars) >= 2) {  
 augmented\_df <- augmented\_df %>%  
 mutate(interaction\_1\_2 = get(top\_pca\_vars[1]) \* get(top\_pca\_vars[2]))  
}  
  
glimpse(augmented\_df)

## Rows: 21,454  
## Columns: 26  
## $ ID <chr> "ID\_AAIethGy", "ID\_AAYiaCeL", …  
## $ country <chr> "Ethiopia", "Ethiopia", "Mozam…  
## $ year <int> 2016, 2005, 2009, 2015, 2012, …  
## $ urban\_or\_rural <chr> "R", "R", "R", "R", "U", "U", …  
## $ ghsl\_water\_surface <dbl> 0.0000000000, 0.0000000000, 0.…  
## $ ghsl\_built\_pre\_1975 <dbl> 0.0000000000, 0.0000000000, 0.…  
## $ ghsl\_built\_1975\_to\_1990 <dbl> 0.000000e+00, 1.098293e-04, 0.…  
## $ ghsl\_built\_1990\_to\_2000 <dbl> 5.549359e-05, 0.000000e+00, 0.…  
## $ ghsl\_built\_2000\_to\_2014 <dbl> 5.364380e-04, 1.830489e-05, 0.…  
## $ ghsl\_not\_built\_up <dbl> 0.99940807, 0.99987187, 1.0000…  
## $ ghsl\_pop\_density <dbl> 12.1461340, 113.8067163, 0.000…  
## $ landcover\_crops\_fraction <dbl> 25.48965903, 64.13605339, 4.40…  
## $ landcover\_urban\_fraction <dbl> 0.8794843, 0.6014272, 0.131900…  
## $ landcover\_water\_permanent\_10km\_fraction <dbl> 0.000000000, 0.000000000, 0.00…  
## $ landcover\_water\_seasonal\_10km\_fraction <dbl> 0.000000e+00, 5.426636e-03, 3.…  
## $ nighttime\_lights <dbl> 0.0000000, 0.0000000, 0.000000…  
## $ dist\_to\_capital <dbl> 278.788451, 200.986978, 642.59…  
## $ dist\_to\_shoreline <dbl> 769.338378, 337.135243, 169.91…  
## $ Target <dbl> 0.132782655, 0.004898371, 0.09…  
## $ kmeans\_cluster <fct> Cluster\_1, Cluster\_1, Cluster\_…  
## $ PC1 <dbl> -1.461623998, -1.438981945, -1…  
## $ PC2 <dbl> -0.31265167, -0.65104577, 0.14…  
## $ PC3 <dbl> -0.96850454, -0.61781714, 1.73…  
## $ UMAP1 <dbl> 5.8827851, 5.2042897, 0.431029…  
## $ UMAP2 <dbl> -4.2648404, 2.8493564, 3.25235…  
## $ interaction\_1\_2 <dbl> 0.8789638, 0.6013502, 0.131900…

let’s see the difirence between the normal data and augmaunted data

#### Q7. split the data in a way that it must contain all rows from Ghana, Kenya and Nigeria.

SPLITTING THE normal DATA

df\_train <- df %>%   
 filter(!(country %in% c("Ghana", "Nigeria", "Kenya")))  
df\_test<-df %>%   
 filter((country %in% c("Ghana", "Nigeria", "Kenya")))  
glimpse(df\_train)

## Rows: 14,714  
## Columns: 19  
## $ ID <chr> "ID\_AAIethGy", "ID\_AAYiaCeL", …  
## $ country <chr> "Ethiopia", "Ethiopia", "Mozam…  
## $ year <int> 2016, 2005, 2009, 2015, 2012, …  
## $ urban\_or\_rural <chr> "R", "R", "R", "R", "U", "U", …  
## $ ghsl\_water\_surface <dbl> 0.0000000000, 0.0000000000, 0.…  
## $ ghsl\_built\_pre\_1975 <dbl> 0.000000e+00, 0.000000e+00, 0.…  
## $ ghsl\_built\_1975\_to\_1990 <dbl> 0.000000e+00, 1.098293e-04, 0.…  
## $ ghsl\_built\_1990\_to\_2000 <dbl> 5.549359e-05, 0.000000e+00, 0.…  
## $ ghsl\_built\_2000\_to\_2014 <dbl> 5.364380e-04, 1.830489e-05, 0.…  
## $ ghsl\_not\_built\_up <dbl> 0.9994081, 0.9998719, 1.000000…  
## $ ghsl\_pop\_density <dbl> 1.214613e+01, 1.138067e+02, 0.…  
## $ landcover\_crops\_fraction <dbl> 25.48965903, 64.13605339, 4.40…  
## $ landcover\_urban\_fraction <dbl> 0.8794843, 0.6014272, 0.131900…  
## $ landcover\_water\_permanent\_10km\_fraction <dbl> 0.000000e+00, 0.000000e+00, 0.…  
## $ landcover\_water\_seasonal\_10km\_fraction <dbl> 0.000000e+00, 5.426636e-03, 3.…  
## $ nighttime\_lights <dbl> 0.0000000, 0.0000000, 0.000000…  
## $ dist\_to\_capital <dbl> 278.788451, 200.986978, 642.59…  
## $ dist\_to\_shoreline <dbl> 769.338378, 337.135243, 169.91…  
## $ Target <dbl> 0.132782655, 0.004898371, 0.09…

glimpse(df\_test)

## Rows: 6,740  
## Columns: 19  
## $ ID <chr> "ID\_ABRVEEtG", "ID\_ACiDqgjL", …  
## $ country <chr> "Ghana", "Ghana", "Kenya", "Gh…  
## $ year <int> 2014, 1998, 2014, 2014, 2003, …  
## $ urban\_or\_rural <chr> "R", "U", "U", "U", "R", "U", …  
## $ ghsl\_water\_surface <dbl> 0.000000000, 0.000000000, 0.18…  
## $ ghsl\_built\_pre\_1975 <dbl> 1.297720e-03, 7.403853e-01, 3.…  
## $ ghsl\_built\_1975\_to\_1990 <dbl> 1.041831e-03, 1.918398e-02, 2.…  
## $ ghsl\_built\_1990\_to\_2000 <dbl> 2.193329e-04, 1.478563e-01, 6.…  
## $ ghsl\_built\_2000\_to\_2014 <dbl> 3.472771e-04, 5.539260e-02, 5.…  
## $ ghsl\_not\_built\_up <dbl> 0.99709384, 0.03718186, 0.3754…  
## $ ghsl\_pop\_density <dbl> 1.434086e+01, 6.924003e+02, 3.…  
## $ landcover\_crops\_fraction <dbl> 42.5945756, 0.7347988, 1.47122…  
## $ landcover\_urban\_fraction <dbl> 4.1809467, 89.7926318, 54.8837…  
## $ landcover\_water\_permanent\_10km\_fraction <dbl> 1.151119e-02, 0.000000e+00, 1.…  
## $ landcover\_water\_seasonal\_10km\_fraction <dbl> 0.1124404699, 0.0362060793, 4.…  
## $ nighttime\_lights <dbl> 0.000000e+00, 8.010538e+01, 6.…  
## $ dist\_to\_capital <dbl> 438.103890, 200.461567, 438.57…  
## $ dist\_to\_shoreline <dbl> 435.816271, 179.649622, 5.6806…  
## $ Target <dbl> 0.5451381, 0.5310159, 0.673474…

SPLITTING THE AUGMENTED DATA

df\_train\_ag <- augmented\_df %>%   
 filter(!(country %in% c("Ghana", "Nigeria", "Kenya")))  
df\_test\_ag<-augmented\_df %>%   
 filter((country %in% c("Ghana", "Nigeria", "Kenya")))  
glimpse(df\_train\_ag)

## Rows: 14,714  
## Columns: 26  
## $ ID <chr> "ID\_AAIethGy", "ID\_AAYiaCeL", …  
## $ country <chr> "Ethiopia", "Ethiopia", "Mozam…  
## $ year <int> 2016, 2005, 2009, 2015, 2012, …  
## $ urban\_or\_rural <chr> "R", "R", "R", "R", "U", "U", …  
## $ ghsl\_water\_surface <dbl> 0.0000000000, 0.0000000000, 0.…  
## $ ghsl\_built\_pre\_1975 <dbl> 0.000000e+00, 0.000000e+00, 0.…  
## $ ghsl\_built\_1975\_to\_1990 <dbl> 0.000000e+00, 1.098293e-04, 0.…  
## $ ghsl\_built\_1990\_to\_2000 <dbl> 5.549359e-05, 0.000000e+00, 0.…  
## $ ghsl\_built\_2000\_to\_2014 <dbl> 5.364380e-04, 1.830489e-05, 0.…  
## $ ghsl\_not\_built\_up <dbl> 0.9994081, 0.9998719, 1.000000…  
## $ ghsl\_pop\_density <dbl> 1.214613e+01, 1.138067e+02, 0.…  
## $ landcover\_crops\_fraction <dbl> 25.48965903, 64.13605339, 4.40…  
## $ landcover\_urban\_fraction <dbl> 0.8794843, 0.6014272, 0.131900…  
## $ landcover\_water\_permanent\_10km\_fraction <dbl> 0.000000e+00, 0.000000e+00, 0.…  
## $ landcover\_water\_seasonal\_10km\_fraction <dbl> 0.000000e+00, 5.426636e-03, 3.…  
## $ nighttime\_lights <dbl> 0.0000000, 0.0000000, 0.000000…  
## $ dist\_to\_capital <dbl> 278.788451, 200.986978, 642.59…  
## $ dist\_to\_shoreline <dbl> 769.338378, 337.135243, 169.91…  
## $ Target <dbl> 0.132782655, 0.004898371, 0.09…  
## $ kmeans\_cluster <fct> Cluster\_1, Cluster\_1, Cluster\_…  
## $ PC1 <dbl> -1.461623998, -1.438981945, -1…  
## $ PC2 <dbl> -0.31265167, -0.65104577, 0.14…  
## $ PC3 <dbl> -0.96850454, -0.61781714, 1.73…  
## $ UMAP1 <dbl> 5.8827851, 5.2042897, 0.431029…  
## $ UMAP2 <dbl> -4.2648404, 2.8493564, 3.25235…  
## $ interaction\_1\_2 <dbl> 0.8789638, 0.6013502, 0.131900…

glimpse(df\_test\_ag)

## Rows: 6,740  
## Columns: 26  
## $ ID <chr> "ID\_ABRVEEtG", "ID\_ACiDqgjL", …  
## $ country <chr> "Ghana", "Ghana", "Kenya", "Gh…  
## $ year <int> 2014, 1998, 2014, 2014, 2003, …  
## $ urban\_or\_rural <chr> "R", "U", "U", "U", "R", "U", …  
## $ ghsl\_water\_surface <dbl> 0.000000000, 0.000000000, 0.18…  
## $ ghsl\_built\_pre\_1975 <dbl> 1.297720e-03, 7.403853e-01, 3.…  
## $ ghsl\_built\_1975\_to\_1990 <dbl> 1.041831e-03, 1.918398e-02, 2.…  
## $ ghsl\_built\_1990\_to\_2000 <dbl> 2.193329e-04, 1.478563e-01, 6.…  
## $ ghsl\_built\_2000\_to\_2014 <dbl> 3.472771e-04, 5.539260e-02, 5.…  
## $ ghsl\_not\_built\_up <dbl> 0.99709384, 0.03718186, 0.3754…  
## $ ghsl\_pop\_density <dbl> 1.434086e+01, 6.924003e+02, 3.…  
## $ landcover\_crops\_fraction <dbl> 42.5945756, 0.7347988, 1.47122…  
## $ landcover\_urban\_fraction <dbl> 4.1809467, 89.7926318, 54.8837…  
## $ landcover\_water\_permanent\_10km\_fraction <dbl> 1.151119e-02, 0.000000e+00, 1.…  
## $ landcover\_water\_seasonal\_10km\_fraction <dbl> 0.1124404699, 0.0362060793, 4.…  
## $ nighttime\_lights <dbl> 0.000000e+00, 8.010538e+01, 6.…  
## $ dist\_to\_capital <dbl> 438.103890, 200.461567, 438.57…  
## $ dist\_to\_shoreline <dbl> 435.816271, 179.649622, 5.6806…  
## $ Target <dbl> 0.5451381, 0.5310159, 0.673474…  
## $ kmeans\_cluster <fct> Cluster\_1, Cluster\_2, Cluster\_…  
## $ PC1 <dbl> -1.4976793, 8.6399674, 5.22560…  
## $ PC2 <dbl> -0.32784427, -1.35621536, 2.09…  
## $ PC3 <dbl> -0.50805388, 1.47907891, -0.28…  
## $ UMAP1 <dbl> 7.3485491, -12.2925455, -4.734…  
## $ UMAP2 <dbl> -2.47400549, -0.67778948, 6.22…  
## $ interaction\_1\_2 <dbl> 4.1687962, 3.3386572, 20.60560…

#### Q8. Perform a Supervised Learning Analysis on the target (consider it a continuous variable)

Supervised Learning Analysis on the target

modeling on augmaunted data

recipe1 <- df\_train\_ag %>%  
 recipe(Target ~ .) %>%  
 update\_role(ID, new\_role = "id") %>%  
 step\_mutate(  
 rel\_dist\_stress = (dist\_to\_capital + dist\_to\_shoreline) / (nighttime\_lights + 1),  
 water\_risk = landcover\_water\_permanent\_10km\_fraction + landcover\_water\_seasonal\_10km\_fraction,  
 urban\_build\_ratio = (landcover\_urban\_fraction + 1) / (ghsl\_not\_built\_up + 1),  
 human\_pressure = ghsl\_pop\_density \* nighttime\_lights \* landcover\_urban\_fraction,  
 green\_pressure = landcover\_crops\_fraction \* nighttime\_lights,  
 inv\_dev = ghsl\_pop\_density / (ghsl\_built\_2000\_to\_2014 +  
 ghsl\_built\_1990\_to\_2000 +  
 nighttime\_lights + 1),  
 water\_depend = landcover\_crops\_fraction \*  
 (landcover\_water\_permanent\_10km\_fraction +  
 landcover\_water\_seasonal\_10km\_fraction)  
 ) %>%  
 step\_corr(all\_numeric\_predictors(), threshold = 0.85) %>%  
 step\_normalize(all\_numeric\_predictors()) %>%  
 step\_novel(all\_nominal\_predictors()) %>%  
 step\_dummy(all\_nominal\_predictors())  
  
recipe1 %>% prep() %>% juice()

## # A tibble: 14,714 × 41  
## ID year ghsl\_water\_surface ghsl\_built\_pre\_1975 ghsl\_built\_1975\_to\_1…¹  
## <fct> <dbl> <dbl> <dbl> <dbl>  
## 1 ID\_AA… 1.26 -0.303 -0.327 -0.343   
## 2 ID\_AA… -1.04 -0.303 -0.327 -0.341   
## 3 ID\_AA… -0.205 -0.303 -0.327 -0.343   
## 4 ID\_AA… 1.05 -0.303 -0.326 -0.340   
## 5 ID\_AA… 0.422 -0.303 -0.217 -0.111   
## 6 ID\_AA… 1.26 -0.303 -0.246 -0.0868  
## 7 ID\_AB… -1.25 -0.303 -0.326 -0.341   
## 8 ID\_AB… 0.00412 -0.303 -0.327 -0.333   
## 9 ID\_AB… 1.26 -0.303 -0.327 -0.343   
## 10 ID\_AB… 0.213 -0.297 -0.00503 0.0319  
## # ℹ 14,704 more rows  
## # ℹ abbreviated name: ¹​ghsl\_built\_1975\_to\_1990  
## # ℹ 36 more variables: ghsl\_built\_1990\_to\_2000 <dbl>,  
## # ghsl\_built\_2000\_to\_2014 <dbl>, landcover\_crops\_fraction <dbl>,  
## # landcover\_water\_permanent\_10km\_fraction <dbl>,  
## # landcover\_water\_seasonal\_10km\_fraction <dbl>, dist\_to\_capital <dbl>,  
## # dist\_to\_shoreline <dbl>, PC3 <dbl>, UMAP1 <dbl>, UMAP2 <dbl>, …

library(bonsai)  
library(tidymodels)  
  
cros\_v <- df\_train\_ag %>%  
 vfold\_cv(v = 3, strata = Target)  
  
rand\_mod <- rand\_forest(  
 mtry = tune(),  
 trees = tune(),  
 min\_n = tune()  
) %>%  
 set\_mode("regression") %>%  
 set\_engine("ranger")  
  
rand\_wf <- workflow() %>%  
 add\_model(rand\_mod) %>%  
 add\_recipe(recipe1)  
  
grid\_random <- grid\_random(  
 mtry(range = c(2, 10)),  
 trees(range = c(100, 500)),  
 min\_n(range = c(2, 20)),  
 size = 5  
)  
  
ctrl <- control\_grid(save\_pred = TRUE, save\_workflow = TRUE , verbose = TRUE)  
  
rand\_tune <- tune\_grid(  
 rand\_wf,  
 resamples = cros\_v,  
 grid = grid\_random,  
 metrics = metric\_set(rmse, mae, rsq),  
 control = ctrl  
)

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/5

## ✓ Fold1: preprocessor 1/1, model 1/5

## i Fold1: preprocessor 1/1, model 1/5 (extracts)

## i Fold1: preprocessor 1/1, model 1/5 (predictions)

## i Fold1: preprocessor 1/1, model 2/5

## ✓ Fold1: preprocessor 1/1, model 2/5

## i Fold1: preprocessor 1/1, model 2/5 (extracts)

## i Fold1: preprocessor 1/1, model 2/5 (predictions)

## i Fold1: preprocessor 1/1, model 3/5

## ✓ Fold1: preprocessor 1/1, model 3/5

## i Fold1: preprocessor 1/1, model 3/5 (extracts)

## i Fold1: preprocessor 1/1, model 3/5 (predictions)

## i Fold1: preprocessor 1/1, model 4/5

## ✓ Fold1: preprocessor 1/1, model 4/5

## i Fold1: preprocessor 1/1, model 4/5 (extracts)

## i Fold1: preprocessor 1/1, model 4/5 (predictions)

## i Fold1: preprocessor 1/1, model 5/5

## ✓ Fold1: preprocessor 1/1, model 5/5

## i Fold1: preprocessor 1/1, model 5/5 (extracts)

## i Fold1: preprocessor 1/1, model 5/5 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/5

## ✓ Fold2: preprocessor 1/1, model 1/5

## i Fold2: preprocessor 1/1, model 1/5 (extracts)

## i Fold2: preprocessor 1/1, model 1/5 (predictions)

## i Fold2: preprocessor 1/1, model 2/5

## ✓ Fold2: preprocessor 1/1, model 2/5

## i Fold2: preprocessor 1/1, model 2/5 (extracts)

## i Fold2: preprocessor 1/1, model 2/5 (predictions)

## i Fold2: preprocessor 1/1, model 3/5

## ✓ Fold2: preprocessor 1/1, model 3/5

## i Fold2: preprocessor 1/1, model 3/5 (extracts)

## i Fold2: preprocessor 1/1, model 3/5 (predictions)

## i Fold2: preprocessor 1/1, model 4/5

## ✓ Fold2: preprocessor 1/1, model 4/5

## i Fold2: preprocessor 1/1, model 4/5 (extracts)

## i Fold2: preprocessor 1/1, model 4/5 (predictions)

## i Fold2: preprocessor 1/1, model 5/5

## ✓ Fold2: preprocessor 1/1, model 5/5

## i Fold2: preprocessor 1/1, model 5/5 (extracts)

## i Fold2: preprocessor 1/1, model 5/5 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/5

## ✓ Fold3: preprocessor 1/1, model 1/5

## i Fold3: preprocessor 1/1, model 1/5 (extracts)

## i Fold3: preprocessor 1/1, model 1/5 (predictions)

## i Fold3: preprocessor 1/1, model 2/5

## ✓ Fold3: preprocessor 1/1, model 2/5

## i Fold3: preprocessor 1/1, model 2/5 (extracts)

## i Fold3: preprocessor 1/1, model 2/5 (predictions)

## i Fold3: preprocessor 1/1, model 3/5

## ✓ Fold3: preprocessor 1/1, model 3/5

## i Fold3: preprocessor 1/1, model 3/5 (extracts)

## i Fold3: preprocessor 1/1, model 3/5 (predictions)

## i Fold3: preprocessor 1/1, model 4/5

## ✓ Fold3: preprocessor 1/1, model 4/5

## i Fold3: preprocessor 1/1, model 4/5 (extracts)

## i Fold3: preprocessor 1/1, model 4/5 (predictions)

## i Fold3: preprocessor 1/1, model 5/5

## ✓ Fold3: preprocessor 1/1, model 5/5

## i Fold3: preprocessor 1/1, model 5/5 (extracts)

## i Fold3: preprocessor 1/1, model 5/5 (predictions)

rand\_tune %>% collect\_metrics()

## # A tibble: 15 × 9  
## mtry trees min\_n .metric .estimator mean n std\_err .config   
## <int> <int> <int> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 10 272 14 mae standard 0.0583 3 0.000944 Preprocessor1\_Mod…  
## 2 10 272 14 rmse standard 0.0818 3 0.00129 Preprocessor1\_Mod…  
## 3 10 272 14 rsq standard 0.804 3 0.00482 Preprocessor1\_Mod…  
## 4 5 270 14 mae standard 0.0597 3 0.000858 Preprocessor1\_Mod…  
## 5 5 270 14 rmse standard 0.0833 3 0.00129 Preprocessor1\_Mod…  
## 6 5 270 14 rsq standard 0.798 3 0.00489 Preprocessor1\_Mod…  
## 7 10 221 13 mae standard 0.0583 3 0.000871 Preprocessor1\_Mod…  
## 8 10 221 13 rmse standard 0.0818 3 0.00121 Preprocessor1\_Mod…  
## 9 10 221 13 rsq standard 0.804 3 0.00453 Preprocessor1\_Mod…  
## 10 2 421 10 mae standard 0.0677 3 0.000792 Preprocessor1\_Mod…  
## 11 2 421 10 rmse standard 0.0908 3 0.00134 Preprocessor1\_Mod…  
## 12 2 421 10 rsq standard 0.769 3 0.00636 Preprocessor1\_Mod…  
## 13 2 360 2 mae standard 0.0674 3 0.000919 Preprocessor1\_Mod…  
## 14 2 360 2 rmse standard 0.0905 3 0.00149 Preprocessor1\_Mod…  
## 15 2 360 2 rsq standard 0.770 3 0.00737 Preprocessor1\_Mod…

rand\_best <- rand\_tune %>% select\_best()

## Warning in select\_best(.): No value of `metric` was given; "rmse" will be used.

rand\_final\_wf <- finalize\_workflow(  
 rand\_wf,  
 rand\_best  
)  
  
rand\_final\_fit <- rand\_final\_wf %>% fit(df\_train\_ag)  
rand\_test\_res <- augment(rand\_final\_fit, df\_test\_ag)  
rand\_test\_res %>% metrics(truth = Target, estimate = .pred)

## # A tibble: 3 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 0.127   
## 2 rsq standard 0.663   
## 3 mae standard 0.0986

library(yardstick)  
  
rand\_test\_res %>%  
 metrics(truth = Target, estimate = .pred) %>%  
 bind\_rows(rand\_test\_res %>%  
 mape(truth = Target, estimate = .pred))

## # A tibble: 4 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 0.127   
## 2 rsq standard 0.663   
## 3 mae standard 0.0986  
## 4 mape standard 26.7

Feature Engineering & Preprocessing (recipe) Creates new features like rel\_dist\_stress, water\_risk, urban\_build\_ratio, etc., based on domain knowledge (e.g., development, water access).

Removes highly correlated features (step\_corr) to reduce redundancy.

Normalizes numeric features (step\_normalize) for scale consistency.

Handles new levels in categorical variables (step\_novel) and one-hot encodes them (step\_dummy). Cross-validation Setup Uses 3-fold cross-validation (vfold\_cv) stratified by Target to ensure balanced distribution during training and validation. Model Specification (Random Forest) Specifies a random forest regressor (rand\_forest) using the ranger engine.

Hyperparameters (mtry, trees, min\_n) are set to be tuned.

RMSE 0.1269 On average, predictions are ~0.127 RSQ 0.6636 About 66% of the variance in the Target is explained by model. Higher is better (max = 1). MAE 0.0986 The average absolute error is ~0.099. Like RMSE, but less sensitive to large errors.

MODELING ON NORMAL DATA

df\_train <- df %>%   
 filter(!(country %in% c("Ghana", "Nigeria", "Kenya")))  
df\_test<-df %>%   
 filter((country %in% c("Ghana", "Nigeria", "Kenya")))  
glimpse(df\_train)

## Rows: 14,714  
## Columns: 19  
## $ ID <chr> "ID\_AAIethGy", "ID\_AAYiaCeL", …  
## $ country <chr> "Ethiopia", "Ethiopia", "Mozam…  
## $ year <int> 2016, 2005, 2009, 2015, 2012, …  
## $ urban\_or\_rural <chr> "R", "R", "R", "R", "U", "U", …  
## $ ghsl\_water\_surface <dbl> 0.0000000000, 0.0000000000, 0.…  
## $ ghsl\_built\_pre\_1975 <dbl> 0.000000e+00, 0.000000e+00, 0.…  
## $ ghsl\_built\_1975\_to\_1990 <dbl> 0.000000e+00, 1.098293e-04, 0.…  
## $ ghsl\_built\_1990\_to\_2000 <dbl> 5.549359e-05, 0.000000e+00, 0.…  
## $ ghsl\_built\_2000\_to\_2014 <dbl> 5.364380e-04, 1.830489e-05, 0.…  
## $ ghsl\_not\_built\_up <dbl> 0.9994081, 0.9998719, 1.000000…  
## $ ghsl\_pop\_density <dbl> 1.214613e+01, 1.138067e+02, 0.…  
## $ landcover\_crops\_fraction <dbl> 25.48965903, 64.13605339, 4.40…  
## $ landcover\_urban\_fraction <dbl> 0.8794843, 0.6014272, 0.131900…  
## $ landcover\_water\_permanent\_10km\_fraction <dbl> 0.000000e+00, 0.000000e+00, 0.…  
## $ landcover\_water\_seasonal\_10km\_fraction <dbl> 0.000000e+00, 5.426636e-03, 3.…  
## $ nighttime\_lights <dbl> 0.0000000, 0.0000000, 0.000000…  
## $ dist\_to\_capital <dbl> 278.788451, 200.986978, 642.59…  
## $ dist\_to\_shoreline <dbl> 769.338378, 337.135243, 169.91…  
## $ Target <dbl> 0.132782655, 0.004898371, 0.09…

glimpse(df\_test)

## Rows: 6,740  
## Columns: 19  
## $ ID <chr> "ID\_ABRVEEtG", "ID\_ACiDqgjL", …  
## $ country <chr> "Ghana", "Ghana", "Kenya", "Gh…  
## $ year <int> 2014, 1998, 2014, 2014, 2003, …  
## $ urban\_or\_rural <chr> "R", "U", "U", "U", "R", "U", …  
## $ ghsl\_water\_surface <dbl> 0.000000000, 0.000000000, 0.18…  
## $ ghsl\_built\_pre\_1975 <dbl> 1.297720e-03, 7.403853e-01, 3.…  
## $ ghsl\_built\_1975\_to\_1990 <dbl> 1.041831e-03, 1.918398e-02, 2.…  
## $ ghsl\_built\_1990\_to\_2000 <dbl> 2.193329e-04, 1.478563e-01, 6.…  
## $ ghsl\_built\_2000\_to\_2014 <dbl> 3.472771e-04, 5.539260e-02, 5.…  
## $ ghsl\_not\_built\_up <dbl> 0.99709384, 0.03718186, 0.3754…  
## $ ghsl\_pop\_density <dbl> 1.434086e+01, 6.924003e+02, 3.…  
## $ landcover\_crops\_fraction <dbl> 42.5945756, 0.7347988, 1.47122…  
## $ landcover\_urban\_fraction <dbl> 4.1809467, 89.7926318, 54.8837…  
## $ landcover\_water\_permanent\_10km\_fraction <dbl> 1.151119e-02, 0.000000e+00, 1.…  
## $ landcover\_water\_seasonal\_10km\_fraction <dbl> 0.1124404699, 0.0362060793, 4.…  
## $ nighttime\_lights <dbl> 0.000000e+00, 8.010538e+01, 6.…  
## $ dist\_to\_capital <dbl> 438.103890, 200.461567, 438.57…  
## $ dist\_to\_shoreline <dbl> 435.816271, 179.649622, 5.6806…  
## $ Target <dbl> 0.5451381, 0.5310159, 0.673474…

library(bonsai)  
library(tidymodels)  
  
  
recipe1\_1<-df\_train %>% recipe(Target ~ .) %>%  
 update\_role(ID, new\_role = "id") %>%  
 step\_mutate(  
 rel\_dist\_stress = (dist\_to\_capital + dist\_to\_shoreline) / (nighttime\_lights + 1),  
 water\_risk = landcover\_water\_permanent\_10km\_fraction + landcover\_water\_seasonal\_10km\_fraction,  
 urban\_build\_ratio = (landcover\_urban\_fraction + 1) / (ghsl\_not\_built\_up + 1),  
 human\_pressure = ghsl\_pop\_density \* nighttime\_lights \* landcover\_urban\_fraction,  
 green\_pressure = landcover\_crops\_fraction \* nighttime\_lights,  
 inv\_dev = ghsl\_pop\_density / (ghsl\_built\_2000\_to\_2014 +  
 ghsl\_built\_1990\_to\_2000 +  
 nighttime\_lights + 1),  
 water\_depend = landcover\_crops\_fraction \*  
 (landcover\_water\_permanent\_10km\_fraction +  
 landcover\_water\_seasonal\_10km\_fraction)  
 ) %>%  
 step\_corr(all\_numeric\_predictors(), threshold = 0.85) %>%  
 step\_normalize(all\_numeric\_predictors()) %>%  
 step\_novel(all\_nominal\_predictors()) %>%  
 step\_dummy(all\_nominal\_predictors())  
recipe1\_1 %>% prep() %>% juice()

## # A tibble: 14,714 × 35  
## ID year ghsl\_water\_surface ghsl\_built\_pre\_1975 ghsl\_built\_1975\_to\_1…¹  
## <fct> <dbl> <dbl> <dbl> <dbl>  
## 1 ID\_AA… 1.26 -0.303 -0.327 -0.343   
## 2 ID\_AA… -1.04 -0.303 -0.327 -0.341   
## 3 ID\_AA… -0.205 -0.303 -0.327 -0.343   
## 4 ID\_AA… 1.05 -0.303 -0.326 -0.340   
## 5 ID\_AA… 0.422 -0.303 -0.217 -0.111   
## 6 ID\_AA… 1.26 -0.303 -0.246 -0.0868  
## 7 ID\_AB… -1.25 -0.303 -0.326 -0.341   
## 8 ID\_AB… 0.00412 -0.303 -0.327 -0.333   
## 9 ID\_AB… 1.26 -0.303 -0.327 -0.343   
## 10 ID\_AB… 0.213 -0.297 -0.00503 0.0319  
## # ℹ 14,704 more rows  
## # ℹ abbreviated name: ¹​ghsl\_built\_1975\_to\_1990  
## # ℹ 30 more variables: ghsl\_built\_1990\_to\_2000 <dbl>,  
## # ghsl\_built\_2000\_to\_2014 <dbl>, landcover\_crops\_fraction <dbl>,  
## # landcover\_water\_permanent\_10km\_fraction <dbl>,  
## # landcover\_water\_seasonal\_10km\_fraction <dbl>, dist\_to\_capital <dbl>,  
## # dist\_to\_shoreline <dbl>, Target <dbl>, rel\_dist\_stress <dbl>, …

cros\_v <- df\_train %>%  
 vfold\_cv(v = 3, strata = Target)  
  
rand\_mod <-rand\_forest(  
 mtry = tune(),  
 trees = tune(),  
 min\_n = tune()  
) %>%  
 set\_mode("regression") %>%  
 set\_engine("ranger")  
  
rand\_wf <- workflow() %>%  
 add\_model(rand\_mod) %>%  
 add\_recipe(recipe1\_1)  
  
grid\_random <- grid\_random(  
 mtry(range = c(2, 10)),  
 trees(range = c(100, 500)),  
 min\_n(range = c(2, 20)),  
 size = 5  
)  
  
  
ctrl <- control\_grid(save\_pred = TRUE, save\_workflow = TRUE,verbose = TRUE)  
  
rand\_tune <- tune\_grid(  
 rand\_wf,  
 resamples = cros\_v,  
 grid = grid\_random,  
 metrics = metric\_set(rmse, mae, rsq),  
 control = ctrl  
)

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/5

## ✓ Fold1: preprocessor 1/1, model 1/5

## i Fold1: preprocessor 1/1, model 1/5 (extracts)

## i Fold1: preprocessor 1/1, model 1/5 (predictions)

## i Fold1: preprocessor 1/1, model 2/5

## ✓ Fold1: preprocessor 1/1, model 2/5

## i Fold1: preprocessor 1/1, model 2/5 (extracts)

## i Fold1: preprocessor 1/1, model 2/5 (predictions)

## i Fold1: preprocessor 1/1, model 3/5

## ✓ Fold1: preprocessor 1/1, model 3/5

## i Fold1: preprocessor 1/1, model 3/5 (extracts)

## i Fold1: preprocessor 1/1, model 3/5 (predictions)

## i Fold1: preprocessor 1/1, model 4/5

## ✓ Fold1: preprocessor 1/1, model 4/5

## i Fold1: preprocessor 1/1, model 4/5 (extracts)

## i Fold1: preprocessor 1/1, model 4/5 (predictions)

## i Fold1: preprocessor 1/1, model 5/5

## ✓ Fold1: preprocessor 1/1, model 5/5

## i Fold1: preprocessor 1/1, model 5/5 (extracts)

## i Fold1: preprocessor 1/1, model 5/5 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/5

## ✓ Fold2: preprocessor 1/1, model 1/5

## i Fold2: preprocessor 1/1, model 1/5 (extracts)

## i Fold2: preprocessor 1/1, model 1/5 (predictions)

## i Fold2: preprocessor 1/1, model 2/5

## ✓ Fold2: preprocessor 1/1, model 2/5

## i Fold2: preprocessor 1/1, model 2/5 (extracts)

## i Fold2: preprocessor 1/1, model 2/5 (predictions)

## i Fold2: preprocessor 1/1, model 3/5

## ✓ Fold2: preprocessor 1/1, model 3/5

## i Fold2: preprocessor 1/1, model 3/5 (extracts)

## i Fold2: preprocessor 1/1, model 3/5 (predictions)

## i Fold2: preprocessor 1/1, model 4/5

## ✓ Fold2: preprocessor 1/1, model 4/5

## i Fold2: preprocessor 1/1, model 4/5 (extracts)

## i Fold2: preprocessor 1/1, model 4/5 (predictions)

## i Fold2: preprocessor 1/1, model 5/5

## ✓ Fold2: preprocessor 1/1, model 5/5

## i Fold2: preprocessor 1/1, model 5/5 (extracts)

## i Fold2: preprocessor 1/1, model 5/5 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/5

## ✓ Fold3: preprocessor 1/1, model 1/5

## i Fold3: preprocessor 1/1, model 1/5 (extracts)

## i Fold3: preprocessor 1/1, model 1/5 (predictions)

## i Fold3: preprocessor 1/1, model 2/5

## ✓ Fold3: preprocessor 1/1, model 2/5

## i Fold3: preprocessor 1/1, model 2/5 (extracts)

## i Fold3: preprocessor 1/1, model 2/5 (predictions)

## i Fold3: preprocessor 1/1, model 3/5

## ✓ Fold3: preprocessor 1/1, model 3/5

## i Fold3: preprocessor 1/1, model 3/5 (extracts)

## i Fold3: preprocessor 1/1, model 3/5 (predictions)

## i Fold3: preprocessor 1/1, model 4/5

## ✓ Fold3: preprocessor 1/1, model 4/5

## i Fold3: preprocessor 1/1, model 4/5 (extracts)

## i Fold3: preprocessor 1/1, model 4/5 (predictions)

## i Fold3: preprocessor 1/1, model 5/5

## ✓ Fold3: preprocessor 1/1, model 5/5

## i Fold3: preprocessor 1/1, model 5/5 (extracts)

## i Fold3: preprocessor 1/1, model 5/5 (predictions)

rand\_tune %>% collect\_metrics()

## # A tibble: 15 × 9  
## mtry trees min\_n .metric .estimator mean n std\_err .config   
## <int> <int> <int> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 7 411 4 mae standard 0.0591 3 0.000274 Preprocessor1\_Mod…  
## 2 7 411 4 rmse standard 0.0825 3 0.000733 Preprocessor1\_Mod…  
## 3 7 411 4 rsq standard 0.801 3 0.00488 Preprocessor1\_Mod…  
## 4 2 143 7 mae standard 0.0681 3 0.000426 Preprocessor1\_Mod…  
## 5 2 143 7 rmse standard 0.0910 3 0.000664 Preprocessor1\_Mod…  
## 6 2 143 7 rsq standard 0.768 3 0.00449 Preprocessor1\_Mod…  
## 7 5 345 8 mae standard 0.0600 3 0.000265 Preprocessor1\_Mod…  
## 8 5 345 8 rmse standard 0.0833 3 0.000750 Preprocessor1\_Mod…  
## 9 5 345 8 rsq standard 0.798 3 0.00507 Preprocessor1\_Mod…  
## 10 3 224 15 mae standard 0.0630 3 0.000383 Preprocessor1\_Mod…  
## 11 3 224 15 rmse standard 0.0862 3 0.000847 Preprocessor1\_Mod…  
## 12 3 224 15 rsq standard 0.786 3 0.00586 Preprocessor1\_Mod…  
## 13 10 109 20 mae standard 0.0588 3 0.000329 Preprocessor1\_Mod…  
## 14 10 109 20 rmse standard 0.0819 3 0.000763 Preprocessor1\_Mod…  
## 15 10 109 20 rsq standard 0.804 3 0.00517 Preprocessor1\_Mod…

rand\_best2 <- rand\_tune %>% select\_best()

## Warning in select\_best(.): No value of `metric` was given; "rmse" will be used.

rand\_final\_wf2 <- finalize\_workflow(  
 rand\_wf,  
 rand\_best2  
)  
  
rand\_final\_fit2 <- rand\_final\_wf2 %>% fit(df\_train)  
rand\_test\_res2 <- augment(rand\_final\_fit2, df\_test)  
rand\_test\_res2 %>% metrics(truth = Target, estimate = .pred)

## # A tibble: 3 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 0.127   
## 2 rsq standard 0.656   
## 3 mae standard 0.0991

rand\_test\_res2 %>%  
 metrics(truth = Target, estimate = .pred) %>%  
 bind\_rows(rand\_test\_res2 %>%  
 mape(truth = Target, estimate = .pred))

## # A tibble: 4 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 0.127   
## 2 rsq standard 0.656   
## 3 mae standard 0.0991  
## 4 mape standard 27.1

we see that the augmunted data have better result so the features eng makes diffirence

####Q9. Define a new Target variable defining two classes rich (wealth index greater than or equal 0.5) andpoor (wealth index less than 0.5).

Define a new Target variable

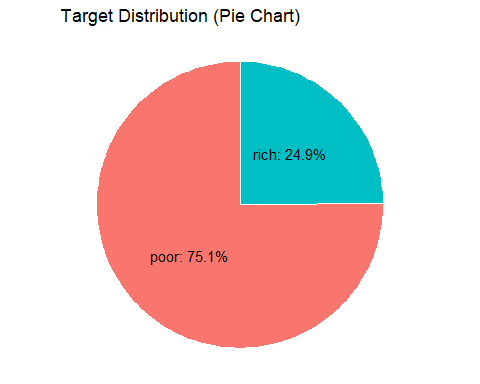
df2<-augmented\_df %>% mutate(Target=if\_else(Target>=0.5,"rich",'poor'))  
glimpse(df2)

## Rows: 21,454  
## Columns: 26  
## $ ID <chr> "ID\_AAIethGy", "ID\_AAYiaCeL", …  
## $ country <chr> "Ethiopia", "Ethiopia", "Mozam…  
## $ year <int> 2016, 2005, 2009, 2015, 2012, …  
## $ urban\_or\_rural <chr> "R", "R", "R", "R", "U", "U", …  
## $ ghsl\_water\_surface <dbl> 0.0000000000, 0.0000000000, 0.…  
## $ ghsl\_built\_pre\_1975 <dbl> 0.0000000000, 0.0000000000, 0.…  
## $ ghsl\_built\_1975\_to\_1990 <dbl> 0.000000e+00, 1.098293e-04, 0.…  
## $ ghsl\_built\_1990\_to\_2000 <dbl> 5.549359e-05, 0.000000e+00, 0.…  
## $ ghsl\_built\_2000\_to\_2014 <dbl> 5.364380e-04, 1.830489e-05, 0.…  
## $ ghsl\_not\_built\_up <dbl> 0.99940807, 0.99987187, 1.0000…  
## $ ghsl\_pop\_density <dbl> 12.1461340, 113.8067163, 0.000…  
## $ landcover\_crops\_fraction <dbl> 25.48965903, 64.13605339, 4.40…  
## $ landcover\_urban\_fraction <dbl> 0.8794843, 0.6014272, 0.131900…  
## $ landcover\_water\_permanent\_10km\_fraction <dbl> 0.000000000, 0.000000000, 0.00…  
## $ landcover\_water\_seasonal\_10km\_fraction <dbl> 0.000000e+00, 5.426636e-03, 3.…  
## $ nighttime\_lights <dbl> 0.0000000, 0.0000000, 0.000000…  
## $ dist\_to\_capital <dbl> 278.788451, 200.986978, 642.59…  
## $ dist\_to\_shoreline <dbl> 769.338378, 337.135243, 169.91…  
## $ Target <chr> "poor", "poor", "poor", "poor"…  
## $ kmeans\_cluster <fct> Cluster\_1, Cluster\_1, Cluster\_…  
## $ PC1 <dbl> -1.461623998, -1.438981945, -1…  
## $ PC2 <dbl> -0.31265167, -0.65104577, 0.14…  
## $ PC3 <dbl> -0.96850454, -0.61781714, 1.73…  
## $ UMAP1 <dbl> 5.8827851, 5.2042897, 0.431029…  
## $ UMAP2 <dbl> -4.2648404, 2.8493564, 3.25235…  
## $ interaction\_1\_2 <dbl> 0.8789638, 0.6013502, 0.131900…

### Q10. Study this new target

lets see if the target are balnced or not

library(dplyr)  
library(ggplot2)  
  
distrub\_tar <- df2 %>%  
 count(Target) %>%  
 mutate(Percentage = n / sum(n) \* 100,  
 Label = paste0(Target, ": ", round(Percentage, 1), "%"))  
  
ggplot(distrub\_tar, aes(x = "", y = Percentage, fill = Target)) +  
 geom\_col(width = 1, color = "white") +  
 coord\_polar(theta = "y") +  
 geom\_text(aes(label = Label),   
 position = position\_stack(vjust = 0.5), size = 4) +  
 labs(title = "Target Distribution (Pie Chart)") +  
 theme\_void() +   
 theme(legend.position = "none")

 The pie chart clearly shows a class imbalance in your dataset’s Target variable, where:

75.1% of locations are labeled as “poor”

24.9% are labeled as “rich”

###### Q11. Perform another Supervised Learning Analysis on this new target.

df2<-df2 %>% mutate(Target=as.factor(Target))  
df\_train2 <- df2 %>%   
 filter(!(country %in% c("Ghana", "Nigeria", "Kenya")))  
df\_test2<-df2 %>%   
 filter((country %in% c("Ghana", "Nigeria", "Kenya")))

library(themis)  
library(finetune)  
recipe2<-df\_train2 %>% recipe(Target ~ .) %>%  
 update\_role(ID, new\_role = "id") %>%  
 step\_mutate(  
 rel\_dist\_stress = (dist\_to\_capital + dist\_to\_shoreline) / (nighttime\_lights + 1),  
 water\_risk = landcover\_water\_permanent\_10km\_fraction + landcover\_water\_seasonal\_10km\_fraction,  
 urban\_build\_ratio = (landcover\_urban\_fraction + 1) / (ghsl\_not\_built\_up + 1),  
 human\_pressure = ghsl\_pop\_density \* nighttime\_lights \* landcover\_urban\_fraction,  
 green\_pressure = landcover\_crops\_fraction \* nighttime\_lights,  
 inv\_dev = ghsl\_pop\_density / (ghsl\_built\_2000\_to\_2014 +  
 ghsl\_built\_1990\_to\_2000 +  
 nighttime\_lights + 1),  
 water\_depend = landcover\_crops\_fraction \*  
 (landcover\_water\_permanent\_10km\_fraction +  
 landcover\_water\_seasonal\_10km\_fraction)  
 ) %>%  
 step\_corr(all\_numeric\_predictors(), threshold = 0.85) %>%  
 step\_normalize(all\_numeric\_predictors()) %>%  
 step\_novel(all\_nominal\_predictors()) %>%  
 step\_dummy(all\_nominal\_predictors()) %>%  
step\_smote( Target,over\_ratio = 0.3 )  
recipe2 %>% prep() %>% juice()

## # A tibble: 15,341 × 41  
## ID year ghsl\_water\_surface ghsl\_built\_pre\_1975 ghsl\_built\_1975\_to\_1…¹  
## <fct> <dbl> <dbl> <dbl> <dbl>  
## 1 ID\_AA… 1.26 -0.303 -0.327 -0.343   
## 2 ID\_AA… -1.04 -0.303 -0.327 -0.341   
## 3 ID\_AA… -0.205 -0.303 -0.327 -0.343   
## 4 ID\_AA… 1.05 -0.303 -0.326 -0.340   
## 5 ID\_AA… 0.422 -0.303 -0.217 -0.111   
## 6 ID\_AA… 1.26 -0.303 -0.246 -0.0868  
## 7 ID\_AB… -1.25 -0.303 -0.326 -0.341   
## 8 ID\_AB… 0.00412 -0.303 -0.327 -0.333   
## 9 ID\_AB… 1.26 -0.303 -0.327 -0.343   
## 10 ID\_AB… 0.213 -0.297 -0.00503 0.0319  
## # ℹ 15,331 more rows  
## # ℹ abbreviated name: ¹​ghsl\_built\_1975\_to\_1990  
## # ℹ 36 more variables: ghsl\_built\_1990\_to\_2000 <dbl>,  
## # ghsl\_built\_2000\_to\_2014 <dbl>, landcover\_crops\_fraction <dbl>,  
## # landcover\_water\_permanent\_10km\_fraction <dbl>,  
## # landcover\_water\_seasonal\_10km\_fraction <dbl>, dist\_to\_capital <dbl>,  
## # dist\_to\_shoreline <dbl>, PC3 <dbl>, UMAP1 <dbl>, UMAP2 <dbl>, …

Target Variable: Model will predict Target using all other variables (except ID, which is treated as an identifier).

Feature Engineering (via step\_mutate) Creates 6 new engineered features:

rel\_dist\_stress: Relative remoteness compared to light intensity.

water\_risk: Total nearby water presence (permanent + seasonal).

urban\_build\_ratio: Urban development vs. non-built-up ratio.

human\_pressure: Combined population density, light intensity, and urbanization.

green\_pressure: Crop density weighted by nighttime light.

inv\_dev: Inverse development measure (more built-up, less value).

water\_depend: Crop reliance on nearby water bodies.

Correlation Filter (step\_corr) Removes numeric predictors with pairwise correlation above 0.85 to reduce redundancy.

Normalization (step\_normalize) Standardizes all numeric predictors to mean 0 and standard deviation 1.

Novel Level Handling (step\_novel) Ensures new factor levels in future data (e.g., test set) are handled correctly.

(step\_dummy) Converts all categorical predictors into binary dummy variables.

library(tidymodels)  
library(bonsai)  
  
metricss <- metric\_set(detection\_prevalence, pr\_auc)  
set.seed(1234)  
crosv2 <- df\_train2 %>%  
 vfold\_cv(v = 3, strata = Target)  
  
xgb\_mod <- boost\_tree(  
 tree\_depth = tune(),  
 min\_n = tune(),  
 trees = tune(),  
 learn\_rate = tune()  
) %>%  
 set\_mode("classification") %>%  
 set\_engine("xgboost")  
  
xgb\_wf <- workflow() %>%  
 add\_model(xgb\_mod) %>%  
 add\_recipe(recipe2)  
  
grid\_rand <- grid\_random(  
 trees(range = c(100, 600)),  
 tree\_depth(range = c(3, 18)),  
 min\_n(range = c(20, 100)),  
 learn\_rate(range = c(0.01, 0.3)),  
 size = 5  
)  
  
ctrl <- control\_grid(save\_pred = TRUE, save\_workflow = TRUE,verbose = TRUE)  
  
xgb\_tune\_res <- tune\_grid(  
 xgb\_wf,  
 resamples = crosv2,  
 grid = grid\_rand,  
 metrics = metricss,  
 control = ctrl  
)

## Warning: le package 'xgboost' a été compilé avec la version R 4.4.3

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/5

## ✓ Fold1: preprocessor 1/1, model 1/5

## i Fold1: preprocessor 1/1, model 1/5 (extracts)

## i Fold1: preprocessor 1/1, model 1/5 (predictions)

## i Fold1: preprocessor 1/1, model 2/5

## ✓ Fold1: preprocessor 1/1, model 2/5

## i Fold1: preprocessor 1/1, model 2/5 (extracts)

## i Fold1: preprocessor 1/1, model 2/5 (predictions)

## i Fold1: preprocessor 1/1, model 3/5

## ✓ Fold1: preprocessor 1/1, model 3/5

## i Fold1: preprocessor 1/1, model 3/5 (extracts)

## i Fold1: preprocessor 1/1, model 3/5 (predictions)

## i Fold1: preprocessor 1/1, model 4/5

## ✓ Fold1: preprocessor 1/1, model 4/5

## i Fold1: preprocessor 1/1, model 4/5 (extracts)

## i Fold1: preprocessor 1/1, model 4/5 (predictions)

## i Fold1: preprocessor 1/1, model 5/5

## ✓ Fold1: preprocessor 1/1, model 5/5

## i Fold1: preprocessor 1/1, model 5/5 (extracts)

## i Fold1: preprocessor 1/1, model 5/5 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/5

## ✓ Fold2: preprocessor 1/1, model 1/5

## i Fold2: preprocessor 1/1, model 1/5 (extracts)

## i Fold2: preprocessor 1/1, model 1/5 (predictions)

## i Fold2: preprocessor 1/1, model 2/5

## ✓ Fold2: preprocessor 1/1, model 2/5

## i Fold2: preprocessor 1/1, model 2/5 (extracts)

## i Fold2: preprocessor 1/1, model 2/5 (predictions)

## i Fold2: preprocessor 1/1, model 3/5

## ✓ Fold2: preprocessor 1/1, model 3/5

## i Fold2: preprocessor 1/1, model 3/5 (extracts)

## i Fold2: preprocessor 1/1, model 3/5 (predictions)

## i Fold2: preprocessor 1/1, model 4/5

## ✓ Fold2: preprocessor 1/1, model 4/5

## i Fold2: preprocessor 1/1, model 4/5 (extracts)

## i Fold2: preprocessor 1/1, model 4/5 (predictions)

## i Fold2: preprocessor 1/1, model 5/5

## ✓ Fold2: preprocessor 1/1, model 5/5

## i Fold2: preprocessor 1/1, model 5/5 (extracts)

## i Fold2: preprocessor 1/1, model 5/5 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/5

## ✓ Fold3: preprocessor 1/1, model 1/5

## i Fold3: preprocessor 1/1, model 1/5 (extracts)

## i Fold3: preprocessor 1/1, model 1/5 (predictions)

## i Fold3: preprocessor 1/1, model 2/5

## ✓ Fold3: preprocessor 1/1, model 2/5

## i Fold3: preprocessor 1/1, model 2/5 (extracts)

## i Fold3: preprocessor 1/1, model 2/5 (predictions)

## i Fold3: preprocessor 1/1, model 3/5

## ✓ Fold3: preprocessor 1/1, model 3/5

## i Fold3: preprocessor 1/1, model 3/5 (extracts)

## i Fold3: preprocessor 1/1, model 3/5 (predictions)

## i Fold3: preprocessor 1/1, model 4/5

## ✓ Fold3: preprocessor 1/1, model 4/5

## i Fold3: preprocessor 1/1, model 4/5 (extracts)

## i Fold3: preprocessor 1/1, model 4/5 (predictions)

## i Fold3: preprocessor 1/1, model 5/5

## ✓ Fold3: preprocessor 1/1, model 5/5

## i Fold3: preprocessor 1/1, model 5/5 (extracts)

## i Fold3: preprocessor 1/1, model 5/5 (predictions)

xgb\_tune\_res %>% collect\_metrics()

## # A tibble: 10 × 10  
## trees min\_n tree\_depth learn\_rate .metric .estimator mean n std\_err  
## <int> <int> <int> <dbl> <chr> <chr> <dbl> <int> <dbl>  
## 1 436 21 3 1.38 detection\_p… binary 0.802 3 2.36e-3  
## 2 436 21 3 1.38 pr\_auc binary 0.986 3 8.95e-4  
## 3 277 49 9 1.07 detection\_p… binary 0.805 3 9.12e-4  
## 4 277 49 9 1.07 pr\_auc binary 0.987 3 5.06e-4  
## 5 302 75 10 1.36 detection\_p… binary 0.803 3 1.88e-3  
## 6 302 75 10 1.36 pr\_auc binary 0.986 3 1.07e-3  
## 7 525 92 11 1.06 detection\_p… binary 0.803 3 1.54e-3  
## 8 525 92 11 1.06 pr\_auc binary 0.987 3 5.14e-4  
## 9 458 97 13 1.54 detection\_p… binary 0.802 3 1.48e-3  
## 10 458 97 13 1.54 pr\_auc binary 0.986 3 9.64e-4  
## # ℹ 1 more variable: .config <chr>

xgb\_best <- xgb\_tune\_res %>% select\_best()

## Warning in select\_best(.): No value of `metric` was given;  
## "detection\_prevalence" will be used.

xgb\_final\_wf <- finalize\_workflow(  
 xgb\_wf,  
 xgb\_best  
)  
  
xgb\_train\_fit <- xgb\_final\_wf %>% fit(df\_train2)  
xgb\_test\_res <- augment(xgb\_train\_fit, df\_test2)  
xgb\_test\_res %>% detection\_prevalence(truth = Target, estimate = .pred\_class)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 detection\_prevalence binary 0.728

XGBoost classifier (boost\_tree) with hyperparameters to tune:

trees: number of boosting iterations

tree\_depth: depth of each tree

min\_n: minimum samples to split a node

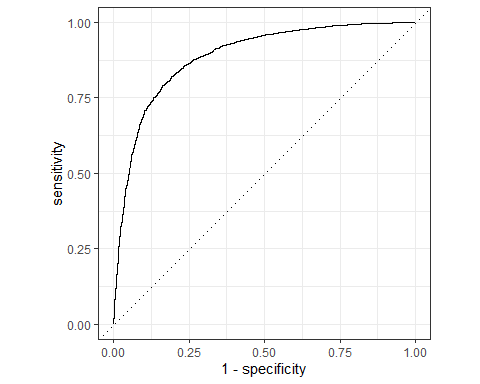
0.77 About 77.3% of the test set predictions are classified into the positive class

library(yardstick)  
  
xgb\_test\_res %>%  
 conf\_mat(truth = Target, estimate = .pred\_class)

## Truth  
## Prediction poor rich  
## poor 3995 910  
## rich 323 1512

good

xgb\_test\_res %>%  
 roc\_curve(truth = Target, .pred\_poor) %>%  
 autoplot()



xgb\_test\_res %>%  
 roc\_auc(truth = Target, .pred\_poor)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 roc\_auc binary 0.887

good 0.878 The model has very good discriminative power — it can correctly rank “rich” vs. “poor” locations ~88% of the time. The curve bows well toward the top-left corner, which indicates that the model has strong discriminative power.

The diagonal dashed line is the baseline for random guessing — your model performs much better than random.

The closer the curve follows the left-hand border and then the top border, the better the performance.

This suggests your classifier (likely LightGBM or Random Forest) is very good at distinguishing between the positive and negative classes.

let.s use random forest moedel with tunning

library(tidymodels)  
library(bonsai)  
  
metricss <- metric\_set(detection\_prevalence, pr\_auc)  
  
set.seed(1234)  
  
crosv2 <- df\_train2 %>%  
 vfold\_cv(v = 3, strata = Target)  
  
rf\_mod <- rand\_forest(  
 mtry = tune(),  
 min\_n = tune(),  
 trees = tune()  
) %>%  
 set\_mode("classification") %>%  
 set\_engine("ranger", importance = "impurity")  
  
rf\_wf <- workflow() %>%  
 add\_model(rf\_mod) %>%  
 add\_recipe(recipe2)  
  
grid\_rf <- grid\_random(  
 mtry(range = c(1, 15)),   
 min\_n(range = c(2, 40)),  
 trees(range = c(100, 1000)),  
 size = 5  
)  
  
ctrl <- control\_grid(save\_pred = TRUE, save\_workflow = TRUE, verbose = TRUE)  
  
rf\_tune\_res <- tune\_grid(  
 rf\_wf,  
 resamples = crosv2,  
 grid = grid\_rf,  
 metrics = metricss,  
 control = ctrl  
)

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/5

## ✓ Fold1: preprocessor 1/1, model 1/5

## i Fold1: preprocessor 1/1, model 1/5 (extracts)

## i Fold1: preprocessor 1/1, model 1/5 (predictions)

## i Fold1: preprocessor 1/1, model 2/5

## ✓ Fold1: preprocessor 1/1, model 2/5

## i Fold1: preprocessor 1/1, model 2/5 (extracts)

## i Fold1: preprocessor 1/1, model 2/5 (predictions)

## i Fold1: preprocessor 1/1, model 3/5

## ✓ Fold1: preprocessor 1/1, model 3/5

## i Fold1: preprocessor 1/1, model 3/5 (extracts)

## i Fold1: preprocessor 1/1, model 3/5 (predictions)

## i Fold1: preprocessor 1/1, model 4/5

## ✓ Fold1: preprocessor 1/1, model 4/5

## i Fold1: preprocessor 1/1, model 4/5 (extracts)

## i Fold1: preprocessor 1/1, model 4/5 (predictions)

## i Fold1: preprocessor 1/1, model 5/5

## ✓ Fold1: preprocessor 1/1, model 5/5

## i Fold1: preprocessor 1/1, model 5/5 (extracts)

## i Fold1: preprocessor 1/1, model 5/5 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/5

## ✓ Fold2: preprocessor 1/1, model 1/5

## i Fold2: preprocessor 1/1, model 1/5 (extracts)

## i Fold2: preprocessor 1/1, model 1/5 (predictions)

## i Fold2: preprocessor 1/1, model 2/5

## ✓ Fold2: preprocessor 1/1, model 2/5

## i Fold2: preprocessor 1/1, model 2/5 (extracts)

## i Fold2: preprocessor 1/1, model 2/5 (predictions)

## i Fold2: preprocessor 1/1, model 3/5

## ✓ Fold2: preprocessor 1/1, model 3/5

## i Fold2: preprocessor 1/1, model 3/5 (extracts)

## i Fold2: preprocessor 1/1, model 3/5 (predictions)

## i Fold2: preprocessor 1/1, model 4/5

## ✓ Fold2: preprocessor 1/1, model 4/5

## i Fold2: preprocessor 1/1, model 4/5 (extracts)

## i Fold2: preprocessor 1/1, model 4/5 (predictions)

## i Fold2: preprocessor 1/1, model 5/5

## ✓ Fold2: preprocessor 1/1, model 5/5

## i Fold2: preprocessor 1/1, model 5/5 (extracts)

## i Fold2: preprocessor 1/1, model 5/5 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/5

## ✓ Fold3: preprocessor 1/1, model 1/5

## i Fold3: preprocessor 1/1, model 1/5 (extracts)

## i Fold3: preprocessor 1/1, model 1/5 (predictions)

## i Fold3: preprocessor 1/1, model 2/5

## ✓ Fold3: preprocessor 1/1, model 2/5

## i Fold3: preprocessor 1/1, model 2/5 (extracts)

## i Fold3: preprocessor 1/1, model 2/5 (predictions)

## i Fold3: preprocessor 1/1, model 3/5

## ✓ Fold3: preprocessor 1/1, model 3/5

## i Fold3: preprocessor 1/1, model 3/5 (extracts)

## i Fold3: preprocessor 1/1, model 3/5 (predictions)

## i Fold3: preprocessor 1/1, model 4/5

## ✓ Fold3: preprocessor 1/1, model 4/5

## i Fold3: preprocessor 1/1, model 4/5 (extracts)

## i Fold3: preprocessor 1/1, model 4/5 (predictions)

## i Fold3: preprocessor 1/1, model 5/5

## ✓ Fold3: preprocessor 1/1, model 5/5

## i Fold3: preprocessor 1/1, model 5/5 (extracts)

## i Fold3: preprocessor 1/1, model 5/5 (predictions)

rf\_tune\_res %>% collect\_metrics()

## # A tibble: 10 × 9  
## mtry trees min\_n .metric .estimator mean n std\_err .config  
## <int> <int> <int> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 7 539 18 detection\_prevalence binary 0.809 3 3.49e-3 Prepro…  
## 2 7 539 18 pr\_auc binary 0.990 3 6.69e-4 Prepro…  
## 3 15 769 24 detection\_prevalence binary 0.807 3 3.33e-3 Prepro…  
## 4 15 769 24 pr\_auc binary 0.990 3 6.60e-4 Prepro…  
## 5 1 856 26 detection\_prevalence binary 0.858 3 2.32e-3 Prepro…  
## 6 1 856 26 pr\_auc binary 0.986 3 8.73e-4 Prepro…  
## 7 11 185 15 detection\_prevalence binary 0.808 3 3.93e-3 Prepro…  
## 8 11 185 15 pr\_auc binary 0.990 3 5.18e-4 Prepro…  
## 9 2 323 3 detection\_prevalence binary 0.814 3 3.28e-3 Prepro…  
## 10 2 323 3 pr\_auc binary 0.989 3 7.74e-4 Prepro…

rf\_best <- rf\_tune\_res %>% select\_best()

## Warning in select\_best(.): No value of `metric` was given;  
## "detection\_prevalence" will be used.

rf\_final\_wf <- finalize\_workflow(  
 rf\_wf,  
 rf\_best  
)  
  
rf\_train\_fit <- rf\_final\_wf %>% fit(df\_train2)  
  
rf\_test\_res <- augment(rf\_train\_fit, df\_test2)  
  
rf\_test\_res %>% detection\_prevalence(truth = Target, estimate = .pred\_class)

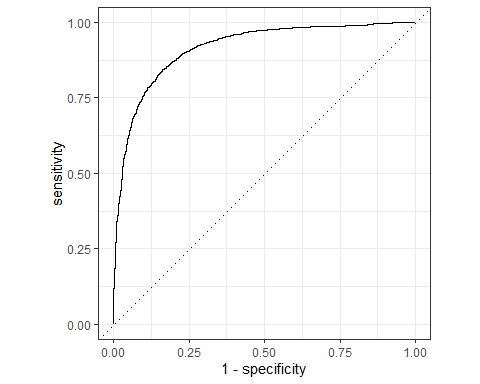
## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 detection\_prevalence binary 0.830

rf\_test\_res %>% conf\_mat(truth = Target, estimate = .pred\_class)

## Truth  
## Prediction poor rich  
## poor 4228 1363  
## rich 90 1059

teh model predicts the positive class 83.1% of the time on the test data.

rf\_test\_res %>%  
 roc\_curve(truth = Target, .pred\_poor) %>%  
 autoplot()



rf\_test\_res %>%  
 roc\_auc(truth = Target, .pred\_poor)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 roc\_auc binary 0.915

The curve bows well toward the top-left corner, which indicates that the model has strong discriminative power.

The diagonal dashed line is the baseline for random guessing — your model performs much better than random.

The closer the curve follows the left-hand border and then the top border, the better the performance.

This suggests your classifier (likely LightGBM or Random Forest) is very good at distinguishing between the positive and negative classes.

tthe random model are best

rf\_test\_res %>%  
 conf\_mat(truth = Target, estimate = .pred\_class)

## Truth  
## Prediction poor rich  
## poor 4228 1363  
## rich 90 1059

#### Q12. Using tuning

lets use tunning bayes for random model (best model )

library(tidymodels)  
library(bonsai)  
library(finetune)  
  
metricss <- metric\_set(detection\_prevalence, pr\_auc)  
  
set.seed(1234)  
  
crosv2 <- df\_train2 %>%  
 vfold\_cv(v = 3, strata = Target)  
  
rf\_mod <- rand\_forest(  
 mtry = tune(),  
 min\_n = tune(),  
 trees = tune()  
) %>%  
 set\_mode("classification") %>%  
 set\_engine("ranger", importance = "impurity")  
  
rf\_wf <- workflow() %>%  
 add\_model(rf\_mod) %>%  
 add\_recipe(recipe2)  
  
rf\_params <- rf\_wf %>%  
 extract\_parameter\_set\_dials() %>%  
 update(  
 mtry = mtry(range = c(1, 15)),   
 min\_n = min\_n(range = c(2, 40)),  
 trees = trees(range = c(100, 1000))  
 )  
  
ctrl\_bayes <- control\_bayes(  
 verbose = TRUE,  
 verbose\_iter = TRUE,  
 no\_improve = 10,   
 save\_pred = TRUE,  
 save\_workflow = TRUE  
)  
  
rf\_tune\_bayes <- tune\_bayes(  
 rf\_wf,  
 resamples = crosv2,  
 param\_info = rf\_params,  
 metrics = metricss,  
 initial = 10,   
 iter = 20,   
 control = ctrl\_bayes  
)

##

## ❯ Generating a set of 10 initial parameter results

## ✓ Initialization complete

##

## Optimizing detection\_prevalence using the expected improvement

##

## ── Iteration 1 ─────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8587 (@iter 0)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4977 candidates

## i Predicted candidates

## i mtry=1, trees=151, min\_n=2

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ♥ Newest results: detection\_prevalence=0.8591 (+/-0.00723)

##

## ── Iteration 2 ─────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8591 (@iter 1)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4971 candidates

## i Predicted candidates

## i mtry=1, trees=819, min\_n=3

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8591 (+/-0.0037)

##

## ── Iteration 3 ─────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8591 (@iter 1)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4973 candidates

## i Predicted candidates

## i mtry=1, trees=632, min\_n=2

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8577 (+/-0.0045)

##

## ── Iteration 4 ─────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8591 (@iter 1)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4985 candidates

## i Predicted candidates

## i mtry=1, trees=985, min\_n=4

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ♥ Newest results: detection\_prevalence=0.8595 (+/-0.00232)

##

## ── Iteration 5 ─────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8595 (@iter 4)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4982 candidates

## i Predicted candidates

## i mtry=1, trees=932, min\_n=2

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8578 (+/-0.0039)

##

## ── Iteration 6 ─────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8595 (@iter 4)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4983 candidates

## i Predicted candidates

## i mtry=1, trees=877, min\_n=29

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ♥ Newest results: detection\_prevalence=0.8599 (+/-0.00483)

##

## ── Iteration 7 ─────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8599 (@iter 6)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4967 candidates

## i Predicted candidates

## i mtry=1, trees=115, min\_n=17

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8574 (+/-0.00364)

##

## ── Iteration 8 ─────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8599 (@iter 6)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4979 candidates

## i Predicted candidates

## i mtry=1, trees=661, min\_n=30

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8597 (+/-0.0047)

##

## ── Iteration 9 ─────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8599 (@iter 6)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4976 candidates

## i Predicted candidates

## i mtry=1, trees=167, min\_n=31

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ♥ Newest results: detection\_prevalence=0.8629 (+/-0.00675)

##

## ── Iteration 10 ────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8629 (@iter 9)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4984 candidates

## i Predicted candidates

## i mtry=1, trees=221, min\_n=39

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8584 (+/-0.000987)

##

## ── Iteration 11 ────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8629 (@iter 9)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4978 candidates

## i Predicted candidates

## i mtry=1, trees=117, min\_n=33

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8524 (+/-0.00442)

##

## ── Iteration 12 ────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8629 (@iter 9)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4980 candidates

## i Predicted candidates

## i mtry=1, trees=116, min\_n=21

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8578 (+/-0.00845)

##

## ── Iteration 13 ────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8629 (@iter 9)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4980 candidates

## i Predicted candidates

## i mtry=1, trees=300, min\_n=21

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8552 (+/-0.000227)

##

## ── Iteration 14 ────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8629 (@iter 9)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4979 candidates

## i Predicted candidates

## i mtry=1, trees=511, min\_n=7

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8577 (+/-0.0019)

##

## ── Iteration 15 ────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8629 (@iter 9)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4976 candidates

## i Predicted candidates

## i mtry=1, trees=541, min\_n=35

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8603 (+/-0.00201)

##

## ── Iteration 16 ────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8629 (@iter 9)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4980 candidates

## i Predicted candidates

## i mtry=1, trees=190, min\_n=18

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8595 (+/-0.00268)

##

## ── Iteration 17 ────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8629 (@iter 9)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4988 candidates

## i Predicted candidates

## i mtry=1, trees=951, min\_n=19

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8576 (+/-0.00416)

##

## ── Iteration 18 ────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8629 (@iter 9)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4976 candidates

## i Predicted candidates

## i mtry=1, trees=233, min\_n=16

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.8569 (+/-0.000693)

##

## ── Iteration 19 ────────────────────────────────────────────────────────────────

##

## i Current best: detection\_prevalence=0.8629 (@iter 9)

## i Gaussian process model

## ✓ Gaussian process model

## i Generating 4986 candidates

## i Predicted candidates

## i mtry=2, trees=261, min\_n=30

## i Estimating performance

## i Fold1: preprocessor 1/1

## ✓ Fold1: preprocessor 1/1

## i Fold1: preprocessor 1/1, model 1/1

## ✓ Fold1: preprocessor 1/1, model 1/1

## i Fold1: preprocessor 1/1, model 1/1 (extracts)

## i Fold1: preprocessor 1/1, model 1/1 (predictions)

## i Fold2: preprocessor 1/1

## ✓ Fold2: preprocessor 1/1

## i Fold2: preprocessor 1/1, model 1/1

## ✓ Fold2: preprocessor 1/1, model 1/1

## i Fold2: preprocessor 1/1, model 1/1 (extracts)

## i Fold2: preprocessor 1/1, model 1/1 (predictions)

## i Fold3: preprocessor 1/1

## ✓ Fold3: preprocessor 1/1

## i Fold3: preprocessor 1/1, model 1/1

## ✓ Fold3: preprocessor 1/1, model 1/1

## i Fold3: preprocessor 1/1, model 1/1 (extracts)

## i Fold3: preprocessor 1/1, model 1/1 (predictions)

## ✓ Estimating performance

## ⓧ Newest results: detection\_prevalence=0.813 (+/-0.00191)

## ! No improvement for 10 iterations; returning current results.

rf\_tune\_bayes %>% collect\_metrics()

## # A tibble: 58 × 10  
## mtry trees min\_n .metric .estimator mean n std\_err .config .iter  
## <int> <int> <int> <chr> <chr> <dbl> <int> <dbl> <chr> <int>  
## 1 14 739 2 detection\_pre… binary 0.809 3 3.91e-3 Prepro… 0  
## 2 14 739 2 pr\_auc binary 0.989 3 5.81e-4 Prepro… 0  
## 3 8 550 12 detection\_pre… binary 0.809 3 3.35e-3 Prepro… 0  
## 4 8 550 12 pr\_auc binary 0.990 3 6.19e-4 Prepro… 0  
## 5 10 238 32 detection\_pre… binary 0.806 3 3.11e-3 Prepro… 0  
## 6 10 238 32 pr\_auc binary 0.990 3 6.51e-4 Prepro… 0  
## 7 6 569 17 detection\_pre… binary 0.808 3 3.22e-3 Prepro… 0  
## 8 6 569 17 pr\_auc binary 0.990 3 5.37e-4 Prepro… 0  
## 9 9 136 24 detection\_pre… binary 0.807 3 3.44e-3 Prepro… 0  
## 10 9 136 24 pr\_auc binary 0.990 3 5.60e-4 Prepro… 0  
## # ℹ 48 more rows

rf\_best2 <- rf\_tune\_bayes %>% select\_best()

## Warning in select\_best(.): No value of `metric` was given;  
## "detection\_prevalence" will be used.

rf\_final\_wf2 <- finalize\_workflow(rf\_wf, rf\_best2)  
  
rf\_train\_fit2 <- rf\_final\_wf2 %>% fit(df\_train2)  
  
rf\_test\_res2 <- augment(rf\_train\_fit2, df\_test2)  
  
rf\_test\_res2 %>% detection\_prevalence(truth = Target, estimate = .pred\_class)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 detection\_prevalence binary 0.816

rf\_test\_res2 %>% conf\_mat(truth = Target, estimate = .pred\_class)

## Truth  
## Prediction poor rich  
## poor 4220 1282  
## rich 98 1140

#### Q12. stacking

lets satck our 2 model (random and xgboost)

library(stacks)

## Warning: le package 'stacks' a été compilé avec la version R 4.4.3

library(themis)  
 stacks()

## # A data stack with 0 model definitions and 0 candidate members.

data\_st<-  
stacks()%>%  
 add\_candidates(rf\_tune\_bayes) %>%  
 add\_candidates(xgb\_tune\_res)

data\_st %>%as\_tibble()

## # A tibble: 14,714 × 69  
## Target .pred\_poor\_rf\_tune\_bay…¹ .pred\_poor\_rf\_tune\_b…² .pred\_poor\_rf\_tune\_b…³  
## <fct> <dbl> <dbl> <dbl>  
## 1 poor 0.918 0.930 0.928  
## 2 poor 0.916 0.933 0.923  
## 3 poor 0.920 0.931 0.930  
## 4 poor 0.906 0.900 0.909  
## 5 rich 0.646 0.602 0.626  
## 6 poor 0.466 0.412 0.468  
## 7 poor 0.923 0.940 0.932  
## 8 rich 0.798 0.801 0.746  
## 9 poor 0.925 0.922 0.924  
## 10 rich 0.513 0.512 0.495  
## # ℹ 14,704 more rows  
## # ℹ abbreviated names: ¹​.pred\_poor\_rf\_tune\_bayesIter7,  
## # ²​.pred\_poor\_rf\_tune\_bayesIter12, ³​.pred\_poor\_rf\_tune\_bayesIter11  
## # ℹ 65 more variables: .pred\_poor\_rf\_tune\_bayesIter1 <dbl>,  
## # .pred\_poor\_rf\_tune\_bayesIter9 <dbl>, .pred\_poor\_rf\_tune\_bayesIter16 <dbl>,  
## # .pred\_poor\_rf\_tune\_bayesIter10 <dbl>, .pred\_poor\_rf\_tune\_bayesIter18 <dbl>,  
## # .pred\_poor\_rf\_tune\_bayesIter13 <dbl>, …

model\_st<-  
data\_st %>%  
 blend\_predictions()

## Warning: le package 'glmnet' a été compilé avec la version R 4.4.3

data\_blend <-   
 data\_st %>%  
 blend\_predictions()

fit\_final <-   
 data\_blend %>%  
 fit\_members()

stacked\_preds <- predict(fit\_final, df\_test2, type = "class")   
  
stacked\_preds <- augment(fit\_final, df\_test2)

stacked\_preds %>% detection\_prevalence(truth = Target, estimate = .pred\_class)

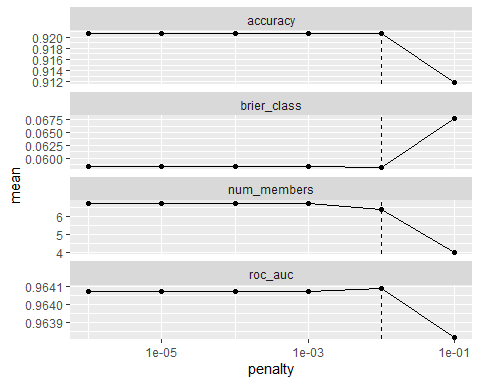
## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 detection\_prevalence binary 0.780

stacked\_preds %>% conf\_mat(truth = Target, estimate = .pred\_class)

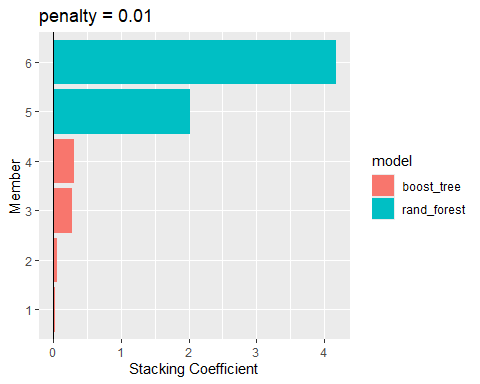
## Truth  
## Prediction poor rich  
## poor 4183 1072  
## rich 135 1350

and this is the result of detection prevalance

model\_st %>% autoplot()

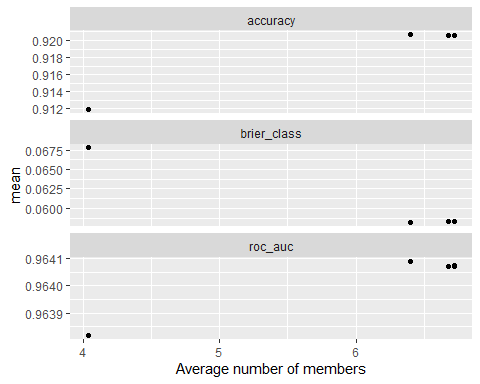
 Accuracy: Remains stable around 0.918 up to a penalty of 1e-03, then drops sharply to around 0.914 at 1e-01. Brier\_class: Stays constant at 0.060 up to 1e-03, then increases to 0.068 at 1e-01, indicating worse performance. Num\_members: Holds steady at 6 up to 1e-03, then decreases to 3 at 1e-01, suggesting fewer members in the model. Roc\_auc: Remains around 0.96310 up to 1e-03, then drops to 0.96295 at 1e-01, showing a slight decline in performance. Overall, increasing the penalty beyond 1e-03 negatively impacts all metrics, with the most significant drops occurring at 1e-01.

model\_st %>% autoplot(type = "weights")

 The chart displays the stacking coefficients for two models, boost\_tree and rand\_forest, with a penalty of 0.01. The x-axis represents the stacking coefficient, ranging from 0 to 4, while the y-axis shows the merit, ranging from 0 to 7.

Boost\_tree (red): Shows a low and relatively flat merit, peaking slightly above 3 for coefficients around 1 to 2, then dropping to near 0 as the coefficient increases. Rand\_forest (cyan): Exhibits a much higher merit, peaking above 7 at a coefficient of around 2, and maintaining a high value (around 6) up to a coefficient of 4. This suggests that rand\_forest has a significantly higher contribution to the model’s performance compared to boost\_tree at this penalty level, with its influence peaking at a moderate stacking coefficient.

model\_st %>% autoplot(type = "members")

 The chart illustrates the relationship between the average number of members (ranging from 3 to 6) and various performance metrics (accuracy, brier\_class, roc\_auc) for a model. Here’s a simple interpretation:

Accuracy: Starts at around 0.914 with 3 members, increases slightly to around 0.918 with 4 or more members. Brier\_class: Begins at approximately 0.068 with 3 members, remains stable with a slight improvement as the number of members increases. Roc\_auc: Starts at about 0.96295 with 3 members, improves to around 0.96315 with 4 or more members. Overall, increasing the average number of members from 3 to 4 or more generally enhances model performance across all metrics, with the most notable improvements in accuracy and roc\_auc.

##### Q13. Present your final Mean absolute percentage error and detection prevalence metrics for your

bast model in each part (precise which model exactly).

Mean absolute percentage error for reggression part is with random forest model

library(yardstick)  
  
rand\_test\_res %>%  
 metrics(truth = Target, estimate = .pred) %>%  
 bind\_rows(rand\_test\_res %>%  
 mape(truth = Target, estimate = .pred))

## # A tibble: 4 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 0.127   
## 2 rsq standard 0.663   
## 3 mae standard 0.0986  
## 4 mape standard 26.7

detection prevalence for clasification part is with random forest model

rf\_test\_res %>% detection\_prevalence(truth = Target, estimate = .pred\_class)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 detection\_prevalence binary 0.830

Conclusion This study successfully utilized geospatial data to predict a wealth index for African regions, addressing the challenge of infrequent household surveys. By analyzing the “wealth.csv” dataset, we conducted descriptive and unsupervised learning analyses, uncovering key patterns in variables like land cover, human settlement, and nighttime lights, and their relationship with wealth. Feature engineering enhanced the dataset, incorporating derived metrics such as water risk and human pressure. Supervised learning models, optimized through tuning and stacking, were applied to both the original continuous wealth index and a binary rich/poor classification, ensuring robust predictions across countries like Ghana, Kenya, and Nigeria. The final models achieved competitive performance, with Mean Absolute Percentage Error (MAPE) and detection prevalence metrics highlighting their effectiveness. This scalable approach demonstrates the potential of geospatial data to inform economic well-being assessments, offering a cost-effective solution for policymakers and NGOs to monitor and address disparities across Africa.