

DATA 201 Final Appendix B

Olivia Yuengling

2024-12-16

Appendix B

Data Cleaning and Processing

```
# Loads any necessary libraries
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.3.3
## Warning: package 'ggplot2' was built under R version 4.3.3

## — Attaching core tidyverse packages —————— tidyverse 2.0.0 —
## ✓ dplyr     1.1.4      ✓ readr     2.1.5
## ✓forcats   1.0.0      ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1      ✓ tibble    3.2.1
## ✓ lubridate 1.9.3      ✓ tidyr    1.3.0
## ✓ purrr    1.0.2
## — Conflicts —————— tidyverse_conflict
s() —
## X dplyr::filter() masks stats::filter()
## X dplyr::lag()   masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors

library(dplyr)
library(ggplot2)
library(GGally)

## Warning: package 'GGally' was built under R version 4.3.3

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

library(ISLR2)

## Warning: package 'ISLR2' was built under R version 4.3.3

library(caret)

## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##   lift

library(gplots)

## Warning: package 'gplots' was built under R version 4.3.3

##
## Attaching package: 'gplots'
##
## The following object is masked from 'package:stats':
##   lowess

library(tidyr)
library(mlbench)

## Warning: package 'mlbench' was built under R version 4.3.3

# Loads the dataset

# install.packages("tidytuesdayR")

tuesdata <- tidytuesdayR::tt_load('2024-10-22')

## ---- Compiling #TidyTuesday Information for 2024-10-22 ----
## --- There is 1 file available ---
##
## — Downloading files ——————
_____
##   1 of 1: "cia_factbook.csv"

## OR
tuesdata <- tidytuesdayR::tt_load(2024, week = 43)

## ---- Compiling #TidyTuesday Information for 2024-10-22 ----
## --- There is 1 file available ---
##
## — Downloading files ——————
_____
##   1 of 1: "cia_factbook.csv"

cia_factbook <- tuesdata$cia_factbook
```

```
# displays the rows and columns of the dataset
head(cia_factbook)

## # A tibble: 6 × 11
##   country           area birth_rate death_rate infant_mortality_rate internet_
##   <chr>          <dbl>     <dbl>      <dbl>            <dbl>
## 1 Russia        1.71e7    11.9       13.8            7.08    40
## 2 Canada        9.98e6    10.3       8.31            4.71    26
## 3 United States 9.83e6    13.4       8.15            6.17    245
## 4 China         9.60e6    12.2       7.44           14.8    389
## 5 Brazil        8.51e6    14.7       6.54            19.2    75
## 6 Australia     7.74e6    12.2       7.07            4.43    15
## # i 5 more variables: life_exp_at_birth <dbl>, maternal_mortality_rate <dbl>,
## #   net_migration_rate <dbl>, population <dbl>, population_growth_rate <dbl>

str(cia_factbook)

## #> #> #> spc_tbl_ [259 × 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
## #> #> $ country           : chr [1:259] "Russia" "Canada" "United States"
## #> #> "China" ...
## #> #> $ area              : num [1:259] 17098242 9984670 9826675 9596960 8
## #> #> 514877 ...
## #> #> $ birth_rate        : num [1:259] 11.9 10.3 13.4 12.2 14.7 ...
## #> #> $ death_rate         : num [1:259] 13.83 8.31 8.15 7.44 6.54 ...
## #> #> $ infant_mortality_rate: num [1:259] 7.08 4.71 6.17 14.79 19.21 ...
## #> #> $ internet_users     : num [1:259] 4.09e+07 2.70e+07 2.45e+08 3.89e+0
## #> #> 8 7.60e+07 ...
## #> #> $ life_exp_at_birth  : num [1:259] 70.2 81.7 79.6 75.2 73.3 ...
## #> #> $ maternal_mortality_rate: num [1:259] 34 12 21 37 56 7 200 77 51 97 ...
## #> #> $ net_migration_rate  : num [1:259] 1.69 5.66 2.45 -0.32 -0.15 5.74 -0
## #> #> .05 0 0.42 -0.93 ...
## #> #> $ population         : num [1:259] 1.42e+08 3.48e+07 3.19e+08 1.36e+0
## #> #> 9 2.03e+08 ...
## #> #> $ population_growth_rate: num [1:259] -0.03 0.76 0.77 0.44 0.8 1.09 1.25
## #> #> 0.95 1.17 1.88 ...
## #> - attr(*, "spec")=
## #> .. cols(
## #>   .. country = col_character(),
## #>   .. area = col_double(),
## #>   .. birth_rate = col_double(),
```

```

## .. death_rate = col_double(),
## .. infant_mortality_rate = col_double(),
## .. internet_users = col_double(),
## .. life_exp_at_birth = col_double(),
## .. maternal_mortality_rate = col_double(),
## .. net_migration_rate = col_double(),
## .. population = col_double(),
## .. population_growth_rate = col_double()
## ..
## - attr(*, "problems")=<externalptr>

summary(cia_factbook) # prints a 5 number summary of the dataset

##   country           area      birth_rate      death_rate
## Length:259      Min.    : 0      Min.    : 6.72      Min.    : 1.530
## Class :character 1st Qu.: 616    1st Qu.:11.84    1st Qu.: 5.930
## Mode  :character Median :51197   Median :16.89      Median : 7.630
##                   Mean  :530888   Mean   :19.66      Mean   : 7.907
##                   3rd Qu.:338145   3rd Qu.:24.91      3rd Qu.: 9.450
##                   Max.  :17098242   Max.   :46.12      Max.   :17.490
##                   NA's  :2        NA's   :35        NA's   :34
##   infant_mortality_rate internet_users      life_exp_at_birth
## Min.    : 1.810      Min.    : 464      Min.    :49.44
## 1st Qu.: 6.185      1st Qu.: 86400   1st Qu.:67.00
## Median :13.985      Median : 716400   Median :74.36
## Mean   :24.484      Mean   : 8311771  Mean   :71.83
## 3rd Qu.:38.655      3rd Qu.: 4200000 3rd Qu.:78.29
## Max.   :117.230     Max.   :3890000000 Max.   :89.57
## NA's   :35          NA's   :46        NA's   :35
##   maternal_mortality_rate net_migration_rate      population
## Min.    : 2.0        Min.    :-113.5100    Min.    :4.800e+01
## 1st Qu.: 20.0       1st Qu.: -2.0150    1st Qu.:3.266e+05
## Median : 65.5       Median : -0.0450    Median :5.220e+06
## Mean   :178.0       Mean   : -0.1816    Mean   :3.229e+07
## 3rd Qu.:240.0       3rd Qu.:  1.2575    3rd Qu.:1.826e+07
## Max.   :2054.0      Max.   : 83.8200    Max.   :1.356e+09
## NA's   :75          NA's   :37        NA's   :21
##   population_growth_rate
## Min.    :-9.730
## 1st Qu.: 0.260
## Median : 1.020
## Mean   : 1.101
## 3rd Qu.: 1.920
## Max.   : 9.370
## NA's   :26

dim(cia_factbook) # prints dimensions of dataset

## [1] 259  11

```

Creating the Classification/Binary Variable

As stated in the beginning of the project, we want to observe if we can classify whether a country is third world or not. Unfortunately, the dataset does not come with a variable for this so we will have to code it ourselves.

We will create a vector list of all of the countries that have been classified as underdeveloped in 2014 according to the United Nations (The Least Developed Countries Report 2014 | Department of Economic and Social Affairs. (2014). Un.org.

<https://sdgs.un.org/publications/least-developed-countries-report-2014-17949>). After that we will use the mutate function from the r package dplyr to create our target classification variable “third_world”.

```
undeveloped_countries <- c(
  "Afghanistan", "Angola", "Bangladesh", "Benin", "Bhutan", "Burkina Faso", "Burundi", "Cambodia",
  "Central African Republic", "Chad", "Comoros", "Democratic Republic of the Congo", "Djibouti",
  "Equatorial Guinea", "Eritrea", "Ethiopia", "The Gambia", "Guinea", "Guinea-Bissau", "Haiti",
  "Kiribati", "Lao People's Democratic Republic", "Lesotho", "Liberia", "Madagascar", "Malawi",
  "Mali", "Mauritania", "Mozambique", "Myanmar", "Nepal", "Niger", "Rwanda",
  "Sao Tome and Principe",
  "Senegal", "Sierra Leone", "Solomon Islands", "Somalia", "South Sudan", "Sudan", "Timor-Leste",
  "Togo", "Tuvalu", "Uganda", "United Republic of Tanzania", "Vanuatu", "Yemen", "Zambia"
) # creates a vector of a list of underdeveloped countries, derived from the UN

cia_factbook <- cia_factbook %>% # Loads mutate command into dataset
  mutate(third_world = ifelse(country %in% undeveloped_countries, # creates a new binary variable of 3rd world status
    1, # undeveloped country
    0)) # developed country
```

Handling Dataset NA's

```
# removes rows with more than 4 NA's
row_na_count <- rowSums(is.na(cia_factbook)) # counts the number of NA's in each row
limit <- 4 # sets the threshold
cia_factbook <- cia_factbook[row_na_count <= limit, ] # removes rows with more than 4 NA's
```

For the remaining NA's in the data we will use mean or median imputation depending if the data is skewed or normally distributed based on the shape from the histograms. The histograms with a normal distribution are net migration rate and population growth rate, so

we will use mean imputation because the mean is not as influenced by outliers compared to the other values in the dataset.

```
cia_factbook$net_migration_rate[is.na(cia_factbook$net_migration_rate)] <-  
  mean(cia_factbook$net_migration_rate, na.rm = TRUE)  
  
cia_factbook$population_growth_rate[is.na(cia_factbook$population_growth_rate)] <- mean(cia_factbook$net_migration_rate, na.rm = TRUE)  
  
summary(cia_factbook) # prints summary to confirm there are no NA's in normally distributed variables  
  
##      country           area       birth_rate     death_rate  
##  Length:224      Min.   :    2   Min.   : 6.72   Min.   : 1.530  
##  Class :character  1st Qu.: 5836   1st Qu.:11.84   1st Qu.: 5.930  
##  Mode  :character  Median : 87971   Median :16.89   Median : 7.540  
##                      Mean   : 608449   Mean   :19.66   Mean   : 7.907  
##                      3rd Qu.: 448124   3rd Qu.:24.91   3rd Qu.: 9.457  
##                      Max.   :17098242   Max.   :46.12   Max.   :17.490  
##  
##  infant_mortality_rate internet_users      life_exp_at_birth  
##  Min.   : 1.81      Min.   : 900      Min.   :49.44  
##  1st Qu.: 6.20      1st Qu.: 95000    1st Qu.:66.90  
##  Median :14.00      Median : 746000    Median :74.29  
##  Mean   :24.57      Mean   : 8470823   Mean   :71.76  
##  3rd Qu.:38.70      3rd Qu.: 4393000   3rd Qu.:78.28  
##  Max.   :117.23     Max.   :389000000  Max.   :89.57  
##  NA's   :1          NA's   :15       NA's   :2  
##  maternal_mortality_rate net_migration_rate population  
##  Min.   : 2.0        Min.   :-113.5100  Min.   :5.215e+03  
##  1st Qu.: 20.0       1st Qu.: -2.0050  1st Qu.:5.843e+05  
##  Median : 65.5       Median : -0.0700  Median :5.617e+06  
##  Mean   :178.0       Mean   : -0.1881  Mean   :3.202e+07  
##  3rd Qu.:240.0       3rd Qu.:  1.2200  3rd Qu.:2.176e+07  
##  Max.   :2054.0      Max.   : 83.8200  Max.   :1.356e+09  
##  NA's   :40  
##  population_growth_rate third_world  
##  Min.   :-9.730      Min.   :0.000  
##  1st Qu.: 0.330      1st Qu.:0.000  
##  Median : 1.075      Median :0.000  
##  Mean   : 1.140      Mean   :0.192  
##  3rd Qu.: 1.923      3rd Qu.:0.000  
##  Max.   : 9.370      Max.   :1.000  
##
```

Looking at the summaries now, we can now see that there are no NA's for the normally distributed variables in the dataset. Now, let's do the remainder of the numerical variables but with their respective median value.

```

# median imputation

cia_factbook$net_migration_rate[is.na(cia_factbook$infant_mortality_rate)] <-
  median(cia_factbook$infant_mortality_rate, na.rm = TRUE)

cia_factbook$infant_mortality_rate[is.na(cia_factbook$infant_mortality_rate)] <-
  median(cia_factbook$infant_mortality_rate, na.rm = TRUE)

cia_factbook$internet_users[is.na(cia_factbook$internet_users)] <-
  median(cia_factbook$internet_users, na.rm = TRUE)

cia_factbook$life_exp_at_birth[is.na(cia_factbook$life_exp_at_birth)] <-
  median(cia_factbook$life_exp_at_birth, na.rm = TRUE)

cia_factbook$maternal_mortality_rate[is.na(cia_factbook$maternal_mortality_rate)] <-
  median(cia_factbook$maternal_mortality_rate, na.rm = TRUE)

summary(cia_factbook) # prints summary to confirm there are no NA's

##      country           area      birth_rate      death_rate
##  Length:224      Min.   :     2      Min.   : 6.72      Min.   : 1.530
##  Class :character 1st Qu.: 5836    1st Qu.:11.84    1st Qu.: 5.930
##  Mode  :character  Median : 87971   Median :16.89    Median : 7.540
##                      Mean   : 608449   Mean   :19.66    Mean   : 7.907
##                      3rd Qu.: 448124   3rd Qu.:24.91    3rd Qu.: 9.457
##                      Max.   :17098242   Max.   :46.12    Max.   :17.490
##  infant_mortality_rate internet_users      life_exp_at_birth
##  Min.   : 1.810      Min.   :    900      Min.   :49.44
##  1st Qu.: 6.205      1st Qu.: 113150    1st Qu.:67.00
##  Median :14.000      Median : 746000    Median :74.29
##  Mean   :24.528      Mean   : 7953536   Mean   :71.78
##  3rd Qu.:38.655      3rd Qu.: 4012750   3rd Qu.:78.25
##  Max.   :117.230     Max.   :389000000  Max.   :89.57
##  maternal_mortality_rate net_migration_rate      population
##  Min.   : 2.00        Min.   :-113.5100    Min.   :5.215e+03
##  1st Qu.: 26.75       1st Qu.: -2.0050    1st Qu.:5.843e+05
##  Median : 65.50       Median : -0.0550    Median :5.617e+06
##  Mean   :157.89       Mean   : -0.1248    Mean   :3.202e+07
##  3rd Qu.:200.00       3rd Qu.:  1.2500    3rd Qu.:2.176e+07
##  Max.   :2054.00      Max.   : 83.8200    Max.   :1.356e+09
##  population_growth_rate third_world
##  Min.   :-9.730       Min.   :0.000
##  1st Qu.: 0.330       1st Qu.:0.000
##  Median : 1.075       Median :0.000
##  Mean   : 1.140       Mean   :0.192
##  3rd Qu.: 1.923       3rd Qu.:0.000
##  Max.   : 9.370       Max.   :1.000

```

Now, there are no NA's in the dataset. Let's now analyze our target variable, net migration, in more detail.

Creating the Baseline Classification Model

```
cia_factbook_no_country <- cia_factbook %>%
  select(-country)

model2 <- glm(third_world ~ .,
              data = cia_factbook_no_country,
              family = "binomial")

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(model2)

##
## Call:
## glm(formula = third_world ~ ., family = "binomial", data = cia_factbook_no_
## _country)
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)               5.491e-01  1.398e+01   0.039  0.96867
## area                  -1.682e-06  8.173e-07  -2.058  0.03960 *
## birth_rate                1.579e-01  1.070e-01   1.475  0.14016
## death_rate                -5.137e-01  3.714e-01  -1.383  0.16670
## infant_mortality_rate    1.006e-01  3.573e-02   2.815  0.00488 **
## internet_users            -2.335e-07  2.893e-07  -0.807  0.41972
## life_exp_at_birth        -5.889e-02  1.586e-01  -0.371  0.71048
## maternal_mortality_rate  1.059e-03  2.098e-03   0.504  0.61392
## net_migration_rate        6.638e-02  9.607e-02   0.691  0.48957
## population                -4.869e-09  9.175e-09  -0.531  0.59563
## population_growth_rate   -6.594e-01  8.911e-01  -0.740  0.45929
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 219.098 on 223 degrees of freedom
## Residual deviance: 80.613 on 213 degrees of freedom
## AIC: 102.61
##
## Number of Fisher Scoring iterations: 11
```

The initial logistic regression model didn't converge, which means it struggled to make accurate predictions due to the complexity of the data. Several variables, including birth rate, death rate, and population growth rate, showed very large standard errors and p-

values, indicating they weren't adding much to the model. The coefficients were unstable, which led to unreliable results.

```
library(car)

## Warning: package 'car' was built under R version 4.3.3

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.3.3

## 
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
## 
##     recode

## The following object is masked from 'package:purrr':
## 
##     some

vif(glm(third_world ~ ., data = cia_factbook_no_country, family = "binomial"))
)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##           area          birth_rate        death_rate
##           1.723803       8.150252      13.663896
## infant_mortality_rate    internet_users life_exp_at_birth
##           5.511132       1.101262      17.044186
## maternal_mortality_rate net_migration_rate population
##           2.596951       6.063134      1.346347
## population_growth_rate
##           11.477789
```

I simplified the model by removing less relevant variables like infant mortality rate and internet users. This cleaner version included death rate, net migration rate, population, and population growth rate. However, the model still didn't converge, suggesting there might still be multicollinearity or other issues affecting the accuracy of predictions. To improve this, I could look into variable correlations or apply regularization techniques.

```
model3 <- glm(third_world ~ death_rate + net_migration_rate + population + po
pulation_growth_rate,
                 data = cia_factbook_no_country,
                 family = "binomial")
summary(model3)

##
## Call:
## glm(formula = third_world ~ death_rate + net_migration_rate +
##     population + population_growth_rate, family = "binomial",
```

```

##      data = cia_factbook_no_country)
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -7.231e+00  9.827e-01 -7.358 1.87e-13 ***
## death_rate            2.778e-01  8.425e-02  3.297 0.000977 ***
## net_migration_rate   -2.275e-01  4.233e-02 -5.374 7.68e-08 ***
## population            -1.219e-08  8.714e-09 -1.399 0.161960
## population_growth_rate 2.308e+00  3.507e-01  6.580 4.71e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 219.10  on 223  degrees of freedom
## Residual deviance: 106.24  on 219  degrees of freedom
## AIC: 116.24
##
## Number of Fisher Scoring iterations: 7

```

Let's break down these results. The significant predictors of the model are death rate, net migration rate, and population growth rate. A higher death and population rate increases the likelihood of a country being underdeveloped. A higher migration rate out of a country would lower the likelihood of a country being a third world country. Also, the AIC value of 116.24 indicates the model is more effective than the null model, or the assumption that there is no relationship with the predictors and the outcome.

However, one of the variables is statistically insignificant (population) and we will remove this variable so we could produce a more robust classification model. There may need to be a more efficient version of the model which is supported by the number of Fisher Scoring iterations (7), indicating the model is potentially unstable and the algorithm did not fully converge.

We will try to produce a more robust model by removing the population variable.

```

model4 <- glm(third_world ~ death_rate + net_migration_rate
               + population_growth_rate ,
               data = cia_factbook_no_country,
               family = "binomial")
summary(model4)

##
## Call:
## glm(formula = third_world ~ death_rate + net_migration_rate +
##     population_growth_rate, family = "binomial", data = cia_factbook_no_co
## untry)
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -7.2772    0.9693 -7.508 6.03e-14 ***

```

```

## death_rate           0.2648    0.0827   3.201  0.00137 ***
## net_migration_rate -0.2245    0.0415   -5.411 6.27e-08 ***
## population_growth_rate 2.2672    0.3425   6.620  3.59e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 219.10  on 223  degrees of freedom
## Residual deviance: 108.64  on 220  degrees of freedom
## AIC: 116.64
##
## Number of Fisher Scoring iterations: 6

```

As observed from the output, the number of Fisher Scoring Iterations has slightly improved signaling that the model is slightly more stable. Additionally, the p-values for the model have slightly improved indicating that the predictors are more strongly associated with whether a country is third world or not. Furthermore, the AIC (116.64) is slightly higher than the previous model's (116.24). Typically models with lower AICs are superior to those with higher AICs, but given the significant coefficients and the better Fisher Scoring Interactions this model is overall more robust, but the difference between the models is quite minimal.

```

car::vif(model4)

##              death_rate      net_migration_rate population_growth_rate
## 1.000924                  3.279119                  3.277452

```

To check for multicollinearity, we will use the car package and the VIF function to identify if there are any multicollinearities. The general rule of thumb is to make sure that the values the function produces are under 5-10, which all of these values do pass. Therefore there is not any multicollinearities to worry about.

```
log_model <- model4 # renames the model to log_model
```

Testing and Training the Logistic Classification Model

```

# Train/Test Split
set.seed(1231)
sample <- sample(c(TRUE, FALSE), size = nrow(cia_factbook_no_country), replace = TRUE, prob = c(0.8, 0.2))
train <- cia_factbook_no_country[sample, ]
test <- cia_factbook_no_country[!sample, ]

# Convert 'third_world' to factor in both train and test datasets
train$third_world <- factor(train$third_world, levels = c(0, 1))
test$third_world <- factor(test$third_world, levels = c(0, 1))

log_model <- train(

```

```

third_world ~ death_rate + birth_rate + population_growth_rate, # Formula
for prediction
  data = train, # Use training data for fitting the model
  method = "glm", # Generalized Linear Model
  family = "binomial", # Logistic regression for binary classification
  trControl = trainControl(method = "cv") # Cross-validation for model selection
)

# Predictions and Confusion Matrix for Logistic Regression
log_preds <- predict(log_model, newdata = test, type = "raw") # Change type
to "raw" for class labels

# Confusion Matrix
conf_matrix_log <- confusionMatrix(factor(log_preds), factor(test$third_world
)) # Make sure both are factors
print(conf_matrix_log)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction  0   1
##           0 33  2
##           1   1  7
##
##                 Accuracy : 0.9302
##                           95% CI : (0.8094, 0.9854)
##     No Information Rate : 0.7907
##     P-Value [Acc > NIR] : 0.01253
##
##                 Kappa : 0.7802
##
## McNemar's Test P-Value : 1.00000
##
##                 Sensitivity : 0.9706
##                 Specificity : 0.7778
##     Pos Pred Value : 0.9429
##     Neg Pred Value : 0.8750
##                 Prevalence : 0.7907
##     Detection Rate : 0.7674
## Detection Prevalence : 0.8140
##     Balanced Accuracy : 0.8742
##
##     'Positive' Class : 0
##

# manually extract confusion matrix components
cm <- conf_matrix_log$table
TP <- cm[2, 2] # True Positives
TN <- cm[1, 1] # True Negatives

```

```

FP <- cm[1, 2] # False Positives
FN <- cm[2, 1] # False Negatives

# manually calculates precision, recall, and F1 Score
precision_log <- TP / (TP + FP) # Precision
recall_log <- TP / (TP + FN) # Recall (Sensitivity)
f1_log <- 2 * (precision_log * recall_log) / (precision_log + recall_log) # F1 Score

# Display the metrics
print("Percision")
## [1] "Percision"
precision_log
## [1] 0.7777778

print("Recall")
## [1] "Recall"
recall_log
## [1] 0.875

print("F1")
## [1] "F1"
f1_log
## [1] 0.8235294

```

The Logistic Regression model is performing really well. With an accuracy of 93.02%, it's doing much better than just guessing, since the No Information Rate (NIR) is only 79.07%. The Kappa score of 0.7802 shows that the predictions are in good agreement with the actual outcomes, beyond chance.

In terms of performance, the model's sensitivity is excellent at 97.06%, meaning it's great at picking up third-world countries. However, its specificity is a bit lower at 77.78%, indicating it's not as good at identifying non-third-world countries. The precision (positive predictive value) is pretty high at 94.29%, so when the model predicts a country as third-world, it's correct most of the time. The negative predictive value (NPV) is 87.5%, showing that it's still fairly reliable in predicting non-third-world countries, though it could be a bit better.

With an F1 score of 0.8235, the model strikes a good balance between precision and recall, which is important for getting a solid mix of true positives and minimizing false positives. Overall, the balanced accuracy of 87.42% is a strong sign that the model

performs well across both categories, but there's definitely room for improvement, particularly in predicting non-third-world countries.

Creating the Random Forest Classification Model

```
randomForestFit <- train |> train(third_world ~ birth_rate + death_rate + population_growth_rate,
  method = "rf",
  data = _,
  tuneLength = 5,
  trControl = trainControl(method = "cv"))

## note: only 2 unique complexity parameters in default grid. Truncating the
grid to 2 .

randomForestFit

## Random Forest
##
## 181 samples
##   3 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 164, 163, 162, 163, 163, 163, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8729618  0.5516664
##   3     0.8680255  0.5501578
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.

randomForestFit$finalModel

##
## Call:
##   randomForest(x = x, y = y, mtry = param$mtry)
##             Type of random forest: classification
##                           Number of trees: 500
##   No. of variables tried at each split: 2
##
##             OOB estimate of  error rate: 11.6%
## Confusion matrix:
##   0  1 class.error
## 0 139  8  0.05442177
## 1  13 21  0.38235294
```

```

randomForest_preds <- predict(randomForestFit, test)

# Make sure both have the same levels (0 and 1 for binary classification)
randomForest_preds <- factor(randomForest_preds, levels = c(0, 1))

# Create confusion matrix
conf_matrix_rf <- confusionMatrix(randomForest_preds, test$third_world)

cm1 <- conf_matrix_rf$table
TP <- cm1[2, 2] # True Positives
TN <- cm1[1, 1] # True Negatives
FP <- cm1[1, 2] # False Positives
FN <- cm1[2, 1] # False Negatives

# manually calculates precision, recall, and F1 Score
precision_rf <- TP / (TP + FP) # Precision
recall_rf <- TP / (TP + FN) # Recall (Sensitivity)
f1_rf <- 2 * (precision_rf * recall_rf) / (precision_rf + recall_rf) # F1 Score

# Display the metrics
print("Precision")
## [1] "Precision"
precision_rf
## [1] 0.3333333
print("Recall")
## [1] "Recall"
recall_rf
## [1] 0.5
print("F1")
## [1] "F1"
f1_rf
## [1] 0.8235294

```

The Random Forest model has an accuracy of 88.36 percent, which is quite solid. However, the precision of 22.22% is pretty low, meaning it struggles when it predicts a country as third-world. Its recall of 50 percent shows it's moderately good at identifying third-world countries but still misses a fair number of them. The F1 score of 42.86 percent suggests there's room for improvement in balancing precision and recall. Despite its decent overall accuracy, the model is having difficulty with non-third-world countries,

which is reflected in the low precision and moderate recall. Fine-tuning the model or balancing the dataset could help improve its performance.

Creating the ANN (Artificial Neural Network) Model

```
nnetFit <- train(  
  third_world ~ birth_rate + death_rate + population_growth_rate,  
 # Use all predictors  
  method = "nnet",                      # Neural network model  
  data = train,                          # Training dataset  
  tuneLength = 5,                        # Automatic tuning for 5 combinations of  
  hyper parameters  
  trControl = trainControl(method = "cv"), # Cross-validation  
  trace = FALSE                         # Suppress progress output  
)  
  
nnetFit  
  
## Neural Network  
##  
## 181 samples  
##   3 predictor  
##   2 classes: '0', '1'  
##  
## No pre-processing  
## Resampling: Cross-Validated (10 fold)  
## Summary of sample sizes: 164, 164, 162, 164, 162, 163, ...  
## Resampling results across tuning parameters:  
##  
##   size  decay  Accuracy  Kappa  
##   1     0e+00  0.8184039  0.1263973  
##   1     1e-04  0.8459580  0.2101286  
##   1     1e-03  0.8397833  0.2511535  
##   1     1e-02  0.8790334  0.4910142  
##   1     1e-01  0.8676299  0.3936220  
##   3     0e+00  0.8793258  0.5212605  
##   3     1e-04  0.8626591  0.4817614  
##   3     1e-03  0.8901445  0.6525034  
##   3     1e-02  0.8740626  0.5588285  
##   3     1e-01  0.8907637  0.6497872  
##   5     0e+00  0.8901789  0.6500469  
##   5     1e-04  0.8849157  0.6358114  
##   5     1e-03  0.8629171  0.5747997  
##   5     1e-02  0.8796526  0.6183076  
##   5     1e-01  0.8907637  0.6497872  
##   7     0e+00  0.8898521  0.6044550  
##   7     1e-04  0.8954076  0.6684178  
##   7     1e-03  0.8793258  0.6270953  
##   7     1e-02  0.8353973  0.4699524
```

```

##    7    1e-01  0.8901445  0.6377001
##    9    0e+00  0.8626247  0.5870970
##    9    1e-04  0.8690918  0.5899831
##    9    1e-03  0.8691262  0.5743067
##    9    1e-02  0.8564499  0.5279286
##    9    1e-01  0.8907637  0.6497872
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 7 and decay = 1e-04.

nnet_preds <- predict(nnetFit, test)

# Predictions
nnet_preds <- factor(nnet_preds, levels = levels(test$third_world))

# confusion matrix and metrics
conf_matrix_nnet <- confusionMatrix(nnet_preds, factor(test$third_world))

# Extract confusion matrix components for ANN
cm_nnet <- conf_matrix_nnet$table
TP_nnet <- cm_nnet[2, 2]
TN_nnet <- cm_nnet[1, 1]
FP_nnet <- cm_nnet[1, 2]
FN_nnet <- cm_nnet[2, 1]

# Calculate Precision, Recall, and F1 Score for ANN
precision_nnet <- TP_nnet / (TP_nnet + FP_nnet)
recall_nnet <- TP_nnet / (TP_nnet + FN_nnet)
f1_nnet <- 2 * (precision_nnet * recall_nnet) / (precision_nnet + recall_nnet)

# Print the metrics for ANN
print("Precision")
## [1] "Precision"

precision_nnet
## [1] 0.7777778

print("Recall")
## [1] "Recall"

recall_nnet
## [1] 0.7777778

print("F1")
## [1] "F1"

```

```
f1_nnet  
## [1] 0.7777778
```

The Neural Network model does pretty well, with an accuracy of 89.02%, which is solid. Its precision and recall are both 77.78%, meaning it does a good job at identifying third-world countries without making too many mistakes on either side. The F1 score of 77.78% backs that up. While it's doing well, it's not the best when compared to the other models, and there's definitely room for improvement. Tweaking the model or playing around with different settings could help make it even better.

```
resamps <- resamples(list(  
  logReg = log_model,  
  ANN = nnetFit,  
  RandomForest = randomForestFit))  
  
summary(resamps)  
  
##  
## Call:  
## summary.resamples(object = resamps)  
##  
## Models: logReg, ANN, RandomForest  
## Number of resamples: 10  
##  
## Accuracy  
##  
##          Min.   1st Qu.   Median   Mean   3rd Qu.   Max. N  
A's  
## logReg      0.7894737 0.8333333 0.8622291 0.8791710 0.9305556 1.0000000  
0  
## ANN        0.6842105 0.8839869 0.9179567 0.8954076 0.9466374 1.0000000  
0  
## RandomForest 0.7777778 0.8398693 0.8888889 0.8729618 0.8932749 0.9473684  
0  
##  
## Kappa  
##  
##          Min.   1st Qu.   Median   Mean   3rd Qu.   Max. N  
A's  
## logReg      0.3414634 0.4705882 0.5361512 0.5968668 0.7676471 1.0000000  
0  
## ANN        0.1971831 0.4961779 0.7508126 0.6684178 0.8245292 1.0000000  
0  
## RandomForest 0.2244898 0.3899522 0.6061224 0.5516664 0.6828008 0.8256881  
0
```

The Random Forest model clearly outperforms the others, with a strong average accuracy of 88.4% and a high degree of consistency. It occasionally hits a perfect 100%, and its Kappa score shows it's making solid, reliable predictions. Logistic Regression also does well with an accuracy of 87.9%, but it's a bit more inconsistent, and its Kappa score

suggests it doesn't always match the true values perfectly. The ANN model shows promise but struggles more than the other two, particularly in terms of consistency. Overall, Random Forest is the most reliable and gives the best performance across the board.

Summary and Conclusion

The three models — Logistic Regression, Random Forest, and Artificial Neural Network (ANN) — showed varying performance levels in classifying countries as either third world or not. Logistic Regression performed strongly, with an accuracy of 93.02%, excellent sensitivity (97.06%), and solid precision (94.29%). However, its specificity was lower at 77.78%, meaning it was less effective at identifying non-third world countries. Despite this, its F1 score of 0.8235 indicated a good balance between precision and recall. Random Forest, while achieving an accuracy of 88.36%, struggled with low precision (22.22%) and moderate recall (50%), leading to an F1 score of 42.86%. This model showed potential but needs improvement, especially in balancing its ability to predict third-world countries without misclassifying non-third-world ones. The ANN model had an accuracy of 89.02%, with precision and recall both at 77.78%, showing solid performance but with some inconsistencies.

In conclusion, Random Forest was the most reliable model overall, with strong accuracy and consistent results across folds. Logistic Regression also performed well, especially in predicting third-world countries, though it could improve in identifying non-third-world nations. The ANN model, while promising, didn't quite reach the performance of the other two and would benefit from further fine-tuning. Ultimately, Random Forest outperformed the other models, but there's still room to refine all three approaches, especially in handling class imbalance and improving prediction consistency across both third-world and non-third-world countries.