

# Final Project Report

Anmol Sachdev

## Table of Contents

Introduction .....	3
Data Exploration.....	4
Data cleaning: .....	4
Exploratory Data Analysis:.....	7
Multiple Linear Regression: .....	10
LASSO Regression.....	15
Logistic Regression .....	18
Conclusion .....	25
References .....	27

## Introduction

In this report, we have selected a dataset that provides a comprehensive summary of health and economic indicators for eastern European countries over the period of 1997 to 2021. Our objective is to use the data presented to answer the following questions: *What health and economic factors influence child mortality rate in Eastern European countries? And is it possible to predict whether a country is majority urban or rural by looking at different health and economic indicators?* To do so, we built a multiple linear regression and a LASSO regression model to predict the mortality rate based off our dataset features. Then, we built a logistic regression to address the second question concerning the population type.

The first section of this report includes a preliminary analysis of the dataset, providing an overview of the available features. Subsequently, we proceed to the data cleaning process, where we assess the data integrity and apply appropriate methods to address any potential issues, such as missing values. Next, we perform an exploratory data analysis (EDA) to gain a deeper understanding of the feature space. This involves a detailed examination of each feature's distribution and density to understand their behavior and patterns. In the next sections of the report, we focus on model development by employing multiple linear regression, LASSO regression and Logistic regression, and Logistic regression to address the questions we formulated based on our data.

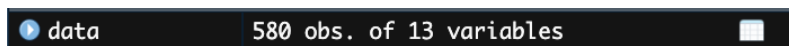
## Data Exploration

The data set used in the study was collected from the World Bank's World Development Indicators Database. This data set contains health and country data from the years 1997 – 2021 for 23 countries in Eastern Europe. The variables in the data set are the following: Mortality rate of children under 5 years old, population (total number), population (ages 0-14 total number), rural population (% of total population), urban population (% of total population), current health expenditure (% of GDP), GDP per capita, % of births attended by skilled health staff.

The data was loaded into R using the following R code:

```
# Read data set. When prompted navigate and open eastern_data.csv file
data <- read.table(file.choose(),
  sep = ",",
  header = TRUE,
  stringsAsFactors = FALSE)
```

The initial data set consisted of 580 observations and 13 variables as shown below:



A screenshot of the RStudio interface showing a data frame named 'data' with 580 observations and 13 variables. The text 'data' is on the left, '580 obs. of 13 variables' is in the center, and a small icon is on the right.

### Data cleaning:

The next step after loading the data was to perform data cleaning. The following data cleaning methods were used:

1. The last 5 records were null values, so we dropped them from the data set.
2. Renamed the column names to names that were easier to use while coding. We used the `rename()` function.
3. Select only the relevant columns for our analysis using the `select()` function. There were extra columns that were redundant, such as country abbreviations, year was written twice.
4. Adjusted the variables class to numeric if appropriate.
5. Normalized the children population value by dividing them by the total population number of the country.
6. Created a new binary variable called `population_type` classifying observations based on type of population (Urban or Rural) by comparing the rural and urban population percentage variables.
7. Excluded the non-normalized variables from our data set, such as the total population.

Please find below the R Code used to perform the changes described above:

```
data <- data[1:(nrow(data) - 5), ] # last 5 records are Null records, dropping the Null records
#cleaning the data using dplyr
data_new <- data %>%
  rename("Mortality_rate" = "Mortality.rate..under.5..per.1.000.live.births...SH.DYN.MORT.",
         "Population_0_14" = "Population.ages.0.14..total..SP.POP.0014.TO.",
         "Population_total" = "Population..total..SP.POP.TOTL.",
         "Rural_pop_pct" = "Rural.population...of.total.population...SP.RUR.TOTL.ZS.",
         "Urban_pop_pct" = "Urban.population...of.total.population...SP.URB.TOTL.IN.ZS.",
         "current_health_exp" = "Current.health.expenditure...of.GDP...SH.XPD.CHEX.GD.ZS.",
         "births_attended_by_skilled_professionals" = "Births.attended.by.skilled.health.staff...of.total...SH.STA.BRTC.ZS.",
         "GDP_per_capita" = "GDP.per.capita..current.US...NY.GDP.PCAP.CD.",
         "Country_name" = "Country.Name") %>% #rename column names to shorter names easier to handle
  select(Mortality_rate, Population_0_14, Population_total, Rural_pop_pct, Urban_pop_pct, current_health_exp,
         births_attended_by_skilled_professionals, GDP_per_capita,
         Country_name, Time) %>% #we exclude columns we will not use such as country code, time code
  mutate(Mortality_rate = as.numeric(Mortality_rate),
         current_health_exp = as.numeric(current_health_exp),
         births_attended_by_skilled_professionals = as.numeric(births_attended_by_skilled_professionals),
         GDP_per_capita = as.numeric(GDP_per_capita)) %>% #adjust class to numeric if appropriate
  mutate(child_pop_pct = Population_0_14 / Population_total, #create new column normalizing number of child population into percentages
         population_type = if_else(Urban_pop_pct > Rural_pop_pct, "Urban", "Rural")) %>% #create new column classifying population type based on rural and urban pop
  select(-c(Population_0_14, Population_total)) #drop non normalized columns
```

The new data set contained 575 observations and 10 variables as shown below:

data\_new 575 obs. of 10 variables

The next step is to determine how we will treat the missing values in our data. First, we calculated the number of missing values in each variable. Please find the R code and corresponding output below:

```
> # getting the Null value count of each attribute
> colSums(is.na(data_new))
```

Mortality_rate	23	Rural_pop_pct	0
Urban_pop_pct	0	current_health_exp	127
births_attended_by_skilled_professionals	99	GDP_per_capita	3
Country_name	0	Time	0
child_pop_pct	0	population_type	0

According to our observations, the following variables have missing values: mortality\_rate, births\_attended\_by\_skilled\_professionals, current\_health\_exp, GDP\_per\_capita. To better understand which are these missing values, we created 2 tables summarizing the number of missing values for the attribute per country and the mean value of the attribute per country. We followed the same procedure for mortality\_rate and current\_health\_expenditure.

### 1. Mortality rate:

```
> (na_mortality_country <- data_new %>%
+   group_by(Country_name) %>%
+   summarise(na_mortality = sum(is.na(Mortality_rate))))
# A tibble: 23 x 2
  Country_name      na_mortality
  <chr>            <int>
1 Albania                1
2 Azerbaijan             1
3 Belarus                1
4 Bosnia and Herzegovina 1
5 Bulgaria               1
6 Croatia                1
7 Czechia                1
8 Estonia                1
9 Finland                1
10 Greece                 1
# ... with 13 more rows
```

```
> (mortality_country <- data_new %>%
+   group_by(Country_name) %>%
+   summarise(mean_mortality=mean(Mortality_rate, na.rm=TRUE)))
# A tibble: 23 x 2
  Country_name      mean_mortality
  <chr>              <dbl>
1 Albania             17.2
2 Azerbaijan          45.9
3 Belarus              7.52
4 Bosnia and Herzegovina 7.95
5 Bulgaria            12.0
6 Croatia              6.27
7 Czechia              4.10
8 Estonia              6.26
9 Finland              3.32
10 Greece              4.87
# ... with 13 more rows
```

For mortality rate, we observe that for the first 10 countries there is 1 missing value.

Additionally, the mean mortality rate for different countries varies significantly. For instance, the mean mortality rate for Azerbaijan is 45.9 which is significantly higher than the mean mortality rate for Czechia, which is 4.10.

## 2. Health Expenditure:

```
> #view number of na values per country for variable current_health_exp
> (na_health_exp_country <- data_new %>%
+   group_by(Country_name) %>%
+   summarise(na_health_exp = sum(is.na(current_health_exp))))
# A tibble: 23 x 2
  Country_name      na_health_exp
  <chr>            <int>
1 Albania                6
2 Azerbaijan             5
3 Belarus                5
4 Bosnia and Herzegovina 5
5 Bulgaria               5
6 Croatia                5
7 Czechia                5
8 Estonia                5
9 Finland                5
10 Greece                 5
# ... with 13 more rows
```

```
> (health_exp_country <- data_new %>%
+   group_by(Country_name) %>%
+   summarise(mean_heal_exp= mean(current_health_exp, na.rm=TRUE)))
# A tibble: 23 x 2
  Country_name      mean_heal_exp
  <chr>              <dbl>
1 Albania             5.74
2 Azerbaijan           2.92
3 Belarus              5.61
4 Bosnia and Herzegovina 8.89
5 Bulgaria             7.00
6 Croatia              7.12
7 Czechia              6.78
8 Estonia              5.72
9 Finland              8.70
10 Greece              8.39
```

For health expenditure, we observe that for the first 10 countries there is at least 5 missing values per country. Additionally, the mean health expenditure for different countries varies as well. For instance, the mean health expenditure for Azerbaijan is 2.92 which is lower than the mean health expenditure for Finland, which is 8.70.

This prior analysis gave us more insights into ways to impute the missing values for the 4 columns identified above. Since there is a difference in behavior between countries, we decided to impute the missing values with the mean of the attribute per country. We used the following R Code to achieve said results:

```
data_complete <- data_new %>%
  group_by(Country_name) %>%
  mutate(Mortality_rate=ifelse(is.na(Mortality_rate),mean(Mortality_rate,na.rm=TRUE),Mortality_rate),
         births_attended_by_skilled_professionals=ifelse(is.na(births_attended_by_skilled_professionals),
                                                         mean(births_attended_by_skilled_professionals,
                                                            na.rm=TRUE),
                                                         births_attended_by_skilled_professionals),
         current_health_exp=ifelse(is.na(current_health_exp),mean(current_health_exp,na.rm=TRUE),
                                   current_health_exp),
         GDP_per_capita=ifelse(is.na(GDP_per_capita),mean(GDP_per_capita,na.rm=TRUE),GDP_per_capita))
```

To make sure we did not have missing values in our analysis, we ran the following R code again:

```
> colSums(is.na(data_complete))
      Mortality_rate      Rural_pop_pct      0
      Urban_pop_pct      current_health_exp      0
      births_attended_by_skilled_professionals      GDP_per_capita      0
      Country_name      Time      0
      child_pop_pct      population_type      0
```

With no missing values left, our data was ready to be modeled.

## Exploratory Data Analysis

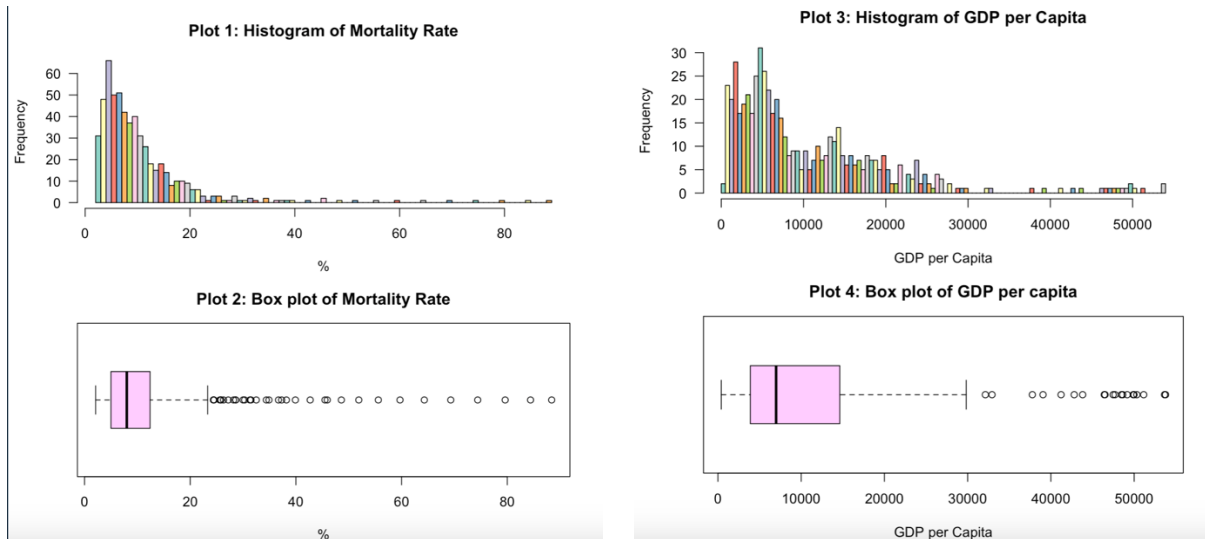
The next step was to do an exploratory data analysis to understand how our variables data values were distributed. We created histograms, boxplots, and density plots for various variables.

R code:

```
#closer look into mortality rate, check outliers and summary statistics
par(mfcol=c(2,1),
    mai=c(1,1,0.2,0.4),
    mar=c(4,4,2,3,2))#fix margins to fit 2 plots on top of each other
#Histogram of lot area in order to view distribution of data points
hist(data_complete$Mortality_rate,
     breaks=100,
     main="Plot 1: Histogram of Mortality Rate",
     xlab="Mortality Rate",
     #ylim= c(1300.0,17671.0),
     col= brewer.pal(9, "Set3"),
     las = 1) #data is skewed to the right (log transformation might have to be used)
#boxplot of lot area in order to observe IQR and outliers
boxplot(data_complete$Mortality_rate,
        horizontal = T,
        # ylim= c(1300.0,17671.0),
        col= "#FFCCFF",
        xlab= "Mortality Rate",
        main= "Plot 2: Box plot of Mortality Rate") #presence of outliers to the right

#closer look into urban GDP per capita, check outliers and summary statistics
par(mfcol=c(2,1),
    mai=c(1,1,0.2,0.4),
    mar=c(4,4,2,3,2))#fix margins to fit 2 plots on top of each other
#Histogram in order to view distribution of data points
hist(data_complete$GDP_per_capita,
     breaks=100,
     main="Plot 3: Histogram of GDP per Capita",
     xlab="GDP per Capita",
     #ylim= c(1300.0,17671.0),
     col= brewer.pal(9, "Set3"),
     las = 1)
#boxplot in order to observe IQR and outliers
boxplot(data_complete$GDP_per_capita,
        horizontal = T,
        # ylim= c(1300.0,17671.0),
        col= "#FFCCFF",
        xlab= "GDP per Capita",
        main= "Plot 4: Box plot of GDP per capita") #Indicates presence of outliers to the right
```

Results:



The plots above show the distribution of the variables: mortality rate and GDP per capita. In general, both histograms (plot 1 and 2) show that the distribution is skewed to the right. One solution is to apply log transformation to the variables for the distribution to be approximately normal. Additionally, we observe several outliers to the right of the boxplots (plots 2 and 4). Finally, the data points for mortality rate seems to be more concentrated given that it has a smaller range than GDP per capita by looking at the boxplots.

We created two additional plots showing the density for the variables `current_health_expenditure` and `child_pop_pct`.

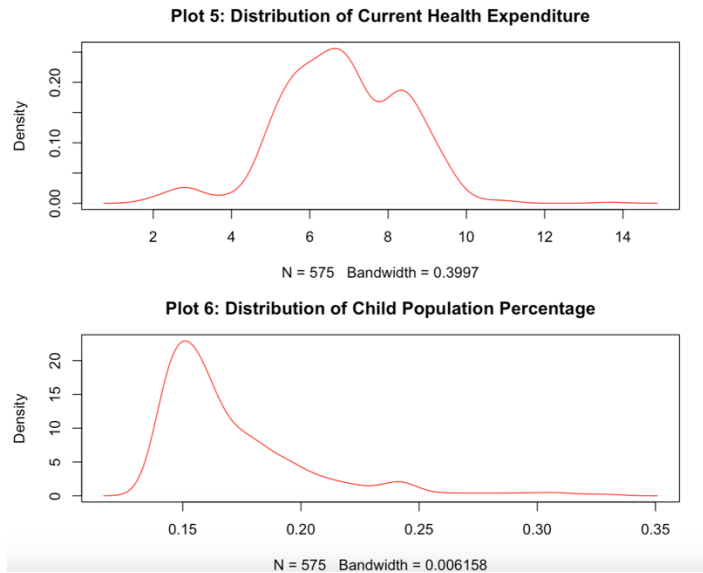
R code:

```
#density plot for current health expenditure
polygon(plot(density(data_complete$current_health_exp),
  main = "Plot 5: Distribution of Current Health Expenditure", col = 'red'))

#density plot for child population percentage
polygon(plot(density(data_complete$child_pop_pct),
  main = "Plot 6: Distribution of Child Population Percentage", col = 'red'))
```

Results:



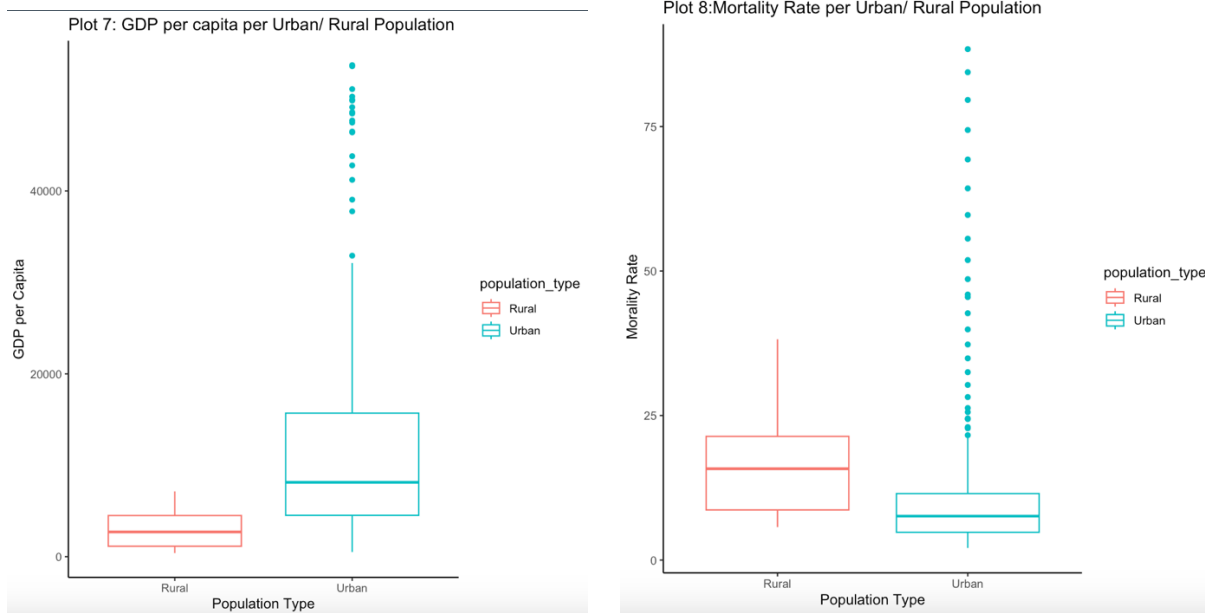


Plot 5 shows that the current health expenditure distribution is approximately normal. However, plot 6 shows that the distribution of the child population percentage is skewed to the right.

To conclude our EDA, we graphed 2 boxplots comparing 2 different variables (GDP per capita, and Child mortality rate) for the two different types of populations: Rural and Urban. Please find the R codes and corresponding outputs below:

```
#create boxplots to compare GDP_per_capita for Urban/ Rural Population
ggplot(data_complete, aes(x=population_type, y=GDP_per_capita, color=population_type)) +
  geom_boxplot() +
  labs(title="Plot 7: GDP per capita per Urban/ Rural Population",
       y = "GDP per Capita", x = "Population Type")+
  theme_classic()

#create boxplots to compare Mortality_rate for Urban/ Rural Population
ggplot(data_complete, aes(x=population_type, y=Mortality_rate, color=population_type)) +
  geom_boxplot() +
  labs(title="Plot 8: Mortality Rate per Urban/ Rural Population",
       y = "Mortality Rate", x = "Population Type")+
  theme_classic()
```



When we look at plot 7, the GDP per capita range of values for Rural is much concentrated than the range for Urban. Additionally, the median GDP per capita value for Urban is higher compared to Rural. On the other side, for plot 8, the median mortality rate for rural is higher than compared to Urban. In both plots, the Urban population type has more outliers than the rural population type. Because the median number for both variables are different for the 2 population types, it is a good indication that these can be used in the logistic regression with population type as the dependent variable.

### Multiple Linear Regression

The next step is to fit the multiple linear regression model using Mortality\_rate as our response variable. To choose which variables to include as predictors, we ran a stepwise regression model to choose the best model. However, before we ran the model, we divided our data set into training and test set. The model will first be trained on the training data set that contains 70% of the observations. Afterwards, we will test the model using the test set that has the remaining 30% of the observations. Please find below the R code used:

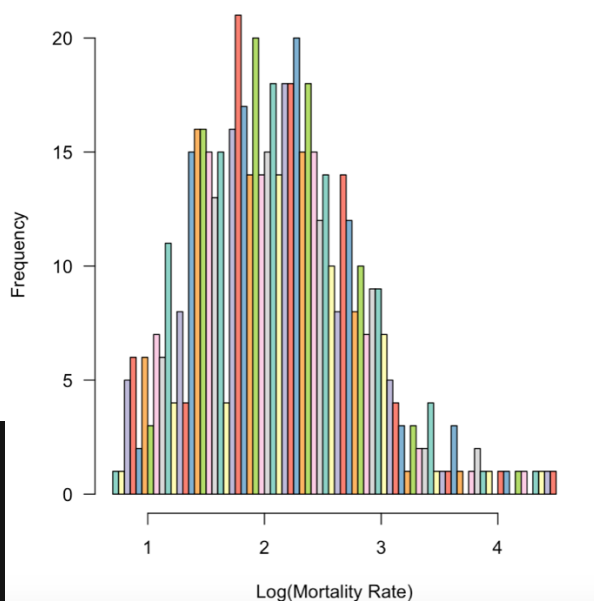
```
# Create Train and Test set - (70/30 split)
set.seed(100) # for repeatability of samples
# create indexing for 70% of the data
trainIndex <- createDataPartition(data_complete$Mortality_rate, p = 0.7, list = FALSE, times = 1)
train <- data_complete[trainIndex,] #70% of data
test <- data_complete[-trainIndex,] #30% of data
```

Now we have 2 new data sets: train and test. The test set has 170 observations and 10 variables, and the train set has 405 observations and 10 variables.

test	170 obs. of 10 variables	
train	405 obs. of 10 variables	

Next, we ran a stepwise regression model to choose the best model. Please find the R code below. We automatically saved the preferred model by sending the output of the step() function to a variable called best\_model. Additionally, we used our train set as the chosen data set for the regression model and performed a log transformation to our response variable Mortality\_rate to deal with its skewness identified above. To determine whether this log transformation was needed, we created a histogram of the log of mortality\_rate.

Plot 9: Histogram of Log(Mortality Rate)



```
#Should we log transform the response variable?
#histogram of log area in order to view distribution of data points
hist(log(data_complete$Mortality_rate),
     breaks=100,
     main="Plot 9: Histogram of Log(Mortality Rate)",
     xlab="Log(Mortality Rate)",
     #xlim= c(1300.0,17671.0),
     col = brewer.pal(9, "Set3"),
     las = 1) #data follows an approxiamtely normal distribution
```

Because the distribution of values of mortality rate follows an approximately normal distribution, we will choose this transformation for the linear regression below.

```
#stepwise regression:
#define intercept-only model
intercept_only <- lm(log(Mortality_rate) ~ 1, data=train)
#define model with all predictors
all <- lm(log(Mortality_rate) ~ . -Country_name -population_type, data=train)
#perform stepwise regression
best_model <- step(intercept_only, direction='both', scope=formula(all))
```

Please note that for the stepwise regression we indicated two models: the one with only the intercept and the other one including all possible predictors (but excluding the country names and population\_type to deal with multicollinearity). Please find the results below:

```
Start: AIC=-313.07
log(Mortality_rate) ~ 1
```

	Df	Sum of Sq	RSS	AIC
+ GDP_per_capita	1	92.471	93.565	-589.42
+ child_pop_pct	1	89.024	97.012	-574.76
+ Time	1	38.535	147.501	-405.07
+ current_health_exp	1	34.852	151.184	-395.08
+ Rural_pop_pct	1	34.447	151.589	-394.00
+ Urban_pop_pct	1	34.447	151.589	-394.00
+ births_attended_by_skilled_professionals	1	26.875	159.161	-374.26
<none>			186.036	-313.07

```
Step: AIC=-589.42
log(Mortality_rate) ~ GDP_per_capita
```

	Df	Sum of Sq	RSS	AIC
+ child_pop_pct	1	39.298	54.267	-808.04
+ births_attended_by_skilled_professionals	1	12.764	80.801	-646.82
+ current_health_exp	1	11.734	81.831	-641.69
+ Time	1	8.782	84.783	-627.33
+ Rural_pop_pct	1	2.825	90.740	-599.83
+ Urban_pop_pct	1	2.825	90.740	-599.83
<none>			93.565	-589.42
- GDP_per_capita	1	92.471	186.036	-313.07

```
Step: AIC=-808.04
log(Mortality_rate) ~ GDP_per_capita + child_pop_pct
```

	Df	Sum of Sq	RSS	AIC
+ current_health_exp	1	3.533	50.734	-833.30
+ Time	1	1.727	52.540	-819.13
+ births_attended_by_skilled_professionals	1	1.316	52.951	-815.98
<none>			54.267	-808.04
+ Rural_pop_pct	1	0.000	54.267	-806.04
+ Urban_pop_pct	1	0.000	54.267	-806.04
- child_pop_pct	1	39.298	93.565	-589.42
- GDP_per_capita	1	42.745	97.012	-574.76

```
Step: AIC=-833.3
log(Mortality_rate) ~ GDP_per_capita + child_pop_pct + current_health_exp
```

	Df	Sum of Sq	RSS	AIC
+ Time	1	2.466	48.269	-851.48
+ births_attended_by_skilled_professionals	1	0.838	49.896	-838.05
+ Rural_pop_pct	1	0.565	50.170	-835.83
+ Urban_pop_pct	1	0.565	50.170	-835.83
<none>			50.734	-833.30
- current_health_exp	1	3.533	54.267	-808.04
- child_pop_pct	1	31.097	81.831	-641.69
- GDP_per_capita	1	37.600	88.335	-610.72

```
Step: AIC=-851.48
log(Mortality_rate) ~ GDP_per_capita + child_pop_pct + current_health_exp + Time
```

	Df	Sum of Sq	RSS	AIC
+ Rural_pop_pct	1	1.2649	47.004	-860.23
+ Urban_pop_pct	1	1.2649	47.004	-860.23
+ births_attended_by_skilled_professionals	1	1.1432	47.126	-859.18
<none>			48.269	-851.48
- Time	1	2.4657	50.734	-833.30
- current_health_exp	1	4.2716	52.540	-819.13
- child_pop_pct	1	23.6942	71.963	-691.73
- GDP_per_capita	1	29.9623	78.231	-657.91

```
Step: AIC=-860.23
log(Mortality_rate) ~ GDP_per_capita + child_pop_pct + current_health_exp + Time + Rural_pop_pct
```

	Df	Sum of Sq	RSS	AIC
+ births_attended_by_skilled_professionals	1	0.9085	46.095	-866.14
<none>			47.004	-860.23
- Rural_pop_pct	1	1.2649	48.269	-851.48
- Time	1	3.1659	50.170	-835.83
- current_health_exp	1	5.4899	52.494	-817.49
- child_pop_pct	1	15.8703	62.874	-744.41
- GDP_per_capita	1	18.5724	65.576	-727.37

```
Step: AIC=-866.14
log(Mortality_rate) ~ GDP_per_capita + child_pop_pct + current_health_exp + Time + Rural_pop_pct + births_attended_by_skilled_professionals
```

	Df	Sum of Sq	RSS	AIC
<none>			46.095	-866.14
- births_attended_by_skilled_professionals	1	0.9085	47.004	-860.23
- Rural_pop_pct	1	1.0301	47.126	-859.18
- Time	1	3.3895	49.485	-839.40
- current_health_exp	1	4.7364	50.832	-828.52
- child_pop_pct	1	12.1431	58.239	-773.43
- GDP_per_capita	1	18.8461	64.941	-729.31

According to our results above, the best model is the one with the lowest AIC, which in our case is AIC = -866.14. The best model is the one that includes:

births\_attended\_by\_skilled\_professionals, rural\_pop\_pct, time, current\_health\_exp, child\_pop\_pct, GDP\_per\_capita. Our resulting regression equation is:

$$\log(\text{Mortality Rate}) = B0 + B1 * \text{GDP\_per\_capita} + B2 * \text{child\_pop\_pct} + B3 * \text{current\_health\_exp} + B4 * \text{time} + B5 * \text{rural\_pop\_pct} + B6 * \text{births\_attended\_by\_skilled\_professionals}$$

The R code and results of the summary of the logistic regression model can be found below:

```
> summary(best_model)

Call:
lm(formula = log(Mortality_rate) ~ GDP_per_capita + child_pop_pct +
  current_health_exp + Time + Rural_pop_pct + births_attended_by_skilled_professionals,
  data = train)

Residuals:
    Min       1Q   Median       3Q      Max
-1.03546 -0.21718 -0.03687  0.26175  0.91466

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.408e+01  5.662e+00   6.020 3.97e-09 ***
GDP_per_capita -2.943e-05  2.307e-06 -12.756 < 2e-16 ***
child_pop_pct   6.993e+00  6.830e-01  10.239 < 2e-16 ***
current_health_exp -8.110e-02  1.268e-02  -6.395 4.51e-10 ***
Time          -1.462e-02  2.702e-03  -5.410 1.09e-07 ***
Rural_pop_pct   6.109e-03  2.048e-03   2.982 0.00304 **
births_attended_by_skilled_professionals -3.188e-02  1.138e-02  -2.801 0.00535 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3403 on 398 degrees of freedom
Multiple R-squared:  0.7522,    Adjusted R-squared:  0.7485
F-statistic: 201.4 on 6 and 398 DF,  p-value: < 2.2e-16
```

If we look at the p-values, all the coefficients are statistically significant with respect to the significance level = 0.05. To better interpret the coefficients, we take the exponents of the coefficients as shown below:

```
> (exp(coef(best_model)) - 1) * 100

              (Intercept)              GDP_per_capita              child_pop_pct
              6.329642e+16              -2.942907e-03              1.088445e+05
              current_health_exp              Time              Rural_pop_pct
              -7.790209e+00              -1.450901e+00              6.127406e-01
births_attended_by_skilled_professionals
              -3.137951e+00
```

The interpretations of the coefficients are as follows:

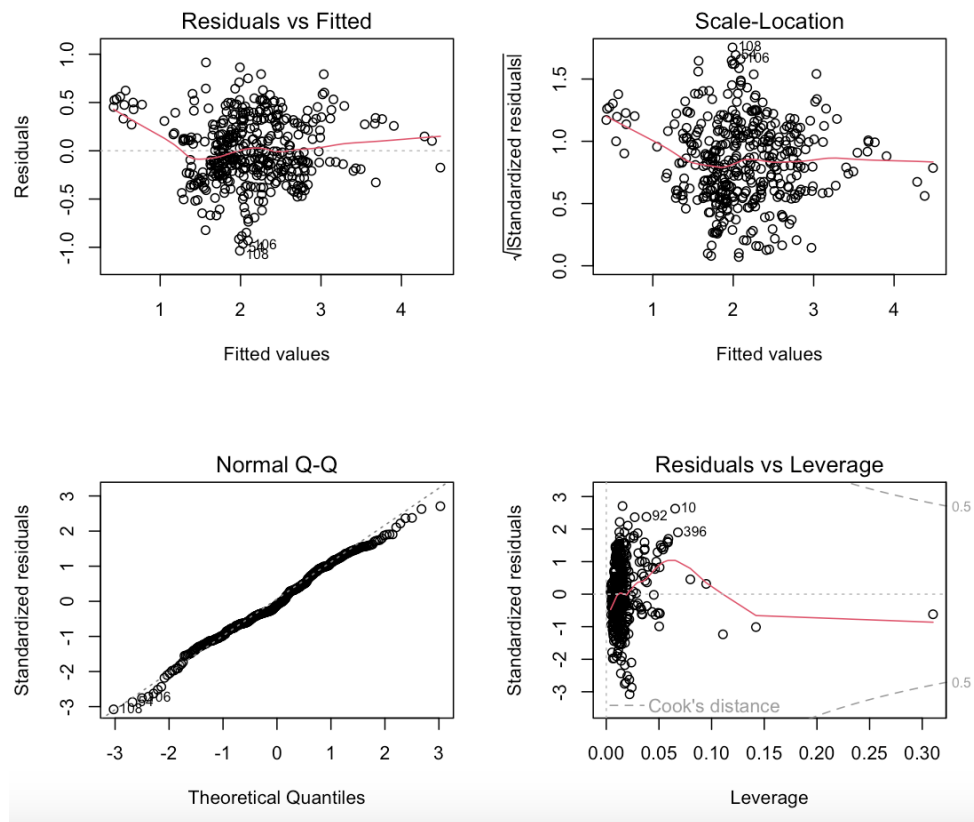
- For every one-unit increase in health expenditure, child mortality rate decreases by about 7.79%.
- For every one-unit increase in births attended by skilled staff, child mortality rate decreases by about 3.14%.
- For every one-unit increase in GDP per capita, child mortality rate decreases by about 2.94\*e-3%.
- For every one-unit increase in time, child mortality rate decreases by about 1.45%.
- For every one-unit increase in child population percentage, child mortality rate increases by about 1.088\*e5%.

- For every one-unit increase in health expenditure, child mortality rate increases by about 6.13\*10<sup>-1</sup>%.

### Regression Diagnostics:

There are several assumptions for Ordinary Least Squares regression, including Normality, Independence of errors, Linearity, and Homoscedasticity. To check these assumptions, we use the plot() function. The R code used and output can be found below:

```
#Use the "plot()" function to plot your regression model.
par(mfcol=c(2,2)) #fit 4 plots in one screen
plot(best_model)#the 4 graphs are used to analyze the linear regression assumptions.
```



The first graph (Residual vs Fitted) is used to check the linearity assumption given that there should be no relationship between the residuals and the fitted values. The red line should be flat, however, it has a “u” shape showing a slight pattern. The second plot (Normal Q-Q Plot) is used to check for the normality assumption where all residuals should be normally distributed. The points should fall closely to the diagonal line to indicate that the dependent variable is normally

distributed for any fixed values of the predictor variables. In our plot only a few plots fall outside of the diagonal line, which is a good sign for our model. The third plot (Scale-Location) is used to determine the homoscedasticity assumption, which means that the variance of the residuals is similar across the values of the independent variables. If the assumption is met, the points should be randomly spread around the horizontal line with no obvious pattern. In our plot there is not an obvious pattern which is a good sign for our model. Finally, the fourth graph (Residual vs Leverage) is used to check for influential points. If there are points outside of the dotted line, they have high influence. In our graph, no points fall outside of the dotted line which is a good sign for our model.

Next, we wanted to check for multicollinearity, which is when independent variables in a regression model are correlated. To do so, we calculate the variance inflation factor (VIF) in R.

```
> vif(best_model) #all less than 5, no concern of multicollinearity
              GDP_per_capita      child_pop_pct
              1.661737          1.818887
    current_health_exp              Time
              1.423956          1.329173
    Rural_pop_pct births_attended_by_skilled_professionals
              1.668120          1.276572
```

All values are less than 5. Hence, none of the variables in our multiple linear regression model are highly correlated.

Finally, we wanted to determine the performance of our model by comparing the root mean square error (RMSE) values of the train and test set as shown below:

```
> #determine performance of train set by calculating the root mean square error
> preds.train.ols <- predict(best_model, new=train)
> rmse(train$Mortality_rate,preds.train.ols)
[1] 13.21977
> #determine performance of test set by calculating the root mean square error
> preds.test.ols <- predict(best_model, new=test)
> rmse(test$Mortality_rate,preds.test.ols)
[1] 12.85539
```

The RMSE for the train set was 13.21977 and for the test set 12.85539. The RMSE of the test set is smaller than the RMSE for the train set, indicating that there is a small likelihood of overfitting.

### LASSO Regression

For the next step, we wanted to use LASSO Regression which uses L1 regularization method to compare our results with the Linear Regression model above. The first step was to create a

matrix of the train and test sets which includes all the variables except for Country\_name, population\_type (to avoid multicollinearity), and the response variable Mortality\_rate.

```
# Convert Training data into matrix
train_X <- model.matrix(Mortality_rate~.-Country_name-population_type, train)[-1] #excl
# Convert Testing data into matrix
test_X <- model.matrix(Mortality_rate~.-Country_name-population_type, test)[-1] #excl
```

To make sure the class of train\_X and test\_X was matrix and not dataframe we ran the following code:

```
class(train_X) #check class is matrix and not dataframe
class(test_X) #check class is matrix and not dataframe
```

Once we verified the class was matrix for both, we assign the response rate Mortality\_rate to new values given the train and test set as shown below:

```
#assign Mortality_rate values to train and test
train_y <- train$Mortality_rate
test_y <- test$Mortality_rate
```

In L2 Regularization, we need to choose the penalty value (Lambda) that will force some of the coefficient estimates to be exactly zero. We used cross-validation to find the optimal values of lambda, and we assigned our alpha value in the glmnet function in R to be 1. Please see the R code used to estimate the lambda.min and lambda.1se values:

```
set.seed(100) # for repeatability of samples
# Run cross validation for different lambda values using K-Fold CV where K=10.
cv.lasso <- cv.glmnet(train_X, train_y, nfolds = 10)
```

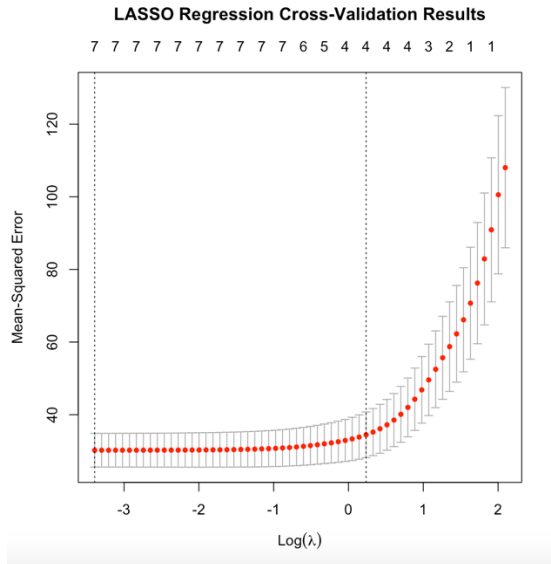
```
> cv.lasso$lambda.min # MIN lambda
[1] 0.03361361
> cv.lasso$lambda.1se # MAX lambda within 1 standard error from MIN
[1] 1.265531
```

Note that we used the cross-validation method with 10 folds. The minimum log lambda is equal to 0.03361361 and the maximum log lambda that is within 1 standard deviation of the minimum is 1.255531. As lambda increases, the shrinkage penalty grows.

Next, we plotted the cross-validation results. Please find the R code and corresponding output below:

```
# Visualize cross-validation data.
plot(cv.lasso) # The vertical lines are the confidence interval for the error.
title("LASSO Regression Cross-Validation Results", line=3) #add title to plot
```





The red dots are the intervals estimate variance of the loss metric computed using cross-validation. The numbers at the top are the number of nonzero coefficient estimates, which go from 7 to 1. The X-axis corresponds to the values of log lambda and the vertical lines show the locations of  $\lambda_{\min}$  (left) and  $\lambda_{1se}$  (right). These values were calculated in R using the code below:

```
> log(cv.lasso$lambda.min) # MIN log lambda
[1] -3.392824
> log(cv.lasso$lambda.1se) # MAX log lambda within 1 standard error from MIN
[1] 0.2354918
```

Finally, the Y-axis corresponds to the mean-squared error. As we increase lambda, the MSE increases at an exponential rate and more variables are reduced to zero.

Next, we fit the LASSO Regression model. We chose  $\lambda_{1se}$  as the lambda for this model because this model “is considered the most parsimonious model and helps to avoid over-fitting” (Valeriy, 2023). We also identified  $\alpha=1$  to differentiate between the LASSO and Ridge regression.

Please see R code and corresponding output below:

```
#fit final model on the training data using lambda.min
model.fit.lasso <- glmnet(train_X, train_y, alpha =1, lambda = cv.lasso$lambda.1se)
#display regresion coef
coef(model.fit.lasso)
```

```
8 x 1 sparse Matrix of class "dgCMatrix"
              s0
(Intercept)  1.040206e+02
Rural_pop_pct      .
Urban_pop_pct      .
current_health_exp -7.011639e-01
births_attended_by_skilled_professionals -1.158809e+00
GDP_per_capita     -8.627973e-05
Time              .
child_pop_pct      1.602532e+02
```

We can observe that the following variable coefficients in the model were reduced to 0, including Rural\_pop\_pct, Urban\_pop\_pct, and Time. We were left with 4 non-zero coefficients.

Finally, to determine the performance of the model, we compare the root mean square error (RMSE) our training and test sets. The R code and output can be observed below:

```
> #determine performance of train set by calculating the root mean square error
> preds.train.lasso <- predict(model.fit.lasso, newx = train_X)
> (train.rmse <- rmse(train_y, preds.train.lasso))
[1] 5.678812
> #determine performance of test set by calculating the root mean square error
> preds.test.lasso <- predict(model.fit.lasso, newx = test_X)
> (test.rmse <- rmse(test_y, preds.test.lasso))
[1] 5.383268
```

The RMSE for the training set was 5.678812 and for the test set 5.383268. The RMSE of the test set is slightly smaller than the RMSE for the train set, indicating that there is a small likelihood of overfitting.

Finally, if we compare the RMSE values for the test sets in the Linear and LASSO regression, the LASSO regression has the smallest one. This indicates that the LASSO regression is performing better. Please see the code below where we observe the RMSE values together:

```
> rmse(test$Mortality_rate, preds.test.ols) #linear regression RMSE
[1] 12.85539
> (test.rmse <- rmse(test_y, preds.test.lasso)) #LASSO RMSE
[1] 5.383268
```

### Logistic Regression

The next step is to fit the logistic regression model using Population\_type as our response variable. To choose which variables to include as predictors, we ran a stepwise regression model to choose the best model. As inputs to the stepwise regression model we had 2 models: the model with only the intercept and the model with all variables (we excluded the variables Country\_name, Rural\_pop\_pct, Urban\_pop\_pct).

```
#define intercept-only model
intercept_only <- glm(as.factor(population_type) ~ 1,
  data = train,
  family= binomial(link= "logit"))
#define model with all predictors
all <- glm(as.factor(population_type) ~ . -Country_name -Rural_pop_pct -Urban_pop_pct,
  data = train,
  family= binomial(link= "logit"))
```

Additionally, we used our train set as the chosen data set for the regression model. The results of the stepwise regression can be found below:

Start: AIC=280.37  
as.factor(population\_type) ~ 1

	Df	Deviance	AIC
+ GDP_per_capita	1	204.45	208.45
+ current_health_exp	1	225.45	229.45
+ child_pop_pct	1	256.82	260.82
+ Mortality_rate	1	268.87	272.87
<none>		278.37	280.37
+ births_attended_by_skilled_professionals	1	276.67	280.67
+ Time	1	277.79	281.79

Step: AIC=208.45  
as.factor(population\_type) ~ GDP\_per\_capita

	Df	Deviance	AIC
+ current_health_exp	1	137.45	143.45
+ Time	1	174.68	180.68
+ births_attended_by_skilled_professionals	1	192.31	198.31
+ Mortality_rate	1	200.78	206.78
<none>		204.45	208.45
+ child_pop_pct	1	204.45	210.45
- GDP_per_capita	1	278.37	280.37

Step: AIC=143.45  
as.factor(population\_type) ~ GDP\_per\_capita + current\_health\_exp

	Df	Deviance	AIC
+ Time	1	111.52	119.52
+ births_attended_by_skilled_professionals	1	124.29	132.29
+ child_pop_pct	1	129.73	137.73
+ Mortality_rate	1	130.91	138.91
<none>		137.45	143.45
- current_health_exp	1	204.45	208.45
- GDP_per_capita	1	225.45	229.45

Step: AIC=119.52  
as.factor(population\_type) ~ GDP\_per\_capita + current\_health\_exp + Time

	Df	Deviance	AIC
+ child_pop_pct	1	91.905	101.91
+ Mortality_rate	1	101.354	111.35
+ births_attended_by_skilled_professionals	1	106.388	116.39
<none>		111.521	119.52
- Time	1	137.453	143.45
- current_health_exp	1	174.682	180.68
- GDP_per_capita	1	224.128	230.13

Step: AIC=101.9  
as.factor(population\_type) ~ GDP\_per\_capita + current\_health\_exp + Time + child\_pop\_pct

	Df	Deviance	AIC
+ births_attended_by_skilled_professionals	1	68.760	80.76
<none>		91.905	101.91
+ Mortality_rate	1	90.550	102.55
- child_pop_pct	1	111.521	119.52
- Time	1	129.727	137.73
- GDP_per_capita	1	164.976	172.98
- current_health_exp	1	174.479	182.48

Step: AIC=80.76  
as.factor(population\_type) ~ GDP\_per\_capita + current\_health\_exp + Time + child\_pop\_pct + births\_attended\_by\_skilled\_professionals

	Df	Deviance	AIC
+ Mortality_rate	1	58.785	72.785
<none>		68.760	80.760
- births_attended_by_skilled_professionals	1	91.905	101.905
- Time	1	97.588	107.588
- child_pop_pct	1	106.388	116.388
- GDP_per_capita	1	137.670	147.670
- current_health_exp	1	158.131	168.131

Step: AIC=72.79  
as.factor(population\_type) ~ GDP\_per\_capita + current\_health\_exp + Time + child\_pop\_pct + births\_attended\_by\_skilled\_professionals + Mortality\_rate

	Df	Deviance	AIC
<none>		58.785	72.785
- Mortality_rate	1	68.760	80.760
- child_pop_pct	1	71.281	83.281
- Time	1	86.246	98.246
- GDP_per_capita	1	87.855	99.855
- births_attended_by_skilled_professionals	1	90.550	102.550
- current_health_exp	1	157.418	169.418

According to our results above, the best model is the one with the lowest AIC, which in our case is AIC = 72.790. The best model is the one that includes: mortality\_rate, child\_pop\_pct, time,

GDP\_per\_capita, births\_attended\_by\_skilled\_professionals, and current\_health\_expenditure. Our resulting regression equation is:

$$P = B0 + B1 * GDP\_per\_capita + B2 * current\_health\_expenditure + B3 * time + B4 * child\_pop\_pct + B5 * births\_attended\_by\_skilled\_professionals + B6 * Mortality\_rate,$$

where  $P = \log(\text{Population\_type})$ .

The R code and results of the summary of the logistic regression model can be found below:

```
> summary(log_model) #view summary of logistic regression

Call:
glm(formula = as.factor(population_type) ~ GDP_per_capita + current_health_exp +
    Time + child_pop_pct + births_attended_by_skilled_professionals +
    Mortality_rate, family = binomial(link = "logit"), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.90411  0.00004  0.00143  0.02945  2.26581

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.023e+03  2.183e+02  4.686 2.78e-06 ***
GDP_per_capita  9.422e-04  2.964e-04  3.178 0.001481 ***
current_health_exp -2.833e+00  5.816e-01 -4.871 1.11e-06 ***
Time           -3.349e-01  8.543e-02 -3.921 8.83e-05 ***
child_pop_pct   -4.545e+01  1.613e+01 -2.818 0.004827 **
births_attended_by_skilled_professionals -3.196e+00  8.983e-01 -3.557 0.000374 ***
Mortality_rate  -3.731e-01  1.331e-01 -2.803 0.005066 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 278.370  on 404  degrees of freedom
Residual deviance:  58.785  on 398  degrees of freedom
AIC: 72.785

Number of Fisher Scoring iterations: 10
```

If we look at the p-values, all the coefficients are statistically significant with respect to the significance level = 0.05 because they are smaller than the significance level. To better interpret the coefficients, we take the exponents of the coefficients as shown below:

```
> #view final model
> exp(coef(log_model)) #display regression coef ( odds) to interpret coeffs
              (Intercept)      GDP_per_capita
              Inf          1.000943e+00
current_health_exp      Time
5.882777e-02      7.153869e-01
child_pop_pct births_attended_by_skilled_professionals
1.824834e-20      4.094250e-02
Mortality_rate
6.885802e-01
```

The results are interpreted the following way:

- The odds of the population type being Urban compared to Rural increases by a factor of  $5.8838 \times 10^{-2}$  for each 1 unit increase in health expenditure.

- The odds of the population type being Urban compared to Rural increases by a factor of  $1.825 \times 10^{-20}$  for each 1 unit increase in child population percent.
- The odds of the population type being Urban compared to Rural increases by a factor of  $6.885 \times 10^{-1}$  for each 1 unit increase in mortality rate.
- The odds of the population type being Urban compared to Rural increases by a factor of 1.0009 for each 1 unit increase in GDP per capita.
- The odds of the population type being Urban compared to Rural increases by a factor of  $7.154 \times 10^{-1}$  for each 1 unit increase in Time.
- The odds of the population type being Urban compared to Rural increases by a factor of  $4.095 \times 10^{-2}$  for each 1 unit increase in births attended by skilled professionals.

#### Confusion matrix for train set:

The next step is to create a confusion matrix for the train set. To do so, we first must calculate the predicted values and assign our prediction to either Urban or Rural with a 0.5 cutoff for the probabilities. If the prediction in the train data using our logistic regression model is greater than 0.5, then assign it to Urban, otherwise to Rural. See the R code used below:

```
#make predictions in train data
probabilities.train <- predict(log_model, newdata = train, type = "response")
predicted.min <- as.factor(ifelse(probabilities.train > 0.5, "Urban", "Rural"))
```

Then, we create the confusion matrix using the predicted.min values and the Population Type values from our train set. Please find the R code and corresponding results below:

```
> confusionMatrix(predicted.min, as.factor(train$population_type), positive= "Urban")
Confusion Matrix and Statistics

      Reference
Prediction Rural Urban
Rural      39      7
Urban       5     354

    Accuracy : 0.9704
   95% CI : (0.9488, 0.9846)
 No Information Rate : 0.8914
 P-Value [Acc > NIR] : 2.892e-09

    Kappa : 0.85

McNemar's Test P-Value : 0.7728

    Sensitivity : 0.9806
   Specificity : 0.8864
  Pos Pred Value : 0.9861
  Neg Pred Value : 0.8478
    Prevalence : 0.8914
  Detection Rate : 0.8741
Detection Prevalence : 0.8864
 Balanced Accuracy : 0.9335

'Positive' Class : Urban
```

From our results, we can observe that we have 354 true positive values (Urban), 39 true negative values (Rural), 5 false positives (Type 1 error), and 7 false negatives (Type 2 error). The true values obtained were much greater than the false values.

The confusion matrix results also give us the values for Accuracy, Precision, Recall, and Specificity.

They are interpreted as follows:

- **Accuracy:** 0.9704. For all the classes, we predicted 97.04% correctly.
- **Precision** (Pos Pred Value): 0.9861. From all the classes we have predicted as positive (Urban), 98.61% are actually positive (Urban).
- **Recall** (Sensitivity): 0.9806. From all the positive classes (Urban), we predicted 98.06% correctly.
- **Specificity:** 0.8864. Out of all the negative classes (Rural), we predicted 88.64% correctly

Confusion matrix for test set:

The next step is to create a confusion matrix and report the results of the model for the test set. To do so, we first must calculate the predicted values and assign our prediction to either Urban or Rural with a 0.5 cutoff for the probabilities. If the prediction in the test data using our logistic regression model is greater than 0.5, then assign it to Urban, otherwise to Rural. Please see the R code used below:

```
#make predictions in test data
probabilities.test <- predict(log_model, newdata = test, type = "response")
predicted.min <- as.factor(ifelse(probabilities.test >= 0.5, "Urban", "Rural"))
```

Then, we create the confusion matrix using the predicted.min values and the Population\_Type values from our test set. Please find the R code and corresponding results below:

```
> confusionMatrix(predicted.min, as.factor(test$population_type), positive = "Urban")
Confusion Matrix and Statistics

          Reference
Prediction Rural Urban
   Rural      12     1
   Urban       6    151

    Accuracy : 0.9588
    95% CI   : (0.917, 0.9833)
  No Information Rate : 0.8941
  P-Value [Acc > NIR] : 0.001929

    Kappa : 0.7522

  Mcnemar's Test P-Value : 0.130570

    Sensitivity : 0.9934
    Specificity : 0.6667
   Pos Pred Value : 0.9618
   Neg Pred Value : 0.9231
    Prevalence : 0.8941
   Detection Rate : 0.8882
  Detection Prevalence : 0.9235
   Balanced Accuracy : 0.8300

 'Positive' Class : Urban
```

From our results, we can observe that we have 151 true positive values (Urban), 12 true negative values (Rural), 6 false positives (Type 1 error), and 1 false negatives (Type 2 error). The true values obtained were much greater than the false values.

The confusion matrix results also give us the values for Accuracy, Precision, Recall, and Specificity.

They are interpreted as follows:

- **Accuracy:** 0.9588. For all the classes, we predicted 95.88% correctly.

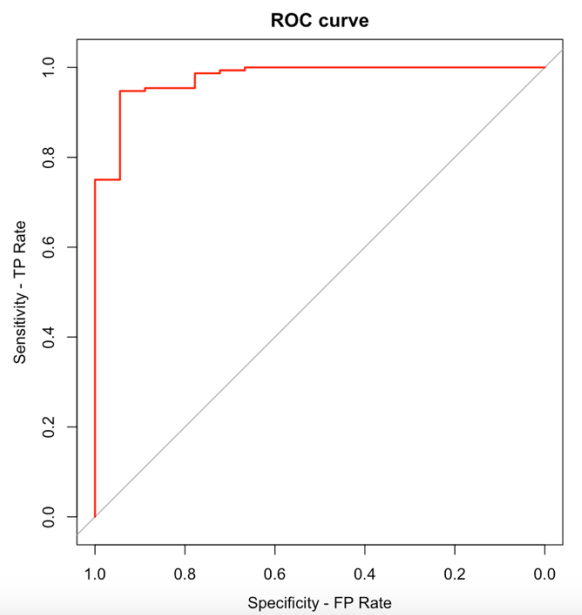
- **Precision** (Pos Pred Value): 0.9618. From all the classes we have predicted as positive (Urban), 96.18% are actually positive (Urban).
- **Recall** (Sensitivity): 0.9934. From all the positive classes (Urban), we predicted 99.34% correctly.
- **Specificity**: 0.6667. Out of all the negative classes (Rural), we predicted 66.67% correctly.

Finally, we compared the accuracy values for both models (using the train and test set). The accuracy for the training set was 97.04% and 95.88% for the test set. Given that they are not far apart, this suggests that there is not a likelihood of overfitting.

### ROC curve and AUC:

The receiver operator characteristic (ROC) curve is used to show the performance of the classification model. The X-axis corresponds to the false positive rate (or Specificity), and the Y-axis corresponds the true positive rate (or Sensitivity). Please see the R code used to plot the curve and the corresponding output:

```
#plot the ROC curve
ROC <- roc(as.factor(test$population_type), probabilities.test)
plot(ROC, col="red", ylab = "Sensitivity - TP Rate",
     xlab = "Specificity - FP Rate",
     main= "ROC curve")
```



The ROC curve's ideal scenario is when the edges of the red line are straight and closer to the top-left corner. This indicates that the model is perfectly able to distinguish between positive class and negative class. In our test model, our red curve is significantly closer to the top-left



corner than to the 45-degree grey diagonal. This suggests that our model is a good classifier. Finally, we calculated the area under the ROC curve. This can be seen in the last line of the code below.

```
> ROC # in case the code above doesnt work, you can observe the AUC value here as well

Call:
roc.default(response = as.factor(test$population_type), predictor = probabilities.test)

Data: probabilities.test in 18 controls (as.factor(test$population_type) Rural) < 152 cases (as.factor(test$population_type) Urban).
Area under the curve: 0.977
```

AUC values range from 0 to 1. A model whose predictions are “100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0” (*Classification: Roc curve and AUC | machine learning | google developers 2022*). Given that our AUC value is 97.7% we can conclude that our predictions are very close to being 100% correct. In other words, there is a 97.7% chance that the model will be able to distinguish between Urban class and Rural class.

### Conclusion

This report aimed analysis relationships between several health and economic factors in Eastern European countries. More specifically, we used the data presented to answer the following questions: What health and economic factors influence child mortality rate in Eastern European countries? And is it possible to predict whether a country is majority urban or rural by looking at different health and economic indicators?

The regression model development and stepwise selection process provided valuable insights into the key determinants of child mortality rate. Our results indicate that child mortality rate has decreased over time, but investment in healthcare services can have a substantial impact on reducing child mortality. Specifically, our findings reveal that a one-unit increase in healthcare expenditure leads to a 7.79% decrease in child mortality rate, while a one-unit increase in births attended by skilled staff results in a 3.14% decrease in child mortality rate. In conclusion, our study highlights the importance of investment in healthcare services for reducing child mortality rate.

The use of logistic regression in this context can help to identify important predictors of population types (rural or urban). By modeling the relationship between several economic and health characteristics and population type, researchers can develop a more nuanced understanding of how these factors interact and contribute to regional health disparities.

Moreover, such an analysis can have significant implications for public health policy and practice.

By identifying key predictors of mortality rates, policymakers and public health officials can develop targeted interventions to address specific population groups or geographic regions that may be at higher risk of mortality. This approach has the potential to improve health outcomes for vulnerable populations, reduce healthcare costs, and promote greater equity in health outcomes. The results of this analysis can serve as valuable input for policy makers and further research in the field.

## References

Google. (2022). *Classification: Roc curve and AUC | machine learning | google developers*.

Google. Retrieved January 31, 2023, from <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>

Shevchenko, V. (2023). *Lecture 4 (Week 4) Regularization with Ridge/LASSO. Northeastern University*.

World Bank Group - International Development, Poverty, & Sustainability. World Bank. (n.d.). Retrieved January 24, 2023, from <https://www.worldbank.org/en/home>