Global Suicide Rate Analysis: A Machine Learning Approach

Choose Your Own Capstone Project

Jason Hoskin

2025-12-04

Contents

1 Introduction

This report describes creating a model that can predict suicide rates per 100,000 people based on year, a country's Gross Domestic Product (GDP) and ascribed Human Development Index (HDI) score, age, generation, and sex. The utilized database is located on Github and collates data from the United Nations, the World Health Organization, World Bank, and previous data analysis from the Github user Rusty.[1] I accomplished this goal through preparing the data set by splitting the dataset into training and testing datasets, transforming GDP and the suicide rate per 100,000 people variables, and imputing the HDI variable because of missing data. Once the data was prepared, I proposed a baseline linear regression model and justified a random forest model, and then trained machine learning algorithms to predict the ratings and minimize model error. I then calculated the root mean square error (RMSE) to determine which model was more effective in prediction.

2 Methods

The following section describes the process of loading and preparing the data, including splitting the data into a holdout set and a training set and justifying which variables need mutation or transformation to improve the data results.

2.1 Data Preparation and Cleaning

First, I loaded the necessary libraries for my analysis, and included code that would download the libraries into R if they were not already downloaded. Afterwards, I downloaded the data from its online hyperlink access, and renamed the variables for standardization and ease of use in R, and then categorized all of the independent variables as a factor for clean linear regression modeling.

```r
### SETUP & DATA LOADING ###
if (!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if (!require(caret))
  install.packages("caret", repos = "http://cran.us.r-project.org")
if (!require(gridExtra))
  install.packages("gridExtra", repos = "http://cran.us.r-project.org")
if (!require(ranger))
  install.packages("ranger", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(gridExtra)
library(ranger)

# Import dataset
master <- read_csv("https://raw.githubusercontent.com/JasonMHoskin/suicide_gdp/main/master.csv")

# Initial Name Standardization
master <- master %>%
  rename(
    suicide_100k_rate = `suicides/100k pop`,
    country_year = `country-year`,
    hdi_for_year = `HDI for year`,
    gdp_for_year = `gdp_for_year ($)`,
    gdp_per_capita = `gdp_per_capita ($)`,
    suicides_no = `suicides_no`
  ) %>%
  mutate(
    country = as.factor(country),
    sex = as.factor(sex),
    age = as.factor(age),
    generation = as.factor(generation)
  )
```

2.2 Data Splitting Strategy

To split the dataset, I used $p = 0.1$ to create a holdout set with 10% of the dataset, leaving 90% for training. I selected a 90/10 split because the dataset is sufficiently large ($N >$ 27,000), allowing the model to train on maximum variance while retaining a statistically significant sample for validation. Then, to ensure that the models would see every country at least once in the data, I identified countries in the holdout set that were not in the training set and moved them to training. Rows containing countries unique to the holdout set were moved to the training set to prevent model failure due to unseen factor levels during prediction. This resulted in a negligible shift in the split ratio (from 10% to ~10.01%).

```r
# Set seed for consistent results
set.seed(123)
```

```
# Create partition on the data
test_index <- createDataPartition(
  y = master$suicide_100k_rate,
  times = 1,
  p = 0.1,
  list = FALSE
)

train_raw <- master[-test_index, ]
holdout_raw <- master[test_index, ]

# Ensure holdout countries exist in train_raw
holdout_set <- holdout_raw %>% semi_join(train_raw, by = "country")
removed <- anti_join(holdout_raw, holdout_set, by = "country")
train_raw <- rbind(train_raw, removed)

# Determine new partition percentages
nrow(holdout_set) / nrow(master)
```

```
## [1] 0.1001078
```

```
nrow(train_raw) / nrow(master)
```

```
## [1] 0.8998922
```

```
# Remove now superfluous vectors
rm(test_index, holdout_raw, removed, master)
```

2.3 Exploratory Data Analysis

Then, I conducted data visualization to assess ways that the data may need to be filtered or transformed. My results found that HDI was a frequently missing datapoint, with 17,482 data missing, leading me to decide to impute the missing HDI data. Moreover, the figures below depicted GDP and the suicide rate as positively skewed, leading me to decide that log transforming those variables would provide a more normal distribution for more accurate analyses.

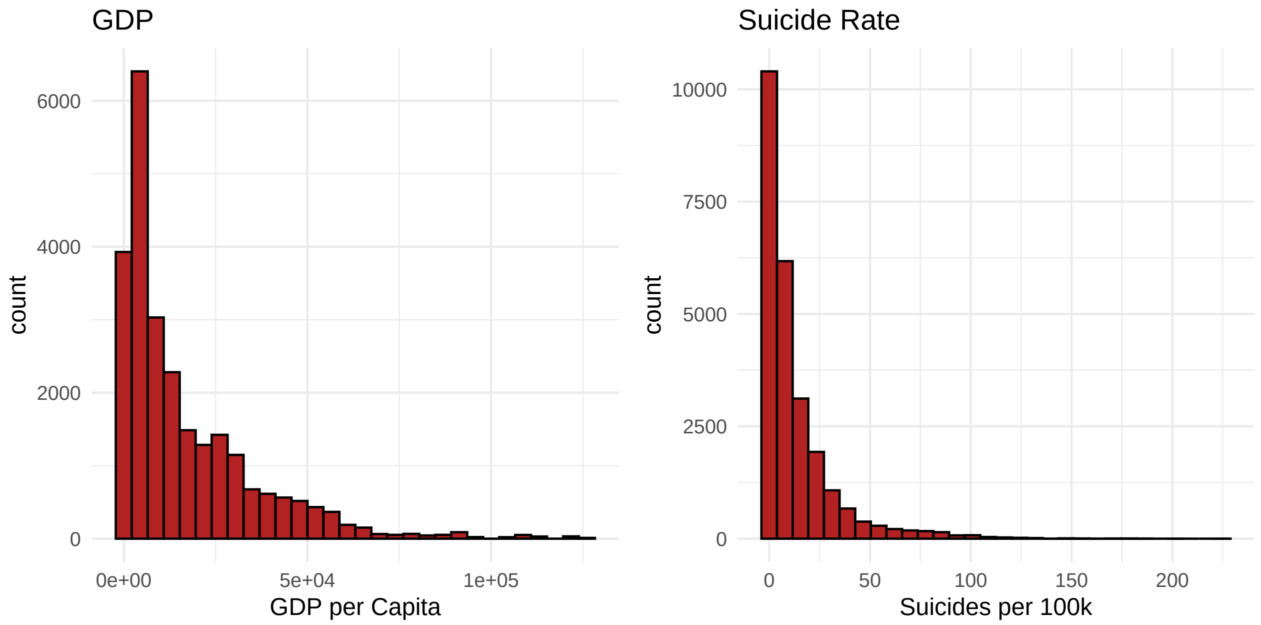
```
# Verify missing data
print(colSums(is.na(train_raw)))
```

```
##           country              year               sex               age
##                 0                 0                 0                 0
##       suicides_no        population suicide_100k_rate      country_year
##                 0                 0                 0                 0
##      hdi_for_year      gdp_for_year    gdp_per_capita        generation
##             17528                 0                 0                 0
```

```
# Visualize data skewness
p_skew_gdp <- ggplot(train_raw, aes(x = gdp_per_capita)) +
  geom_histogram(bins = 30, fill = "firebrick", color = "black") +
  labs(title = "GDP", x = "GDP per Capita") +
  theme_minimal()

p_skew_suicide <- ggplot(train_raw, aes(x = suicide_100k_rate)) +
  geom_histogram(bins = 30, fill = "firebrick", color = "black") +
  labs(title = "Suicide Rate", x = "Suicides per 100k") +
  theme_minimal()
```

```r
grid.arrange(p_skew_gdp, p_skew_suicide, ncol = 2)
```



Thus, I applied imputation by calculating the replacement values as the mean for each country, and including a global HDI mean as a fall back safety net in case a country has zero HDI data. This process is conducted to still allow HDI to be a contributing variable in prediction while minimizing chances of spurious prediction a much as possible.

```r
# Imputation statistics-Calculated on Train Only
hdi_means <- train_raw %>%
  group_by(country) %>%
  summarize(mean_hdi = mean(hdi_for_year, na.rm = TRUE), .groups = "drop")
global_mean <- mean(train_raw$hdi_for_year, na.rm = TRUE)
```

I now have confirmed how I want to clean and transform my data. I will now create a function that will allow me to make these changes to both my test and train datasets. I added one to all of my log transformations to ensure any 0 values could still be included in the data while minimizing my addition. I also included two age variables so that one can provide more clear data visualization and the other can be used for training. This function was then applied to both the train and the holdout sets, so that now both data sets have the same adjusted variables that the model can transfer from train_set to holdout_set.

```r
# Function to apply cleaning and transformation
process_data <- function(df, hdi_ref) {
  df %>%
    left_join(hdi_ref, by = "country") %>%
    mutate(
      hdi_filled = ifelse(is.na(hdi_for_year), mean_hdi, hdi_for_year),
      hdi_filled = ifelse(is.na(hdi_filled), global_mean, hdi_filled),
      log_gdp = log(gdp_per_capita + 1),
      log_suicide_rate = log(suicide_100k_rate + 1),
      age_numeric = case_when(
        age == "5-14 years" ~ 10,
        age == "15-24 years" ~ 20,
        age == "25-34 years" ~ 30,
        age == "35-54 years" ~ 45,
        age == "55-74 years" ~ 65,
        age == "75+ years" ~ 80,
        TRUE ~ NA_real_
      ),
      is_male = ifelse(sex == "male", 1, 0),
      age = factor(age, levels = c("5-14 years", "15-24 years", "25-34 years",
                                   "35-54 years", "55-74 years", "75+ years"))
    ) %>%
    select(country, sex, age, age_numeric, is_male, generation, log_gdp, hdi_filled, log_suicide_rate)
    na.omit()
}

# Apply processing
train_set <- process_data(train_raw, hdi_means)
holdout_set <- process_data(holdout_set, hdi_means)
```

To confirm that the previous issues have been resolved, I ran the same code and confirmed that the HDI data has been properly imputed, and the log GDP and suicide rate transformations are notably more normally distributed.

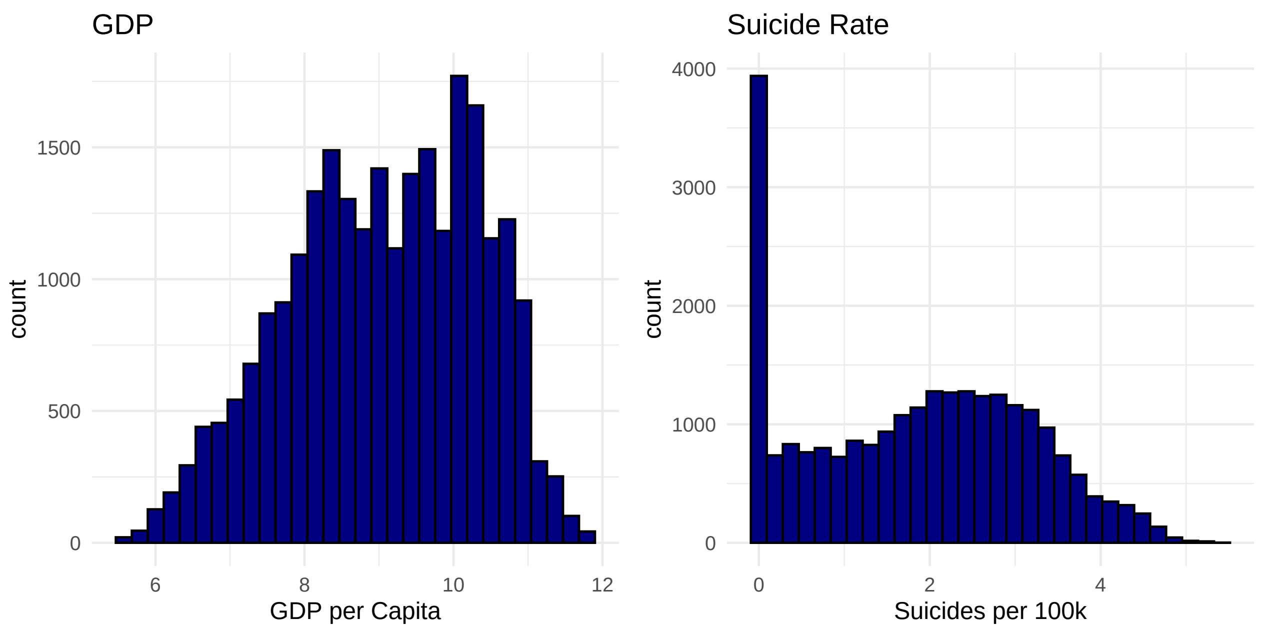
```
# Verify missing data
print(colSums(is.na(train_set)))
```

```
##          country              sex              age      age_numeric
##                0                0                0                0
##          is_male       generation          log_gdp       hdi_filled
##                0                0                0                0
## log_suicide_rate
##                0
```

```
# Visualize data skewness
p_log_gdp <- ggplot(train_set, aes(x = log_gdp)) +
  geom_histogram(bins = 30, fill = "navy", color = "black") +
  labs(title = "GDP", x = "GDP per Capita") +
  theme_minimal()

p_log_suicide <- ggplot(train_set, aes(x = log_suicide_rate)) +
  geom_histogram(bins = 30, fill = "navy", color = "black") +
  labs(title = "Suicide Rate", x = "Suicides per 100k") +
  theme_minimal()

grid.arrange(p_log_gdp, p_log_suicide, ncol = 2)
```



3 Results

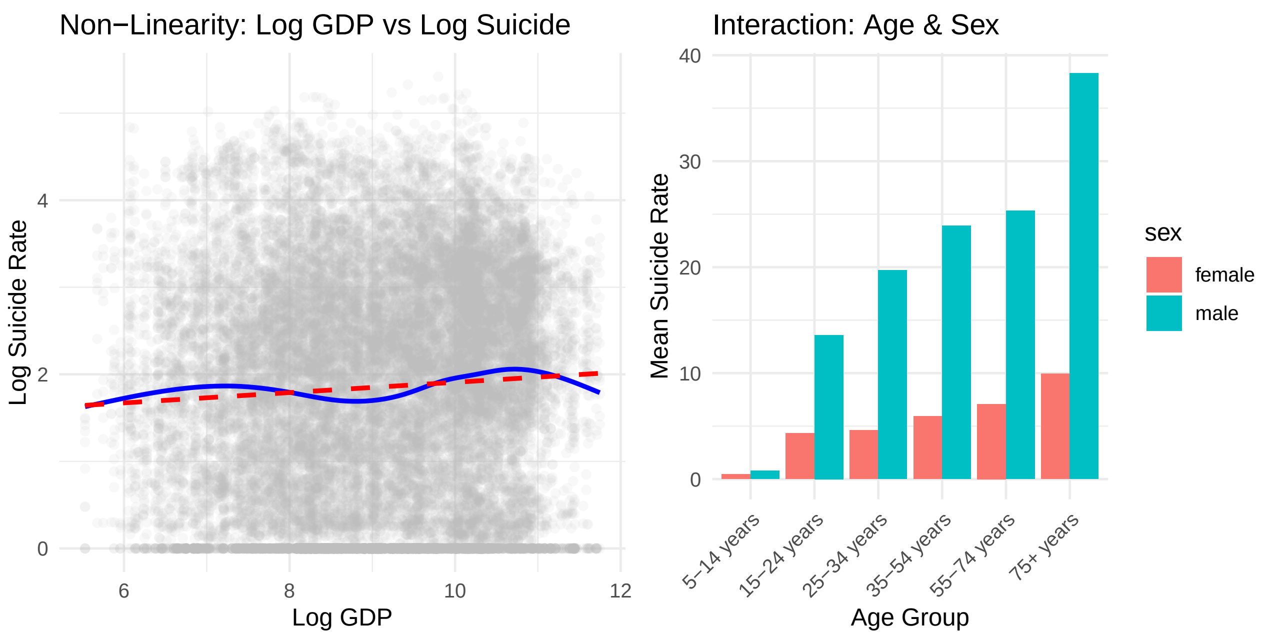
Now that my data is prepared for training, I then established which model aside from linear regression to train and then ran both and assessed their efficacy. Conducting exploratory data analysis revealed nonlinearity in the data and possible existing interactions. I thus decided to run a random forest model to automatically adjust the model.

```r
# Plot 1: Non-Linearity
p1 <- ggplot(train_set, aes(x = log_gdp, y = log_suicide_rate)) +
  geom_point(alpha = 0.1, color = "gray") +
  geom_smooth(method = "gam", color = "blue", se = FALSE) +
  geom_smooth(method = "lm", color = "red", linetype = "dashed", se = FALSE) +
  labs(title = "Non-Linearity: Log GDP vs Log Suicide", x = "Log GDP", y = "Log Suicide Rate") +
  theme_minimal()

# Plot 2: Interaction Effects
p2 <- train_set %>%
  group_by(age, sex) %>%
  summarize(mean_rate = mean(exp(log_suicide_rate) - 1), .groups = "drop") %>%
  ggplot(aes(x = age, y = mean_rate, fill = sex)) +
  geom_col(position = "dodge") +
  labs(title = "Interaction: Age & Sex", y = "Mean Suicide Rate", x = "Age Group") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(p1, p2, ncol = 2)
```

```
## `geom_smooth()` using formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```



3.1 Model Equations

Thus, to predict suicide rates, I implemented two distinct algorithms: a parametric Linear Regression model and a non-parametric Random Forest model.

3.1.1 1. Linear Regression Model

The baseline model is a multiple linear regression. Because the distribution of suicide rates is highly skewed, we apply a log-transformation to the target variable (Y) and the GDP predictor to stabilize variance and improve linearity.

The model is defined as:

$$\log(\text{rate}_i + 1) = \beta_0 + \beta_{sex} X_{i,sex} + \beta_{age} X_{i,age} + \beta_{gdp} \log(\text{GDP}_i + 1) + \beta_{hdi} \text{HDI}_i + \beta_{gen} X_{i,gen} + \varepsilon_i$$

Where:
* Y_i: The suicide rate per 100k population for observation i.
* β_0: The intercept.
* $X_{i,sex}, X_{i,age}, X_{i,gen}$: Categorical predictors for Sex, Age Group, and Generation (dummy encoded).
* $\text{GDP}_i, \text{HDI}_i$: Continuous predictors for GDP per Capita and Human Development Index.
* ε_i: The error term, assumed to be normally distributed $\varepsilon \sim N(0, \sigma^2)$.

3.1.2 2. Random Forest Model

To capture non-linear relationships and complex interactions (e.g., the varying effect of age on suicide rates depending on sex) without manual feature engineering, I utilized a Random Forest algorithm. This ensemble method aggregates the predictions of B decision trees.

The prediction for a new observation x is calculated as the average of the individual trees:

$$\hat{Y} = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$$

Where:
* B: The total number of trees in the forest (tuned via cross-validation).
* $T_b(x)$: The prediction of the b-th decision tree grown on a bootstrap sample of the training data.
* The algorithm randomly selects a subset of features (`mtry`) at each split to decorrelate the trees and reduce variance.

3.2 Training and Analysis

I then trained both models. To determine which model was more accurate, I decided to calculate the root mean squared error (RMSE) to assess the average distance between predicted and actual values. RMSE is the preferred metric for this regression problem because it penalizes large errors more heavily than Mean Absolute Error (MAE), which is critical in a recommendation system where a large prediction miss could have catastrophic failures.

```r
# Define Control
ctrl <- trainControl(method = "cv", number = 5)

# Model 1: Linear Regression
fit_lm <- train(
  log_suicide_rate ~ sex + age_numeric + log_gdp + hdi_filled + generation,
  method = "lm",
  data = train_set,
  trControl = ctrl
)

# Model 2: Random Forest
fit_rf <- train(
  log_suicide_rate ~ sex + age_numeric + log_gdp + hdi_filled + generation,
  method = "ranger",
  data = train_set,
  trControl = ctrl,
  tuneGrid = data.frame(mtry = c(3, 5), splitrule = "variance", min.node.size = 5),
  importance = "impurity"
)

# Predictions
pred_lm <- predict(fit_lm, holdout_set)
pred_rf <- predict(fit_rf, holdout_set)

error_lm <- RMSE(holdout_set$log_suicide_rate, pred_lm)
error_rf <- RMSE(holdout_set$log_suicide_rate, pred_rf)

results <- tibble(
  Model = c("Linear Regression", "Random Forest"),
  RMSE_Holdout = c(error_lm, error_rf)
)

print(results)
```

```
## # A tibble: 2 x 2
##   Model             RMSE_Holdout
##   <chr>                    <dbl>
## 1 Linear Regression        0.996
## 2 Random Forest            0.636
```

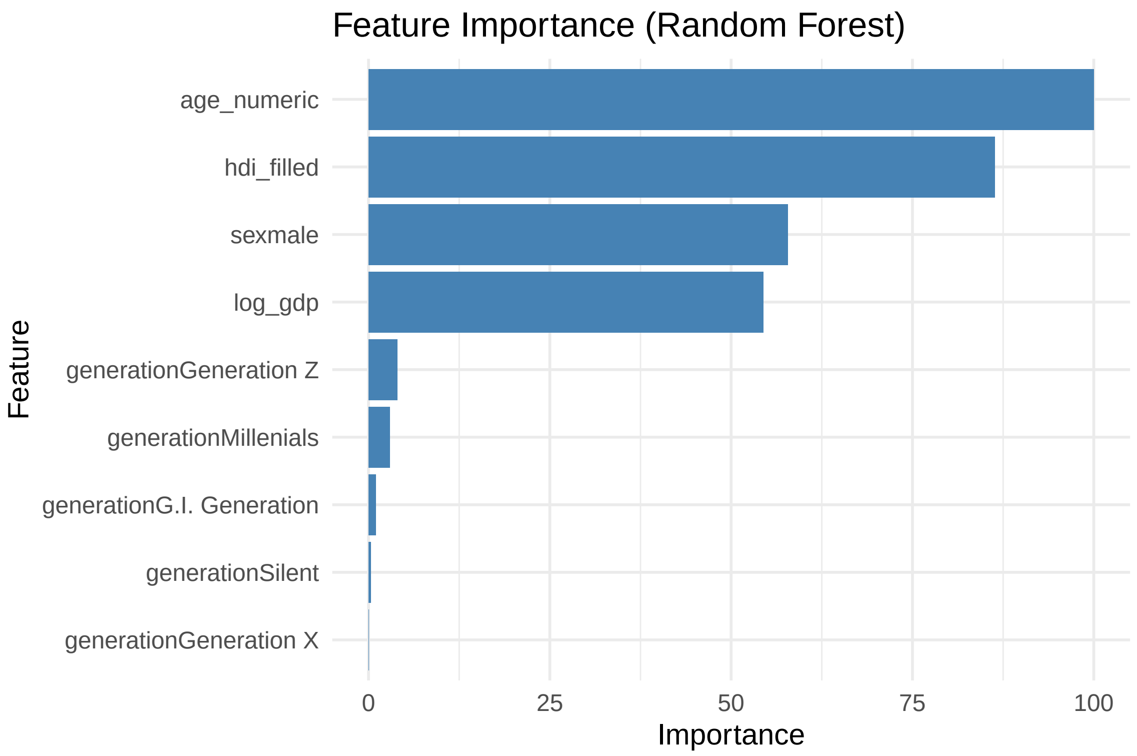
Therefore, the results demonstrate that the random forest model is more accurate with an RMSE of 0.64 compared to the linear regression model's 1.03.

3.3 Variable Importance

I also wanted to analyze which variables were the most predictive in determining the estimated log of suicide rates. I utilized the varImp() function and then graphed the results.

```r
varImp(fit_rf) %>%
  ggplot(aes(x = reorder(row.names(.), Overall), y = Overall)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Feature Importance (Random Forest)", x = "Feature", y = "Importance") +
  theme_minimal()
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```



The findings demonstrate age to be the most predictive variable, with the imputed HDI, being male, and the log of GDP (in order) also having notable important contributions.

4 Conclusion

The purpose of this research was to generate a system that could predict the suicide rate per 100,000 people. The findings indicated that the random forest model outperformed the baseline linear regression model, with age, country HDI and GDP, and being male as the most significant predictors. There are notable limitations to this model, such as the data requiring heavy imputation for the HDI variable and these models not incorporating any change over the years from 1985 to 2016. Thus, future research could incorporate a time-series analysis to see how suicide rates have changed across regions, ages, and sexes over the years. Future research could also consider other factors, such as the mental health expenditures or policies in various countries, or across differing demographics. Despite this study's limitations, this field of research remains vital, as accurate modeling helps to target necessary resources and interventions to high-risk demographics and regions.

5 References

[1] [Rusty]. (2018). Suicide Rates Overview 1985 to 2016 [dataset]. Retrieved from https://www.kaggle.com/datasets/russellyates88/suicide-rates-overview-1985-to-2016/data