Case Study 3 - World GDP Data 2023

Data Science with R, By Aung Soe

Introduction

Overview of Data

Data contains 2023 Gross Domestic Product (GDP) information for a set of countries around the world. The data was downloaded from Kaggle, collected by Nidula Elgiriyewithana. This comprehensive dataset provides a wealth of information about all countries worldwide, covering a wide range of indicators and attributes. It encompasses demographic statistics, economic indicators, environmental factors, healthcare metrics, education statistics, and much more. With every country represented, this dataset offers a complete global perspective on various aspects of nations, enabling in-depth analyses and cross-country comparisons.[1]

Key features used in this case study: 1. Country: Name of the country. 2. Density (P/Km2): Population density measured in persons per square kilometer. 3. Agricultural Land (%): Percentage of land area used for agricultural purposes. 4 .Land Area (Km2): Total land area of the country in square kilometers. 5. Armed Forces Size: Size of the armed forces in the country. 6. Birth Rate: Number of births per 1,000 population per year. 7. CO2 Emissions: Carbon dioxide emissions in tons. 8. CPI: Consumer Price Index, a measure of inflation and purchasing power. 9 .CPI Change (%): Percentage change in the Consumer Price Index compared to the previous year. 10. Fertility Rate: Average number of children born to a woman during her lifetime. 11. Forested Area (%): Percentage of land area covered by forests. 12. Gasoline_Price: Price of gasoline per liter in local currency. 13. GDP: Gross Domestic Product, the total value of goods and services produced in the country. 14. Gross Primary Education Enrollment (%): Gross enrollment ratio for primary education. 15. Gross Tertiary Education Enrollment (%): Gross enrollment ratio for tertiary education. 16. Infant Mortality: Number of deaths per 1,000 live births before reaching one year of age. 17. Largest City: Name of the country's largest city. 18. Life Expectancy: Average number of years a newborn is expected to live. 19. Maternal Mortality Ratio: Number of maternal deaths per 100,000 live births. 20. Minimum Wage: Minimum wage level in local currency. 21. Out of Pocket Health Expenditure (%): Percentage of total health expenditure paid out-of-pocket by individuals. 22. Physicians per Thousand: Number of physicians per thousand people. 23. Population: Total population of the country. 24. Population: Labor Force Participation (%): Percentage of the population that is part of the labor force. 25. Tax Revenue (%): Tax revenue as a percentage of GDP. 26. Total Tax Rate: Overall tax burden as a percentage of commercial profits. 27. Unemployment Rate: Percentage of the labor force that is unemployed. 28. Urban Population: Percentage of the population living in urban areas.

Motivation of study

There are many attributes of each country that could be good predictors of its GDP, however, it is more difficult to predict the GDP per capita, which is GDP for given population. This is because a country can have very high GDP per capita but it may have a very small military. This probably just means that it focuses its resources on other development areas but has very low budget for the military. Another example would be Singapore, since it is a city-state, the land area is one of the lowest, and on the other end of the spectrum, the United States of America is one of the largest countries. They both are at the top of the GDP per capita list. They are both very rich countries and their citizens are highly education and very productive.

Such scenarios make predicting GDP per capita very difficult. Therefore, it makes the case study fun with lots of opportunity for learning.

```
#install.packages('tidyverse')
#install.packages('dplyr')
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2          v readr     2.1.4.9000
## v forcats   1.0.0          v stringr   1.5.0
## v ggplot2   3.4.2          v tibble    3.2.1
## v lubridate 1.9.2          v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## 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(tidytext)
library(stringr)
library(gt)
df = read_csv("./world-data-2023.csv")
```

```
## Rows: 195 Columns: 35
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (19): Country, Abbreviation, Agricultural Land( %), Capital/Major City, ...
## dbl  (9): Birth Rate, Calling Code, Fertility Rate, Infant mortality, Life e...
## num  (7): Density
## (P/Km2), Land Area(Km2), Armed Forces size, Co2-Emissions,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Discovery and Data Preparation

Data Cleaning

There are some missing values in the data and these will prevent from producing a valid correlation matrix. There are two possible options to handle this: 1. Remove the country from the data 2. Fill in missing values with median values of related countries (imputation) 3. Identify features with high number of missing data and drop them

Option-1:

This option would look for all the feature columns and remove any countries that has any N/A values. This is probably not the best solution because there are a lot of countries removed, about 43% of the countries will be removed from the data set. Furthermore, countries such as Austria, Egypt and Singapore will be removed, and one would consider these countries to be important in the model calibration.

```r
# Rows removed (countries) with N/A values
removed_rows <- df[which(rowSums(is.na(df)) > 0),] %>%
  select(Country)
percent_removed = nrow(removed_rows)/nrow(df) * 100
cat("Percentage of countries removed for N/A values: ",percent_removed,"\n")
```

```
## Percentage of countries removed for N/A values:  43.58974
```

Option-2:

This option would require complicated surgery of the data. For instance, finding the right value for missing CPI for certain countries require some knowledge of how "like" countries are performing. This may be based on minimum wage, unemployment rate or GDP per capita itself. This dependency on other variables will not be an issue only when those variables are not missing. Scanning through the data, a lot of countries that have missing CPI data also have missing unemployment data, so imputation in this case will be very difficult. Furthermore, trying to come up with a good imputation model is a project in itself so will not be evaluated for this project.

Option-3:

First, countries or city states that have missing values in, for example, CPI have missing values across other features/variables. These countries are Monaco, Eswatini, Vatican City, Nauru, North Macedonia, Palestinian National Authority, Tuvalu. Removing these countries should not change the model performance because CPI could be a strong predictor of GDP. Following this essence, countries with missing values in these "key" features/variables are removed.

Second, certain features/variables have missing values because they just do not have them implemented or required in their systems. An example is the minimum wage. Such features/variables may not be ignored so the best solution might be to assign them to zero.

Lastly, there will still be some missing values in the data. These should not matter because they exist in features/variables that will be dropped in data manipulation phase.

```r
# Remove countries with missing values in "key" features/variables
keyVar = c("GDP", "CPI", "Unemployment rate", "Tax revenue (%)", "Gasoline Price", "Gross primary educa
df <- df[which(rowSums(is.na(df[,keyVar])) == 0),]

# Assign $0 as minimum wage for countries without a value.
df[, "Minimum wage"][is.na(df[, "Minimum wage"])] <- '$0.0'
```

```r
# Data Manipulation
world_data <- df %>%
  mutate(CountryID = row_number()) %>%
  rename(Pop_Density = `Density
(P/Km2)`) %>%
  rename(Land_Area = `Land Area(Km2)`) %>%
  rename(Birth_Rate = `Birth Rate`) %>%
  rename(Fertility_Rate = `Fertility Rate`) %>%
  rename(Infant_Mortality = `Infant mortality`) %>%
  rename(Life_Expectancy = `Life expectancy`) %>%
  rename(Mat_Mort_Ratio = `Maternal mortality ratio`) %>%
  rename(Phys_per_Thou = `Physicians per thousand`) %>%
```

```
  rename(Emissions = `Co2-Emissions`) %>%
  mutate(Agricultural_Land = as.numeric(gsub('[%]', '', `Agricultural Land( %)`))/100) %>%
  mutate(CPI_delta = as.numeric(gsub('[%]', '',`CPI Change (%)`))) %>%
  mutate(Forested_Area = as.numeric(gsub('[%]', '',`Forested Area (%)`))) %>%
  mutate(Gas_Prices = as.numeric(gsub('[$]', '', `Gasoline Price`))) %>%
  mutate(GDP_num = as.numeric(gsub('[$,]', '', GDP))) %>%
  mutate(GDP_per_capita = GDP_num/Population) %>%
  mutate(Primary_Ed_Enrol = as.numeric(gsub('[%]', '', `Gross primary education enrollment (%)`))/100) %
  mutate(Tert_Ed_Enrol = as.numeric(gsub('[%]', '', `Gross tertiary education enrollment (%)`))/100) %>%
  mutate(Min_Wage = as.numeric(gsub('[$]', '', `Minimum wage`))) %>%
  mutate(OutOfPocket_perc = as.numeric(gsub('[%]', '', `Out of pocket health expenditure`))/100) %>%
  mutate(Labor_Force_perc = as.numeric(gsub('[%]', '', `Population: Labor force participation (%)`))/10
  mutate(Tax_Revenue_perc = as.numeric(gsub('[%]', '', `Tax revenue (%)`))/100) %>%
  mutate(Tax_Rate = as.numeric(gsub('[%]', '', `Total tax rate`))/100) %>%
  mutate(Unemployment = as.numeric(gsub('[%]', '', `Unemployment rate`))/100) %>%
  mutate(Urban_Pop_perc = Urban_population/Population) %>%
  select(CountryID, Country, Pop_Density, Agricultural_Land, Land_Area, Birth_Rate, Fertility_Rate, Infa

#install.packages("countrycode")
library(countrycode)
world_data <- world_data %>%
  mutate(Region = countrycode(world_data$Country, "country.name", "region"))
```

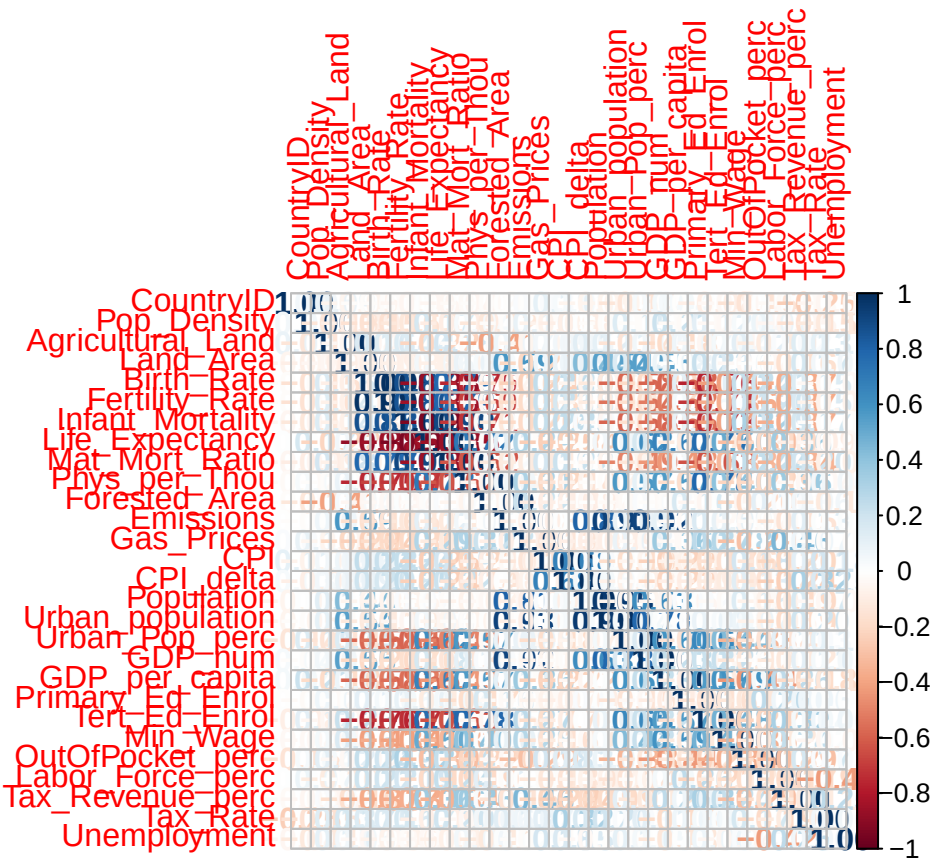
Data Exploration

Before setting up any models, a subset of the data could be extracted. This subset should have the variables that can be the features of the model. Any variables that has correlation above 0.5 would be considered as a feature to the models. Below is the correlation matrix.

```
#install.packages('corrplot')
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
cor_world_data <- subset(world_data, select = -c(Country, Region))
CorrMatrix <- cor(cor_world_data)
corrplot(CorrMatrix, method = 'number')
```



Based on the correlations, the following features have the highest correlations with GDP per capita. Plots below show that most of them have linear correlation and some non-linear correlation with GDP per capita.

```
# Make scatter plots for features that have correlation > 0.5 with GDP ($ Per Capita)
library(ggplot2)
library(gridExtra)
```

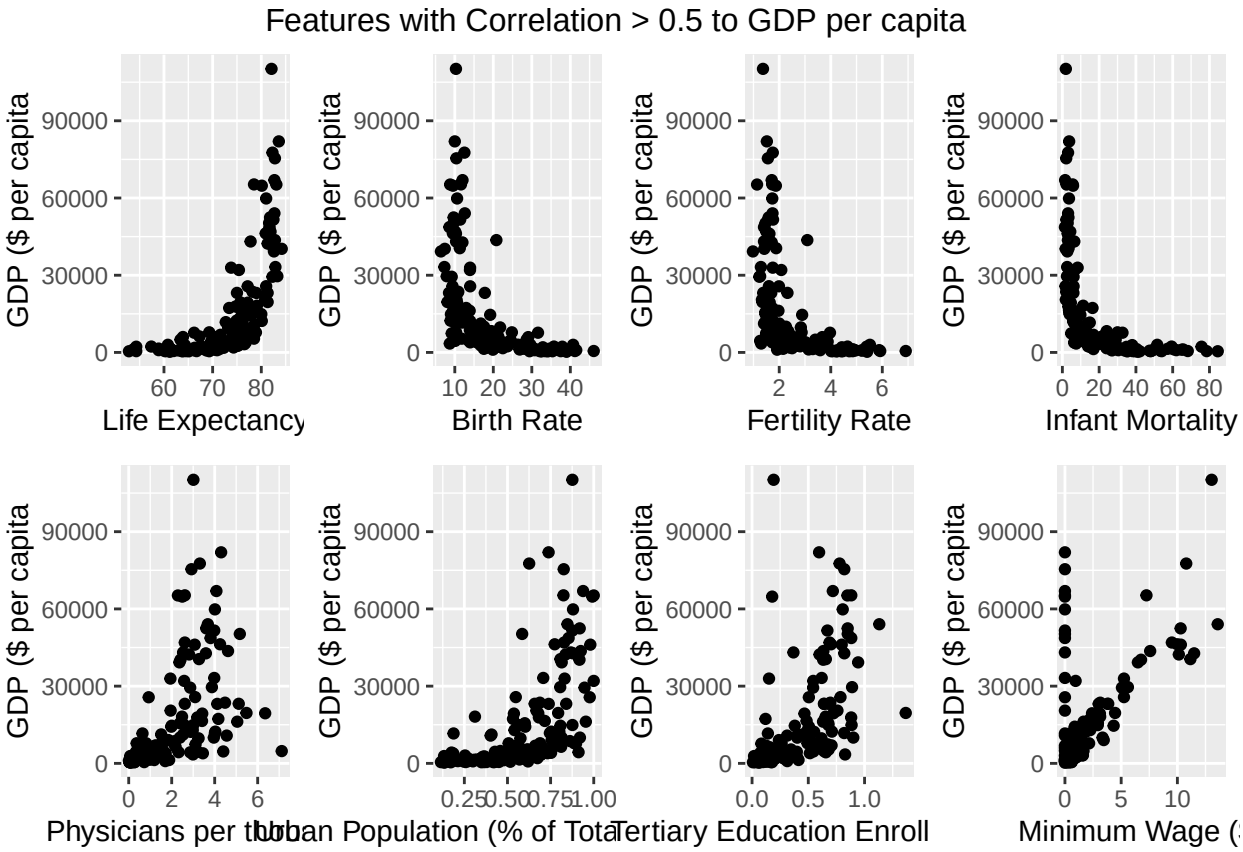
```
##
## Attaching package: 'gridExtra'
```

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

```
LifeExp_plot <- ggplot(data = cor_world_data, mapping = aes(Life_Expectancy, GDP_per_capita)) +
  geom_point(colour = 'black') + labs(x = "Life Expectancy", y = "GDP ($ per capita")
BirthRate_plot <- ggplot(data = cor_world_data, mapping = aes(Birth_Rate, GDP_per_capita)) +
  geom_point(colour = 'black') + labs(x = "Birth Rate", y = "GDP ($ per capita")
Fertilty_plot <- ggplot(data = cor_world_data, mapping = aes(Fertility_Rate, GDP_per_capita)) +
  geom_point(colour = 'black') + labs(x = "Fertility Rate", y = "GDP ($ per capita")
Infant_plot <- ggplot(data = cor_world_data, mapping = aes(Infant_Mortality, GDP_per_capita)) +
  geom_point(colour = 'black') + labs(x = "Infant Mortality", y = "GDP ($ per capita")
Physician_plot <- ggplot(data = cor_world_data, mapping = aes(Phys_per_Thou, GDP_per_capita)) +
  geom_point(colour = 'black') + labs(x = "Physicians per thousand", y = "GDP ($ per capita")
Urban_plot <- ggplot(data = cor_world_data, mapping = aes(Urban_Pop_perc, GDP_per_capita)) +
  geom_point(colour = 'black') + labs(x = "Urban Population (% of Total Population)", y = "GDP ($ per ca
```

```
Education_plot <- ggplot(data = cor_world_data, mapping = aes(Tert_Ed_Enrol, GDP_per_capita)) +
  geom_point(colour = 'black') + labs(x = "Tertiary Education Enrollment (%)", y = "GDP ($ per capita")
Wage_plot <- ggplot(data = cor_world_data, mapping = aes(Min_Wage, GDP_per_capita)) +
  geom_point(colour = 'black') + labs(x = "Minimum Wage ($)", y = "GDP ($ per capita")

grid.arrange(LifeExp_plot, BirthRate_plot, Fertilty_plot, Infant_plot,
             Physician_plot, Urban_plot, Education_plot, Wage_plot, ncol = 4,
             top = "Features with Correlation > 0.5 to GDP per capita")
```



Modeling

Data Partitioning

The data is split into training and test sets. The training set has 70% of the cleaned data, and they are picked randomly. In order to duplicate the answers, the seed was set to a constant. The other 30% will be used for testing the data.

```
#trainData = split world_data into train and test by 80-20 rule
#install.packages("caret", dependencies = c("Depends", "Suggests"))
library(caret)
```

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

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
factors_gdp = world_data %>%
  select(CountryID, GDP_per_capita, Life_Expectancy, Birth_Rate, Fertility_Rate, Infant_Mortality, Phys

set.seed(1988)
splitData = caret::createDataPartition(factors_gdp$CountryID, p = 0.7, list=F, times=1)
trainWorld = factors_gdp[splitData,]
testWorld = factors_gdp[!row.names(factors_gdp) %in% row.names(trainWorld),]
testWorld = factors_gdp[-splitData,]
```

Linear Regression

There are a few models to choose to fit the data. The easiest one is the linear regression model. However, this model is expected to perform poorly because the feature grid plots above showed that not all of the features are linear. The root mean squared errors on the train and test sets are $13907.06 and $9417.50. This proves that the model does not fit the data very well. The expected GDP per capita of a country is plotted against the predicted numbers based on the linear regression model. The gray reference line is when expected is equal to predicted. Any country above this line is predicted higher and below this line is predicted lower than the measured/expected GDP per capita.

```
linear_model <- lm(GDP_per_capita ~ ., data=trainWorld)
summary(linear_model)
```

```
##
## Call:
## lm(formula = GDP_per_capita ~ ., data = trainWorld)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -20493  -8236  -1603   2866  55402
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -176316.82   47899.36  -3.681 0.000388 ***
## CountryID            45.76      35.94   1.273 0.206075
## Life_Expectancy    2184.10     588.00   3.714 0.000345 ***
## Birth_Rate        -1338.85     935.39  -1.431 0.155652
## Fertility_Rate     9354.98    6279.24   1.490 0.139618
## Infant_Mortality    607.26     221.83   2.738 0.007404 **
## Phys_per_Thou      2214.77    1782.63   1.242 0.217173
## Urban_Pop_perc    20997.96    9033.14   2.325 0.022249 *
## Tert_Ed_Enrol     -3212.82    9468.93  -0.339 0.735140
## Min_Wage           1806.62     575.03   3.142 0.002246 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 14630 on 94 degrees of freedom
## Multiple R-squared:  0.608,  Adjusted R-squared:  0.5704
## F-statistic:  16.2 on 9 and 94 DF,  p-value: 1.018e-15
```

```
lm_predicted_train <- predict(linear_model, newdata = trainWorld)
lm_rmse_train <- sqrt(mean((trainWorld$GDP_per_capita - lm_predicted_train)^2))
lm_rmse_train
```

```
## [1] 13907.06
```

```
lm_predicted_test <- predict(linear_model, newdata = testWorld)
lm_rmse_test <- sqrt(mean((testWorld$GDP_per_capita - lm_predicted_test)^2))
lm_rmse_test
```

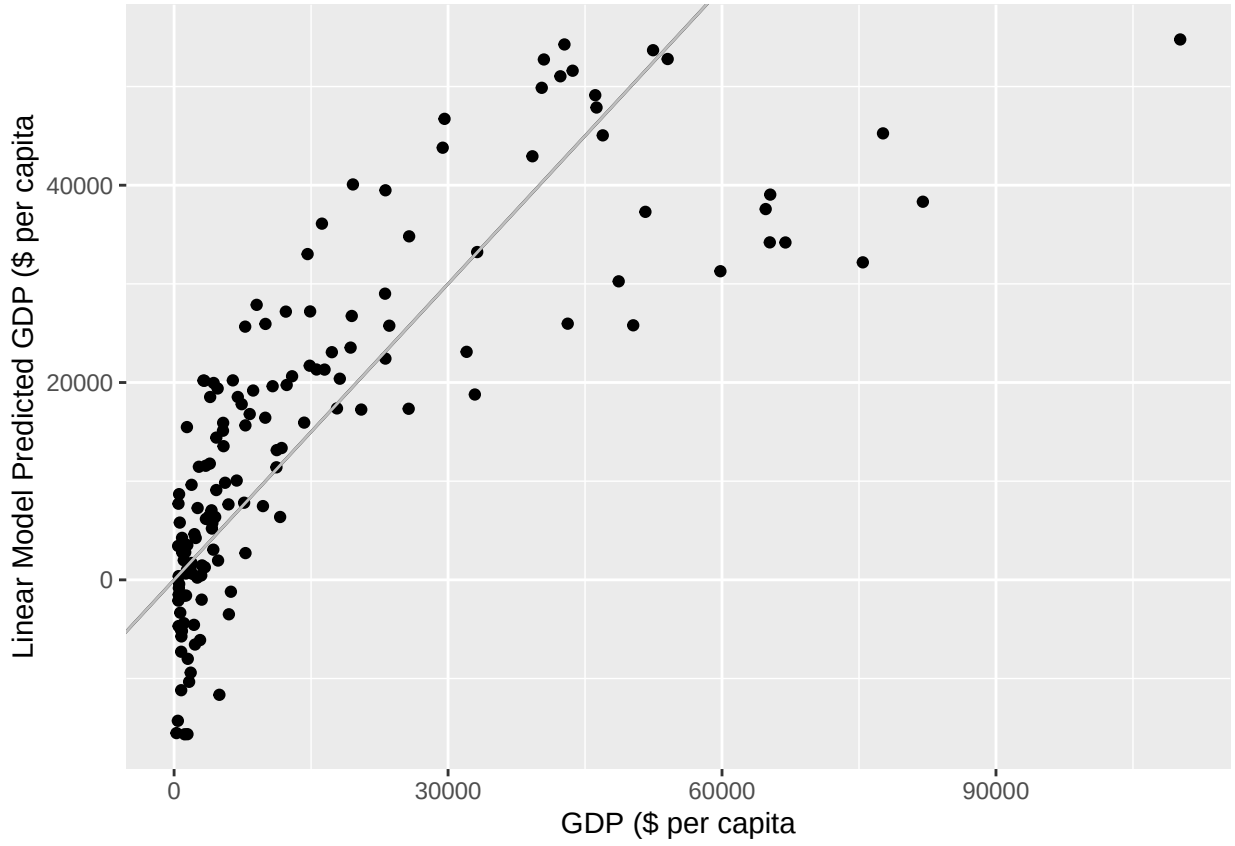
```
## [1] 9417.473
```

```
lm_predict_world <- data.frame("Linear_Model_GDP_per_capita" = predict(linear_model, newdata = factors_

lm_plot_data = data.frame("Measured GDP per capita" = c(factors_gdp$Linear_Model_GDP_per_capita), "Predi
```

```
## Warning: Unknown or uninitialised column: `Linear_Model_GDP_per_capita`.
```

```
ggplot(data = factors_gdp, mapping = aes(GDP_per_capita, lm_predict_world$Linear_Model_GDP_per_capita))
  geom_point(colour = 'black') + labs(x = "GDP ($ per capita", y = "Linear Model Predicted GDP ($ per ca
  geom_abline(slope = 1, intercept = 0) + geom_abline(colour= 'gray')
```



Random Forest Regressor A random forest regressor model is a great option for data that has non-linear

correlations. This model does better with both train and test sets: RMSE on train set is $5152.13 and test set is $8959.48. Plot below shows that most data fall close to the reference line but there are a few countries that are significantly off.

```
# Install the required package for function
#install.packages("randomForest")
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

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

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
train_predictors <- trainWorld %>%
  select(Life_Expectancy, Birth_Rate, Fertility_Rate, Infant_Mortality, Phys_per_Thou, Urban_Pop_perc, 
test_predictors <- testWorld %>%
  select(Life_Expectancy, Birth_Rate, Fertility_Rate, Infant_Mortality, Phys_per_Thou, Urban_Pop_perc, 

randforestReg <- randomForest(x = train_predictors, y = trainWorld$GDP_per_capita , maxnodes = 100, ntr
summary(randforestReg)
```

```
##                 Length Class  Mode
## call              5    -none- call
## type              1    -none- character
## predicted       104    -none- numeric
## mse             300    -none- numeric
## rsq             300    -none- numeric
## oob.times       104    -none- numeric
## importance        8    -none- numeric
## importanceSD      0    -none- NULL
## localImportance   0    -none- NULL
## proximity         0    -none- NULL
## ntree             1    -none- numeric
## mtry              1    -none- numeric
## forest           11    -none- list
## coefs             0    -none- NULL
## y               104    -none- numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
```

```
rf_predicted_train <- predict(randforestReg, newdata = train_predictors)
rf_rmse_train <- sqrt(mean((trainWorld$GDP_per_capita - rf_predicted_train)^2))
rf_rmse_train
```

```
## [1] 5418.429
```

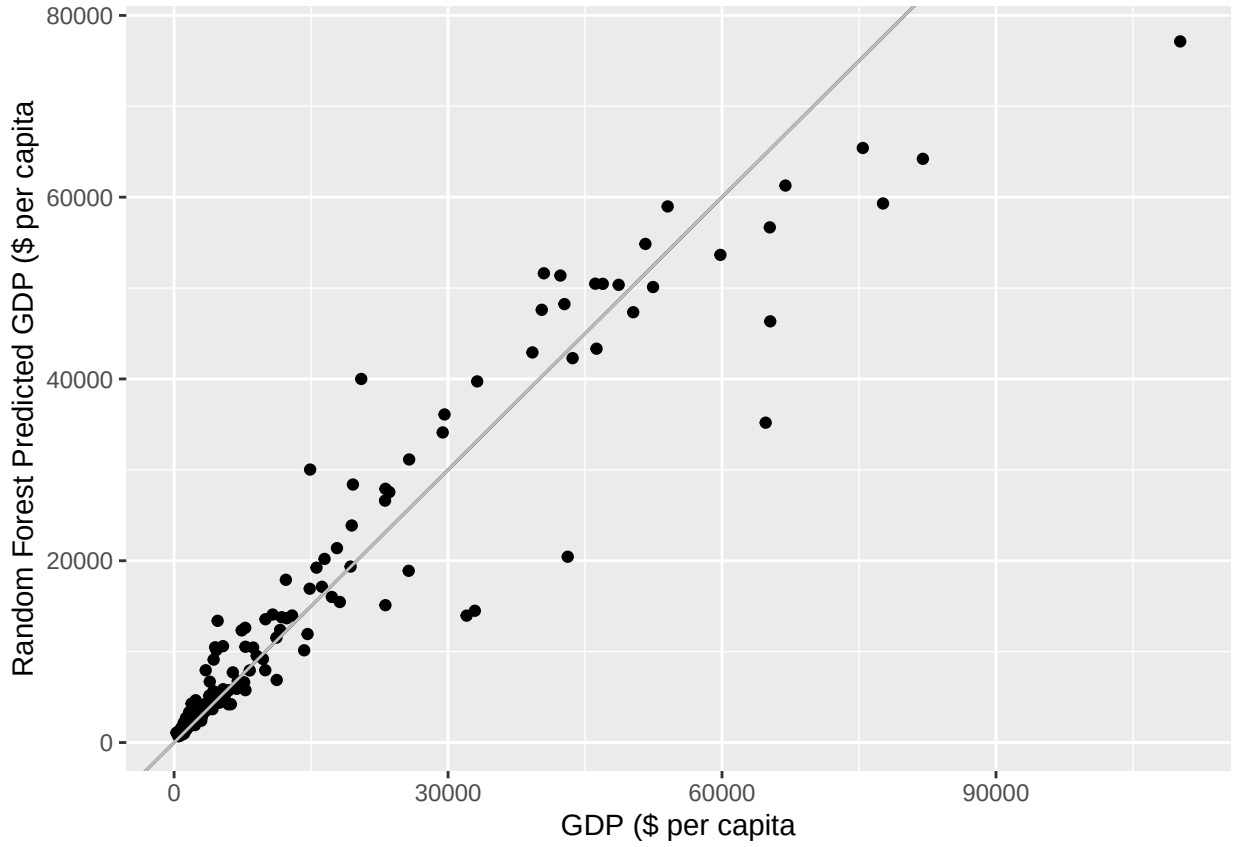
```
rf_predicted_test <- predict(randforestReg, newdata = test_predictors)
rf_rmse_test <- sqrt(mean((testWorld$GDP_per_capita - rf_predicted_test)^2))
rf_rmse_test
```

```
## [1] 8682.094
```

```
factors_only_no_gdp <- factors_gdp %>%
  select(Life_Expectancy, Birth_Rate, Fertility_Rate, Infant_Mortality, Phys_per_Thou, Urban_Pop_perc, 

rf_predict_world <- data.frame("Random_Forest_GDP_per_capita" = predict(randforestReg, newdata = factor

ggplot(data = factors_gdp, mapping = aes(GDP_per_capita, rf_predict_world$Random_Forest_GDP_per_capita)
  geom_point(colour = 'black') + labs(x = "GDP ($ per capita", y = "Random Forest Predicted GDP ($ per 
  geom_abline(slope = 1, intercept = 0) + geom_abline(colour= 'gray')
```




```
rf_plot_data = data.frame("Measured GDP per capita" = c(factors_gdp$GDP_per_capita), "Predicted GDP per
```

```r
gdp_data <- cbind(world_data, lm_predict_world, rf_predict_world)
save(gdp_data, file = "gdp_data.RData", version = 2)
```

Summary

Predicting GDP per capita is indeed challenging. There are quite a few correlated variables that are not linear. Therefore, it is not suprising that the Linear Regression model does a poor job relative to the Random Forest Regressor. The correlation matrix showed that out of 32 variables, 8 features are useful for fitting models. If there were more features or more countries in the data, the prediction with the Random Forest Regressor could be improved.

References

[1] Data downloaded from: https://www.kaggle.com/datasets/nelgiriyewithana/countries-of-the-world-2023 [2] Correlation: https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html [3] https://cran.r-project.org/web/packages/egg/vignettes/Ecosystem.html [4] https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/rowsum [5] Linear Model: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/lm [6] Random Forest: https://www.rdocumentation.org/packages/randomForest/versions/4.7-1.1/topics/randomForest