World Happiness Report 2021

Zaryn Ooi

9/6/2021

What makes people in a country happy?

Task

Create a model that accurately predicts the happiness of countries around the world.

Data Source

World Happiness Report 2021

https://www.kaggle.com/ajaypalsinghlo/world-happiness-report-2021

Context

The World Happiness Report is a landmark survey of the state of global happiness. The report continues to gain global recognition as governments, organizations and civil society increasingly use happiness indicators to inform their policy-making decisions. Leading experts across fields – economics, psychology, survey analysis, national statistics, health, public policy and more – describe how measurements of well-being can be used effectively to assess the progress of nations. The reports review the state of happiness in the world today and show how the new science of happiness explains personal and national variations in happiness.

Content

The happiness scores and rankings use data from the Gallup World Poll . The columns following the happiness score estimate the extent to which each of six factors – Economic Production, Social Support, Life Expectancy, Freedom, Absence of Corruption, and Generosity.

Although there are 20 variables in this dataset, my interest will be on 7 key variables.

- Country Name: The names of the countries.
- Ladder Score: Happiness score or subjective well-being. This is the national average response to the question of life evaluations.
- Logged GDP per capita: The GDP-per-capita time series from 2019 to 2020 using countryspecific forecasts of real GDP growth in 2020.
- Social Support: Social support refers to assistance or support provided by members of social networks to an individual.
- Healthy Life Expectancy: Healthy life expectancy is the average life in good health.
- Freedom to Make Life Choices: Freedom to make life choices is the national average of binary responses to the GWP question "Are you satisfied or dissatisfied with your freedom to choose what you do with your life?"

- Generosity: Generosity is the residual of regressing national average of response to the GWP question "Have you donated money to a charity in the past month?" on GDP per capita.?
- Perceptions of Corruption: The measure is the national average of the survey responses to two questions in the GWP: "Is corruption widespread throughout the government or not" and "Is corruption widespread within businesses or not?"

Install and Load Packages

```r
install.packages("pacman")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

```r
pacman::p_load(
  caret,
  corrplot,
  GGally,
  magrittr,
  pacman,
  parallel,
  randomForest,
  rattle,
  rio,
  tictoc,
  tidyverse,
  rpart,
  rpart.plot
)
```

```r
library(readr)
```

```r
install.packages("olsrr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

```r
library(olsrr)
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

Import Data

```r
data <- read_csv("world-happiness-report-2021.csv")
```

```
## Rows: 149 Columns: 20
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): Country name, Regional indicator
## dbl (18): Ladder score, Standard error of ladder score, upperwhisker, lowerw...
```

```
##
```

```
## 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.
```

```r
# Explore the imported data
head(data)
```

```
## # A tibble: 6 x 20
##   `Country name` `Regional indica~ `Ladder score` `Standard error ~ upperwhisker
##   <chr>          <chr>                      <dbl>             <dbl>        <dbl>
## 1 Finland        Western Europe              7.84             0.032         7.90
## 2 Denmark        Western Europe              7.62             0.035         7.69
## 3 Switzerland    Western Europe              7.57             0.036         7.64
## 4 Iceland        Western Europe              7.55             0.059         7.67
## 5 Netherlands    Western Europe              7.46             0.027         7.52
## 6 Norway         Western Europe              7.39             0.035         7.46
## # ... with 15 more variables: lowerwhisker <dbl>, Logged GDP per capita <dbl>,
## #   Social support <dbl>, Healthy life expectancy <dbl>,
## #   Freedom to make life choices <dbl>, Generosity <dbl>,
## #   Perceptions of corruption <dbl>, Ladder score in Dystopia <dbl>,
## #   Explained by: Log GDP per capita <dbl>, Explained by: Social support <dbl>,
## #   Explained by: Healthy life expectancy <dbl>,
## #   Explained by: Freedom to make life choices <dbl>, ...
```

```r
str(data)
```

```
## spec_tbl_df [149 x 20] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Country name                              : chr [1:149] "Finland" "Denmark" "Switzerland" "Icelan
##  $ Regional indicator                        : chr [1:149] "Western Europe" "Western Europe" "Wester
##  $ Ladder score                              : num [1:149] 7.84 7.62 7.57 7.55 7.46 ...
##  $ Standard error of ladder score            : num [1:149] 0.032 0.035 0.036 0.059 0.027 0.035 0.036
##  $ upperwhisker                              : num [1:149] 7.9 7.69 7.64 7.67 7.52 ...
##  $ lowerwhisker                              : num [1:149] 7.78 7.55 7.5 7.44 7.41 ...
##  $ Logged GDP per capita                     : num [1:149] 10.8 10.9 11.1 10.9 10.9 ...
##  $ Social support                            : num [1:149] 0.954 0.954 0.942 0.983 0.942 0.954 0.934
##  $ Healthy life expectancy                   : num [1:149] 72 72.7 74.4 73 72.4 73.3 72.7 72.6 73.4
##  $ Freedom to make life choices              : num [1:149] 0.949 0.946 0.919 0.955 0.913 0.96 0.945
##  $ Generosity                                : num [1:149] -0.098 0.03 0.025 0.16 0.175 0.093 0.086
##  $ Perceptions of corruption                 : num [1:149] 0.186 0.179 0.292 0.673 0.338 0.27 0.237
##  $ Ladder score in Dystopia                  : num [1:149] 2.43 2.43 2.43 2.43 2.43 2.43 2.43 2.43 2
##  $ Explained by: Log GDP per capita          : num [1:149] 1.45 1.5 1.57 1.48 1.5 ...
##  $ Explained by: Social support              : num [1:149] 1.11 1.11 1.08 1.17 1.08 ...
##  $ Explained by: Healthy life expectancy     : num [1:149] 0.741 0.763 0.816 0.772 0.753 0.782 0.763
##  $ Explained by: Freedom to make life choices: num [1:149] 0.691 0.686 0.653 0.698 0.647 0.703 0.685
##  $ Explained by: Generosity                  : num [1:149] 0.124 0.208 0.204 0.293 0.302 0.249 0.244
##  $ Explained by: Perceptions of corruption   : num [1:149] 0.481 0.485 0.413 0.17 0.384 0.427 0.448
##  $ Dystopia + residual                       : num [1:149] 3.25 2.87 2.84 2.97 2.8 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `Country name` = col_character(),
##   ..   `Regional indicator` = col_character(),
##   ..   `Ladder score` = col_double(),
##   ..   `Standard error of ladder score` = col_double(),
##   ..   upperwhisker = col_double(),
##   ..   lowerwhisker = col_double(),
##   ..   `Logged GDP per capita` = col_double(),
##   ..   `Social support` = col_double(),
```

```
##   ..   `Healthy life expectancy` = col_double(),
##   ..   `Freedom to make life choices` = col_double(),
##   ..   Generosity = col_double(),
##   ..   `Perceptions of corruption` = col_double(),
##   ..   `Ladder score in Dystopia` = col_double(),
##   ..   `Explained by: Log GDP per capita` = col_double(),
##   ..   `Explained by: Social support` = col_double(),
##   ..   `Explained by: Healthy life expectancy` = col_double(),
##   ..   `Explained by: Freedom to make life choices` = col_double(),
##   ..   `Explained by: Generosity` = col_double(),
##   ..   `Explained by: Perceptions of corruption` = col_double(),
##   ..   `Dystopia + residual` = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Data Cleaning

```r
# Remove unnecessary columns
data <- data[ -c(2,4:6,13:20) ]
```

```r
# Rename column names
colnames(data)<-
  c("country","ladder","GDP","social","healthy","freedom","generosity","corruption")

head(data)
```

```
## # A tibble: 6 x 8
##   country     ladder   GDP social healthy freedom generosity corruption
##   <chr>        <dbl> <dbl>  <dbl>   <dbl>   <dbl>      <dbl>      <dbl>
## 1 Finland       7.84  10.8  0.954    72     0.949     -0.098      0.186
## 2 Denmark       7.62  10.9  0.954    72.7   0.946      0.03       0.179
## 3 Switzerland   7.57  11.1  0.942    74.4   0.919      0.025      0.292
## 4 Iceland       7.55  10.9  0.983    73     0.955      0.16       0.673
## 5 Netherlands   7.46  10.9  0.942    72.4   0.913      0.175      0.338
## 6 Norway        7.39  11.1  0.954    73.3   0.96       0.093      0.27
```

```r
# Check NA values
is.null(data)
```

```
## [1] FALSE
```

```r
data %>% na.omit(data)
```

```
## # A tibble: 149 x 8
##    country     ladder   GDP social healthy freedom generosity corruption
##    <chr>        <dbl> <dbl>  <dbl>   <dbl>   <dbl>      <dbl>      <dbl>
##  1 Finland       7.84  10.8  0.954    72     0.949     -0.098      0.186
##  2 Denmark       7.62  10.9  0.954    72.7   0.946      0.03       0.179
##  3 Switzerland   7.57  11.1  0.942    74.4   0.919      0.025      0.292
##  4 Iceland       7.55  10.9  0.983    73     0.955      0.16       0.673
##  5 Netherlands   7.46  10.9  0.942    72.4   0.913      0.175      0.338
##  6 Norway        7.39  11.1  0.954    73.3   0.96       0.093      0.27
##  7 Sweden        7.36  10.9  0.934    72.7   0.945      0.086      0.237
##  8 Luxembourg    7.32  11.6  0.908    72.6   0.907     -0.034      0.386
##  9 New Zealand   7.28  10.6  0.948    73.4   0.929      0.134      0.242
## 10 Austria       7.27  10.9  0.934    73.3   0.908      0.042      0.481
```

```
## # ... with 139 more rows
```

Summary statistics of the data

```
data %>% summary()
```

```
##    country              ladder           GDP              social      
##  Length:149         Min.   :2.523   Min.   : 6.635   Min.   :0.4630  
##  Class :character   1st Qu.:4.852   1st Qu.: 8.541   1st Qu.:0.7500  
##  Mode  :character   Median :5.534   Median : 9.569   Median :0.8320  
##                     Mean   :5.533   Mean   : 9.432   Mean   :0.8147  
##                     3rd Qu.:6.255   3rd Qu.:10.421   3rd Qu.:0.9050  
##                     Max.   :7.842   Max.   :11.647   Max.   :0.9830  
##     healthy         freedom         generosity         corruption    
##  Min.   :48.48   Min.   :0.3820   Min.   :-0.28800   Min.   :0.0820  
##  1st Qu.:59.80   1st Qu.:0.7180   1st Qu.:-0.12600   1st Qu.:0.6670  
##  Median :66.60   Median :0.8040   Median :-0.03600   Median :0.7810  
##  Mean   :64.99   Mean   :0.7916   Mean   :-0.01513   Mean   :0.7274  
##  3rd Qu.:69.60   3rd Qu.:0.8770   3rd Qu.: 0.07900   3rd Qu.:0.8450  
##  Max.   :76.95   Max.   :0.9700   Max.   : 0.54200   Max.   :0.9390  
```

The max ladder score is 7.842, mean is 5.533, median is 5.534, and the min is 2.523. For all our models, the variable "ladder" will be the dependant variable and expressed as the Happiness score for the entire project.

Data Visualization

Happiest Countries

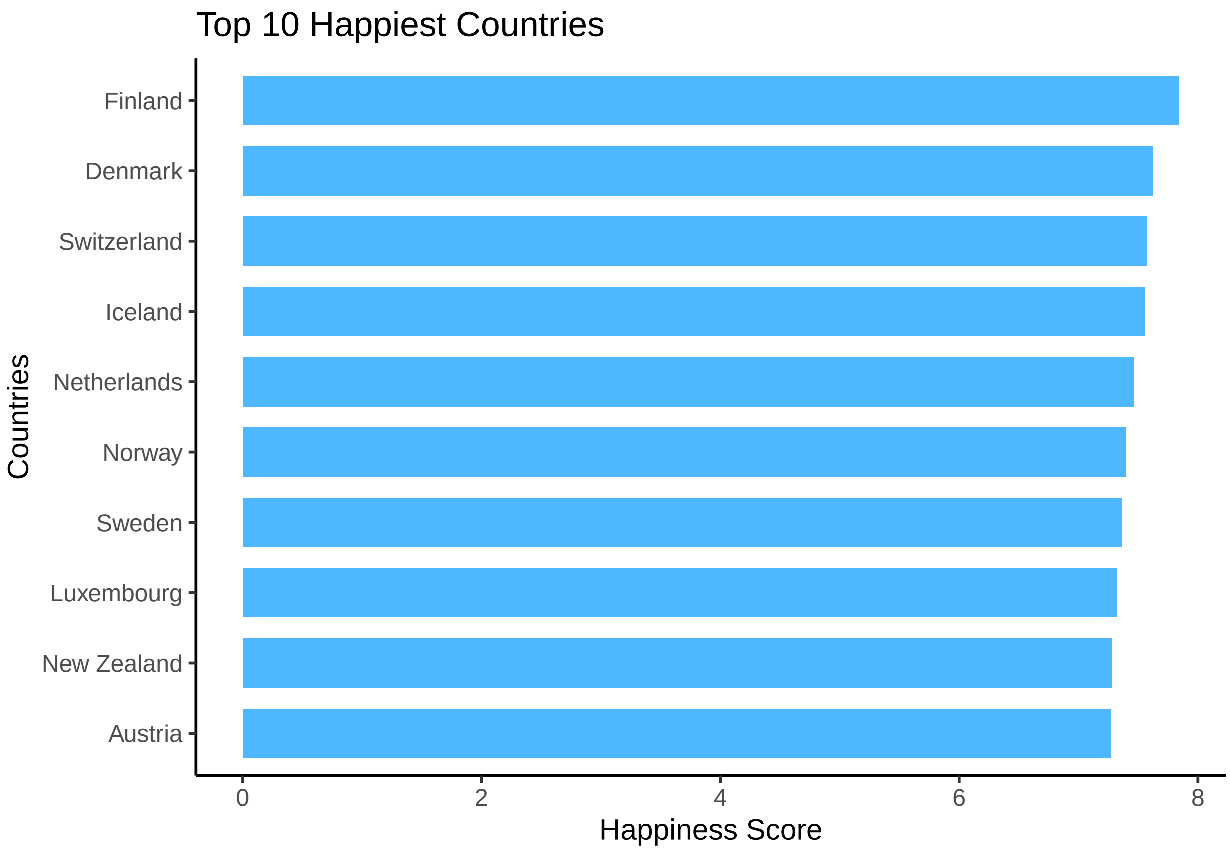
I will create a barchart to visualize the top 5 and bottom 5 rank by cuntries.

```
# top 10
top10 <- data %>%
  select(country, ladder) %>%
  arrange(desc(ladder)) %>%
  slice(1:10)
print(top10)
```

```
## # A tibble: 10 x 2
##    country     ladder
##    <chr>        <dbl>
##  1 Finland       7.84
##  2 Denmark       7.62
##  3 Switzerland   7.57
##  4 Iceland       7.55
##  5 Netherlands   7.46
##  6 Norway        7.39
##  7 Sweden        7.36
##  8 Luxembourg    7.32
##  9 New Zealand   7.28
## 10 Austria       7.27
```

```
# Barchart to visualize the top 10 happiest countries
top10 %>%
  ggplot(aes(fct_reorder(country, ladder), ladder)) +
  geom_bar(stat = "identity", fill = "#0099FF", width = 0.7, alpha = 0.7) + labs( x= "Countries",
        y= "Happiness Score",
```

```
      title =  "Top 10 Happiest Countries")  +
  theme(axis.title.y = element_blank()) +
  theme_classic() +
  coord_flip()
```



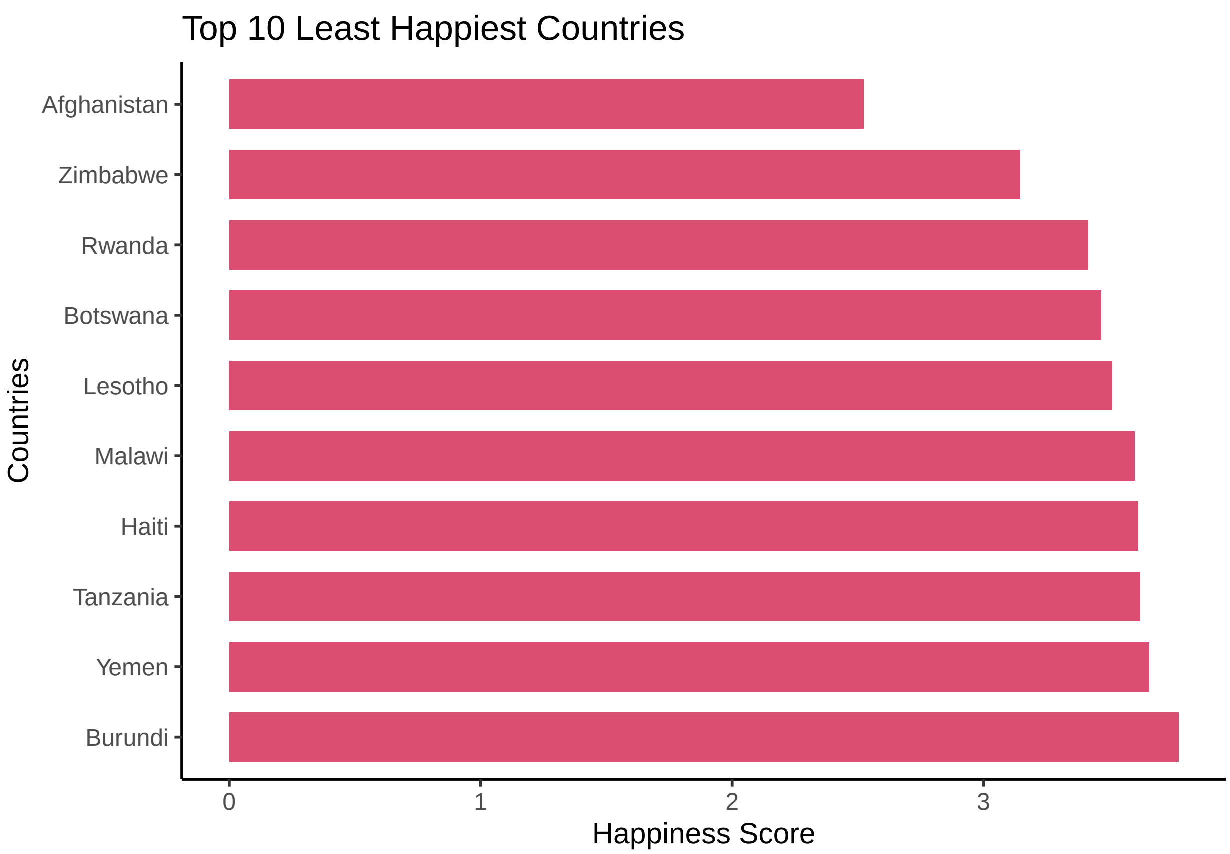
Least Happiest Countries

```
# bottom 10
bottom10 <- data %>%
  select(country, ladder) %>%
  arrange(ladder) %>%
  slice(1:10)
print(bottom10)
```

```
## # A tibble: 10 x 2
##    country     ladder
##    <chr>        <dbl>
##  1 Afghanistan   2.52
##  2 Zimbabwe      3.14
##  3 Rwanda        3.42
##  4 Botswana      3.47
##  5 Lesotho       3.51
##  6 Malawi        3.6
##  7 Haiti         3.62
##  8 Tanzania      3.62
##  9 Yemen         3.66
```

```
## 10 Burundi       3.78
```

```r
# Barchart to visualize the bottom 5 happiest country
bottom10 %>%
  ggplot(aes(fct_reorder(country, -ladder), ladder)) +
  geom_bar(stat = "identity", fill = "#CC0033", width = 0.7, alpha = 0.7) + labs( x= "Countries",
        y= "Happiness Score",
        title =  "Top 10 Least Happiest Countries")  +
  theme(axis.title.y = element_blank()) +
  theme_classic() +
  coord_flip()
```



The happiest country was Finland with a ladder score of 7.84 and the least happy country was Afghanistan with a score of 2.52.

Show Correlation

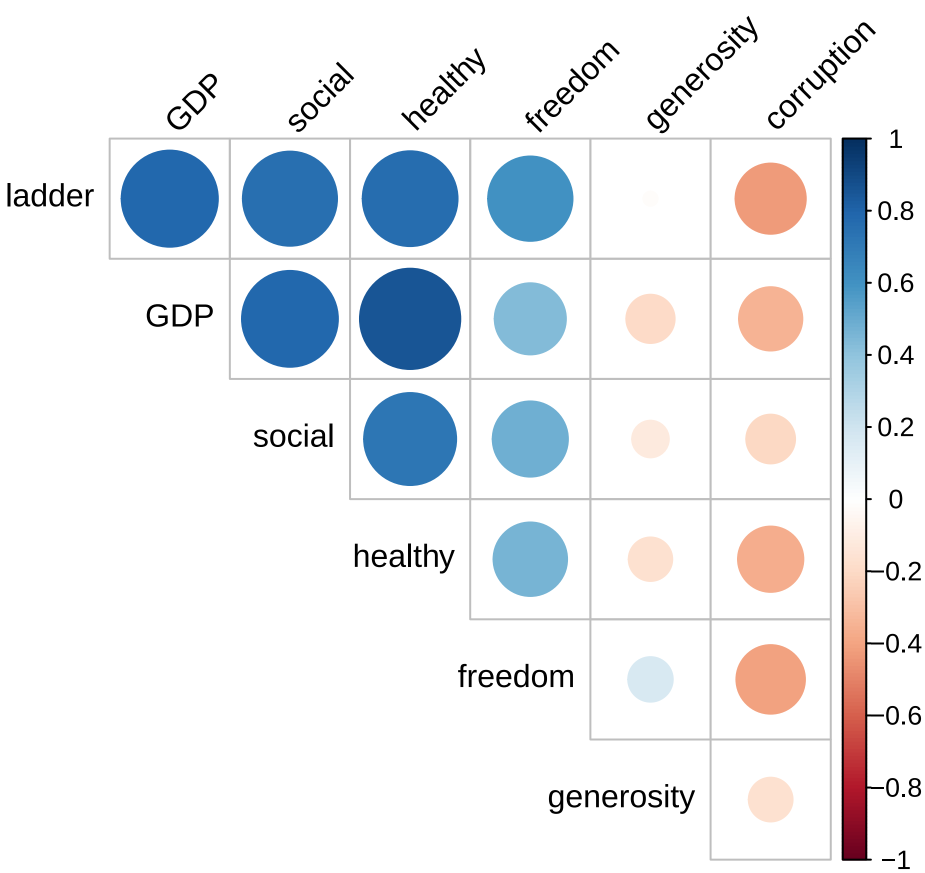
```r
# Remove "country" variable
df <- data %>%
  select(-1)
print(df)
```

```
## # A tibble: 149 x 7
##    ladder   GDP social healthy freedom generosity corruption
##     <dbl> <dbl>  <dbl>   <dbl>   <dbl>      <dbl>      <dbl>
##  1   7.84  10.8  0.954    72     0.949     -0.098      0.186
##  2   7.62  10.9  0.954    72.7   0.946      0.03       0.179
##  3   7.57  11.1  0.942    74.4   0.919      0.025      0.292
```

```
## 4    7.55  10.9  0.983   73     0.955     0.16      0.673
## 5    7.46  10.9  0.942   72.4   0.913     0.175     0.338
## 6    7.39  11.1  0.954   73.3   0.96      0.093     0.27
## 7    7.36  10.9  0.934   72.7   0.945     0.086     0.237
## 8    7.32  11.6  0.908   72.6   0.907    -0.034     0.386
## 9    7.28  10.6  0.948   73.4   0.929     0.134     0.242
## 10   7.27  10.9  0.934   73.3   0.908     0.042     0.481
## # ... with 139 more rows
```

Correlation Matrix

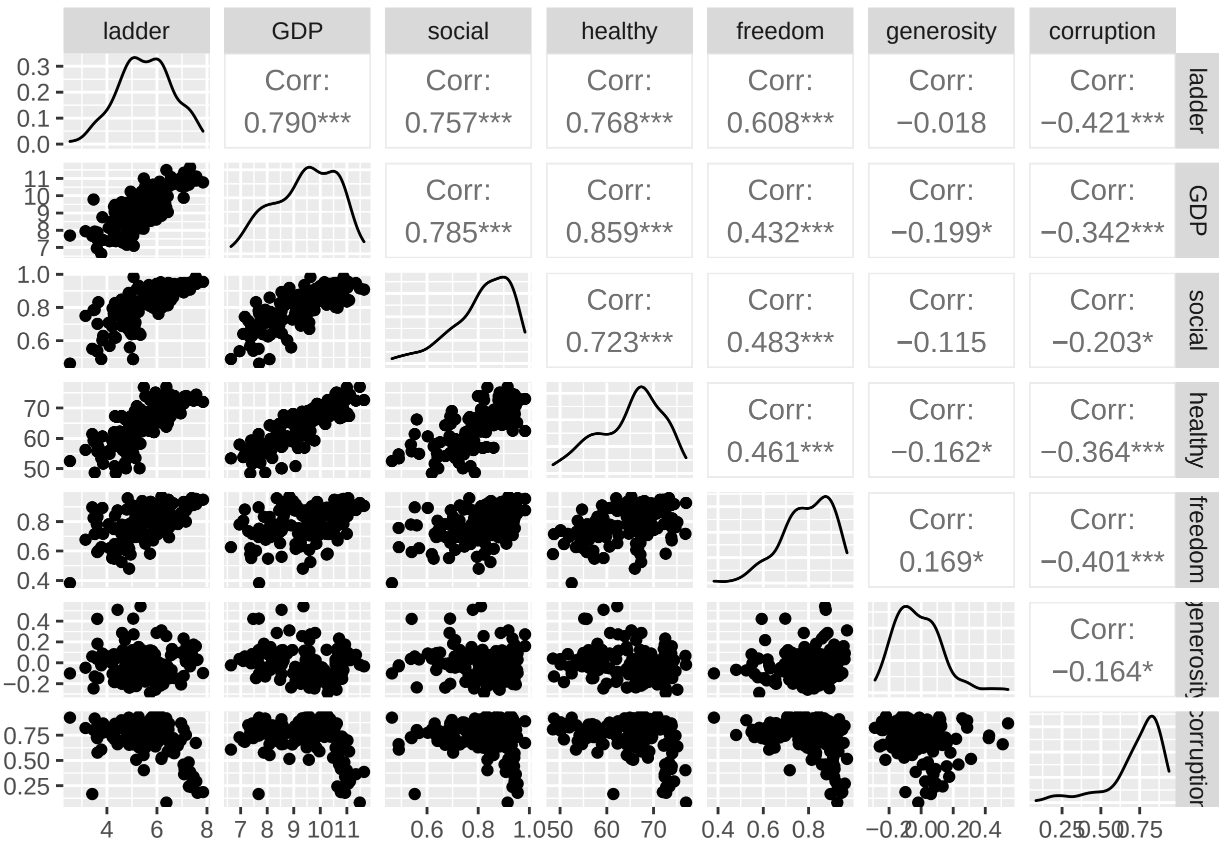
```r
# Visualize Correlation Matrix
df %>%
  cor() %>%
  corrplot(
    type   = "upper",
    diag   = F,
    order  = "original",
    tl.col = "black",
    tl.srt = 45
  )
```



Scatterplot Matrix

```r
# Scatterplot Matrix
df %>%
```

```
select(
    ladder,
    GDP:corruption
    ) %>%
  ggpairs()
```



Based on the Correlation Matrix and Scatterplot Matrix, it can be seen that most variables are positively correlated, except for "Genorisity" and "Perceptions of Corruption".

Model Building

Set Seed

```
set.seed(123)
```

Split data into train data and test data

```
train <- df %>% sample_frac(.70)
test  <- anti_join(df, train)
```

```
## Joining, by = c("ladder", "GDP", "social", "healthy", "freedom", "generosity", "corruption")
```

Linear Regression

For our first model, I will used our training data to perform a linear regression model with ladder score as the outcome variabe and the rest variables as the predictor variables.

```r
# Compute multiple regression with train data
lm <- lm(ladder ~ ., data=train)
```

```r
# Summarize Regression Model
summary(lm)
```

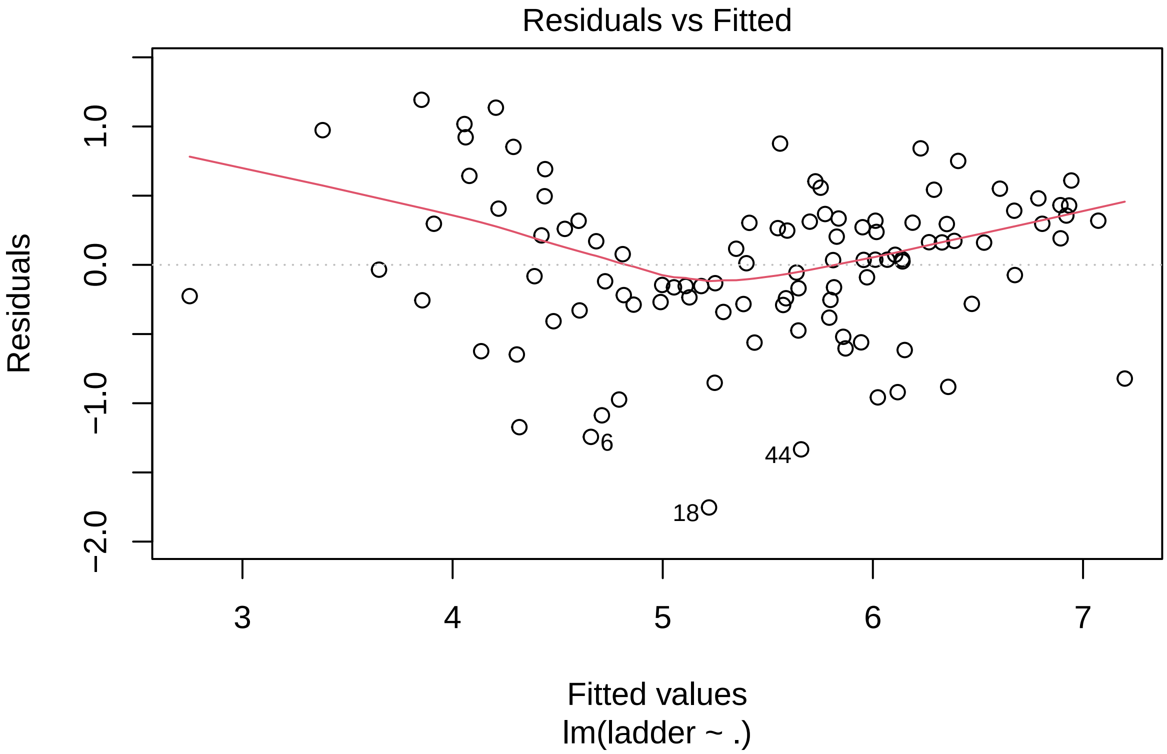
```
##
## Call:
## lm(formula = ladder ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.75318 -0.28217  0.03734  0.31907  1.19293
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.71824    0.81710  -3.327  0.00124 **
## GDP          0.24880    0.10847   2.294  0.02397 *
## social       2.55467    0.78631   3.249  0.00159 **
## healthy      0.03826    0.01716   2.229  0.02810 *
## freedom      1.95679    0.64859   3.017  0.00326 **
## generosity   0.27553    0.42365   0.650  0.51700
## corruption  -0.38734    0.37821  -1.024  0.30832
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5732 on 97 degrees of freedom
## Multiple R-squared:  0.7367, Adjusted R-squared:  0.7204
## F-statistic: 45.23 on 6 and 97 DF,  p-value: < 2.2e-16
```

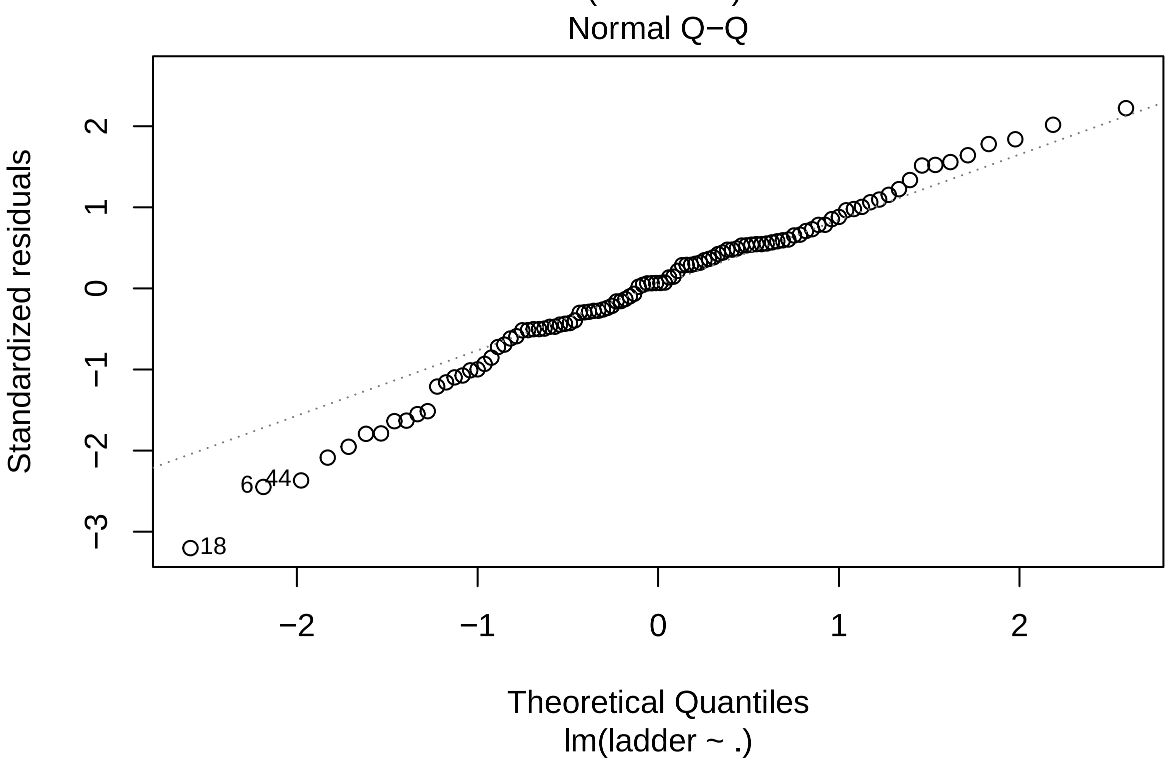
The linear model returned an R-squared value of 73.67% and an adjusted R-squared value of 72.04%, meaning that 73.67% of the variance of ladder can be explained by the predictor variables. This model predicted that "GDP","social support","healthy life expectancy",and "freedom to make life choices" were the most impactful attributes as they have shown to be statistically significant. The p-value is less than .05 which is statistically significant. Therefore, the predictor variables gives us a reliable estimate in determining the ladder score (happiness).

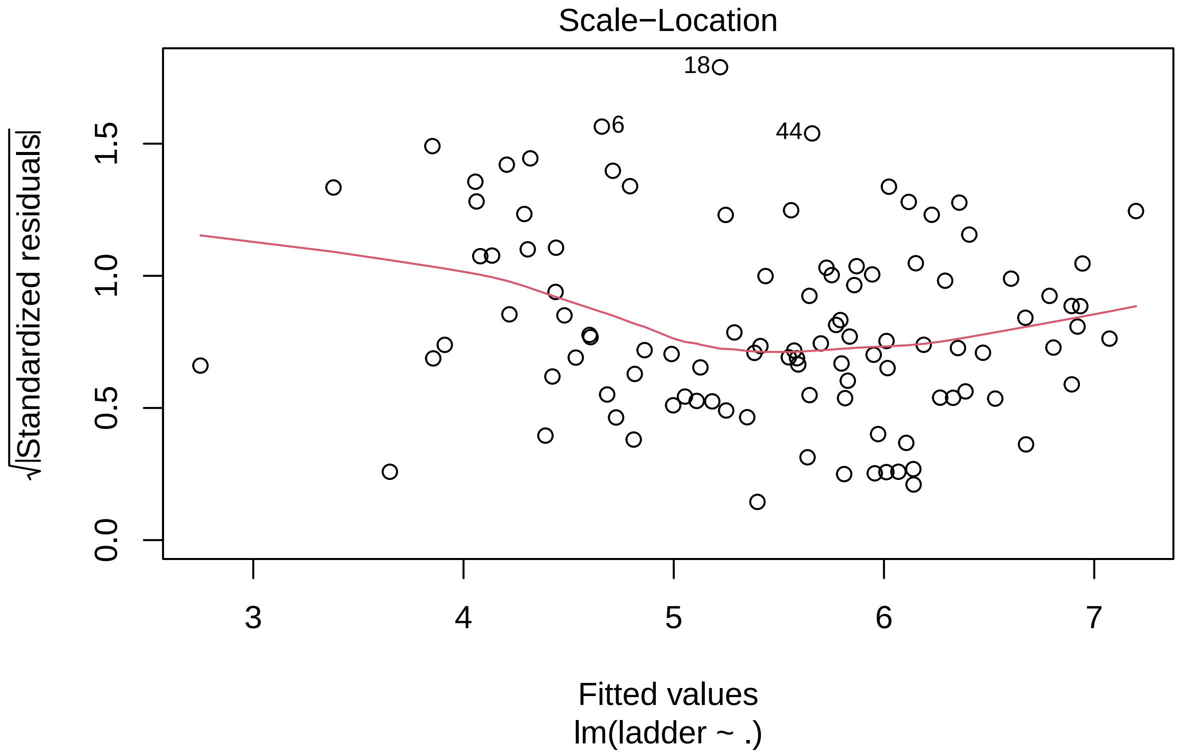
Diagnostic Plot

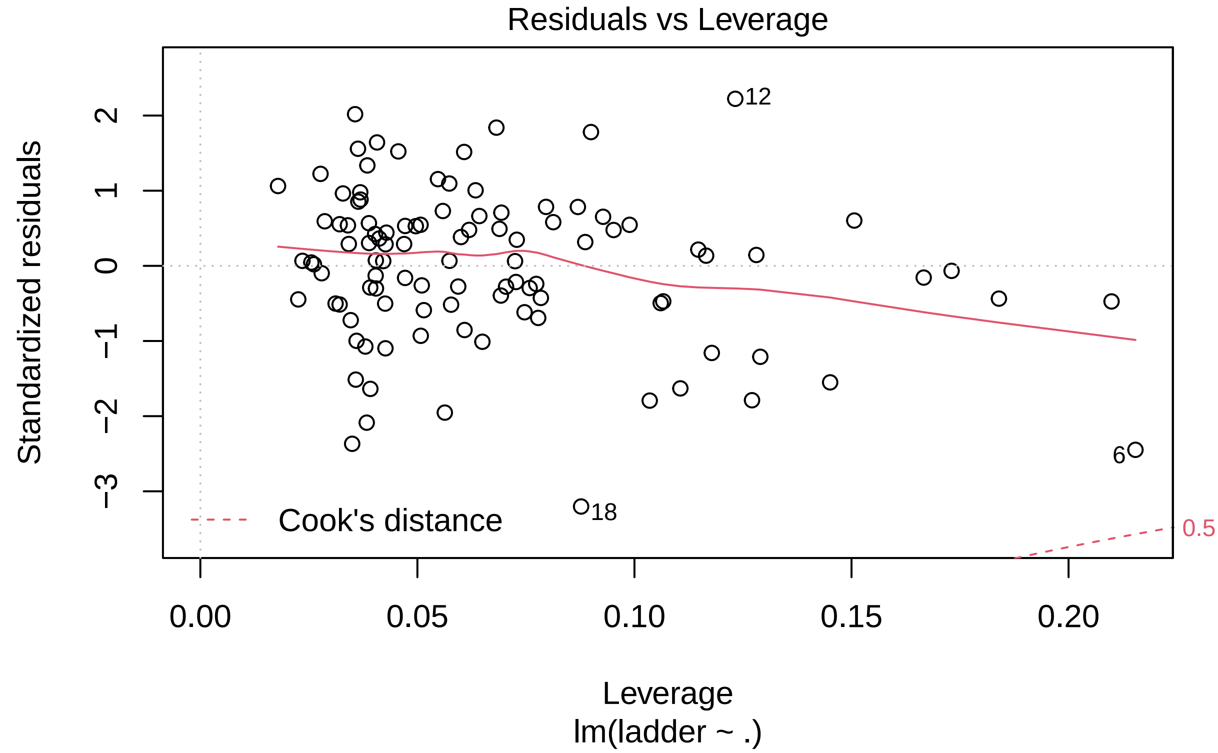
Although this linear regression model appears to be fairly accurate, it needs to be verified that the data it is being applied to is normally distributed. If the data are not normally distributed, the model results cannot be accurately applied when using a linear regression model.

```r
# Diagnostic Plot
lm %>% plot()
```


Residuals vs Fitted
Residuals
1.0
0.0
−1.0
−2.0
3
4
5
6
7
6
44
18
Fitted values
lm(ladder ~ .)


Normal Q−Q
Standardized residuals
2
1
0
−1
−2
−3
−2
−1
0
1
2
6
44
18
Theoretical Quantiles
lm(ladder ~ .)





Diagnostic Plot 2 - Normal Q-Q I will focus on the Normal Q-Q plot. This plot is used to examine whether the residuals are normally distributed. It's good if residuals points follow the straight dashed line. In our example, the points fall roughly along the straight diagonal line. The observations #44 and #6 and #18 deviate a bit from the line at the tail ends, but not enough to declare that the residuals are non-normally distributed. Therefore, Normal Q-Q plot confirms that the data is normally distributed.

```r
# Apply Linear Regression on Test Data
lm_p <- predict(
lm, newdata = test
)
```

```r
# Get predicted happiness values of countries
lm_p
```

```
##        1        2        3        4        5        6        7        8
## 6.912169 7.010368 6.992547 6.881776 6.857481 6.321616 6.378159 6.178791
##        9       10       11       12       13       14       15       16
## 5.993696 6.571705 5.554811 5.818675 6.100920 5.770912 5.462212 6.030545
##       17       18       19       20       21       22       23       24
## 5.606186 6.292585 5.941151 6.232158 5.468386 5.512408 5.567835 5.615210
##       25       26       27       28       29       30       31       32
## 5.661793 5.459828 5.485949 5.572017 4.109677 5.027174 4.761317 4.258754
##       33       34       35       36       37       38       39       40
## 5.075184 4.048186 5.247374 4.324227 5.079223 4.799337 5.264299 3.865485
##       41       42       43       44       45
## 5.007668 4.410174 3.588909 3.807507 3.210472
```

Compute accuracy of Linear Regression Model

I will calculate the Root Mean Squared Error(RMSE), which measures the model prediction error to assess the performance of the models. It corresponds to the average difference between the observed known values of the outcome and the predicted value by the model. The lower the RMSE, the better the model.

```r
# Compute RMSE for linear regression model
linear.rmse <- RMSE(lm_p, test$ladder)
```

```r
linear.rmse
```

```
## [1] 0.4938069
```

The Root Mean-Squared Error (RMSE) for this linear regression model was 44.95%.

```r
# Create a table to save our results for each model
accuracy_results <- tibble(method = "Linear_Regression", RMSE = linear.rmse)
```

```r
# View the accuracy table
accuracy_results %>% knitr::kable()
```

method	RMSE
Linear_Regression	0.4938069

Selecting predictors for multiple regression

All Possible Regression

```r
# Criteria for all possible model combination
all.mod <- ols_step_all_possible(lm)
all.mod
```

```
##    Index N                                              Predictors     R-Square
## 1      1 1                                                     GDP 6.095380e-01
## 3      2 1                                                 healthy 5.916306e-01
## 2      3 1                                                  social 5.484425e-01
```

```
## 4       4 1                                         freedom 3.512322e-01
## 6       5 1                                      corruption 1.501406e-01
## 5       6 1                                      generosity 2.094314e-05
## 9       7 2                                     GDP freedom 6.869448e-01
## 12      8 2                                  social healthy 6.769648e-01
## 7       9 2                                      GDP social 6.634078e-01
## 16     10 2                                 healthy freedom 6.618479e-01
## 8      11 2                                     GDP healthy 6.456668e-01
## 11     12 2                                  GDP corruption 6.277717e-01
## 10     13 2                                  GDP generosity 6.236466e-01
## 13     14 2                                  social freedom 6.230341e-01
## 15     15 2                               social corruption 6.154322e-01
## 18     16 2                              healthy corruption 6.026683e-01
## 17     17 2                              healthy generosity 5.954217e-01
## 14     18 2                               social generosity 5.536086e-01
## 20     19 2                              freedom corruption 3.707823e-01
## 19     20 2                              freedom generosity 3.616716e-01
## 21     21 2                           generosity corruption 1.564083e-01
## 32     22 3                          social healthy freedom 7.169887e-01
## 23     23 3                              GDP social freedom 7.154543e-01
## 26     24 3                             GDP healthy freedom 7.063519e-01
## 34     25 3                       social healthy corruption 6.967414e-01
## 22     26 3                              GDP social healthy 6.905457e-01
## 25     27 3                           GDP social corruption 6.899316e-01
## 29     28 3                          GDP freedom generosity 6.889775e-01
## 30     29 3                          GDP freedom corruption 6.885757e-01
## 33     30 3                       social healthy generosity 6.830025e-01
## 24     31 3                           GDP social generosity 6.760075e-01
## 39     32 3                      healthy freedom corruption 6.622466e-01
## 38     33 3                      healthy freedom generosity 6.618522e-01
## 28     34 3                          GDP healthy corruption 6.558752e-01
## 27     35 3                          GDP healthy generosity 6.556813e-01
## 36     36 3                       social freedom corruption 6.487849e-01
## 31     37 3                       GDP generosity corruption 6.349169e-01
## 35     38 3                       social freedom generosity 6.230446e-01
## 37     39 3                    social generosity corruption 6.155773e-01
## 40     40 3                   healthy generosity corruption 6.039326e-01
## 41     41 3                   freedom generosity corruption 3.863264e-01
## 42     42 4                      GDP social healthy freedom 7.317869e-01
## 53     43 4               social healthy freedom corruption 7.221967e-01
## 46     44 4                   GDP social freedom corruption 7.218166e-01
## 45     45 4                   GDP social freedom generosity 7.181649e-01
## 52     46 4               social healthy freedom generosity 7.176807e-01
## 44     47 4                   GDP social healthy corruption 7.079723e-01
## 48     48 4                  GDP healthy freedom generosity 7.077874e-01
## 49     49 4                  GDP healthy freedom corruption 7.068443e-01
## 43     50 4                   GDP social healthy generosity 6.998540e-01
## 54     51 4            social healthy generosity corruption 6.986154e-01
## 47     52 4                GDP social generosity corruption 6.946762e-01
## 51     53 4               GDP freedom generosity corruption 6.899931e-01
## 56     54 4           healthy freedom generosity corruption 6.622843e-01
## 50     55 4               GDP healthy generosity corruption 6.616481e-01
## 55     56 4            social freedom generosity corruption 6.492007e-01
## 58     57 5           GDP social healthy freedom corruption 7.355192e-01
```

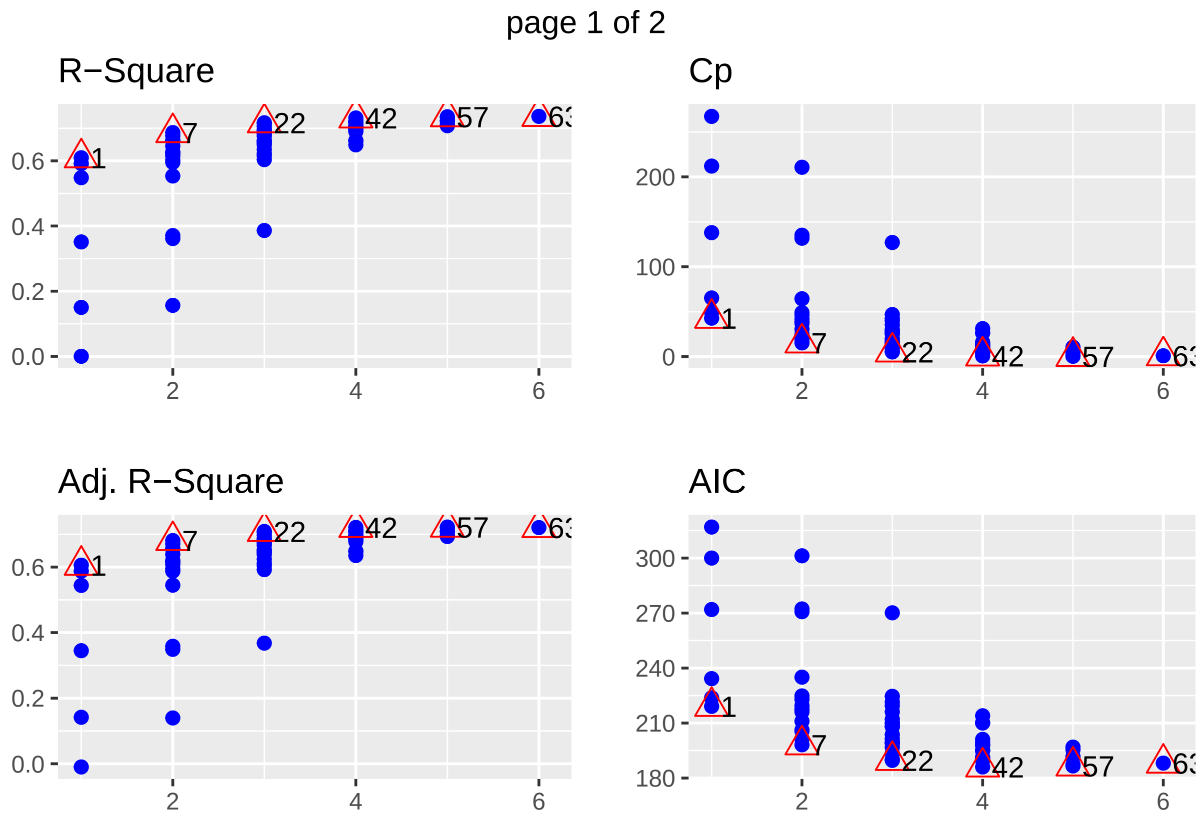
```
## 57    58 5            GDP social healthy freedom generosity 7.338201e-01
## 60    59 5         GDP social freedom generosity corruption 7.231762e-01
## 62    60 5     social healthy freedom generosity corruption 7.223860e-01
## 59    61 5         GDP social healthy generosity corruption 7.119566e-01
## 61    62 5        GDP healthy freedom generosity corruption 7.080110e-01
## 63    63 6 GDP social healthy freedom generosity corruption 7.366674e-01
##    Adj. R-Square Mallow's Cp
## 1    0.605709891   43.828853
## 3    0.587627009   50.425114
## 2    0.544015416   66.333718
## 4    0.344871733  138.977203
## 6    0.141808627  213.050412
## 5   -0.009782773  268.347811
## 9    0.680745675   17.315617
## 12   0.670568054   20.991804
## 7    0.656742619   25.985593
## 16   0.655151823   26.560193
## 8    0.638650293   32.520595
## 11   0.620400797   39.112367
## 10   0.616194011   40.631871
## 13   0.615569387   40.857487
## 15   0.607817006   43.657670
## 18   0.594800391   48.359311
## 17   0.587410210   51.028666
## 14   0.544769181   66.430733
## 20   0.358322537  133.775819
## 19   0.349031392  137.131808
## 21   0.139703532  212.741657
## 32   0.708498323    8.248790
## 23   0.706917934    8.813979
## 26   0.697542475   12.166894
## 34   0.687643653   15.706977
## 22   0.681262117   17.989185
## 25   0.680629583   18.215396
## 29   0.679646828   18.566855
## 30   0.679232948   18.714870
## 33   0.673492572   20.767782
## 24   0.666287715   23.344431
## 39   0.652114047   28.413314
## 38   0.651707787   28.558603
## 28   0.645551428   30.760281
## 27   0.645351705   30.831708
## 36   0.638248430   33.372029
## 31   0.623964411   38.480376
## 35   0.611735902   42.853617
## 37   0.604044568   45.604244
## 40   0.592050558   49.893622
## 41   0.367916144  130.050078
## 42   0.720950058    4.797759
## 53   0.710972297    8.330390
## 46   0.710576857    8.470396
## 45   0.706777589    9.815528
## 52   0.706273871    9.993870
## 44   0.696173190   13.570021
```

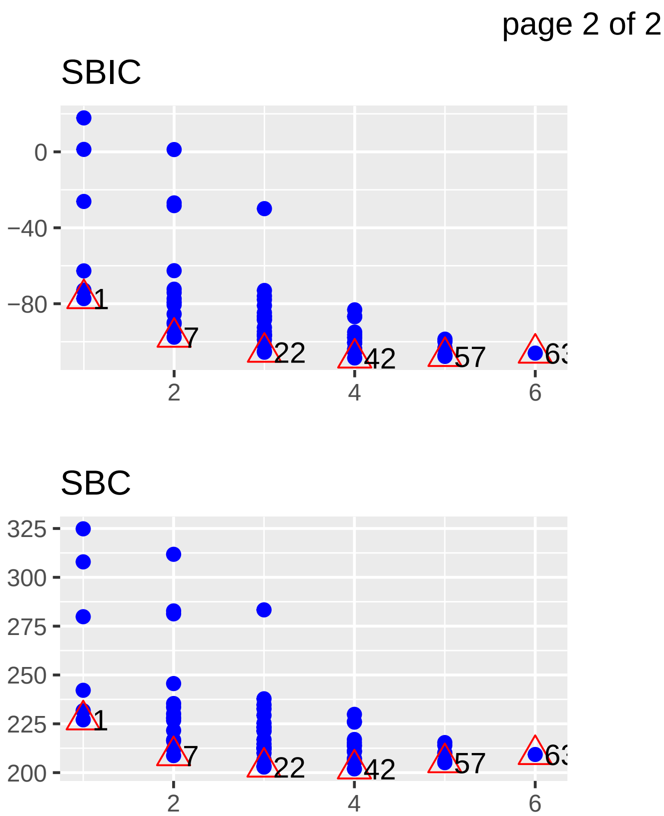
```
## 48   0.695980796   13.638138
## 49   0.694999674   13.985505
## 43   0.687726922   16.560426
## 54   0.686438291   17.016666
## 47   0.682339853   18.467720
## 51   0.677467529   20.192768
## 56   0.648639249   30.399434
## 50   0.647977278   30.633805
## 55   0.635027007   35.218855
## 58   0.722025260    5.422972
## 57   0.720239466    6.048846
## 60   0.709052587    9.969559
## 62   0.708221984   10.260664
## 59   0.697260542   14.102366
## 61   0.693113633   15.555750
## 63   0.720378838    7.000000
```

```
# Visualize the model
plot(all.mod)
```

```
## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.
```

```
## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.

## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.
```





Best Subset Regression

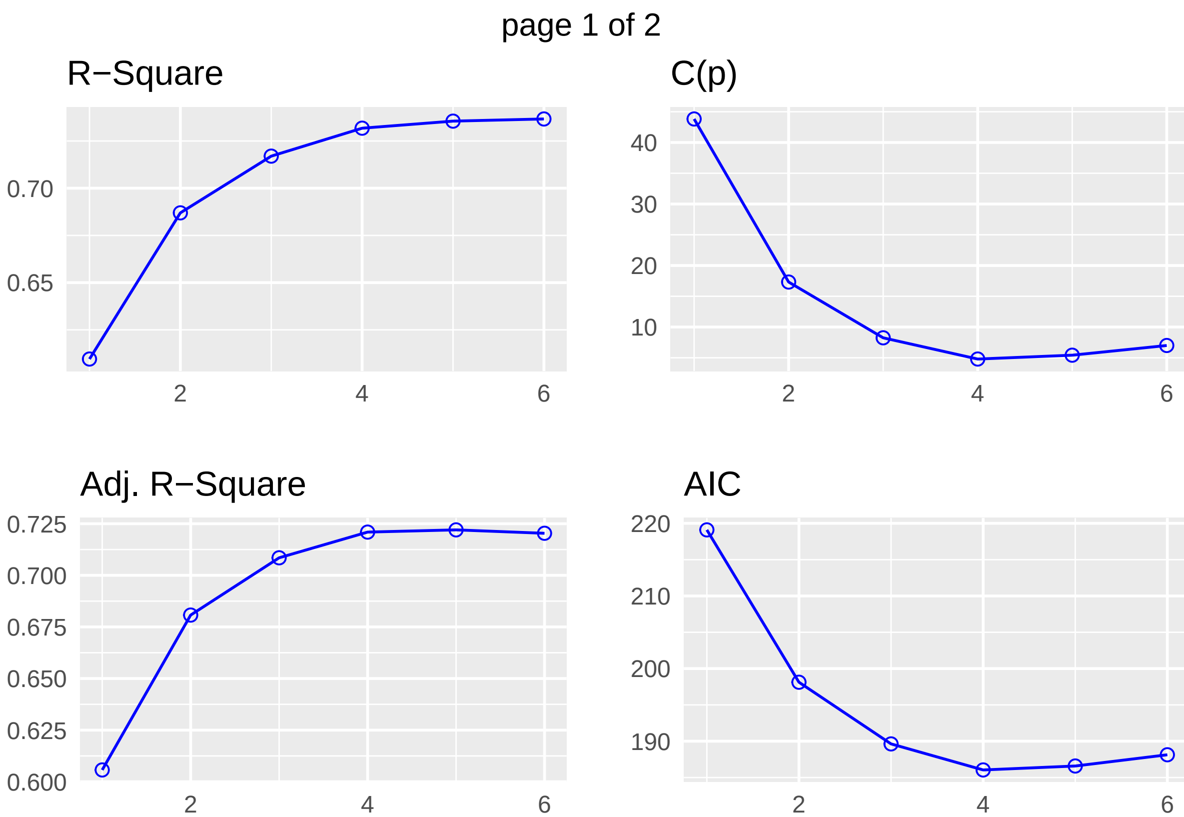
Best Subset Regression select the subset of predictors that do the best at meeting some well-defined objective criterion, such as having the largest R2 value or the smallest MSE, Mallow's Cp or AIC.

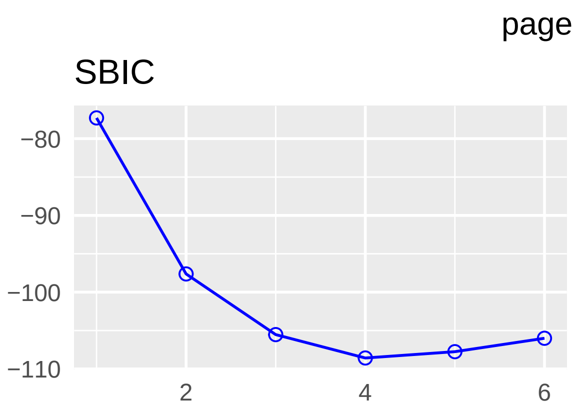
```r
# Criteria of the best subset of the model
best.mod <- ols_step_best_subset(lm)
best.mod
```

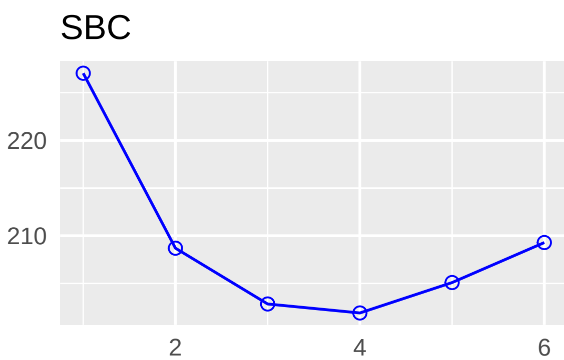
```
##                        Best Subsets Regression
## ---------------------------------------------------------------
## Model Index    Predictors
## ---------------------------------------------------------------
##      1         GDP
##      2         GDP freedom
##      3         social healthy freedom
##      4         GDP social healthy freedom
##      5         GDP social healthy freedom corruption
##      6         GDP social healthy freedom generosity corruption
## ---------------------------------------------------------------
##
##                                                    Subsets Regression Summary
## -------------------------------------------------------------------------------------------------------------
##                        Adj.        Pred
## Model    R-Square    R-Square    R-Square     C(p)        AIC         SBIC         SBC         MSEP
## -------------------------------------------------------------------------------------------------------------
##   1        0.6095      0.6057      0.5933   43.8289    219.1005     -77.2998    227.0337    48.1817
##   2        0.6869      0.6807      0.6676   17.3156    198.1216     -97.6257    208.6991    39.0162
##   3        0.7170      0.7085      0.6901    8.2488    189.6287    -105.5296    202.8507    35.6281
```

```
##   4        0.7318      0.7210      0.7003     4.7978     186.0434     -108.5748     201.9097     34.1097
##   5        0.7355      0.7220      0.6944     5.4230     186.5860     -107.7522     205.0968     33.9818
##   6        0.7367      0.7204      0.6903     7.0000     188.1335     -106.0058     209.2887     34.1867
## -----------------------------------------------------------------------------------------------------------
## AIC: Akaike Information Criteria
##  SBIC: Sawa's Bayesian Information Criteria
##  SBC: Schwarz Bayesian Criteria
##  MSEP: Estimated error of prediction, assuming multivariate normality
##  FPE: Final Prediction Error
##  HSP: Hocking's Sp
##  APC: Amemiya Prediction Criteria
```

```r
# Visualize the model
plot(best.mod)
```







In this case, model with 3 predictors is the best fitting model due to its largest R2 value and the smallest MSE, Mallow's Cp and AIC value.

Stepwise Procedures

Stepwise Forward Regression Forward Regression starts with no predictors in the model, iteratively adds the most contributive predictors, and stops when the improvement is no longer statistically significant.

```
# Forward selection based on p-values
ols_step_forward_p(lm, details = FALSE)
```

```
##
##                              Selection Summary
## ---------------------------------------------------------------------------
##         Variable                  Adj.
## Step     Entered      R-Square   R-Square     C(p)        AIC        RMSE
## ---------------------------------------------------------------------------
##    1     GDP             0.6095     0.6057    43.8289    219.1005    0.6806
##    2     freedom         0.6869     0.6807    17.3156    198.1216    0.6125
##    3     social          0.7155     0.7069     8.8140    190.1910    0.5868
##    4     healthy         0.7318     0.7210     4.7978    186.0434    0.5726
##    5     corruption      0.7355     0.7220     5.4230    186.5860    0.5715
## ---------------------------------------------------------------------------
```

In this case, the most contributive predictors are GDP, Corruption, Social, Freedom, and Healthy.

Stepwise Backward Regression Backward selection starts with all predictors in the model, iteratively removes the least contributive predictors, and stops when all of the predictors are statistically significant.

```
# Backward selection based on p-values
ols_step_backward_p(lm, details = FALSE)
```

```
##
##
##                            Elimination Summary
## ---------------------------------------------------------------------------
##        Variable                       Adj.
## Step    Removed      R-Square    R-Square     C(p)        AIC         RMSE
## ---------------------------------------------------------------------------
##    1    generosity      0.7355       0.722    5.4230    186.5860    0.5715
## ---------------------------------------------------------------------------
```

In this case, Stepwise backward regression has eliminated the least significant variables predictor, "Generosity" .

Stepwise Regression Stepwise Regression is a combination of forward and backward selections.

```
# Variable Selection
step <- ols_step_both_p(lm, details = FALSE)
```

Based on the stepwise regression result, the three predictor model with the three predictors : "GDP", "social", and "corruption" are the best fitting model.

Gradient Boosting Model(GBM)

For the second model, I will try a Gradient Boosting Model. This approaches creates an ensemble where new models are added sequentially rather than simply averaging the predicted values of the models.

```
# Load necessary packages
library(gbm)
```

```
## Loaded gbm 2.1.8
```

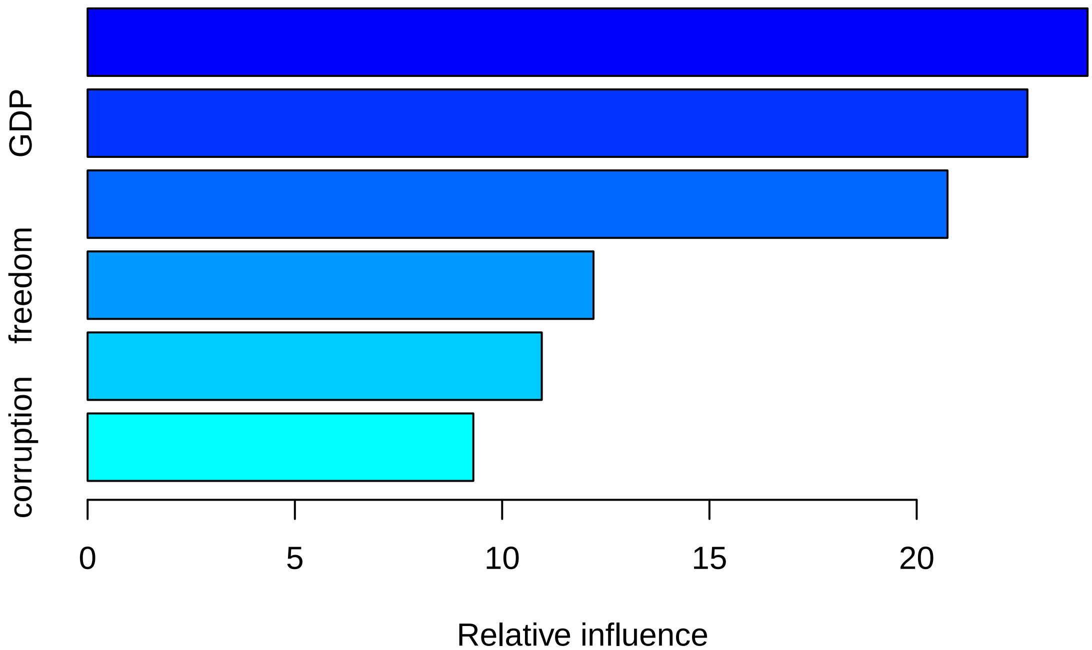
```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:olsrr':
##
##     cement

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

```
Boston.boost <-
  gbm(ladder ~ . ,
      data = train,
      distribution = "gaussian",
      n.trees = 10000,
      shrinkage = 0.01,
      interaction.depth = 4)
Boston.boost
```

```
## gbm(formula = ladder ~ ., distribution = "gaussian", data = train,
##     n.trees = 10000, interaction.depth = 4, shrinkage = 0.01)
## A gradient boosted model with gaussian loss function.
## 10000 iterations were performed.
## There were 6 predictors of which 6 had non-zero influence.
```

```r
#Summary of Variable Importance and a plot of Variable Importance
summary(Boston.boost)
```




```
##                   var   rel.inf
## social         social 24.121402
## GDP               GDP 22.670963
## healthy       healthy 20.743102
## freedom       freedom 12.203351
## generosity generosity 10.955980
## corruption corruption  9.305202
```

The summary of the Model gives a feature importance plot. In the above list is on the top is the most important variable and at last is the least important variable. In this case, "Healthy", "GDP", and "Social" appear to be the top 3 important predictors.

```r
# Predict on Test Set
Boston.boost_p <- predict(Boston.boost, test)
```

```
## Using 10000 trees...
```

```r
# Compute RMSE for gbm model
gbm.rmse <- RMSE(Boston.boost_p, test$ladder)
```

```r
gbm.rmse
```

Compute Accuracy

```
## [1] 0.4423769
```

The Root Mean-Squared Error (RMSE) for this GBM model was 63.83%.

```r
# Save the gbm model RMSE result into our accuracy table
accuracy_results <- bind_rows(accuracy_results, tibble(method = "GBM", RMSE = gbm.rmse))
```

```r
# View the accuracy table
accuracy_results %>% knitr::kable()
```

method	RMSE
Linear_Regression	0.4938069
GBM	0.4423769

Decision Tree for Regression

For the third model, I will train a regression decision tree to predict the happiness of country.

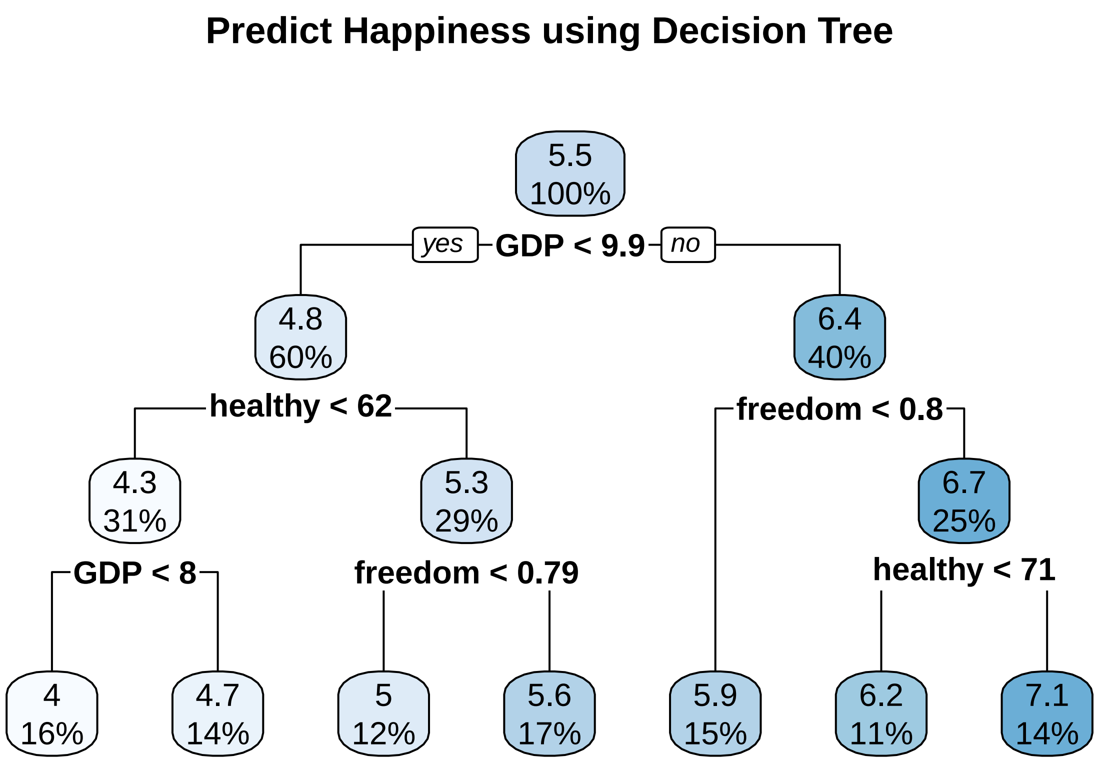
```r
library(rpart)
```

```r
decision_tree <- rpart(ladder~.,
            method = "anova",
            data = train)
```

```r
decision_tree
```

```
## n= 104
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 104 121.023100 5.464500
##    2) GDP< 9.853 62  40.213250 4.824823
##      4) healthy< 61.818 32  15.558320 4.340625
##        8) GDP< 8.015 17   8.340748 4.045000 *
##        9) GDP>=8.015 15   4.048075 4.675667 *
##      5) healthy>=61.818 30   9.150152 5.341300
##       10) freedom< 0.7945 12   1.506416 4.968000 *
##       11) freedom>=0.7945 18   4.856678 5.590167 *
##    3) GDP>=9.853 42  17.989880 6.408786
##      6) freedom< 0.7975 16   3.641138 5.896375 *
##      7) freedom>=0.7975 26   7.562453 6.724115
##       14) healthy< 70.551 11   1.170733 6.213364 *
##       15) healthy>=70.551 15   1.417849 7.098667 *
```

```r
rpart.plot(decision_tree,
         main = "Predict Happiness using Decision Tree")
```



This decision tree is a way of predicting adder score. It has provided 5 understandable decision criteria. The first decision is whether GDP of the country is above or below 9.8. If <9.8, that's yes, then we look at whether the country have a score on healthy that has less than 62. If <62, we look at whether the country's GDP is less than 8.1. If <8.1, then the ladder score of the country is 4.3.

If the first decision in which whether the country's GDP is greater than 9.8, that's no, then we will look at whether the country's GDP is less than or greater than 11. If below 11, then the country will have a ladder score of 6. On the other hand, if the contry's GDP is greater than 11, then the ladder score will be 7. These percentages at the bottom tell us what percentage of total cases fall into that particular cell.

```
# Predict Test Data
ladder_p <- decision_tree %>%
predict(newdata = test)
```

```
# Compute RMSE for decision tree
decision.rmse <- RMSE(ladder_p, test$ladder)
```

```
decision.rmse
```

Compute Accuracy

```
## [1] 0.5266417
```

The Root Mean-Squared Error (RMSE) for this decision tree model was 69.97%.

6

```
# Save the decision tree RMSE result into our accuracy table
accuracy_results <- bind_rows(accuracy_results, tibble(method = "Decision Tree", RMSE = decision.rmse))
```

```
# View the accuracy table
accuracy_results %>% knitr::kable()
```

method	RMSE
Linear_Regression	0.4938069
GBM	0.4423769
Decision Tree	0.5266417

Random Forest

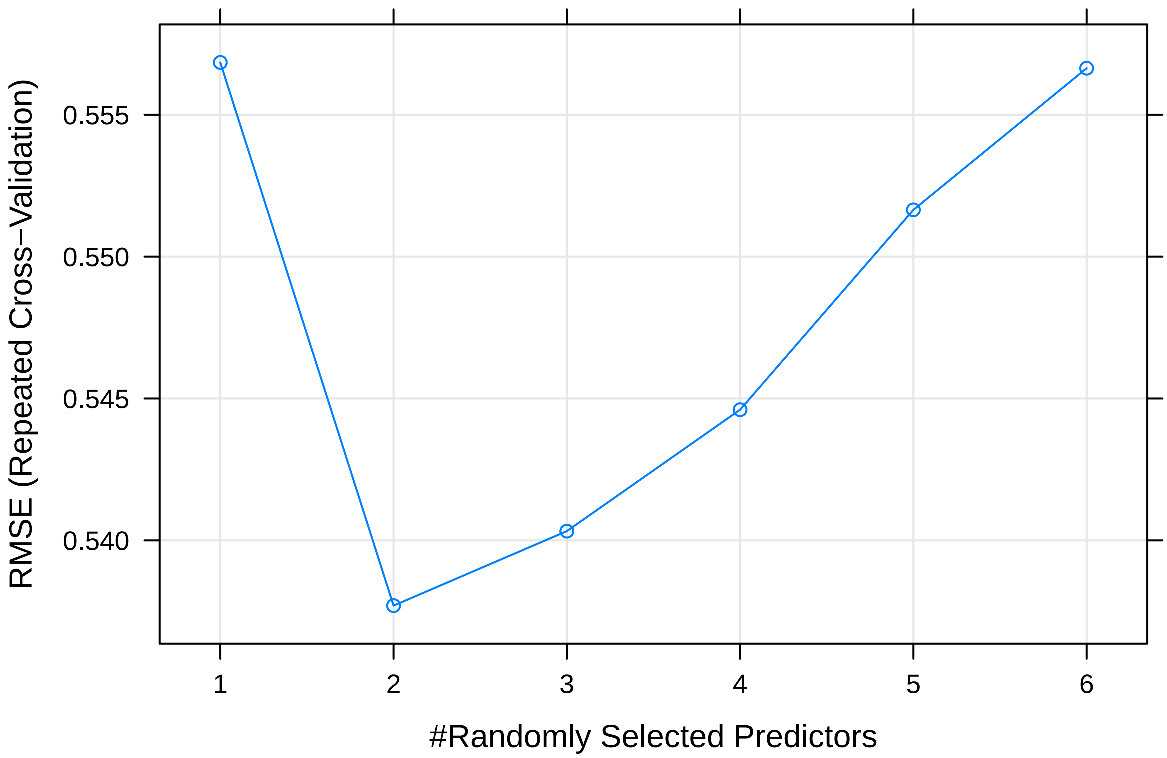
For the last model, I will train a Random Forest model. Random Forest takes the average of multiple decision trees in order to improve predictions.

```
# Define Parameters
control <- trainControl(
method = "repeatedcv",
number = 10,
repeats = 3,
search = "random",
allowParallel = TRUE
)
```

```
# Train random forest model
randomforest <- train(
ladder~.,
data = train,
method = "rf",
trControl = control,
tuneLength = 15,
ntree = 800,
importance = TRUE
)
randomforest
```

```
## Random Forest
##
## 104 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 93, 93, 92, 92, 94, 94, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##   1     0.5568415  0.7760071  0.4420318
##   2     0.5377012  0.7828246  0.4281572
##   3     0.5403259  0.7770715  0.4321242
##   4     0.5446042  0.7714692  0.4339076
##   5     0.5516471  0.7647903  0.4411726
##   6     0.5566380  0.7594125  0.4465668
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
# Plot accuracy by number of predictors
randomforest %>% plot()
```

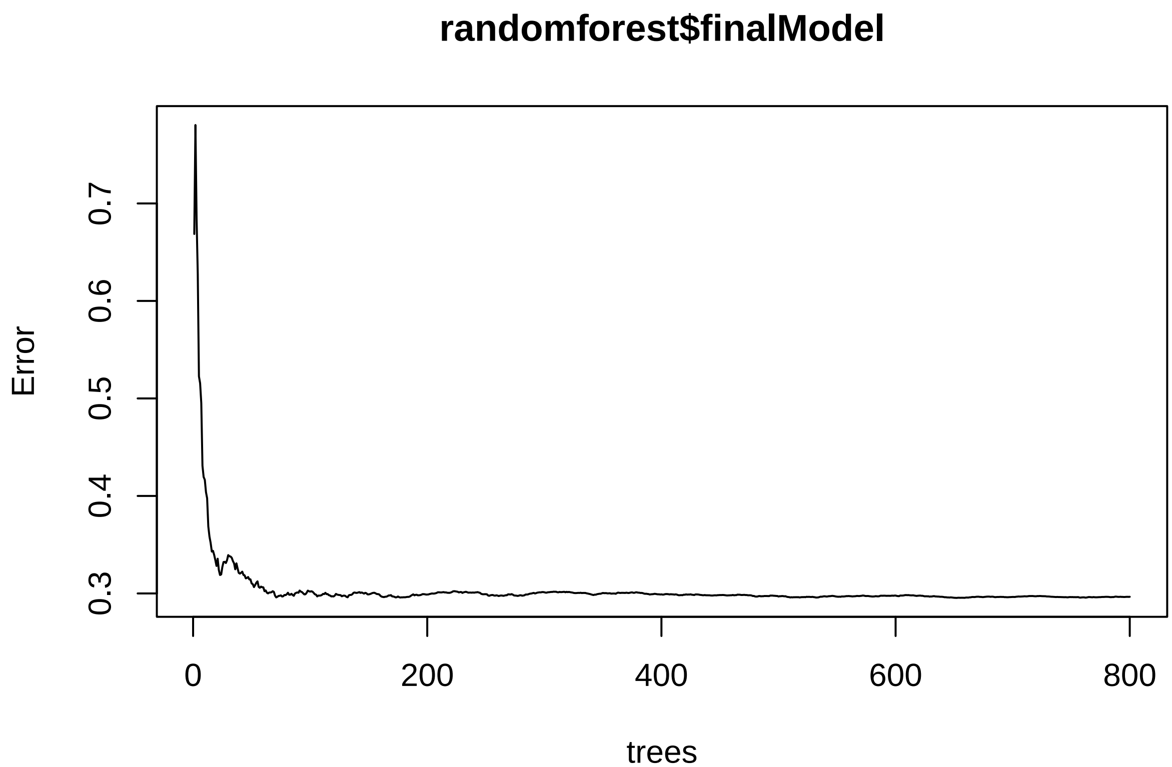


This tells us that in the 800 randomly generated decision trees, how many predictors did they each have? Most decision trees only had 1 predictor, and very few had 2 or 3 predictors. Between 4 and 6 predictors, the number of decision trees slowly increases but slightly drop on having 6 predictors.

```
randomforest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 800, mtry = min(param$mtry,      ncol(x)), importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 800
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.29654
##                     % Var explained: 74.52
```

```
plot(randomforest$finalModel)
```




```
which.min(randomforest$finalModel$mse)
```

```
## [1] 652
```

I need 785 trees to have the lowest error. I will now rerun the model and add an argument called "ntree" to indicating the number of trees I want to generate.

```
randomforest2 <- train(
ladder~.,
data = train,
method = "rf",
trControl = control,
tuneLength = 15,
ntree = 785,
importance = TRUE
)
randomforest2
```

```
## Random Forest
##
## 104 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 93, 93, 95, 92, 95, 94, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##   1     0.5428980  0.7797502  0.4373760
##   2     0.5251434  0.7851992  0.4237672
```

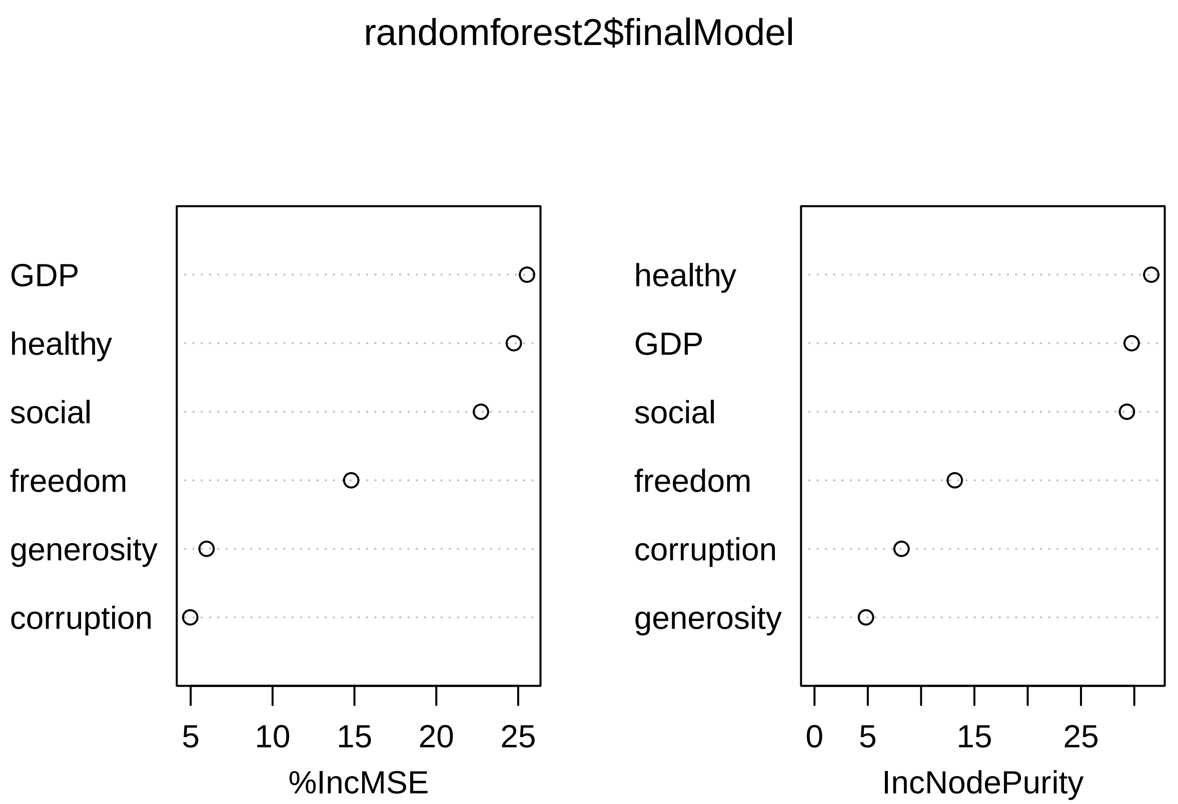
```
##   3      0.5295646  0.7793313  0.4257296
##   4      0.5369284  0.7716918  0.4316204
##   5      0.5447692  0.7648060  0.4365861
##   6      0.5463240  0.7629932  0.4374717
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
randomforest2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 785, mtry = min(param$mtry,      ncol(x)), importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 785
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.2993098
##                     % Var explained: 74.28
```

We can now see which of the predictors in our model are the most useful.

```
varImpPlot(randomforest2$finalModel)
```



The higher the INCMSE and IncNodePurity, the more important the predictor variable. "GDP" is most important followed by "healthy" and then "social".

```
# Predict Test data
randomforest_p <-predict(randomforest2, newdata = test)
```

```r
# Compute RMSE for random forest
randomforest.rmse <- RMSE(randomforest_p, test$ladder)
```

```r
randomforest.rmse
```

Compute Accuracy

```
## [1] 0.4598692
```

The Root Mean-Squared Error (RMSE) for this decision tree model was 61.67%.

```r
# Save the decision tree RMSE result into our accuracy table
accuracy_results <- bind_rows(accuracy_results, tibble(method = "Random Forest", RMSE = randomforest.rms
```

```r
# View the accuracy table
accuracy_results %>% knitr::kable()
```

method	RMSE
Linear_Regression	0.4938069
GBM	0.4423769
Decision Tree	0.5266417
Random Forest	0.4598692

Conclusion

The aim of this project was to create a model that accurately predicts the happiness of countries around the world. I have analyzed 4 different models, linear regression, gbm, decision tree, and random forest. From the results, I can conclude that multiple linear regression model estimated best accuracy as it has the lowest RMSE value. The RMSE of each model is shown in the table below.

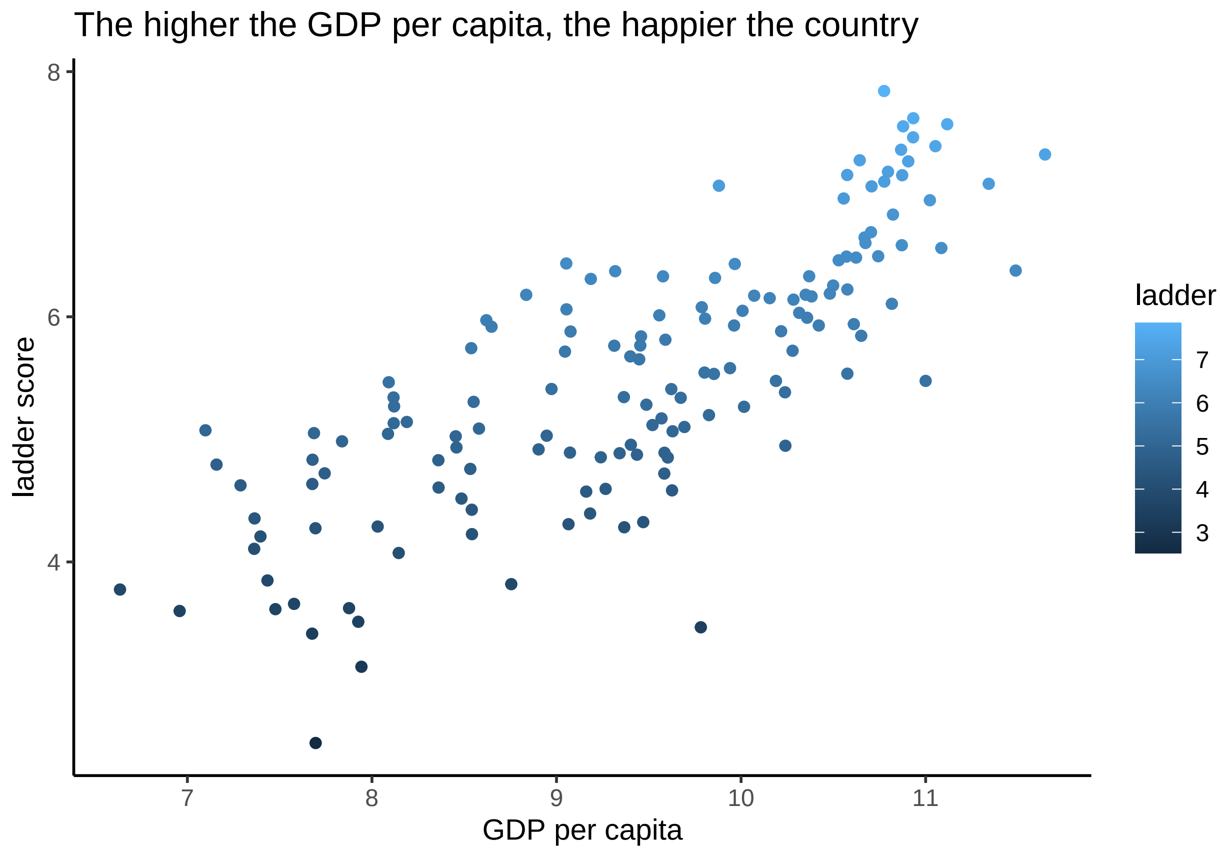
```r
# Determining the Final Model
accuracy_results %>% knitr::kable()
```

method	RMSE
Linear_Regression	0.4938069
GBM	0.4423769
Decision Tree	0.5266417
Random Forest	0.4598692

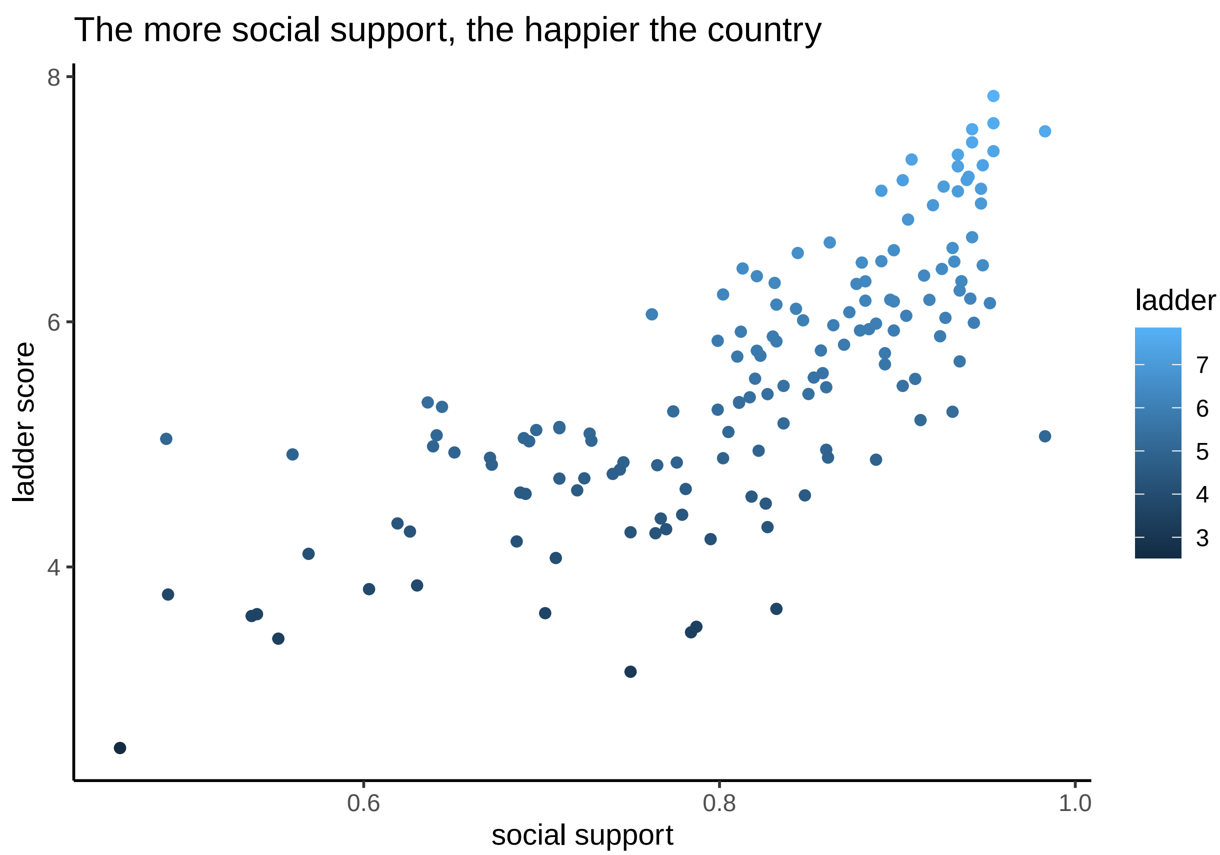
According to the linear regression model, the results conclude that GDP is the single most influential predictor in determining a country's happiness. Other significant attributes are a country's social support and corruption scores. If a country with lower happiness score opts to use the results of this study to drive their upcoming initiative, they should focus on ways to stimulate the economies.

Below are the scatterplot showing how each of the top predictors relate to happiness.

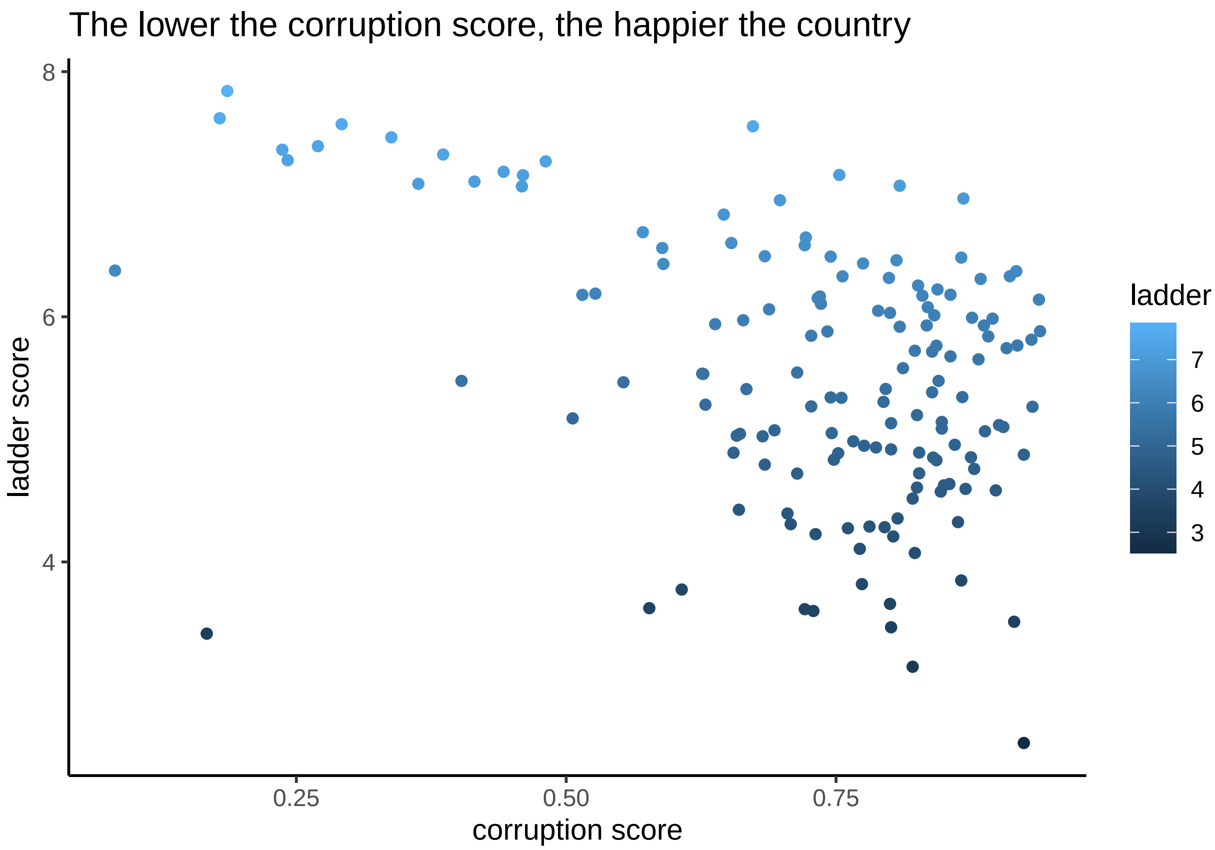
```r
# GDP vs Happiness
ggplot(data = df) +
  geom_point(mapping = aes(x= GDP , y=ladder, color= ladder)) +
  theme_classic() +
  labs( x= "GDP per capita",
        y= "ladder score",
    title = "The higher the GDP per capita, the happier the country")
```




```
# Social Support vs Happiness
ggplot(data = df) +
  geom_point(mapping = aes(x= social , y=ladder, color= ladder)) +
  theme_classic() +
  labs( x= "social support",
        y= "ladder score",
    title = "The more social support, the happier the country")
```




```r
# Corruption score vs Happiness
ggplot(data = df) +
  geom_point(mapping = aes(x= corruption , y=ladder, color= ladder)) +
  theme_classic() +
  labs( x= "corruption score",
        y= "ladder score",
    title = "The lower the corruption score, the happier the country")
```



The lower the corruption score, the happier the country
8
6
4
ladder score
0.25
0.50
0.75
corruption score
ladder
7
6
5
4
3