Happiness Project

Mahip Sharma

6/15/2019

Happiness Project

Goal of this project is to build a machine learning model that will predict Happiness Score of a country based on different features like Economy (GDP per Capita), Family, Health (Life Expectancy), Freedom, Generosity, Government corruption (or Trust on Government) and Dystopia. We will explore Happiness data, plot graphs and implement multiple models to then select the best machine learing model to predit Happiness Score. We will use RMSE function to gauge performance of models.

Initial Setup:

LIBRARIES:

```
  # Following libraries will be used in this project
      library(corrplot)

## corrplot 0.84 loaded

      library(dslabs)
      library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

      library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

      library(ggplot2)
      library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

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

      library(gam)

## Loading required package: splines

## Loading required package: foreach

## Loaded gam 1.16

      library(ggplot2)
```

Load CSV File: Happiness data 2017

```
          # Load csv file into dataframe using "read.csv". "2017.csv" file is
kept in "Documents" folder present in root directory

                      happiness_17_df <- read.csv('~/Documents/2017.csv')
```

We will rename column names:

```
          happiness_17_df <- happiness_17_df %>% rename('HappinessRank' = 'Ha
ppiness.Rank','HappinessScore'='Happiness.Score',
                                                    'WhiskerHigh' = 'Whisker.
high','WhiskerLow' = 'Whisker.low' ,
                                                    'Economy' = 'Economy..GDP
.per.Capita.', 'Health' = 'Health..Life.Expectancy.',
                                                    'GovCorruption' = 'Trust.
.Government.Corruption.', 'Dystopia' = 'Dystopia.Residual')

          # Following are the new column names of happiness_17_df

              names(happiness_17_df) # Columns Names

##  [1] "Country"        "HappinessRank"   "HappinessScore" "WhiskerHigh"
##  [5] "WhiskerLow"     "Economy"         "Family"         "Health"
##  [9] "Freedom"        "Generosity"      "GovCorruption"  "Dystopia"
```

We will create a new column Region in happiness_17_df dataframe and it can have following values: Asia, Europe, North America, South America, Australia, Middle East and Africa:

```
          happiness_17_df <- happiness_17_df %>% mutate(Region = NA)

          happiness_17_df$Region[which(happiness_17_df$Country %in% c('Norway
','Finland', 'Sweden','Austria','Belgium','Russia','Algeria','Romania','North 
Cyprus','Cyprus',
                                                                    'Turkey
','Serbia','Croatia','Azerbaijan','Portugal','Bulgaria','Albania','Armenia','
Georgia','Denmark',
                                                                    'Nether
lands','Luxembourg','Spain','Poland','Latvia','Bolivia','Slovenia','Estonia',
'Kosovo','Bosnia and Herzegovina',
                                                                    'Icelan
d','Ireland','United Kingdom','Czech Republic','France','Belarus','Hungary','
Montenegro','Greece','Ethiopia',
                                                                    'Switze
rland','Germany','Slovakia','Italy','Lithuania','Moldova','Macedonia','Ukrain
e'))] <- 'Europe'

          happiness_17_df$Region[which(happiness_17_df$Country %in% c('Taiwan 
Province of China','Indonesia','Bhutan','Afghanistan','Singapore'
                                                                    ,'Malay
sia','Vietnam','Bangladesh','Myanmar','India','Uzbekistan','Japan','South Kor
ea','Turkmenistan','Hong Kong S.A.R., China'
                                                                    ,'China
','Nepal','Mauritania','Thailand','Kazakhstan','Mauritius','Philippines','Pak
istan','Tajikistan','Mongolia','Sri Lanka'))] <- 'Asia'

          happiness_17_df$Region[which(happiness_17_df$Country %in% c('United 
Arab Emirates','Saudi Arabia','Bahrain','Iraq','Jordan','Kyrgyzstan','Yemen',
'Israel','Qatar','Kuwait',
                                                                    'Palest
inian Territories','Lebanon','Egypt','Iran','Syria'))] <- 'Middle East'

          happiness_17_df$Region[which(happiness_17_df$Country %in% c('Mexico
','United States','Canada'))] <- 'North America'

          happiness_17_df$Region[which(happiness_17_df$Country %in% c('Guatem
ala','Haiti','Brazil','Panama','Belize'
                                                                    ,'Parag
uay','Venezuela','Nicaragua','Peru','Honduras','Costa Rica','Chile','Argentin
a','Uruguay','Colombia','Ivory Coast'))] <- 'South America'

          happiness_17_df$Region[which(happiness_17_df$Country %in% c('Austra
lia', 'New Zealand'))] <- 'Australia'
```

```
        happiness_17_df$Region[which(happiness_17_df$Country %in% c('El Salvador',  'Somalia','South Africa','Mozambique','Cambodia','Uganda','Chad','Guinea','Tanzania','Trinidad and Tobago',
                                                                      'Dominican Republic','Tunisia','Sierra Leone','Gabon','Congo (Kinshasa)','Sudan','Burkina Faso','Zimbabwe','Botswana','Togo',
                                                                      'Burundi','Malta','Nigeria','Cameroon','Namibia','Senegal','Mali','Ghana','Niger','Lesotho',
                                                                      'Benin','South Sudan','Rwanda','Central African Republic','Ecuador','Libya','Jamaica','Morocco','Kenya','Zambia',
                                                                      'Congo (Brazzaville)','Malawi','Angola','Madagascar','Liberia'))] <- 'Africa'

      # Change Region column to factor

        happiness_17_df$Region <- as.factor(happiness_17_df$Region)

      # Following code displays new structure of happiness_17_df

        str(happiness_17_df)

## 'data.frame':    155 obs. of  13 variables:
##  $ Country      : Factor w/ 155 levels "Afghanistan",..: 105 38 58 133 45 99 26 100 132 7 ...
##  $ HappinessRank : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ HappinessScore: num  7.54 7.52 7.5 7.49 7.47 ...
##  $ WhiskerHigh   : num  7.59 7.58 7.62 7.56 7.53 ...
##  $ WhiskerLow    : num  7.48 7.46 7.39 7.43 7.41 ...
##  $ Economy       : num  1.62 1.48 1.48 1.56 1.44 ...
##  $ Family        : num  1.53 1.55 1.61 1.52 1.54 ...
##  $ Health        : num  0.797 0.793 0.834 0.858 0.809 ...
##  $ Freedom       : num  0.635 0.626 0.627 0.62 0.618 ...
##  $ Generosity    : num  0.362 0.355 0.476 0.291 0.245 ...
##  $ GovCorruption : num  0.316 0.401 0.154 0.367 0.383 ...
##  $ Dystopia      : num  2.28 2.31 2.32 2.28 2.43 ...
##  $ Region        : Factor w/ 7 levels "Africa","Asia",..: 4 4 4 4 4 4 6 3 4 3 ...
```

Data Exploration and Data Visualization:

We can see that data is already clean and tidy:

```
head(happiness_17_df)

##       Country HappinessRank HappinessScore WhiskerHigh WhiskerLow  Economy
## 1      Norway             1          7.537    7.594445   7.479556 1.616463
```

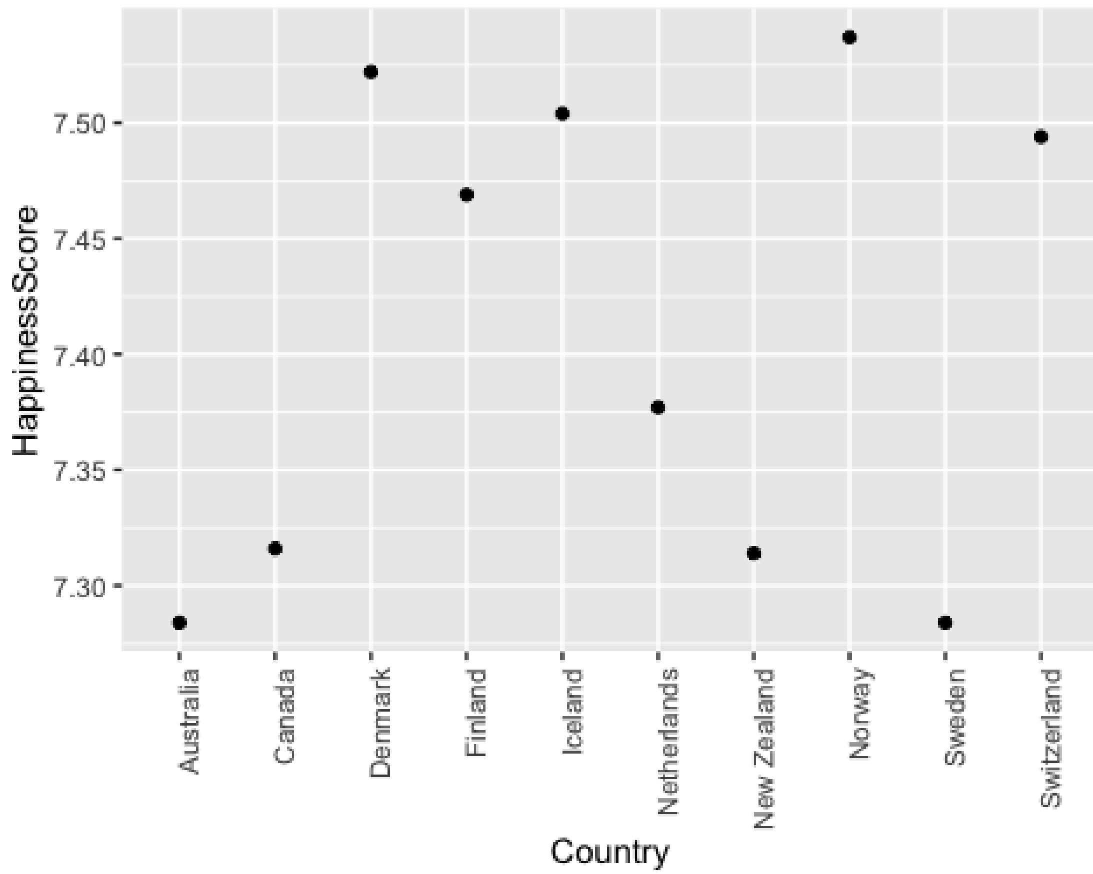
```
## 2     Denmark             2          7.522    7.581728   7.462272 1.482383
## 3     Iceland             3          7.504    7.622030   7.385970 1.480633
## 4 Switzerland             4          7.494    7.561772   7.426227 1.564980
## 5     Finland             5          7.469    7.527542   7.410458 1.443572
## 6 Netherlands             6          7.377    7.427426   7.326574 1.503945
##     Family    Health   Freedom Generosity GovCorruption Dystopia Region
## 1 1.533524 0.7966665 0.6354226  0.3620122     0.3159638 2.277027 Europe
## 2 1.551122 0.7925655 0.6260067  0.3552805     0.4007701 2.313707 Europe
## 3 1.610574 0.8335521 0.6271626  0.4755402     0.1535266 2.322715 Europe
## 4 1.516912 0.8581313 0.6200706  0.2905493     0.3670073 2.276716 Europe
## 5 1.540247 0.8091577 0.6179509  0.2454828     0.3826115 2.430182 Europe
## 6 1.428939 0.8106961 0.5853845  0.4704898     0.2826618 2.294804 Europe
```

Following code provides details about happiness_17_df:

```
        str(happiness_17_df) # Structure of happiness_17_df

## 'data.frame':    155 obs. of  13 variables:
##  $ Country      : Factor w/ 155 levels "Afghanistan",..: 105 38 58 133 45
99 26 100 132 7 ...
##  $ HappinessRank : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ HappinessScore: num  7.54 7.52 7.5 7.49 7.47 ...
##  $ WhiskerHigh   : num  7.59 7.58 7.62 7.56 7.53 ...
##  $ WhiskerLow    : num  7.48 7.46 7.39 7.43 7.41 ...
##  $ Economy       : num  1.62 1.48 1.48 1.56 1.44 ...
##  $ Family        : num  1.53 1.55 1.61 1.52 1.54 ...
##  $ Health        : num  0.797 0.793 0.834 0.858 0.809 ...
##  $ Freedom       : num  0.635 0.626 0.627 0.62 0.618 ...
##  $ Generosity    : num  0.362 0.355 0.476 0.291 0.245 ...
##  $ GovCorruption : num  0.316 0.401 0.154 0.367 0.383 ...
##  $ Dystopia      : num  2.28 2.31 2.32 2.28 2.43 ...
##  $ Region        : Factor w/ 7 levels "Africa","Asia",..: 4 4 4 4 4 4 6 3
4 3 ...

        dim(happiness_17_df)   # 155 rows and 12 columns

## [1] 155  13
```

Following are the top 10 countries based on Happiness Score:

```
        happiness_17_df[c(1:10),] %>% arrange(HappinessRank) %>% ggplot(aes(C
ountry, HappinessScore)) + geom_point() +
          theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

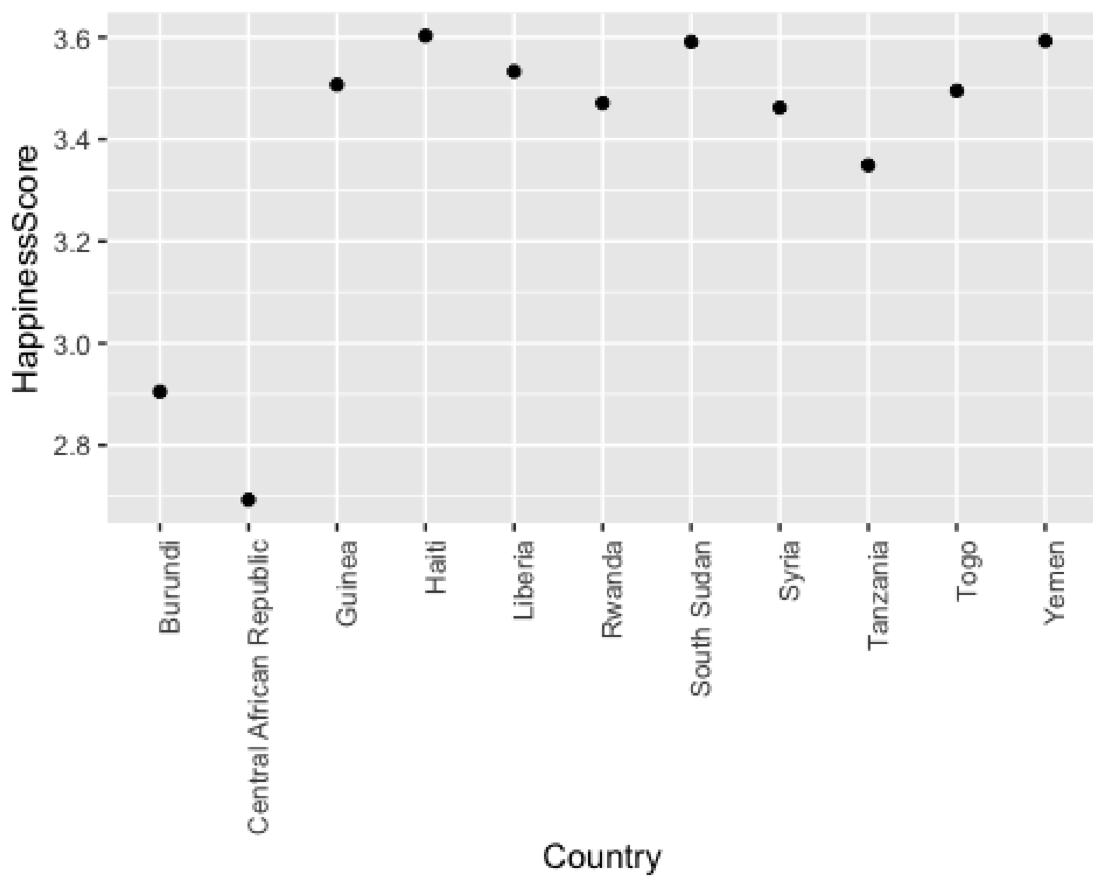


Note that higher the Happiness Score of a country, better the rank. e.g. Norway has the highest Happiness Score and ranks 1st in the list. Happiness Score is the mean value of WhiskerHigh and WhiskerLow.

7 out of top 10 Happy countries in the world are from Europe.

Following are the last 10 countries based on Happiness Score:

```
        happiness_17_df[c(145:155),] %>% arrange(HappinessRank) %>% ggplot(a
es(Country, HappinessScore)) + geom_point() +
          theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



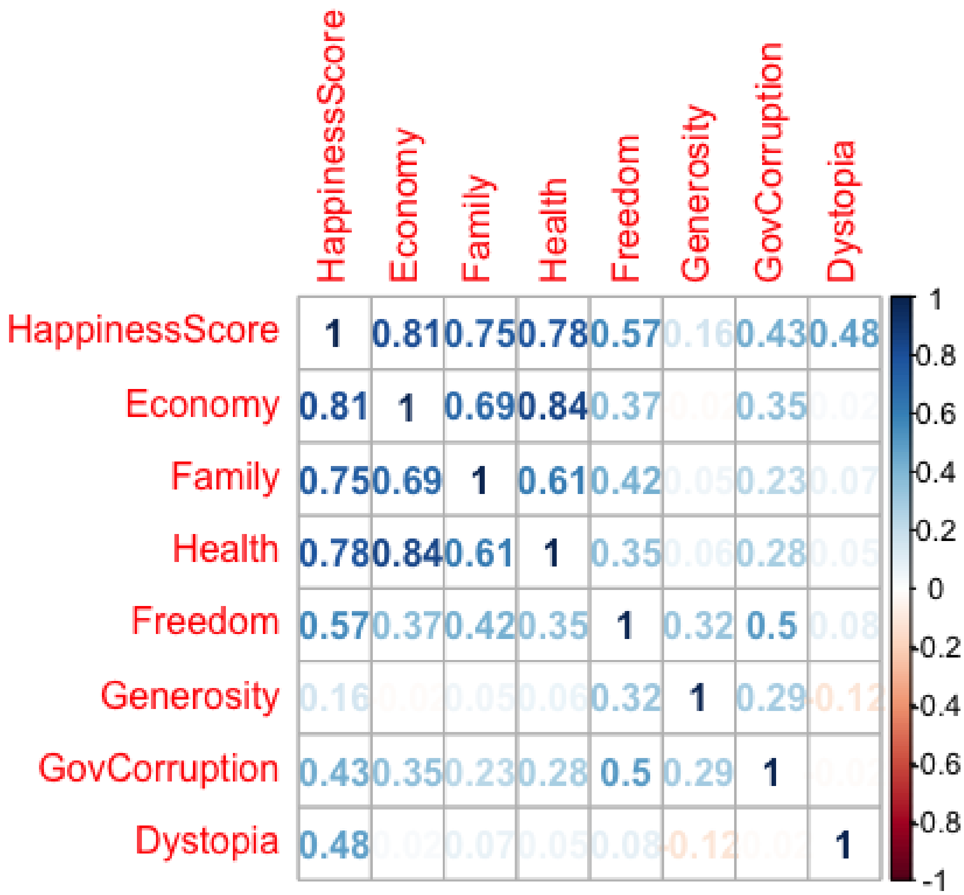
Note that many countries are from Africa region

Correlation between different attributes in happiness_17_df dataframe:

Following code will calculate correlation between different attributes. Note that we will consider only numeric features and we have excluded Happiness Rank as it is inversely propotional to Happiness Score with negative correlation.

```
            # Happiness Rank is negatively correlation to Happiness Score

            cor(happiness_17_df[,c('HappinessScore',"HappinessRank")])

##                 HappinessScore HappinessRank
## HappinessScore      1.0000000    -0.9927745
## HappinessRank      -0.9927745     1.0000000

            happiness_correlation <- cor(happiness_17_df[,c('HappinessScore',
'Economy',
                                                              'Family','Health'
,'Freedom','Generosity',
                                                              'GovCorruption','
Dystopia')])
```

```r
# Round correlation values to nearest 2 decimal places and plot these values against each other

corrplot(round(happiness_correlation,2), method = "number")
```



Findings from Correlation Plot:

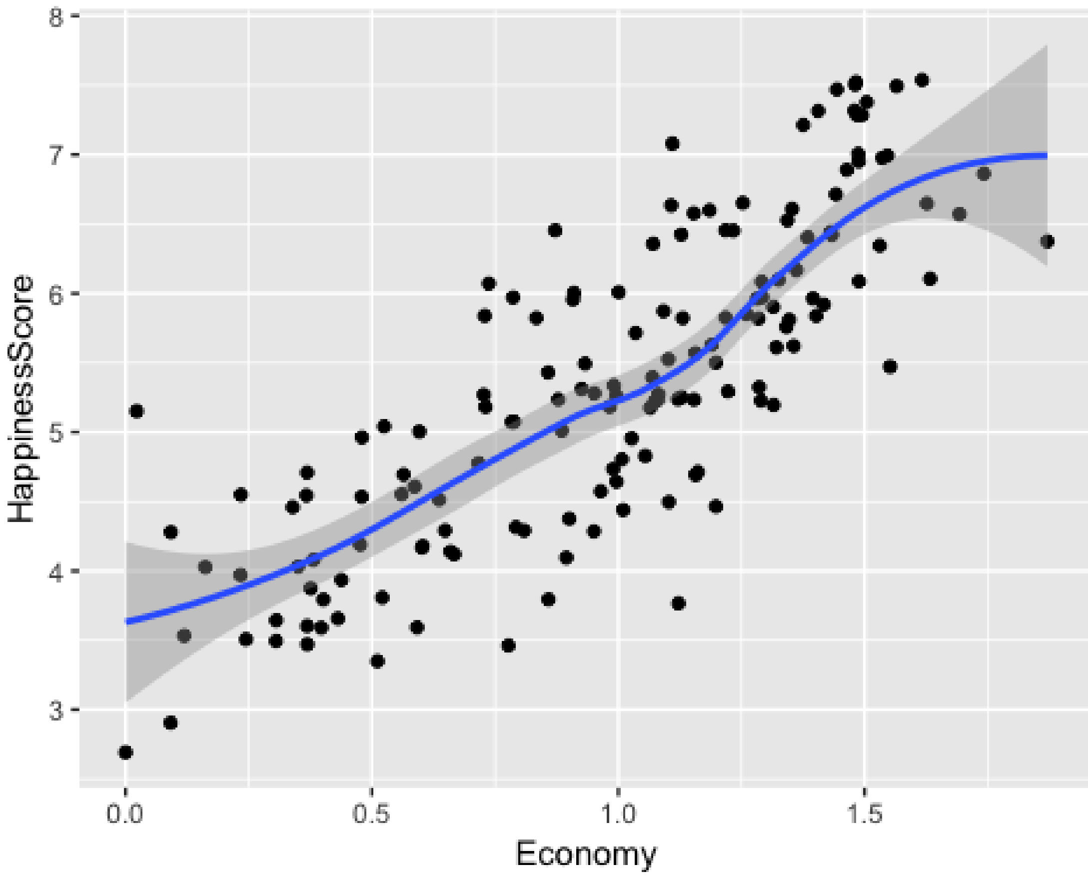
```
  All values of features against Happiness Score are positive.
  We can see that Happiness Score is highly correlated to Economy, Family and Health.
  Happiness Score is moderately correlated to Freedom, Trust (or government corruption) and Dystopla.
  Happiness Score is loosely correlated to Generosity.
```

Following section shows scatter plots with regression line for different features vs Happiness Score:

These figures states that features have linear relationship with Happiness Score. Additionally, higher the correction value of features, better is the linear relationship with Happiness Score.

Following plot shows relationship between Happiness Score and Economy with regression line:

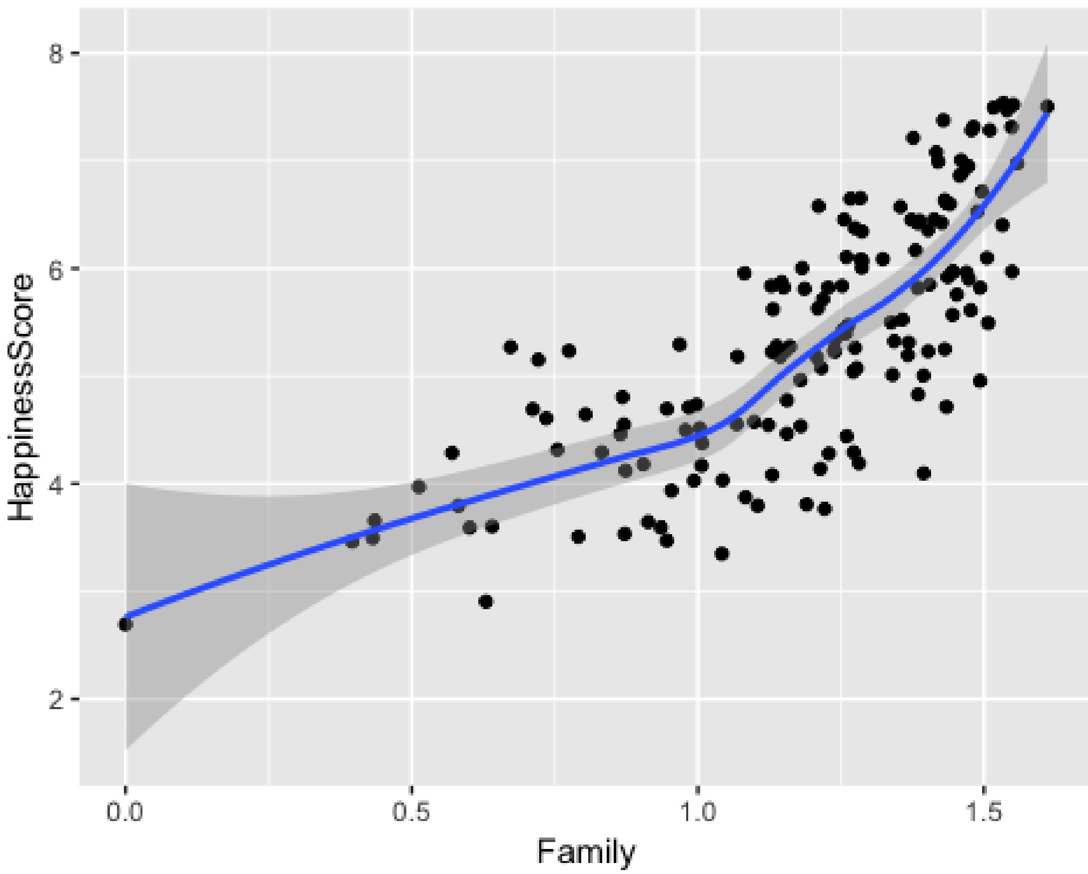
```
            happiness_17_df %>% ggplot(aes(Economy,HappinessScore)) + geom_po
int() + geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
            # Correlation value = 0.81 (High)
```

Following plot shows relationship between Happiness Score and Family with regression line:

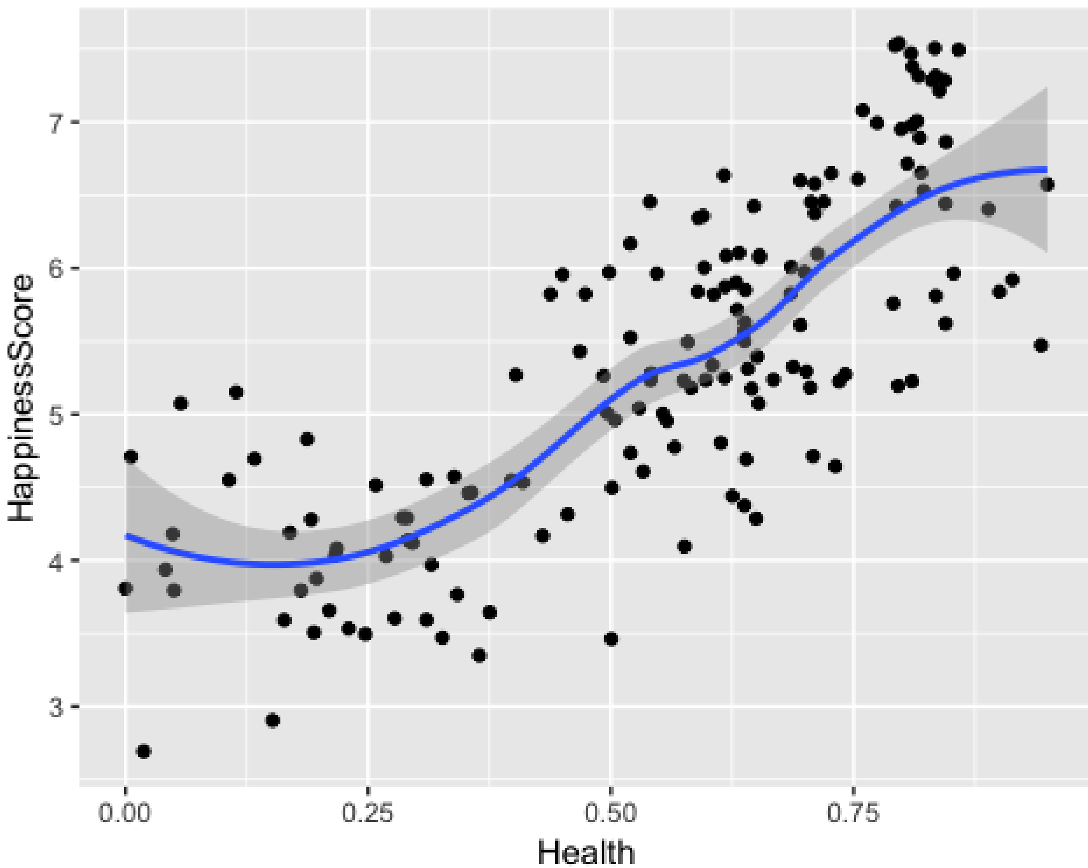
```
            happiness_17_df %>% ggplot(aes(Family,HappinessScore)) + geom_poi
nt() + geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# Correlation value = 0.75 (High)
```

Following plot shows relationship between Happiness Score and Health with regression line:

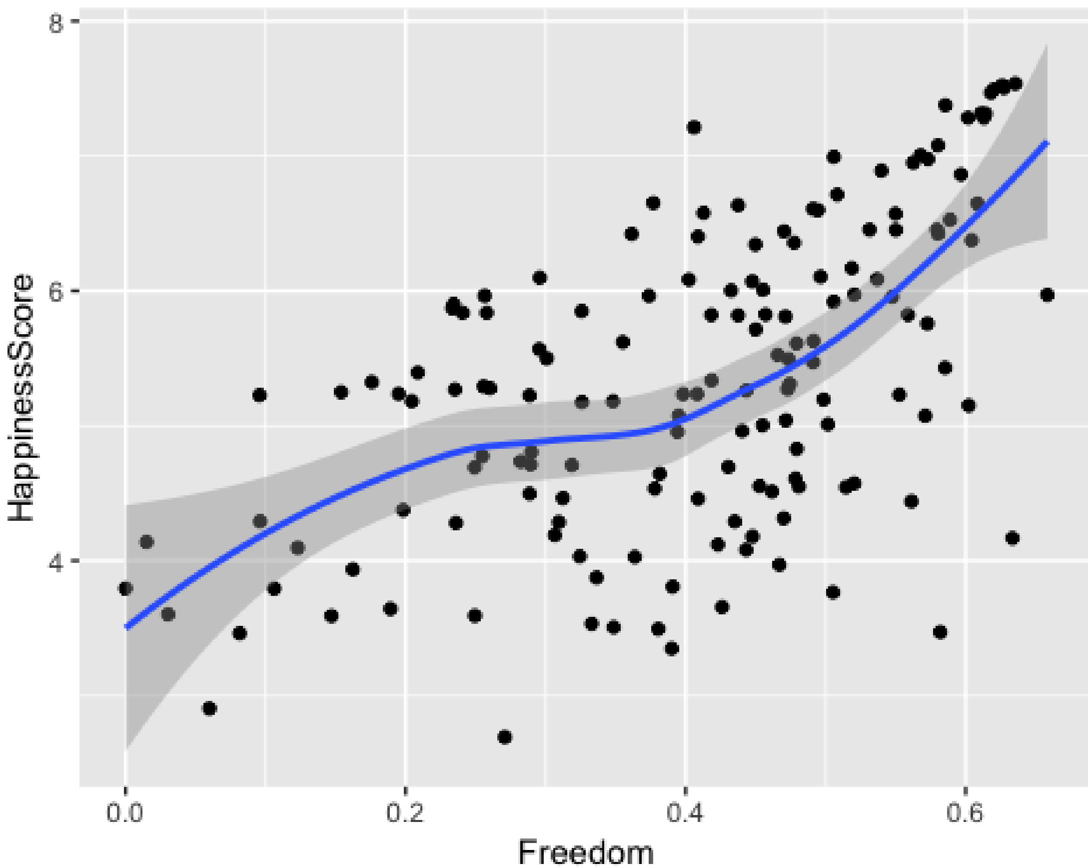
```
happiness_17_df %>% ggplot(aes(Health,HappinessScore)) + geom_point() + geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# Correlation value = 0.78 (High)
```

Following plot shows relationship between Happiness Score and Freedom with regression line:

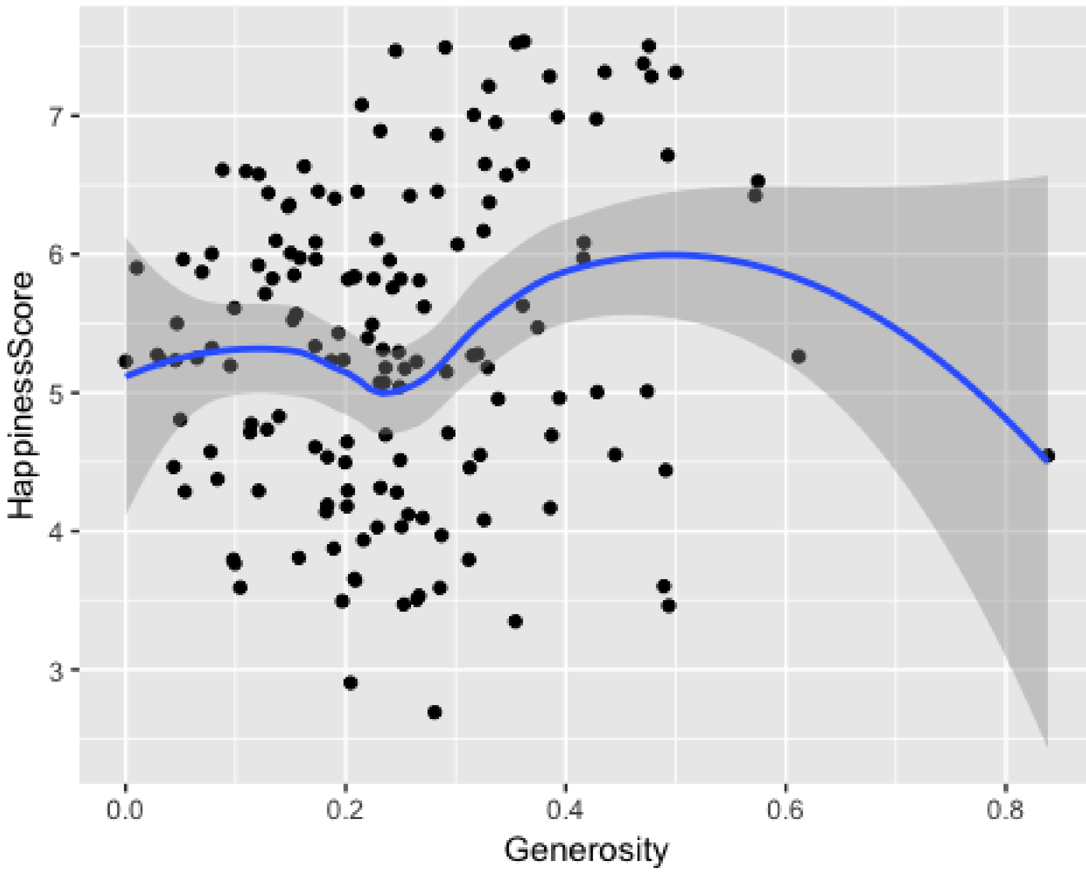
```
happiness_17_df %>% ggplot(aes(Freedom,HappinessScore)) + geom_point() + geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# Correlation value =  0.57 (Moderate)
```

Following plot shows relationship between Happiness Score and Generosity with regression line:

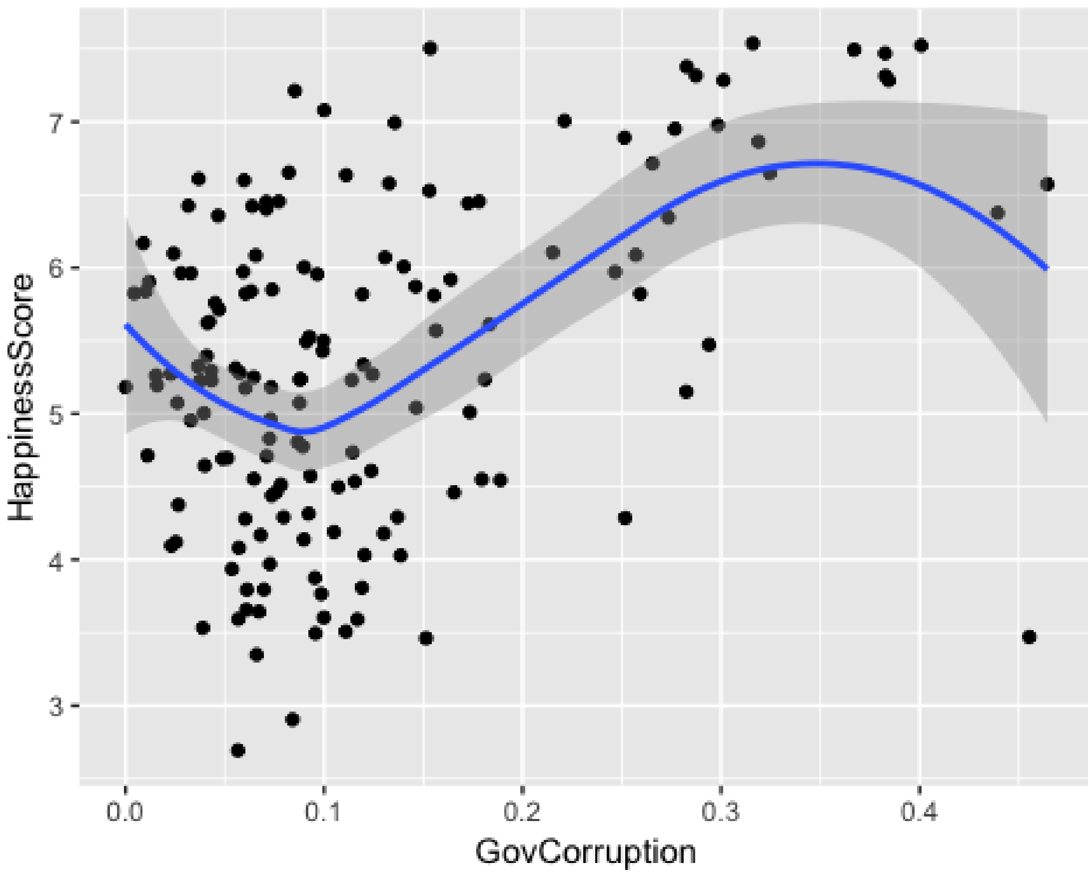
```
happiness_17_df %>% ggplot(aes(Generosity,HappinessScore)) + geom_point() + geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# Correlation value = 0.16 (Low)
```

Following plot shows relationship between Happiness Score and GovCorruption with regression line:

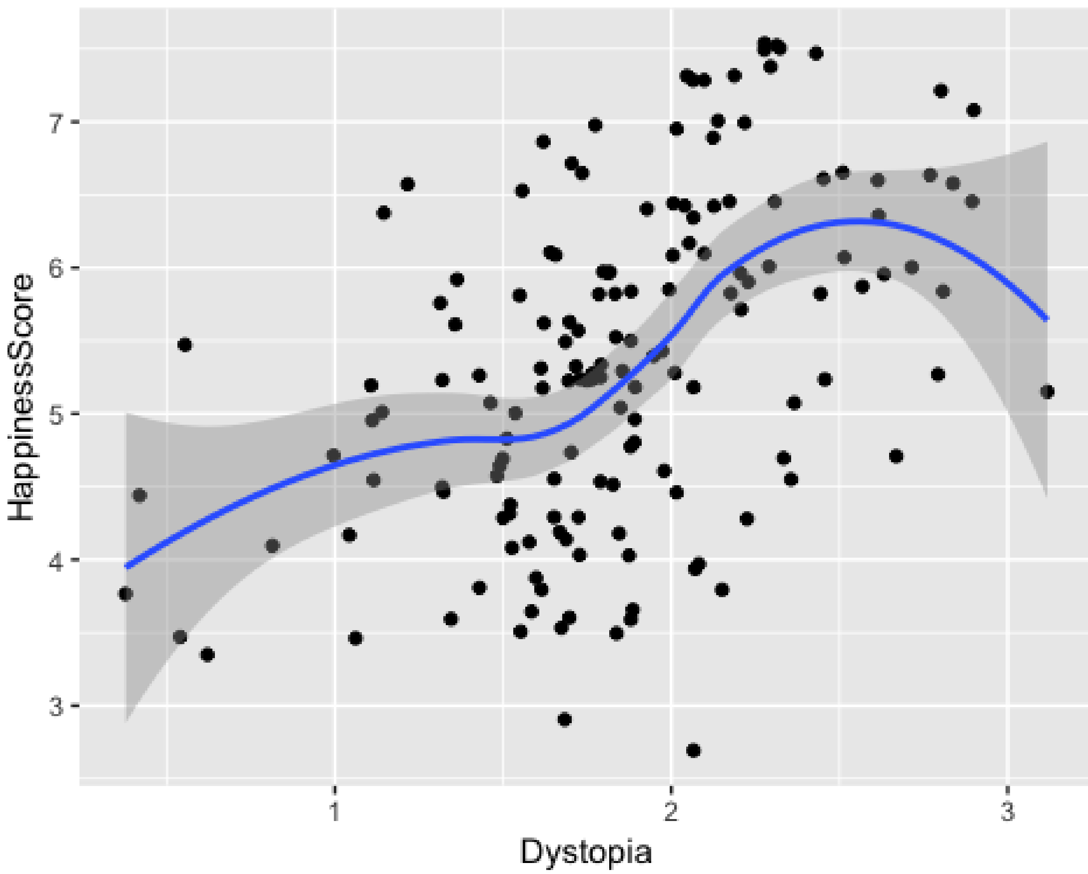
```
happiness_17_df %>% ggplot(aes(GovCorruption,HappinessScore)) + geom_point() + geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# Correlation value = 0.43 (Moderate)
```

Following plot shows relationship between Happiness Score and Dystopia with regression line:

```
happiness_17_df %>% ggplot(aes(Dystopia,HappinessScore)) %>% + geom_point() + geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# Correlation value = 0.48 (Moderate)
```

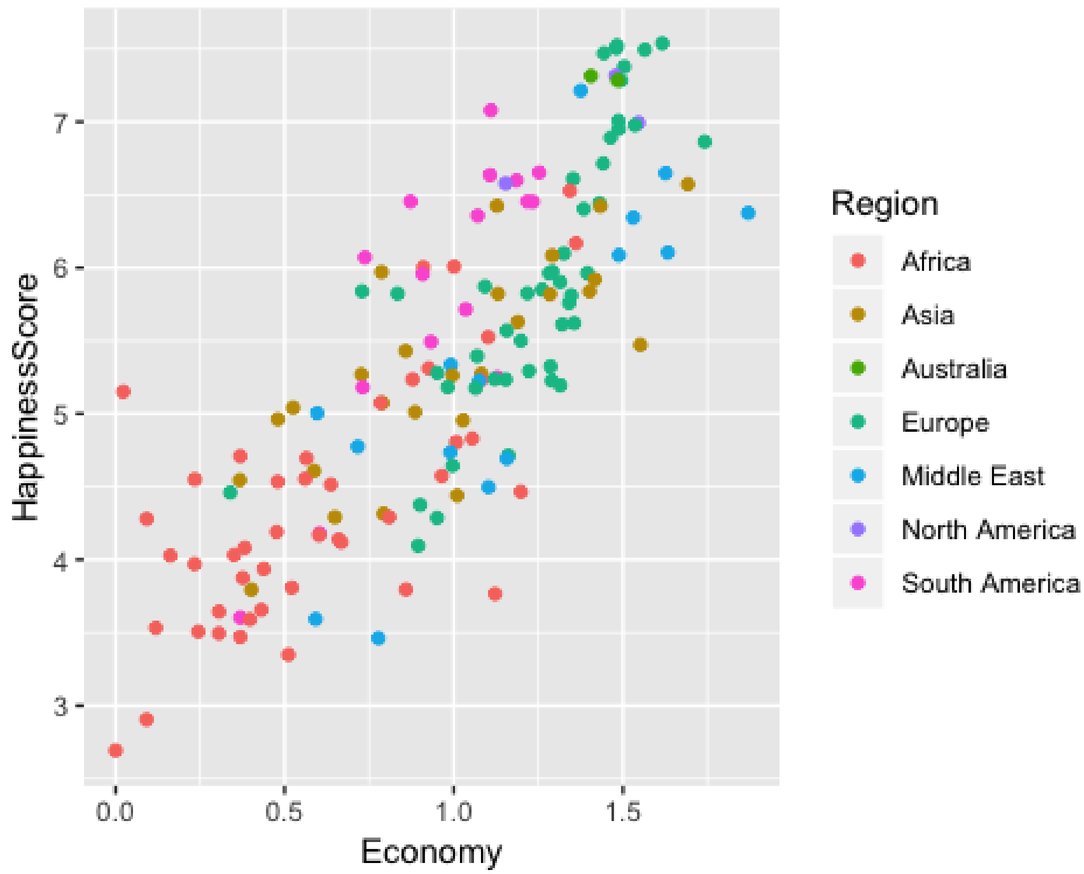
We can see that these graphs show the same result: Happiness Score is highly correlated to Economy, Family and Health. Moderately correlated to Freedom, Trust (or government corruption) and Dystopla. Loosely correlated to Generosity.

Key features of Happiness data follow a bivariate normal distribution with Happiness Score.

Scatter Plot for different regions:

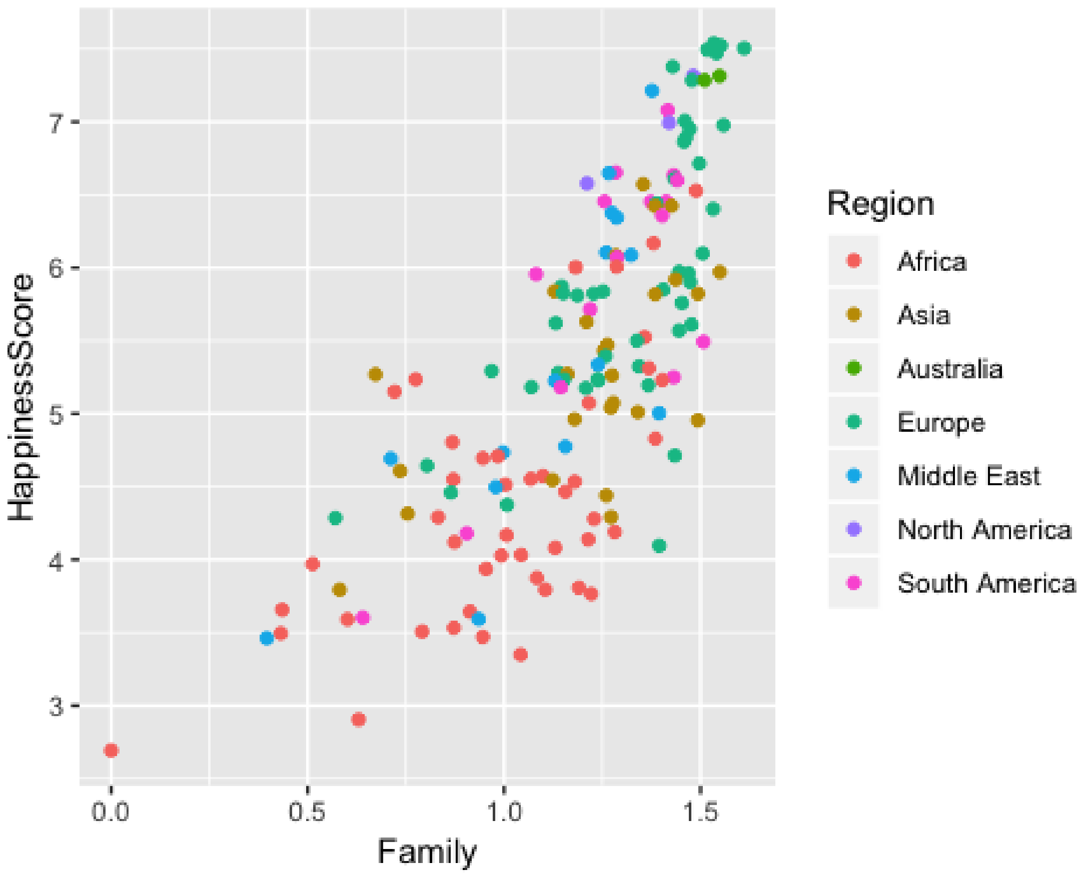
Following plot shows Happiness Score vs Economy for each region:

```
happiness_17_df %>% ggplot(aes(Economy,HappinessScore,color = Reg
ion)) + geom_point()
```



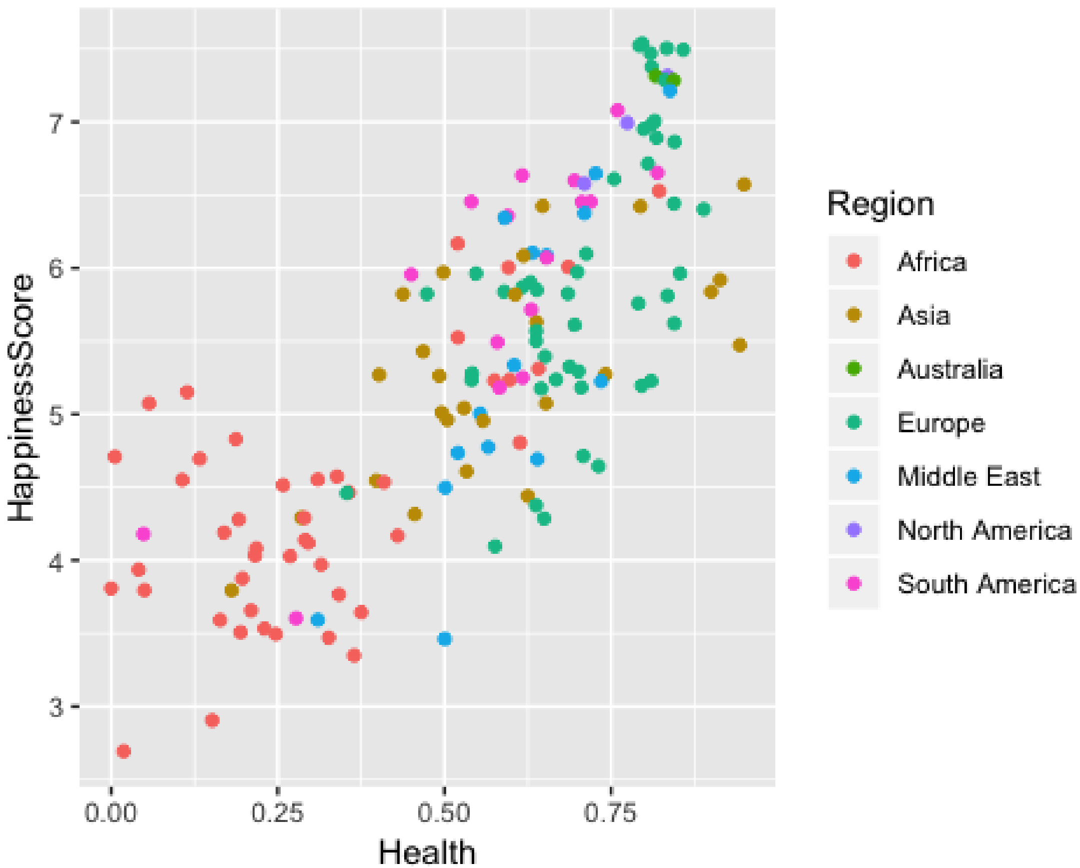
Following plot shows Happiness Score vs Family for each region:

```
happiness_17_df %>% ggplot(aes(Family,HappinessScore,color = Regi
on)) + geom_point()
```



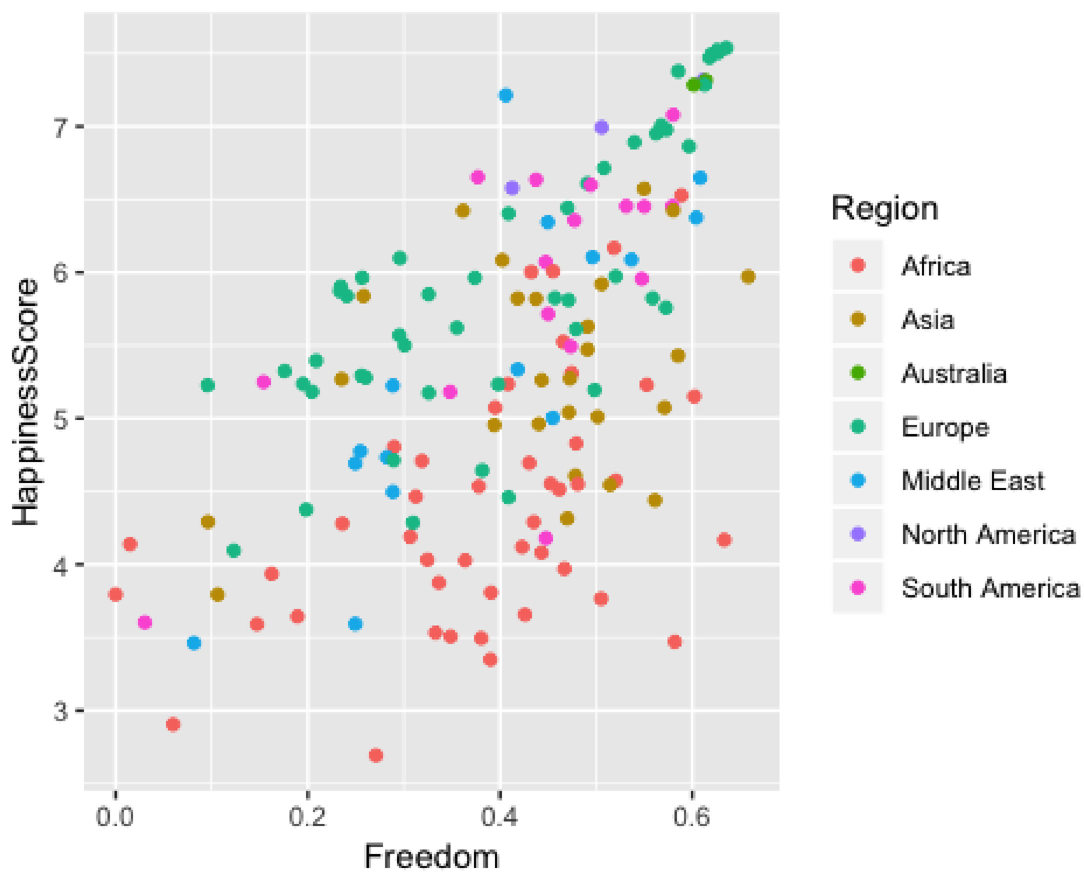
Following plot shows Happiness Score vs Health for each region:

```
            happiness_17_df %>% ggplot(aes(Health,HappinessScore,color = Regi
on)) + geom_point()
```



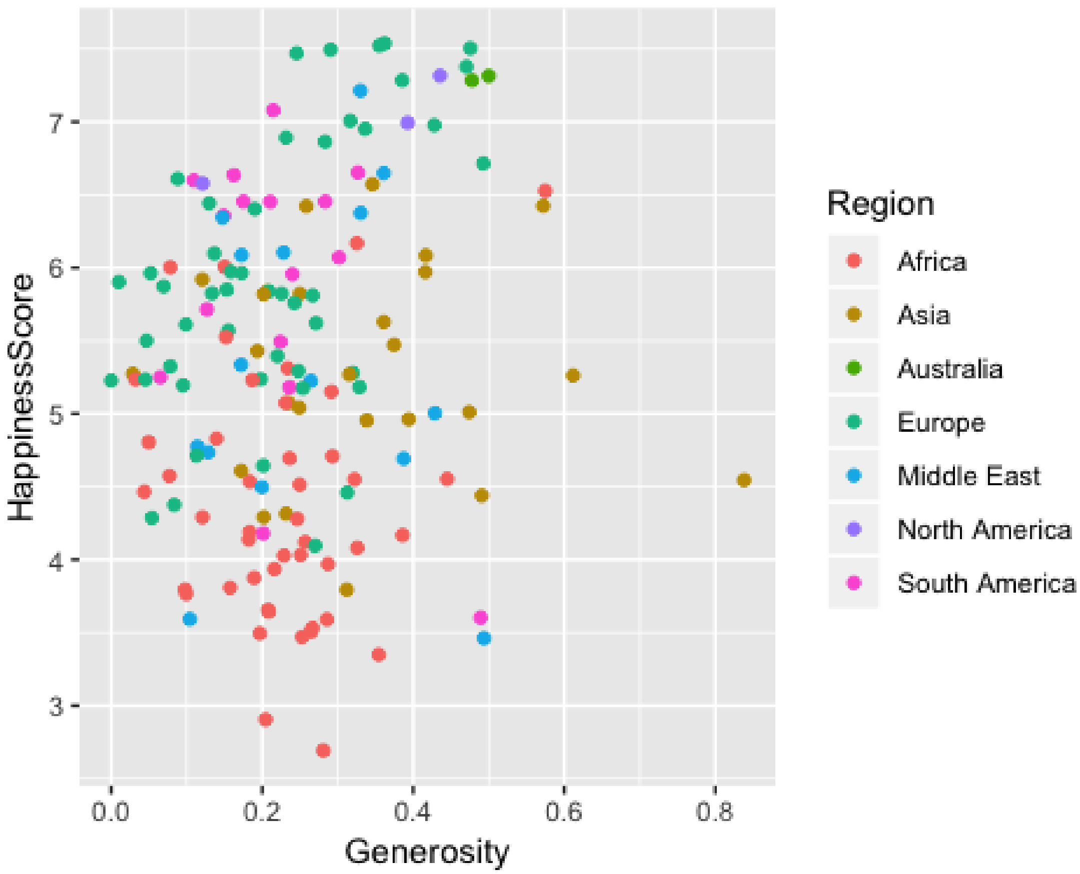
Following plot shows Happiness Score vs Freedom for each region:

```
happiness_17_df %>% ggplot(aes(Freedom,HappinessScore,color = Reg
ion)) + geom_point()
```



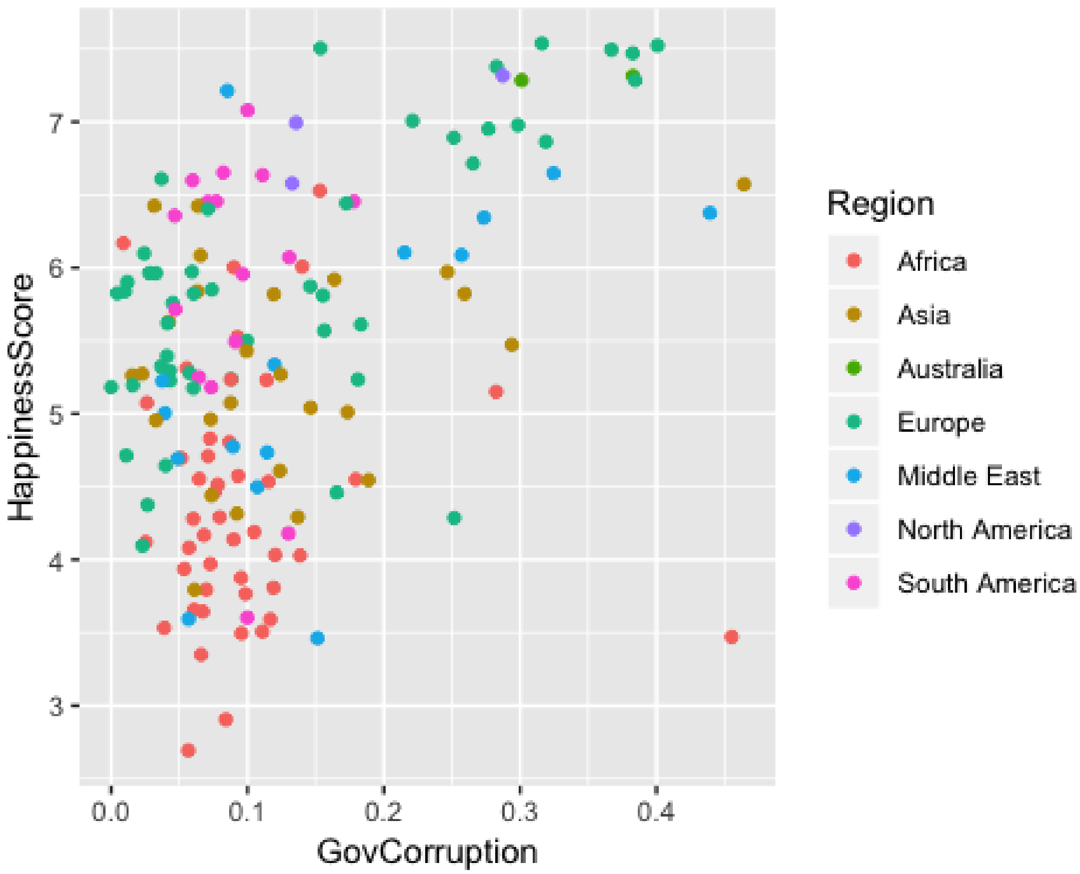
Following plot shows Happiness Score vs Generosity for each region:

```
            happiness_17_df %>% ggplot(aes(Generosity,HappinessScore,color =
Region)) + geom_point()
```



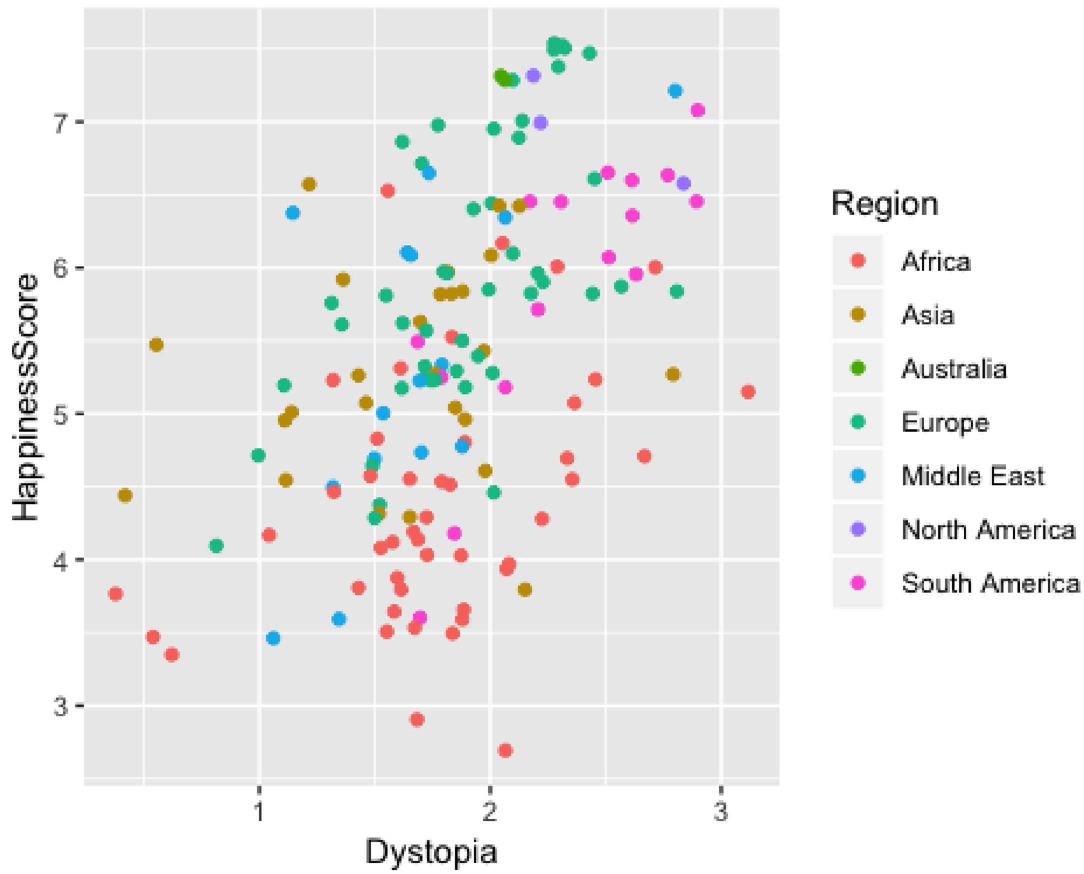
Following plot shows Happiness Score vs GovCorruption for each region:

```
            happiness_17_df %>% ggplot(aes(GovCorruption,HappinessScore,color
= Region)) + geom_point()
```



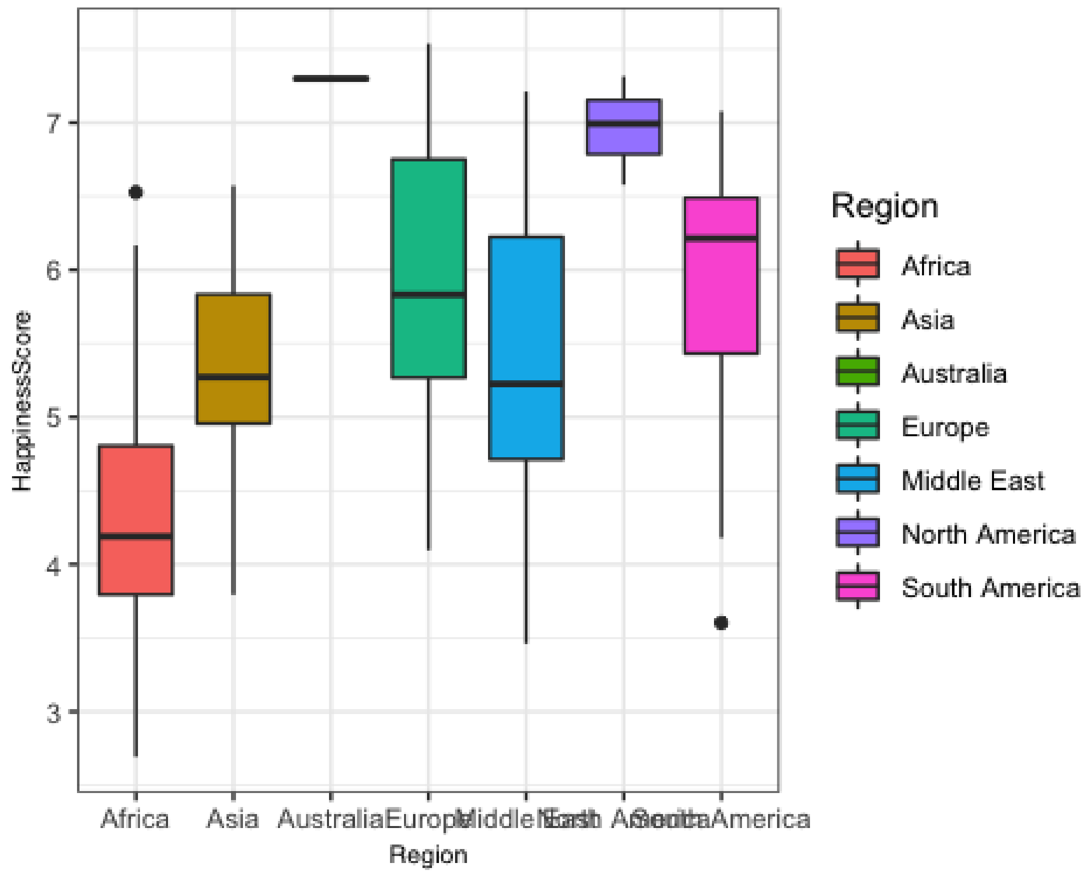
Following plot shows Happiness Score vs Dystopia for each region:

```
          happiness_17_df %>% ggplot(aes(Dystopia,HappinessScore,color = Re
gion)) + geom_point()
```



Following graph displays Happiness Score in different Regions:

```
        ggplot(happiness_17_df , aes(Region, HappinessScore)) + geom_boxplot(
aes(fill=Region)) + theme_bw() +
          theme(axis.title = element_text(family = "Helvetica", size = (8)))
```



We can see that Australia region has highest mean Happiness Score as there are just 2 countries in that region. North America is the region with 2nd highest average Happiness Score as it has just 3 countries. South America comes at the 3rd place based on mean Happiness Score and Europe stands 4th in this list.

Following are the rankings of regions based on mean Happiness Score in the region:

```
1 Australia
2 North America
3 South America
4 Europe
5 and 6 Asia and Middle East are almost the same
7 Africa
```

Training and Test Sets:

We will create Train and Test set using happiness_17_df. Note that we will train our models using 50% of the happiness_17_df data and then predict happiness score on the remaining 50% of the test data.

Following code will create Test and Train set:
Note that we will only consider following features in train and test set: Economy, Family, Health, Freedom, Generosity, GovCorruption and Dystopia

```
        index <- createDataPartition(happiness_17_df$HappinessScore, times = 
1, p = 0.5, list = FALSE)
        train_set <- happiness_17_df[-index,c("HappinessScore","Economy","Fam
ily","Health","Freedom","Generosity","GovCorruption", "Dystopia")]
        test_set <- happiness_17_df[-index,c("HappinessScore","Economy","Fami
ly","Health","Freedom","Generosity","GovCorruption", "Dystopia")]

    # Following code will display dimensions of train and test sets

        dim(train_set)

## [1] 76  8

        dim(test_set)

## [1] 76  8
```

RMSE FUNCTION:

We will use RMSE (Root Mean Square Error) as the loss function to define the best model/approach and gauge model stability.
Note that lower the RMSE, better is the prediction.
Following RMSE function will take 2 inputs, true and predicted ratings, and will return RMSE value. This function will be used to calculate RMSE values for all the models:

```
        RMSE <- function(true_ratings, predicted_ratings){
          sqrt(mean((true_ratings - predicted_ratings)^2))
        }
```

GLOBAL VARIABLES: RMSE result dataframe and Model Name:

```
    # Following code will flush previous rmse_result, if this code is execute
d 2nd time
        if(exists('rmse_result') == TRUE)
        {
          if(length(rmse_result) > 0)
          {
              rm(rmse_result)
          }
        }
```

```
    rmse_result <- data.frame(method = 'datframe initialization' , RMSE = 0)
# Initialization of rmse_result dataframe

    # Remove 1st entry of rmse_result, as its dummy

        rmse_result <- rmse_result[-1,0]

    # Model Name will be stored in following variable

        model_name <- ' '
```

RESULT FUNCTION:

This function will add RMSE values into rmse_result. All models will call this function to report their RMSE values. Display rmse_result if display attribute is true.

```
        add_RMSE_result <- function(model_name,RMSE_value,display){
          rmse_result <<- bind_rows(rmse_result , data.frame(method = model_n
ame , RMSE = RMSE_value))
          if(display){
              rmse_result %>% knitr::kable()
          }
        }
```

Plot Smooth Density Graph Function:

This function will take predicted ratings of a model and will make smooth denisity plot. Note that predicted ratings will be displayed against true ratings (i.e. test_set Happiness Score ratings).

```
    displaySmoothPlot <- function(pred){
      happiness_result <- data.frame(pred = pred, true = test_set$HappinessSc
ore)
      happiness_result %>% ggplot(aes(true,pred)) + geom_point() + geom_smoot
h()
    }
```

Machine Learning Models:

Model 1 - Regression:

```r
model_name <- 'Model 1 : Regression'
```

In Data Exploration and Visualization section, we have seen that different features (Economy, Family, Health, Freedom, Generosity, GovCorruption and Dystopia) have linear relationship with Happiness Score

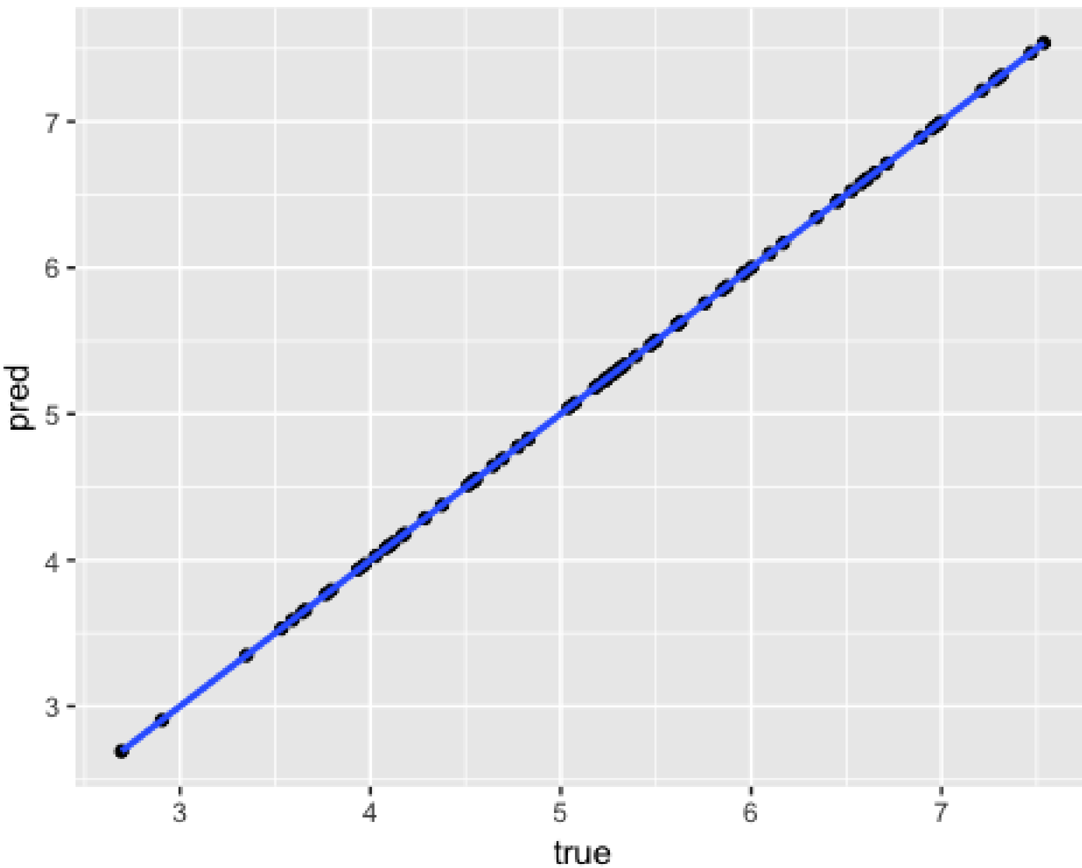
So we will use regression model to predict happiness score:

```r
    # Following code will provide us happiness_lm_hat using regression fu
nction "lm"

        happiness_lm_hat = lm(HappinessScore ~ . , train_set)

    # Following code will display summary of happiness_lm_hat

        summary(happiness_lm_hat)

##
## Call:
## lm(formula = HappinessScore ~ ., data = train_set)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -4.791e-04 -2.345e-04 -5.820e-06  2.381e-04  4.575e-04
##
## Coefficients:
##                 Estimate Std. Error   t value Pr(>|t|)
## (Intercept)   -1.037e-04  1.972e-04    -0.526    0.601
## Economy        1.000e+00  1.760e-04  5681.494   <2e-16 ***
## Family         1.000e+00  1.717e-04  5825.479   <2e-16 ***
## Health         1.000e+00  2.625e-04  3809.712   <2e-16 ***
## Freedom        1.000e+00  3.013e-04  3318.879   <2e-16 ***
## Generosity     1.000e+00  2.832e-04  3533.041   <2e-16 ***
## GovCorruption  9.996e-01  4.279e-04  2335.860   <2e-16 ***
## Dystopia       1.000e+00  6.537e-05 15297.776   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0002822 on 68 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.823e+08 on 7 and 68 DF,  p-value: < 2.2e-16

    # Now we will use this model to predict Happiness Score for test set

        happiness_lm_pred <- predict(happiness_lm_hat, test_set)

    # Following is the mean squared error for liner regression model:

        MSE_lm <- RMSE(test_set$HappinessScore, happiness_lm_pred)
```

```r
        # Add results into rmse_result

            add_RMSE_result(model_name,MSE_lm,TRUE)
```

method	RMSE
Model 1 : Regression	0.000267

```r
        # Following code will generate Predicted vs True ratings graph with r
egression line

            displaySmoothPlot(happiness_lm_pred)

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



GLM Model:

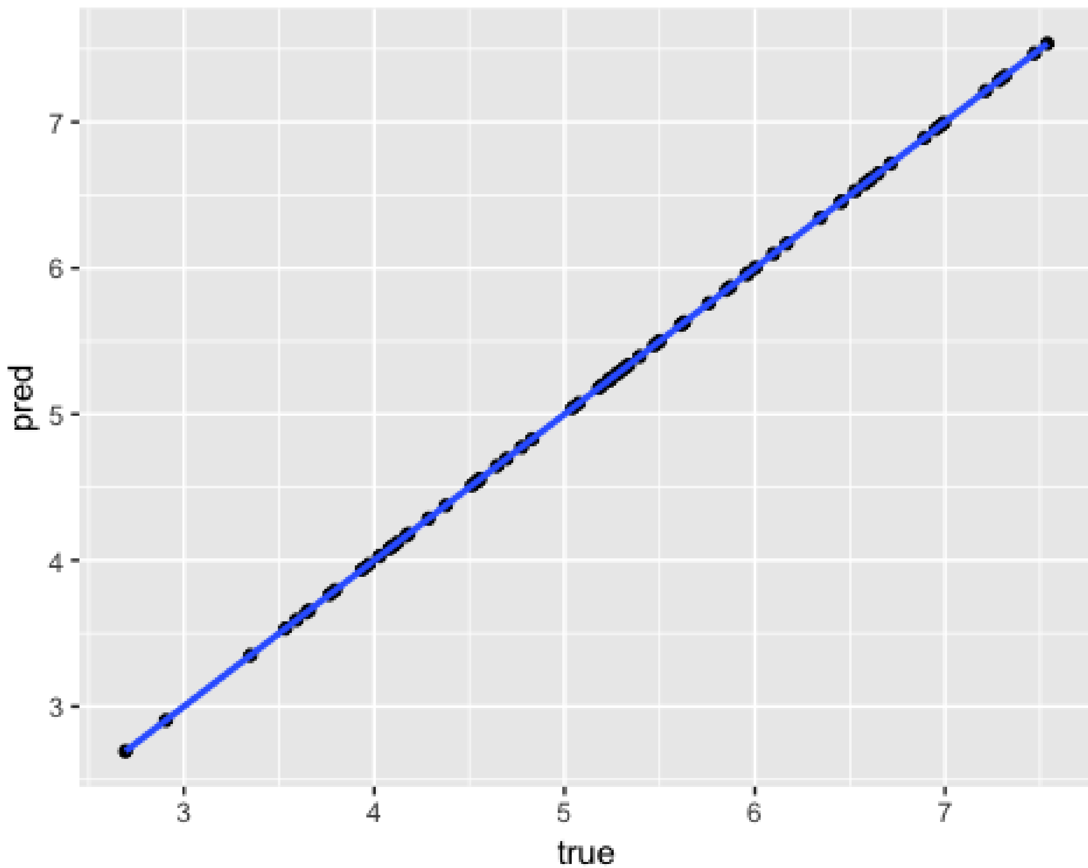
```r
    model_name <- 'Model 2 : GLM'
```

Poisson glm regression is useful when we are predicting an outcome variable that represents counts from a set of continuous predictors.

```r
            # Following code will provide us happiness_lm_hat using train function of carets package

                happiness_glm_hat <- train(HappinessScore ~ . , data = train_set, method = 'glm')

            # Now we will use this model to predict Happiness Score for test set

                happiness_glm_pred <- predict(happiness_glm_hat, test_set)

            # Following is the mean squared error for liner regression model:

                RMSE_glm <- RMSE(test_set$HappinessScore,happiness_glm_pred)

        # Add results into rmse_result

            add_RMSE_result(model_name,RMSE_glm,TRUE)
```

method	RMSE
Model 1 : Regression	0.000267
Model 2 : GLM	0.000267

```r
        # Following code will generate Predicted vs True ratings graph with regression line

            displaySmoothPlot(happiness_glm_pred)

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

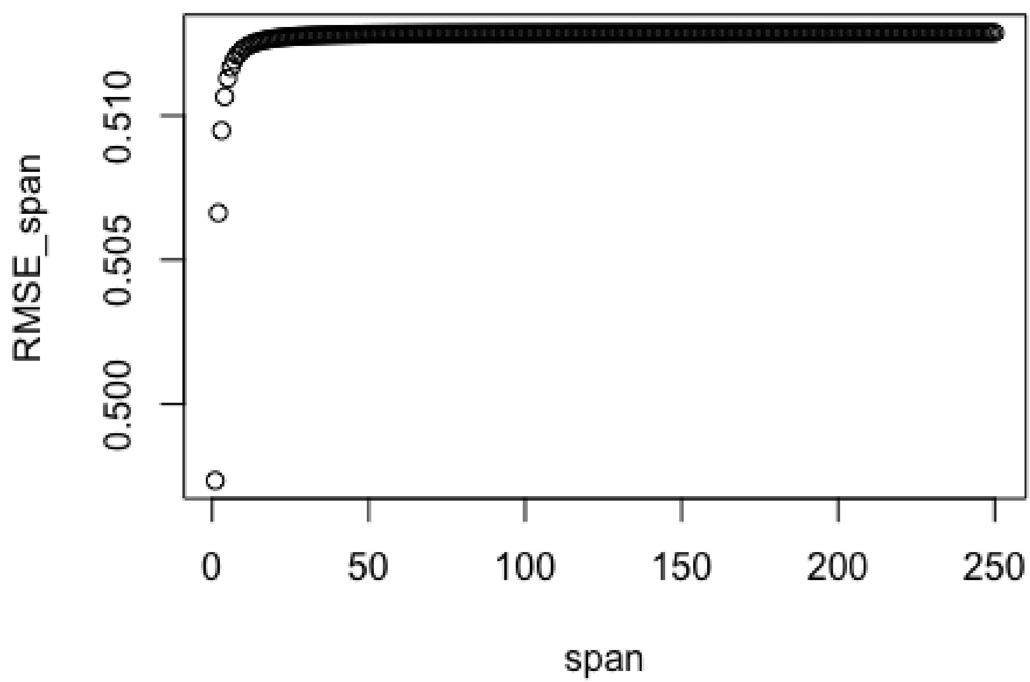


Loess Model:

Local Regression fits multiple regressions in local neighborhood. The size of the neighborhood can be controlled using the span argument. It controls the degree of smoothing.

```
    model_name <- 'Model 3 : Loess'

        # Following code will tune span value that will provide lowest RMSE v
alue
        # Note than loess function can accept maximum of 4 features. So, we h
ave used the ones that are highly correlated to Happiness Score

            span <- c(seq(1:250))

            RMSE_span <- sapply(span, function(s){

              happiness_loess_hat <- loess(HappinessScore ~ Economy + Family
+ Health + Freedom, span = s, degree = 1 , data = train_set)
```

```
          happiness_loess_pred <- predict(happiness_loess_hat, test_set)

          MSE_loess <- RMSE(test_set$HappinessScore,happiness_loess_pred)

          return(MSE_loess)
        })

    # Following is the span that gives us the least RMSE value

        best_span <- span[which.min(RMSE_span)]

    # Following is the plot for RMSE vs span values

        plot(span,RMSE_span)
```

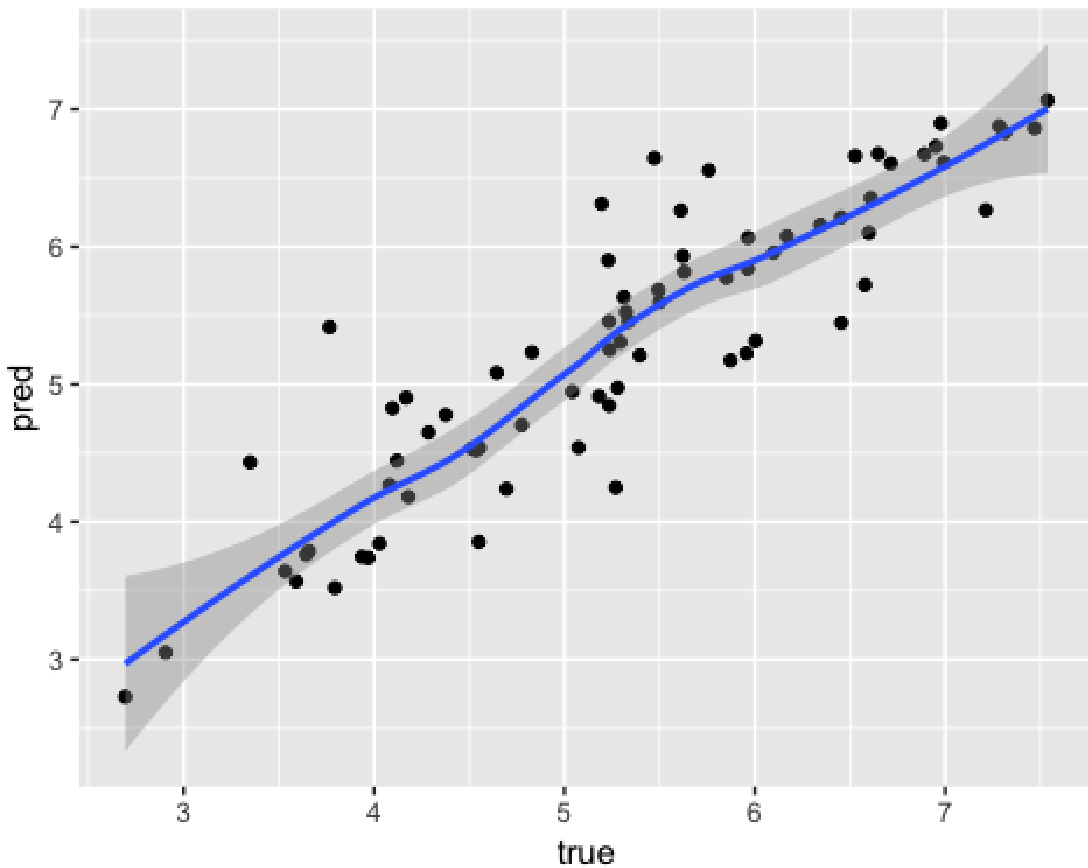



```
    # We will now use the best span value that we calculated above to fin
d predicted and RMSE

        # Following code will provide us happiness_loess_hat using loess 
function
```

```r
                happiness_loess_hat <- loess(HappinessScore ~ Economy + Famil
y + Health + Freedom, data = train_set, span = best_span, degree = 1)

            # Now we will use this model to predict Happiness Score for test
set

                happiness_loess_pred <- predict(happiness_loess_hat, test_set
)

            # Following is the mean squared error:

                RMSE_loess <- RMSE(test_set$HappinessScore,happiness_loess_pr
ed)

        # Add results into rmse_result

            add_RMSE_result(model_name,RMSE_loess,TRUE)
```

method	RMSE
Model 1 : Regression	0.0002670
Model 2 : GLM	0.0002670
Model 3 : Loess	0.4973461

```r
        # Following code will generate Predicted vs True ratings graph with r
egression line

            displaySmoothPlot(happiness_loess_pred)

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



KNN Model:

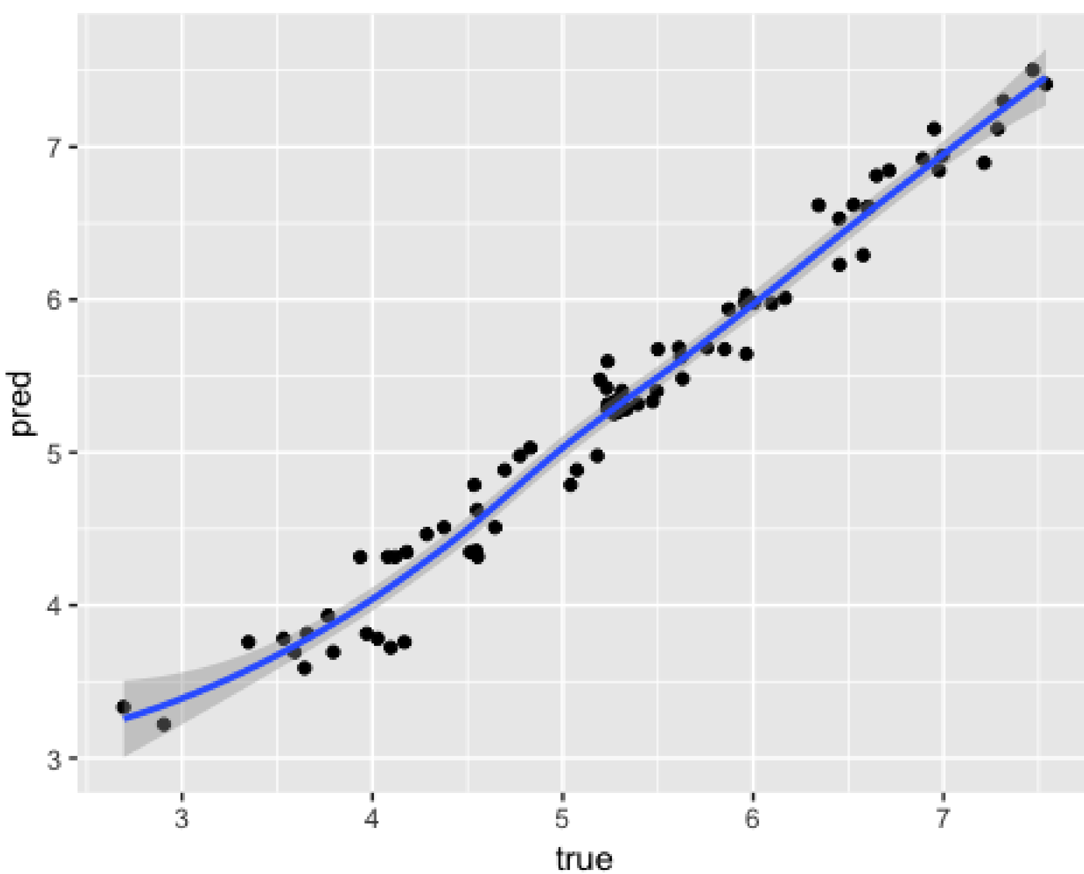
k nearest neighbours is similar to bin smoothing. Model estimates values based on k nearest neighbours. k can be tuned.

```
    model_name <- 'Model 4 : knn'

       # Following code will provide us happiness_lm_hat using train function
of carat package
        # We can tune value of k using tuneGrid attribute of train function
        # After multiple iterations, I found out the best tuneGrid that gives
the least RMSE value.  I we changes tuneGrid, it increase RMSE

          happiness_knn_hat <- train(HappinessScore ~ . , data = train_set, m
ethod = 'knn' , tuneGrid = data.frame(k = seq(1,2,0.1)) )

      # Now we will use this model to predict Happiness Score for test set

          happiness_knn_pred <- predict(happiness_knn_hat, test_set)
```

```r
# Following is the mean squared error

  RMSE_knn <- RMSE(test_set$HappinessScore,happiness_knn_pred)

 # Add results into rmse_result

   add_RMSE_result(model_name,RMSE_knn,TRUE)
```

method	RMSE
Model 1 : Regression	0.0002670
Model 2 : GLM	0.0002670
Model 3 : Loess	0.4973461
Model 4 : knn	0.1986335

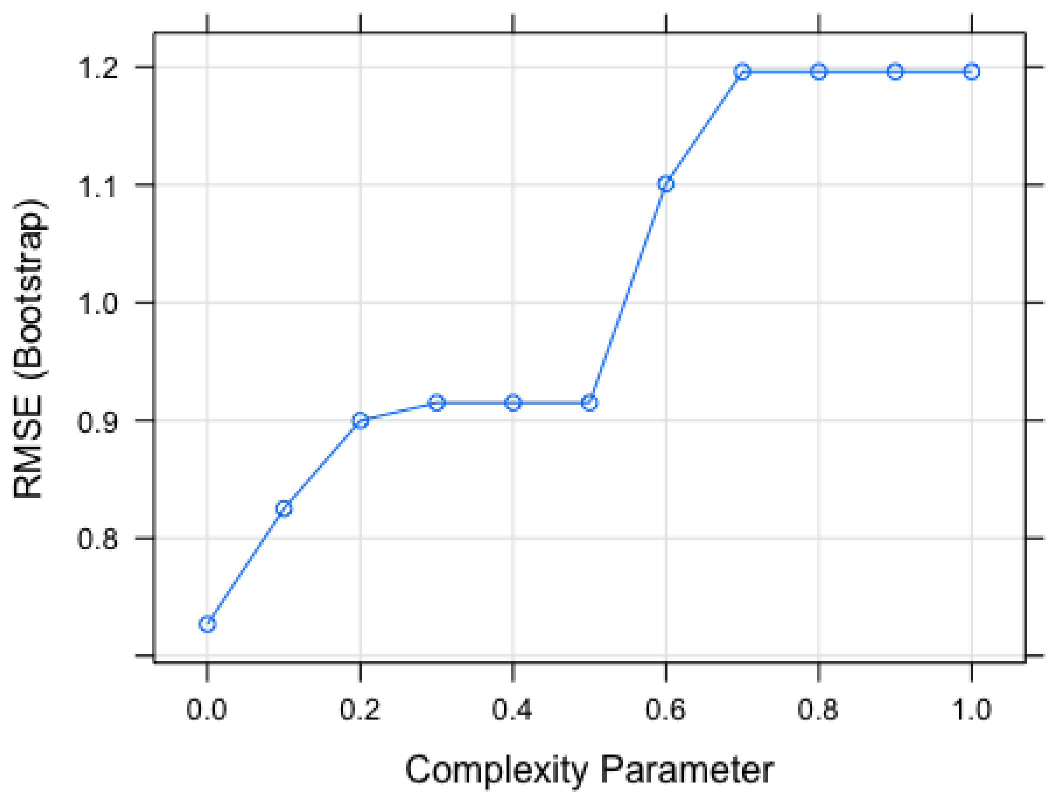
```r
 # Following code will generate Predicted vs True ratings graph with r
egression line

   displaySmoothPlot(happiness_knn_pred)

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



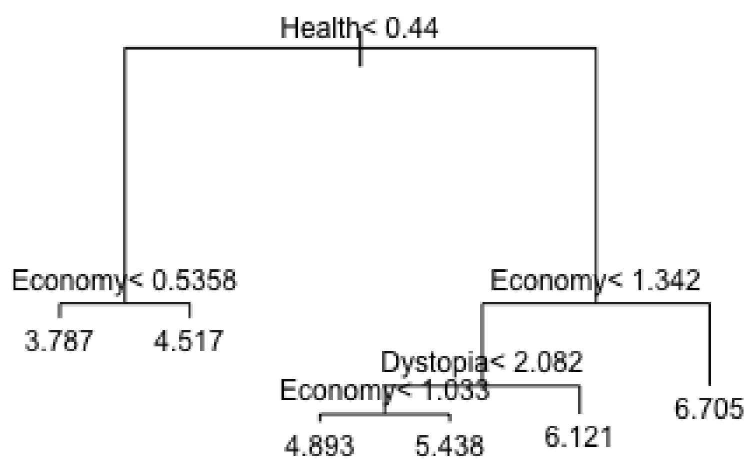
Tree Model:

Tree model divides data into decision tree that will be used to predict values.

```
    model_name <- 'Model 5 : Tree'

        # Following code will tune cp value to give us the best estimate usin
g train function of caret package

            happiness_rpart_hat <- train(HappinessScore ~ . , method = 'rpart
' ,
                                         tuneGrid = data.frame(cp = seq(0,1,0
.1)),
                                         data = train_set)

        # Following plot displays RMSE vs cp (Complexity Parameter)

            plot(happiness_rpart_hat)
```



```r
        # Following is the Tree diagram based on the best model (with best cp value)

            plot(happiness_rpart_hat$finalModel, margin = 0.1)
            text(happiness_rpart_hat$finalModel, cex = 0.75)
```

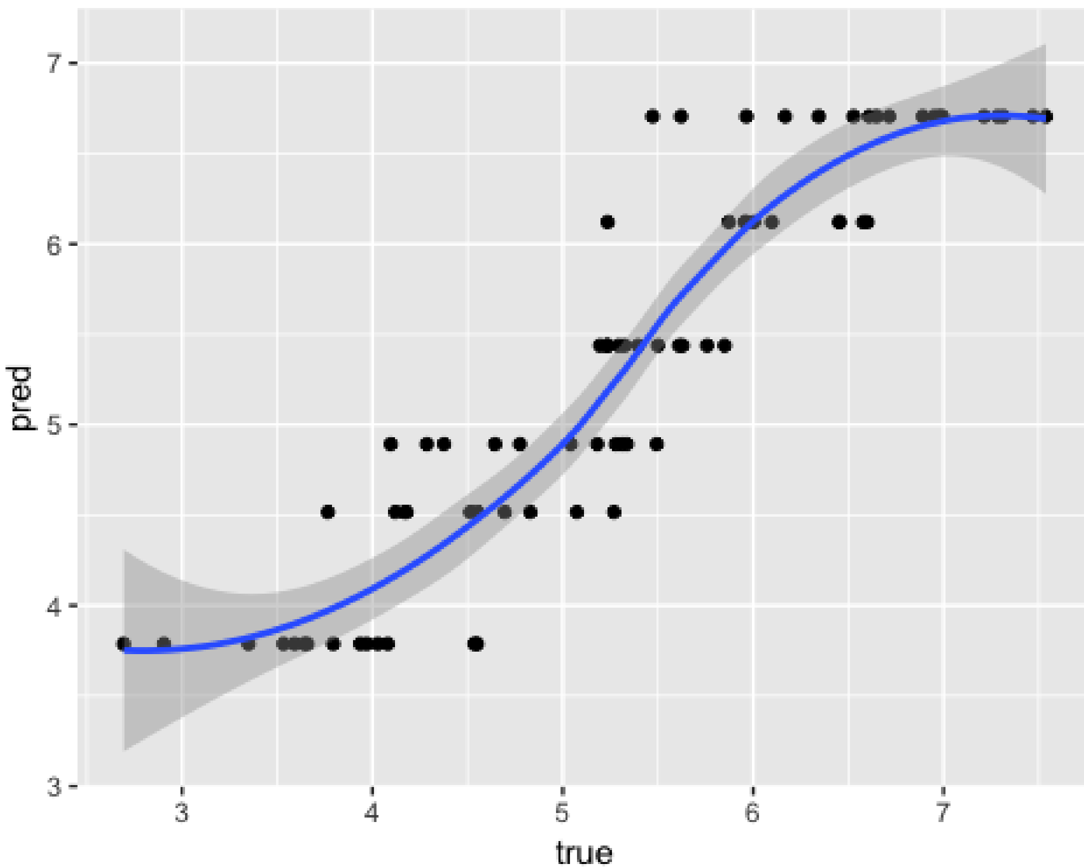



```r
        # Now we will use this model to predict Happiness Score for test set

            happiness_rpart_pred <- predict(happiness_rpart_hat, test_set)

        # Following is the mean squared error

            RMSE_tree <- RMSE(test_set$HappinessScore,happiness_rpart_pred)

        # Add results into rmse_result

            add_RMSE_result(model_name,RMSE_tree,TRUE)
```

method	RMSE
Model 1 : Regression	0.0002670
Model 2 : GLM	0.0002670

Model 3 : Loess	0.4973461
Model 4 : knn	0.1986335
Model 5 : Tree	0.4708843

```
        # Following code will generate Predicted vs True ratings graph with r
egression line

            displaySmoothPlot(happiness_rpart_pred)

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

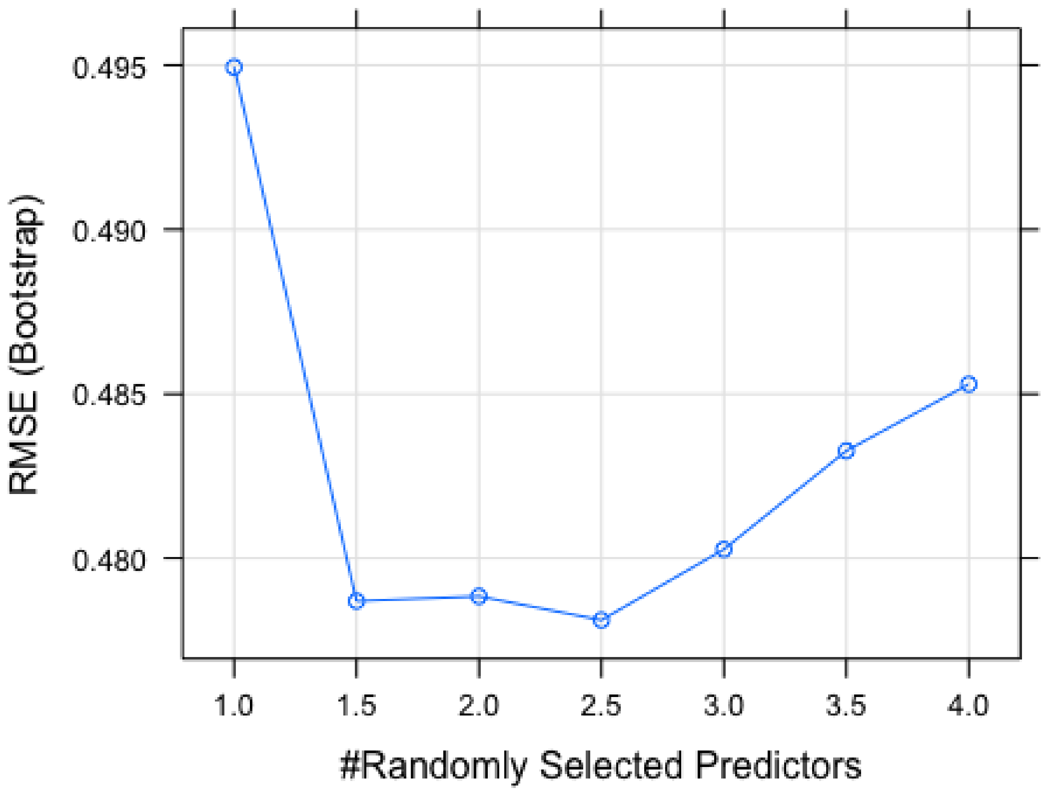


Random Forest:

Random forest improves prediction performance and reduce instability by averaging multiple decision trees, a forest of trees constructed with randomness.
Following are the 2 functions that implement Random Forest model:

Random Forest : "RM" function:

```r
        model_name <- 'Model 6.1 : Random Forest - RF'

            # Following code will provide us happiness_randomforest_rm_hat using train function of caret package.  Note method name will be "rm" and we will tune mtry attribute of "rm" function

                happiness_randomforest_rf_hat <- train(HappinessScore ~ . , data = train_set, method = 'rf', tuneGrid = data.frame(mtry = seq(1,4,0.5)))

            # Following plot displays RMSE vs Randomly selected predictors

                plot(happiness_randomforest_rf_hat)
```

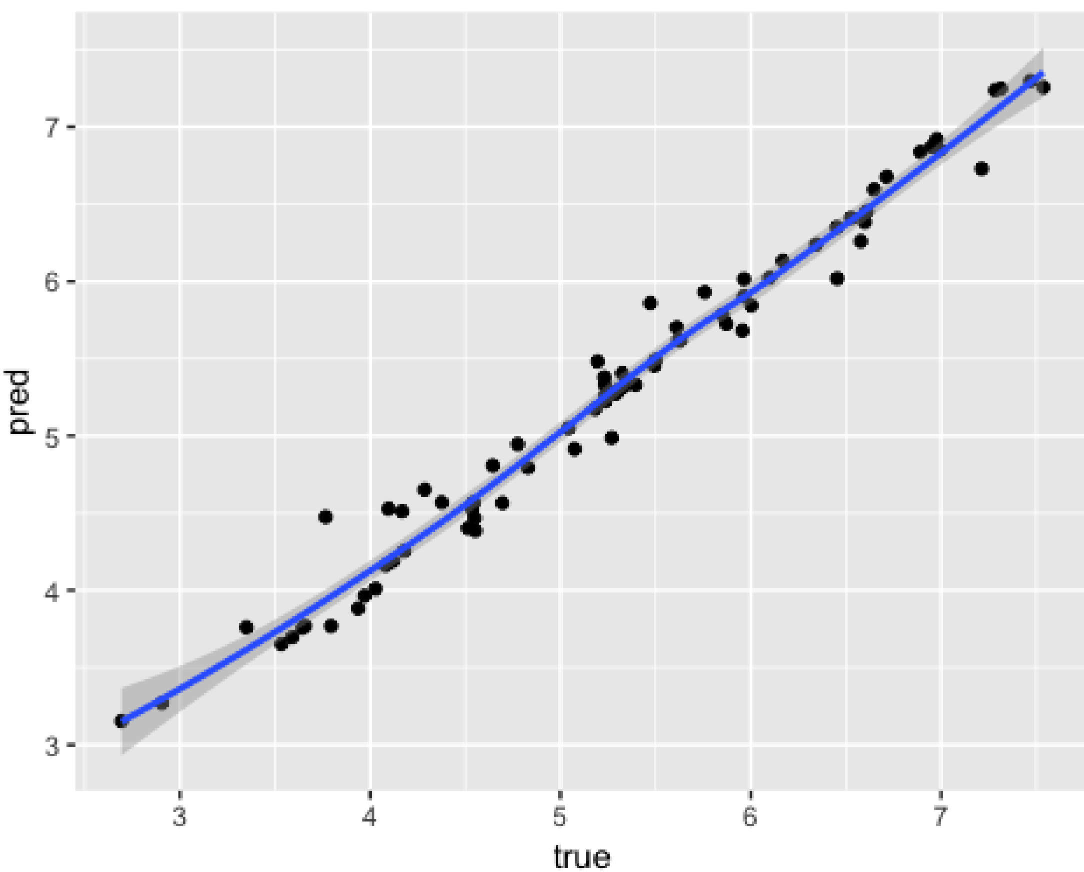



```r
            # Now we will use this model to predict Happiness Score for test set

                happiness_randomforest_rf_pred <- predict(happiness_randomforest_rf_hat, test_set)

            # Following is the mean squared error

                RMSE_randomforest_rf <- RMSE(test_set$HappinessScore,happiness_randomforest_rf_pred)
```

```r
# Add results into rmse_result

add_RMSE_result(model_name,RMSE_randomforest_rf,TRUE)
```

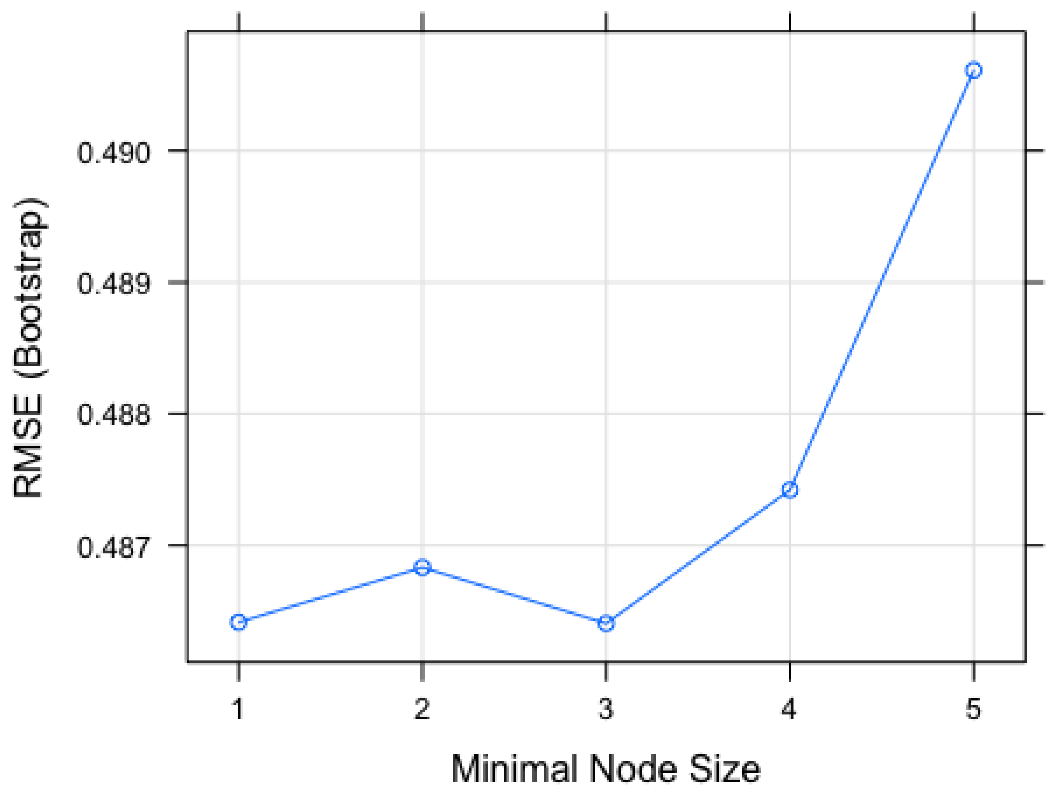
method	RMSE
Model 1 : Regression	0.0002670
Model 2 : GLM	0.0002670
Model 3 : Loess	0.4973461
Model 4 : knn	0.1986335
Model 5 : Tree	0.4708843
Model 6.1 : Random Forest - RF	0.1995343

```r
# Following code will generate Predicted vs True ratings graph wi
th regression line

displaySmoothPlot(happiness_randomforest_rf_pred)

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Random Forest : "RM" function:

```
        model_name <- 'Model 6.2 : Random Forest - Rborist'

            # Following code will provide us happiness_randomforest_rborist_h
at using train function of caret package.  Note method name will be "Rborist"
and we will tune minNode attribute keeping predFixed constant of "Rborist" fu
nction

                happiness_randomforest_rborist_hat <- train(HappinessScore ~
. , data = train_set, method = 'Rborist', tuneGrid = data.frame(predFixed = 3
,

minNode = seq(1,5)))

            # Following plot displays RMSE vs Minimal Node Size

                plot(happiness_randomforest_rborist_hat)
```

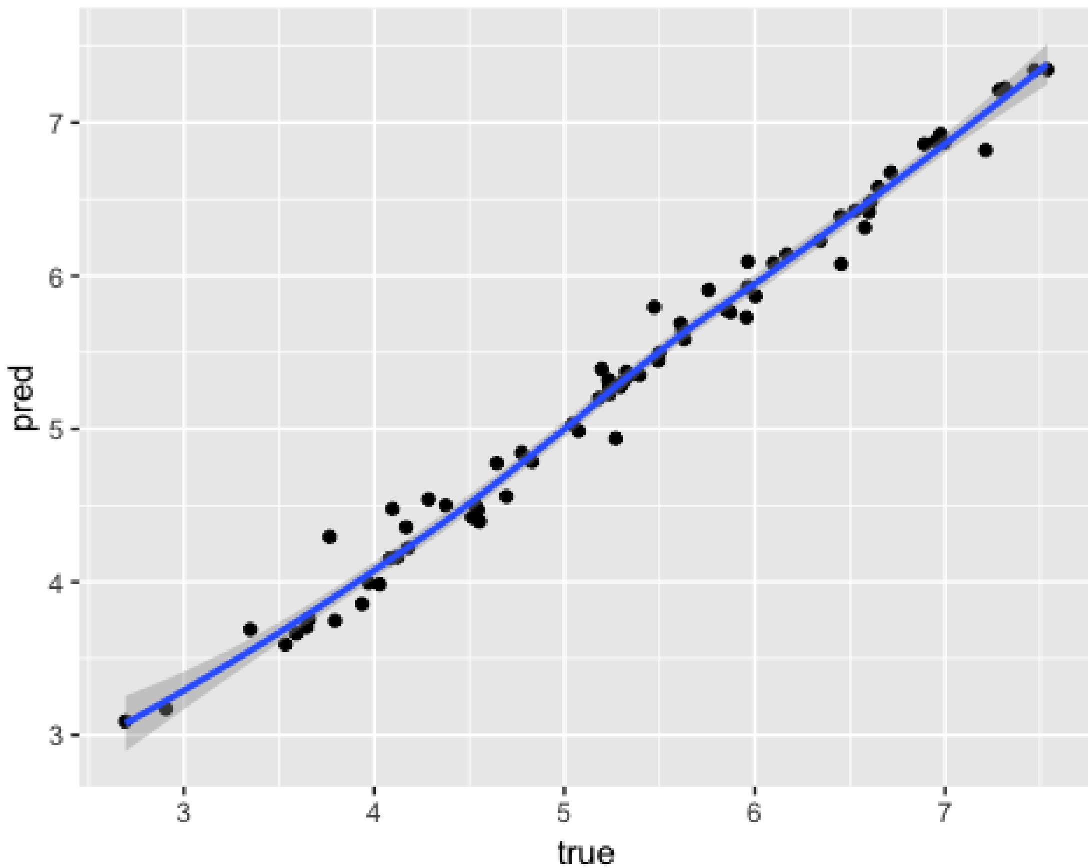



```
            # Now we will use this model to predict Happiness Score for test
set
```

```
                happiness_randomforest_rborist_pred <- predict(happiness_rand
omforest_rborist_hat, test_set)

            # Following is the mean squared error

                RMSE_randomforest_rborist <- RMSE(test_set$HappinessScore,hap
piness_randomforest_rborist_pred)

            # Add results into rmse_result

                add_RMSE_result(model_name,RMSE_randomforest_rborist,TRUE)
```

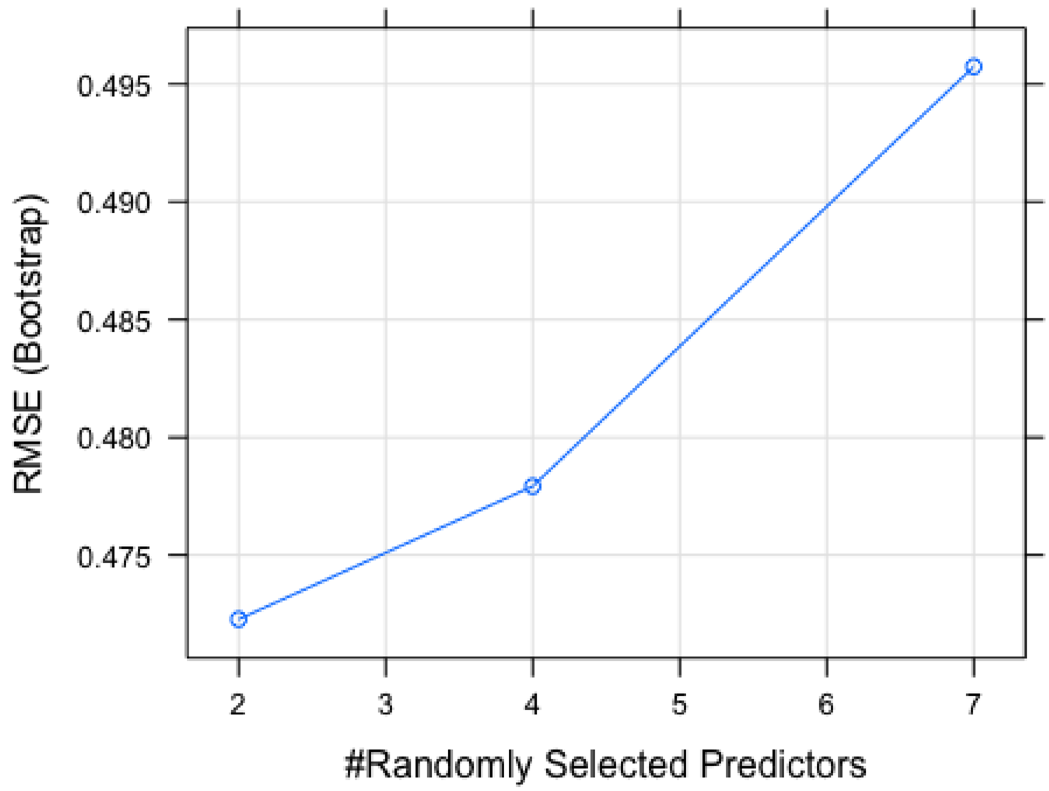
method	RMSE
Model 1 : Regression	0.0002670
Model 2 : GLM	0.0002670
Model 3 : Loess	0.4973461
Model 4 : knn	0.1986335
Model 5 : Tree	0.4708843
Model 6.1 : Random Forest - RF	0.1995343
Model 6.2 : Random Forest - Rborist	0.1618590

```
            # Following code will generate Predicted vs True ratings graph wi
th regression line

                displaySmoothPlot(happiness_randomforest_rborist_pred)

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



GamLoess:

```
    model_name <- 'Model 7 : GamLoess'

        # Following is the grid to tuen span and degree for gamloess function

            gamloess_grid <- expand.grid(span = seq(0,2,0.1), degree = 1)

        # Following code will provide us happiness_gamloess_hat by tuning our
grid

            happiness_gamloess_hat <- train(HappinessScore ~ . , data = train
_set,
                                            model = 'gamLoess',
                                            tuningGrid = gamloess_grid)

        #  Following plot displays RMSE vs Randomly Selected Predictors

            plot(happiness_gamloess_hat)
```

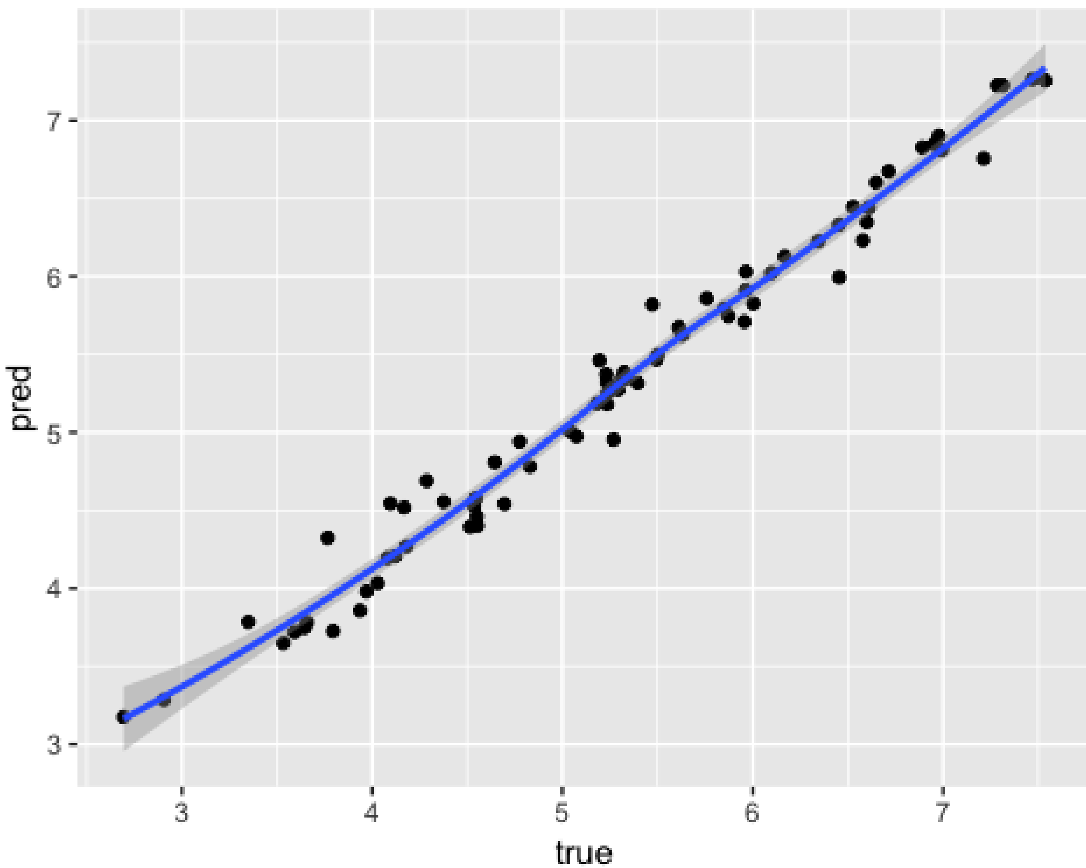



```
        # Now we will use this model to predict Happiness Score for test set

            happiness_gamloess_pred <- predict(happiness_gamloess_hat, test_s
et)

        # Following is the mean squared error

            RMSE_gamloess <- RMSE(test_set$HappinessScore,happiness_gamloess_
pred)

        # Add results into rmse_result

            add_RMSE_result(model_name,RMSE_gamloess,TRUE)
```

method	RMSE
Model 1 : Regression	0.0002670
Model 2 : GLM	0.0002670
Model 3 : Loess	0.4973461
Model 4 : knn	0.1986335
Model 5 : Tree	0.4708843

Model 6.1 : Random Forest - RF	0.1995343
Model 6.2 : Random Forest - Rborist	0.1618590
Model 7 : GamLoess	0.1973784

```
        # Following code will generate Predicted vs True ratings graph with r
egression line

            displaySmoothPlot(happiness_gamloess_pred)

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Multiple Models at a time:

We can evaluate multiple models at a time using train function of caret package:

```
        # We can predict values for multiple models all at once.  Following i
s the code that will do this

        # Following is the list of models
```

```r
            models <- c("svmLinear",
                        "gamboost", "kknn", "gam",
                        "ranger", "avNNet", "mlp", "monmlp", "gbm",
                        "svmRadial", "svmRadialCost", "svmRadialSigma")

        # Following code will apply each model to get happiness_models_hat

            happiness_models_hat <- lapply(models, function(model){
              train(HappinessScore ~ . , data = train_set, method = model)
            })




            names(happiness_models_hat) <- models

        # Now we will predict Happiness Score for test set

            happiness_models_pred <- sapply(happiness_models_hat, function(ob
ject)
              predict(object, newdata = test_set))

        # Dimentions of happiness_models_pred. 76 rows and 17 columns(models)

            dim(happiness_models_pred)

## [1] 76 12

        # Following code will calculate RMSE value of each model and will sto
re it in RMSE_Models

            RMSE_Models <- sapply(models, function(model){
              print(model)
              return(RMSE(test_set$HappinessScore,happiness_models_pred[,mode
l]))
            })

## [1] "svmLinear"
## [1] "gamboost"
## [1] "kknn"
## [1] "gam"
## [1] "ranger"
## [1] "avNNet"
## [1] "mlp"
## [1] "monmlp"
## [1] "gbm"
## [1] "svmRadial"
## [1] "svmRadialCost"
## [1] "svmRadialSigma"
```

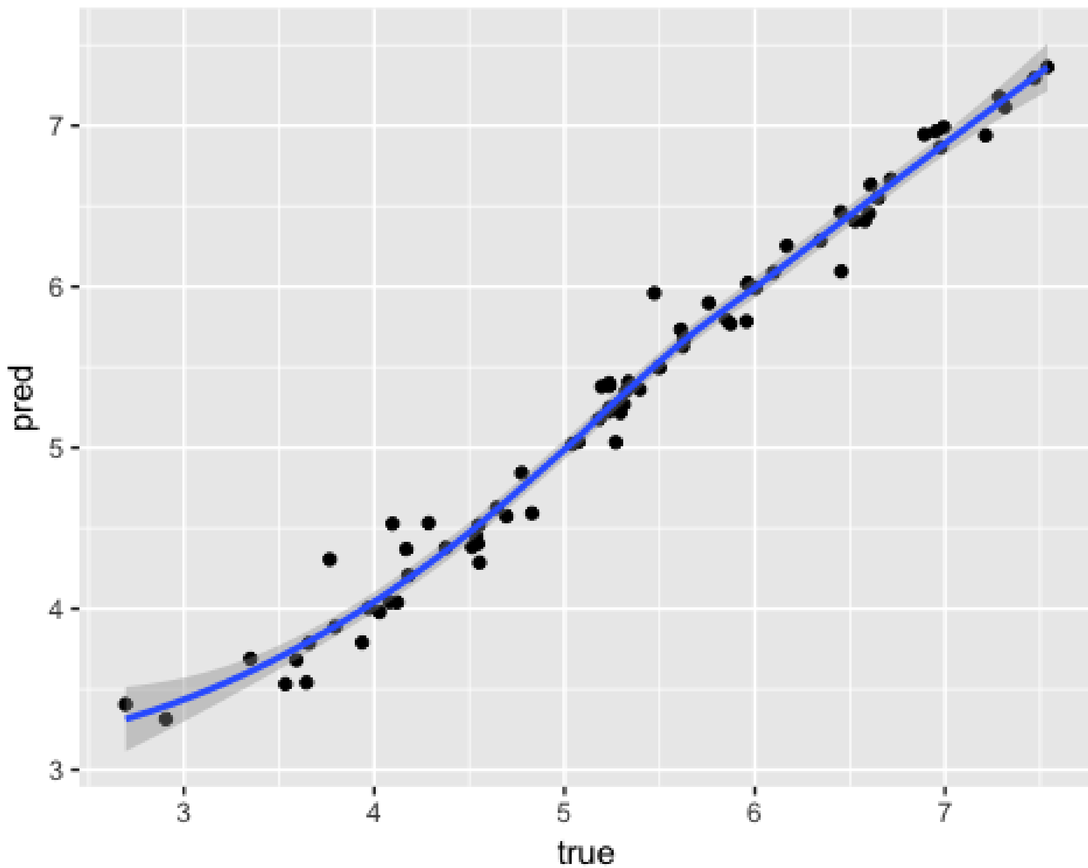
```
# Add results into rmse_result

    for (i in 1:length(models)) {

      model_name <- paste('Model ',models[i])

      add_RMSE_result(model_name,RMSE_Models[models[i]],FALSE)
    }

# rmse_result
rmse_result %>% knitr::kable()
```

method	RMSE
Model 1 : Regression	0.0002670
Model 2 : GLM	0.0002670
Model 3 : Loess	0.4973461
Model 4 : knn	0.1986335
Model 5 : Tree	0.4708843
Model 6.1 : Random Forest - RF	0.1995343
Model 6.2 : Random Forest - Rborist	0.1618590
Model 7 : GamLoess	0.1973784
Model svmLinear	0.0665486
Model gamboost	0.0453855
Model kknn	0.2821341
Model gam	0.0002092
Model ranger	0.1901763
Model avNNet	4.4548272
Model mlp	0.1819612
Model monmlp	0.0024610
Model gbm	0.2267900
Model svmRadial	0.1408112
Model svmRadialCost	0.1449640
Model svmRadialSigma	0.1090391

Ensemble:

Ensemble is used to combine 2 or more models to get better prediction.
There are 3 main types of ensemble: Averaging, Majority Vote and Weighted Average.
We will use Averaging to improve our predictions using 2 models.

```
    model_name <- 'Model 8 : Ensemble'

        # Note that we are using predictions from "Multiple Models at a time"
section.  Please execute that section 1st before executing following code.

        # Ensemble is used to combine 2 or more models to get better predicti
on
        # There are 3 main types of ensemble: Averaging, Majority Vote and We
ighted Average

        # We will use Averaging to see if there are any improvements in our p
rediction

        # Following code will predict Happiness Rating using averaging for 'g
bm' and 'svmRadial' models

            ensemble_models <- c('gbm','ranger')

            happiness_ensemble_pred <- rowMeans(happiness_models_pred[,c(ense
mble_models)])

        # Following code displays individual results of

            # We can see the RMSE reduced after applying Ensemble Averaging

                print(paste('gbm RMSE: ',RMSE(test_set$HappinessScore,happine
ss_models_pred[,'gbm'])))
                print(paste('ranger RMSE: ',RMSE(test_set$HappinessScore,happ
iness_models_pred[,'ranger'])))

                happiness_RMSE_ensemble <- RMSE(test_set$HappinessScore,happi
ness_ensemble_pred)
                print(paste('Ensemble RMSE: ',happiness_RMSE_ensemble))

        # Add results into rmse_result

            add_RMSE_result(model_name,happiness_RMSE_ensemble,TRUE)

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

        # Following code will generate Predicted vs True ratings graph with r
egression line

            displaySmoothPlot(happiness_ensemble_pred)

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



RESULTS:

RMSE values for all the models are displayed in ascending order below:

```
        rmse_result %>% arrange(RMSE)

##                                 method         RMSE
## 1                            Model  gam 0.0002092151
## 2                 Model 1 : Regression 0.0002669756
## 3                         Model 2 : GLM 0.0002669756
## 4                        Model  monmlp 0.0024609986
## 5                      Model  gamboost 0.0453855143
## 6                     Model  svmLinear 0.0665485723
## 7                Model  svmRadialSigma 0.1090390707
## 8                     Model  svmRadial 0.1408111868
## 9                 Model  svmRadialCost 0.1449640021
## 10 Model 6.2 : Random Forest - Rborist 0.1618589786
## 11                           Model  mlp 0.1819612319
## 12                  Model 8 : Ensemble 0.1827372151
## 13                       Model  ranger 0.1901762958
```

```
## 14                 Model 7 : GamLoess 0.1973783612
## 15                      Model 4 : knn 0.1986334791
## 16      Model 6.1 : Random Forest - RF 0.1995342584
## 17                         Model  gbm 0.2267899909
## 18                        Model  kknn 0.2821340920
## 19                     Model 5 : Tree 0.4708843466
## 20                    Model 3 : Loess 0.4973460826
## 21                      Model  avNNet 4.4548272355
```

“gam”, “Regression” and “GLM” are the top 3 models that predict Happiness Score accurately.

CONCLUSION:

We have used 21 models to predict Happiness Score and following are the top 3 models based on our Results:

1) “gam” model (Generalized Additive Model)
2) “lm” model (Linear Model)
3) “glm” model (Generalized Linear Model)

Above results show that liner models are better fit for Happiness data to predict Happiness Score of countries.