Prediction of Life Expectancy using Regression

David

2024-09-19

The main objective of this analysis is to build a Regression model that can accurately predict Life Expectancy for any country in Africa. I'm mainly focusing on Africa because Africa has the highest number of least developed countries. These are low-income countries which are highly vulnerable to economic shocks and have low levels of human assets (UN, 2023). These countries are mainly characterized by slow economic growth and high population growth, which has led to the rising number of people living in extreme poverty.

```
# Load packages
suppressMessages(
  {library(tidyverse)
    library(caret)
    library(janitor)
    library(mlr)
    library(psych)
    library(parallel)
    library(parallelMap)
  }
)

# Import data
Life_Expectancy <- read.csv("Life_Expectancy.csv")

# View the structure of the dataset
Life_Expectancy |> glimpse()

## Rows: 1,904
## Columns: 17
## $ Country                                          <chr> "Albania",
"Algeri…
## $ Year                                             <int> 2000, 2000,
2000, …
## $ Continent                                        <chr> "Europe",
"Africa"…
## $ Least.Developed                                  <lgl> FALSE, FALSE,
TRUE…
## $ Life.Expectancy                                  <dbl> 73.95500,
70.64000…
## $ Population                                       <int> 3089027,
31042238,…
## $ CO2.emissions                                    <dbl> 1.02621311,
2.5787…
```

```
## $ Health.expenditure                                    <dbl> 7.233370, 
3.489033…
## $ Electric.power.consumption                            <dbl> 1414.70378, 
652.80…
## $ Forest.area                                           <dbl> 28.0766423, 
0.6629…
## $ GDP.per.capita                                        <dbl> 3860.8046, 
8436.99…
## $ Individuals.using.the.Internet                        <dbl> 0.114097347, 
0.491…
## $ Military.expenditure                                  <dbl> 1.2463602, 
3.43338…
## $ People.practicing.open.defecation                     <dbl> 0.88885278, 
6.2339…
## $ People.using.at.least.basic.drinking.water.services <dbl> 86.75447, 
89.84883…
## $ Obesity.among.adults                                  <dbl> 12.8, 15.0, 
3.0, 2…
## $ Beer.consumption.per.capita                           <dbl> 1.33431, 
0.11978, …
```

The data has 1904 observations of 17 variables. Country and Continent are character variables, least developed is a logical variable while the rest of the variables are numeric.

Clean the data

```
# Clean variable names
Life_Expectancy <- clean_names(Life_Expectancy)

# Rename variables with longer variable names to have shorter names
Life_Expectancy <- Life_Expectancy |> rename(open_defecation = 
"people_practicing_open_defecation",
                          basic_drinking_water = 
"people_using_at_least_basic_drinking_water_services",
                          adults_obesity = "obesity_among_adults",
                          beer_consumption = "beer_consumption_per_capita",
                          internet = "individuals_using_the_internet")

# Check for missing values
map_dbl(Life_Expectancy, ~length(which(is.na(.))))

##                    country                       year 
##                          0                          0 
##                  continent            least_developed 
##                          0                          0 
##            life_expectancy                 population 
##                          0                          0 
##              co2_emissions         health_expenditure 
##                          0                          0 
## electric_power_consumption                forest_area
```

```
##                         0                          0
##            gdp_per_capita                   internet
##                         0                          0
##      military_expenditure            open_defecation
##                         0                          0
##      basic_drinking_water             adults_obesity
##                         0                          0
##          beer_consumption
##                         0
```

There are no missing values in the data.

```r
# Check for duplicated observations
sum(duplicated(Life_Expectancy))

## [1] 0
```

There are no duplicated observations in the data as well.

Exploratory Data Analysis

I'm more interested in Africa, so I will filter the data for Africa. I'd like to obtain the average Life Expectancy in Africa between the period 2000 to 2015, average GDP per Capita, Health Expenditure, the population with access to clean water services and the population which is still practicing open defecation.

```r
# Filter the data for Africa
Africa_Data <- Life_Expectancy |> filter( continent == "Africa")
```

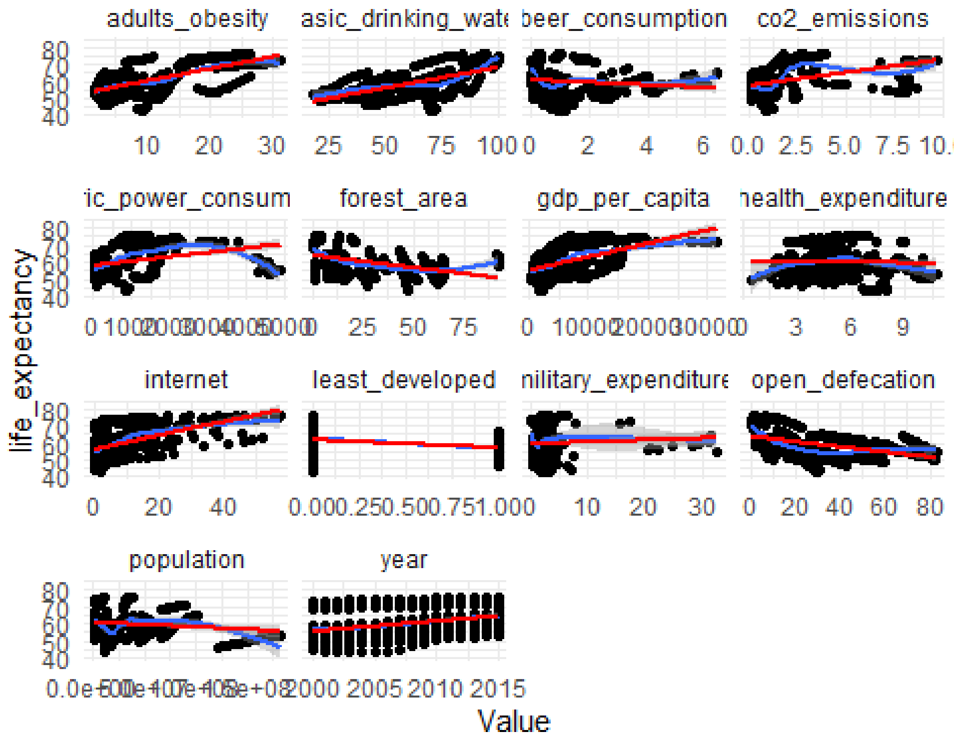
The data for Africa has 448 observations.

```r
# Calculate summary statistics
Africa_Data |> select(-c(country, continent)) |> summary()

##       year      least_developed life_expectancy   population
##  Min.   :2000   Mode :logical   Min.   :43.06   Min.   :  1186873
##  1st Qu.:2004   FALSE:256       1st Qu.:53.49   1st Qu.:  6037752
##  Median :2008   TRUE :192       Median :58.43   Median : 18429400
##  Mean   :2008                   Mean   :59.72   Mean   : 28169437
##  3rd Qu.:2011                   3rd Qu.:64.22   3rd Qu.: 36229780
##  Max.   :2015                   Max.   :76.09   Max.   :181137454
##  co2_emissions     health_expenditure electric_power_consumption
##  Min.   :0.03242   Min.   : 0.410     Min.   :  22.76
##  1st Qu.:0.25974   1st Qu.: 3.561     1st Qu.: 102.25
##  Median :0.57964   Median : 4.466     Median : 215.42
##  Mean   :1.54075   Mean   : 4.796     Mean   : 727.14
##  3rd Qu.:2.17482   3rd Qu.: 5.672     3rd Qu.:1019.67
##  Max.   :9.60845   Max.   :10.716     Max.   :4851.69
##   forest_area       gdp_per_capita       internet
military_expenditure
```

```
##  Min.   : 0.0521   Min.   :  434.8   Min.   : 0.00101   Min.   : 0.1427
##  1st Qu.: 9.0794   1st Qu.: 1828.3   1st Qu.: 0.96989   1st Qu.: 1.0555
##  Median :21.5803   Median : 3428.9   Median : 3.97500   Median : 1.4898
##  Mean   :28.1303   Mean   : 5826.6   Mean   : 8.60403   Mean   : 2.8765
##  3rd Qu.:45.7896   3rd Qu.: 8804.6   3rd Qu.:11.52500   3rd Qu.: 2.7355
##  Max.   :91.9781   Max.   :31702.9   Max.   :57.08000   Max.   :32.6557
##  open_defecation    basic_drinking_water adults_obesity    beer_consumption
##  Min.   : 0.08106   Min.   :18.09        Min.   : 1.600   Min.   :0.0000
##  1st Qu.: 7.47031   1st Qu.:49.20        1st Qu.: 4.300   1st Qu.:0.2396
##  Median :19.08088   Median :64.81        Median : 6.400   Median :0.6300
##  Mean   :25.66821   Mean   :65.63        Mean   : 9.582   Mean   :1.1585
##  3rd Qu.:41.58956   3rd Qu.:82.50        3rd Qu.:12.550   3rd Qu.:1.4234
##  Max.   :82.72894   Max.   :99.87        Max.   :31.000   Max.   :6.4100
```

- The average Life Expectancy in Africa between 2000 to 2015 was 59.79 years (SD = 8.12). The average population was 28,169,437.
- The average GDP Per Capita for Africa (the average income per person) between 2000 to 2015 was about $5,826.58, the average Health Expenditure was 4.796% of GDP, while the average military expenditure was 2.88% of GDP.
- On average, 65.63% of the population have access to basic drinking water services, but 25.67% of the population still practice open defecation. This is still a major challenge for Africa.
- Also on average, 8.6% of the population have access to internet connection, and 9.58% of the population are obese.

```r
# Convert the data to long format for plotting
UntidyData <- gather(Africa_Data, key = "Variable", value = "Value",
                     -c(country, continent, life_expectancy))

# Plot
ggplot(UntidyData, aes(Value, life_expectancy)) +
  facet_wrap(~Variable, scale = "free_x")+
  geom_point() +
  geom_smooth() +
  geom_smooth(method = "lm", col ="red") +
  theme_minimal()

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



Most of the predictor variables have non-linear relationship with the outcome variable. Health expenditure and military expenditure don't seem to have a relationship with Life Expectancy.

Model training

```
# Select the variables to be used for model training (omit country &
continent)
data <- Africa_Data |> select(-c(country, continent))

## Partition the data into training and test sets

# Set seed for reproducibility
set.seed(42)
# Partition the data
splitIndex <- createDataPartition(data$life_expectancy, p = 0.80, list =
FALSE)
# Assign 80% to training set
training_data <- data[splitIndex, ]
# Assign test set the remaining 20%
test_data <- data[-splitIndex, ]

# Convert all the features in training data to numeric
training_data <- mutate_all(training_data, .funs = ~ as.numeric(.))
# Convert all the features in test data to numeric
test_data <- mutate_all(test_data, .funs = ~ as.numeric(.))
```

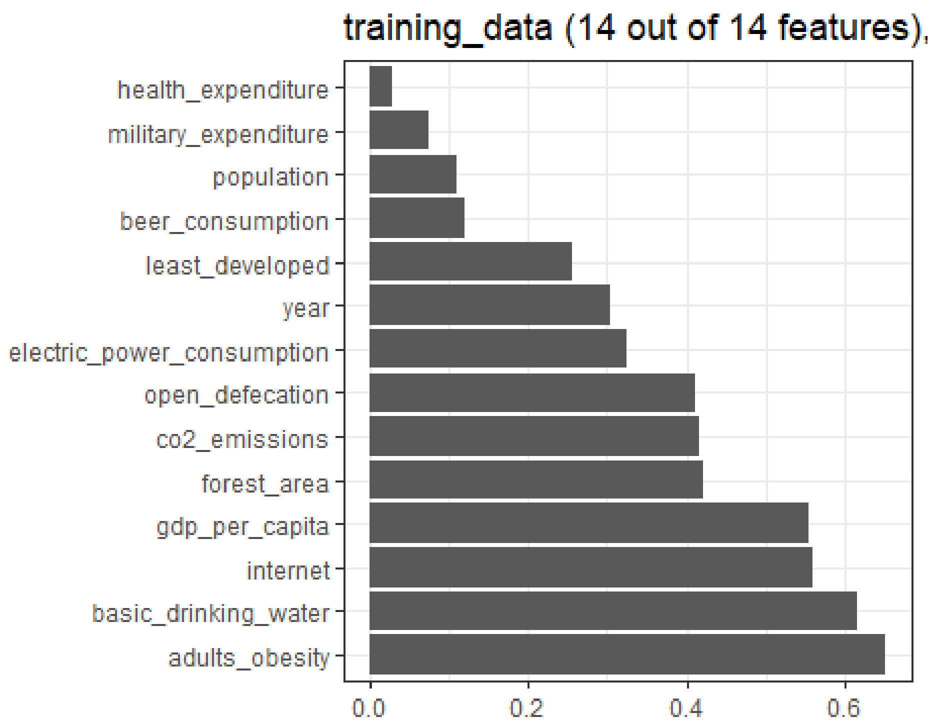
Linear Regression model

```
# Define regression task
reg_task <- makeRegrTask(data = training_data, target = "life_expectancy")
# Define learner
reg_learner <- makeLearner("regr.lm")
```

```
# Estimate variable importance using linear correlation
filterVals <- generateFilterValuesData( reg_task, method =
"linear.correlation")

# Get the correlation coefficients
filterVals$data

##                           name    type             filter      value
##                         <char>  <char>             <fctr>      <num>
##  1:                       year numeric linear.correlation 0.30431729
##  2:             least_developed numeric linear.correlation 0.25650947
##  3:                  population numeric linear.correlation 0.10987719
##  4:               co2_emissions numeric linear.correlation 0.41608276
##  5:          health_expenditure numeric linear.correlation 0.02896530
##  6: electric_power_consumption numeric linear.correlation 0.32426011
##  7:                 forest_area numeric linear.correlation 0.42050964
##  8:              gdp_per_capita numeric linear.correlation 0.55367524
##  9:                    internet numeric linear.correlation 0.55811083
## 10:        military_expenditure numeric linear.correlation 0.07339473
## 11:              open_defecation numeric linear.correlation 0.41195536
## 12:         basic_drinking_water numeric linear.correlation 0.61581017
## 13:              adults_obesity numeric linear.correlation 0.65227749
## 14:            beer_consumption numeric linear.correlation 0.12078963
```

Most of the predictors are moderately correlated with the outcome variable. Population, Health Expenditure, Military Expenditure and beer consumption are weakly correlated with life expectancy. None of the predictors is highly correlated with the outcome variable.

```
# Plot the feature importance
plotFilterValues(filterVals) + theme_bw() + coord_flip()
```



Despite Health Expenditure being a very important determinant of life expectancy, it contributes very little information to the outcome variable. This means that Health is underfunded in Africa and this remains a major concern. Adults obesity, basic drinking water services, access to internet and GDP per Capita, respectively, contribute the highest information to the outcome variable. Open defecation, CO2 emissions and forest area cover contribute nearly the same amount of information to the outcome variable.

From my EDA, most of the predictor variables didn't have a linear relationship with the outcome Life Expectancy, but I'll use all the variables for modelling because they might be of importance to non-linear models.

```
# Fit the Linear Regression model
Linear_model <- train(reg_learner, reg_task)
# Obtain model results
results <- getLearnerModel(Linear_model)

# Have a model summary
summary(results)

##
## Call:
## stats::lm(formula = f, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7832 -2.2830  0.1162  2.0636  9.1261
##
```

```
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                 -1.367e+02  1.124e+02  -1.216   0.2247    
## year                         9.699e-02  5.624e-02   1.725   0.0855 .  
## least_developed              6.990e+00  7.478e-01   9.348  < 2e-16 ***
## population                  -3.582e-08  7.321e-09  -4.892 1.53e-06 ***
## co2_emissions               -2.153e-01  3.434e-01  -0.627   0.5311    
## health_expenditure          -6.609e-01  1.565e-01  -4.222 3.10e-05 ***
## electric_power_consumption -2.683e-03  5.813e-04  -4.615 5.55e-06 ***
## forest_area                 -1.735e-01  1.558e-02 -11.134  < 2e-16 ***
## gdp_per_capita               3.973e-04  8.927e-05   4.451 1.16e-05 ***
## internet                     1.642e-01  2.735e-02   6.005 4.86e-09 ***
## military_expenditure         1.488e-01  3.957e-02   3.759   0.0002 ***
## open_defecation             -1.440e-01  1.516e-02  -9.496  < 2e-16 ***
## basic_drinking_water         1.022e-01  2.221e-02   4.599 5.96e-06 ***
## adults_obesity               3.137e-01  6.339e-02   4.949 1.17e-06 ***
## beer_consumption            -6.205e-02  2.542e-01  -0.244   0.8073    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 3.55 on 345 degrees of freedom
## Multiple R-squared:  0.8142, Adjusted R-squared:  0.8067 
## F-statistic:   108 on 14 and 345 DF,  p-value: < 2.2e-16
```

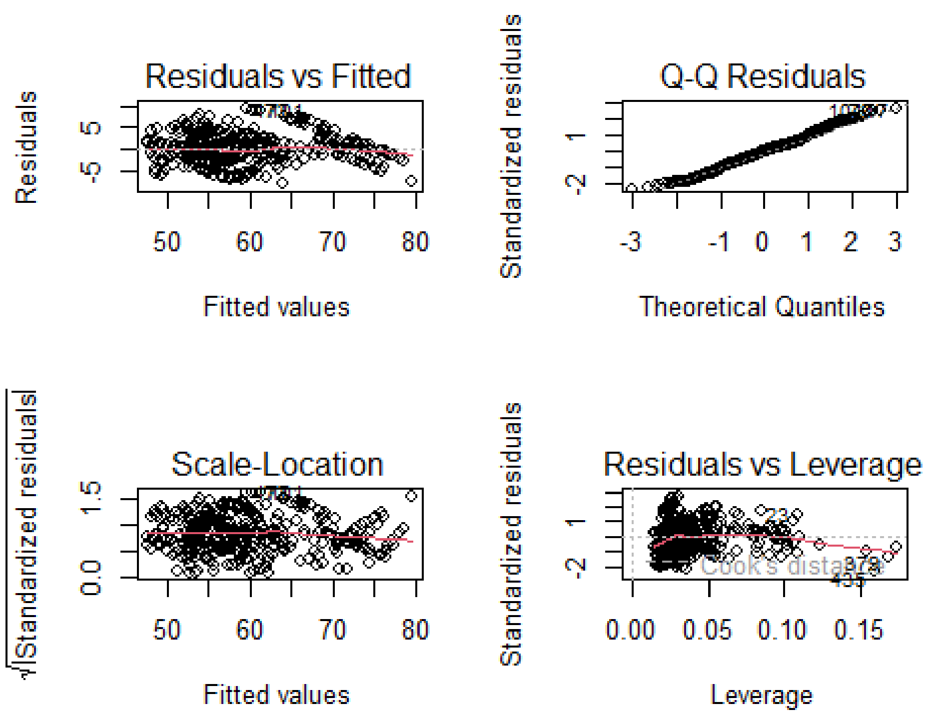
The overall model is significant (p-value < 0.01). The adjusted R-squared is also high (0.8067), implying that the predictors explain 80.67% of the variance in the dependent/outcome variable (Life Expectancy). Year, CO2 emissions and Beer consumption are insignificant predictors of Life Expectancy.

- The intercept is not meaningful in this case.
- On average, life expectancy in countries that aren't least developed is expected to be 6.99 years higher, as compared to life expectancy in countries that are least developed, keeping all other variables constant.
- For every individual increase in population, life expectancy is expected to decrease by (3.582 ^ (10*-8)), keeping all other variables constant.
- For every unit increase in CO2 emissions, life expectancy is expected to decrease by 0.2153 years, keeping all other variables constant. Even though this isn't significant.
- For every 1% increase in health expenditure as a percentage of GDP, life expectancy is expected to decrease by 0.66 years, keeping all other variables constant. This is very opposite with the obvious expectation that increasing health expenditure would increase life expectancy.
- For every unit increase in electric power consumption, life expectancy is expected to decrease by 0.0027 years, keeping all other variables constant.
- For every square unit increase in forest area, life expectancy is expected to decrease by 0.174 years, keeping all other variables constant. This is also opposite with my expectation that, an increase in forest area would improve life expectancy due to cleaner air, biodiversity, and general health benefits linked to green spaces.

- For every one-dollar increase in GDP per ca-pita, life expectancy is expected to increase by 0.000397 years, keeping all other variables constant.
- For every 1% increase in population with access to the internet, life expectancy is expected to increase by 0.16 years, keeping all other variables constant.
- For every 1% increase in military expenditure as a percentage of GDP, life expectancy is expected to increase by 0.14 years, keeping all other variables constant. This matches the expectation that increasing military expenditure would increase the safety levels of a country, mainly from external attacks like terrorism.
- For every 1% increase in population practicing open defecation, life expectancy is expected to decrease by 0.144 years, keeping all other variables constant.
- For every 1% increase in population with access to basic drinking water services, life expectancy is expected to increase by 0.102 years, keeping all other variables constant.
- For every 1% increase in adult population with obesity, life expectancy is expected to increase by 0.31, keeping all other variables constant. This is also opposite with what I expected.

Linear Model Diagnostics

```r
# Plot the model results
par(mfrow = c(2, 2))
plot(results)
```



- There are no patterns in the residuals vs fitted plot, implying that there's a linear relationship between the dependent and the predictor variables.

- The normal Q-Q plot closely resembles a straight line along the diagonal, with little discrepancies on the tails. This implies that the residuals are normally distributed.
- There's no pattern in the Scale-Location plot, implying that there's no heteroscedasticity of the residuals.

Nearly all the assumptions for the Linear model are met.

```
# Cross-validate the linear model to see how it generalizes

# Wrap learner with feature preprocessing
lm_scaled <- makePreprocWrapperCaret(learner = reg_learner, ppc.scale = TRUE,
                                     ppc.center = TRUE)

# Make resampling description
kFold <-makeResampleDesc(method = "RepCV", folds = 7, reps = 30)

# Cross-validate
lmCV <-resample(lm_scaled, reg_task, resampling = kFold,
                measures = list(rmse, rsq),
                show.info = FALSE)

# View CV results
lmCV

## Resample Result
## Task: training_data
## Learner: regr.lm.preproc
## Aggr perf: rmse.test.rmse=3.6336046,rsq.test.mean=0.7899168
## Runtime: 5.48988
```

An RMSE value of 3.63 is a bit high, the model doesn't perform very well.

Non-linear Regression Models

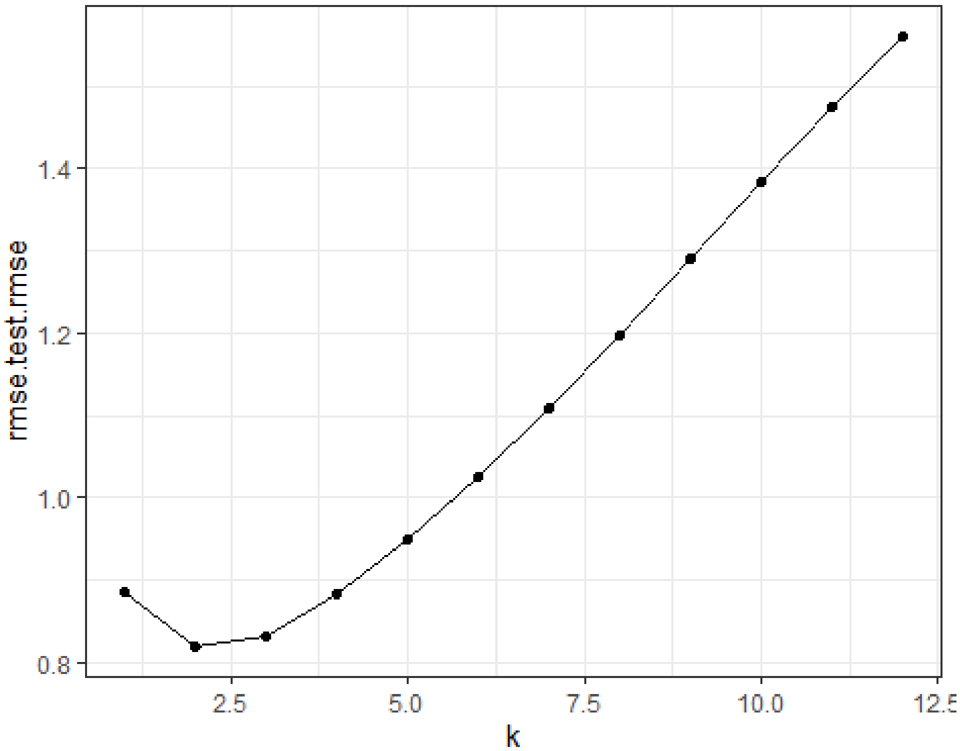
KNN

```
# Define KNN Learner
kknn <- makeLearner("regr.kknn")

## Loading required package: kknn

##
## Attaching package: 'kknn'

## The following object is masked from 'package:caret':
##
##     contr.dummy

# Wrap learner with feature preprocessing
kknn_normalized <- makePreprocWrapperCaret(learner = kknn,
```

```r
                                              ppc.scale = TRUE,
                                              ppc.center = TRUE)

# Define hyperparameter space for tuning k
kknnParamSpace <- makeParamSet(makeDiscreteParam("k", values = 1:12))
# Specify search strategy
gridSearch <- makeTuneControlGrid()
# Tune the model
tunedK <- tuneParams(kknn_normalized, task = reg_task,
                     resampling = kFold,
                     par.set = kknnParamSpace,
                     control = gridSearch,
                     measures = list(rmse, rsq),
                     show.info = FALSE)

# View tuning results
tunedK

## Tune result:
## Op. pars: k=2
## rmse.test.rmse=0.8194258,rsq.test.mean=0.9891001
```

The optimal value of k is 2. RMSE is lower compared to that of the Linear model. R-squared is also closer to 1, which is good. It is however important to note that KNN can overfit at lower values of k.

```r
# Extract model information
knnTuningData <- generateHyperParsEffectData(tunedK)
# Visualize the hyperparameter tuning process
plotHyperParsEffect(knnTuningData, x = "k", y = "rmse.test.rmse",
                    plot.type = "line") + theme_bw()
```



RMSE is lowest at k = 2.

```
# Set the optimal value of k for the final model
tunedKnn <- setHyperPars(kknn, par.vals = tunedK$x)
# Train the final model
tunedKnnModel <- train(tunedKnn, reg_task)
```

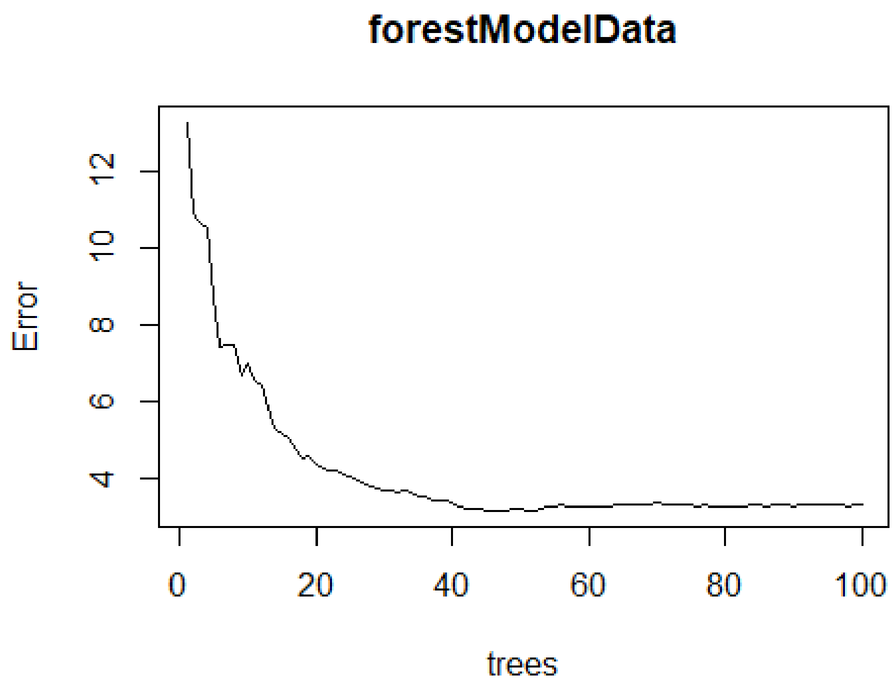
Random Forest

```
# Define learner
rf_learner <- makeLearner("regr.randomForest")

# Define hyperparameter space for tuning the model
forestParamSpace <- makeParamSet(
 makeIntegerParam("ntree", lower = 100, upper = 100),
 makeIntegerParam("mtry", lower = 4, upper = 25),
 makeIntegerParam("nodesize", lower = 1, upper = 30),
 makeIntegerParam("maxnodes", lower = 5, upper = 25))

# Make resampling description
kFold <- makeResampleDesc(method = "RepCV", folds = 7, reps = 20)

# Specify search strategy
randSearch <- makeTuneControlRandom(maxit = 50)

# Begin parallelization (Parallel processing speeds up the hyperparameter
tuning process)
parallelStartSocket(cpus = detectCores())
```

```
## Starting parallelization in mode=socket with cpus=4.

# Perform hyperparameter tuning with cross-validation
tunedForestPars <- tuneParams(rf_learner, task = reg_task, resampling =
kFold,
                              par.set = forestParamSpace, control =
randSearch,
                              measures = list(rmse, rsq),
                              show.info = FALSE)

## Exporting objects to slaves for mode socket: .mlr.slave.options

## Mapping in parallel: mode = socket; level = mlr.tuneParams; cpus = 4;
elements = 50.

# Stop parallelization
parallelStop()

## Stopped parallelization. All cleaned up.

# View tuning results
tunedForestPars

## Tune result:
## Op. pars: ntree=100; mtry=20; nodesize=21; maxnodes=25
## rmse.test.rmse=1.9145249,rsq.test.mean=0.9417979
```

The RF model outperforms the Linear model but is outperformed by KNN. It has a lower RMSE value of 1.91, even though not very closer to zero.

```
# Set the optimal hyperparameters for the final model
tunedForest <- setHyperPars(rf_learner, par.vals = tunedForestPars$x)

# Train the final model using the optimal hyperparameters
tunedForestModel <- train(tunedForest, reg_task)

## Warning in randomForest.default(x = data[["data"]], y = data[["target"]],
:
## invalid mtry: reset to within valid range

# Extract model information
forestModelData <- getLearnerModel(tunedForestModel)

# Check if there are enough trees in the forest
plot(forestModelData)
```



The out-of-bag error stabilizes after about 75 bagged trees, implying that I have enough trees in the forest.

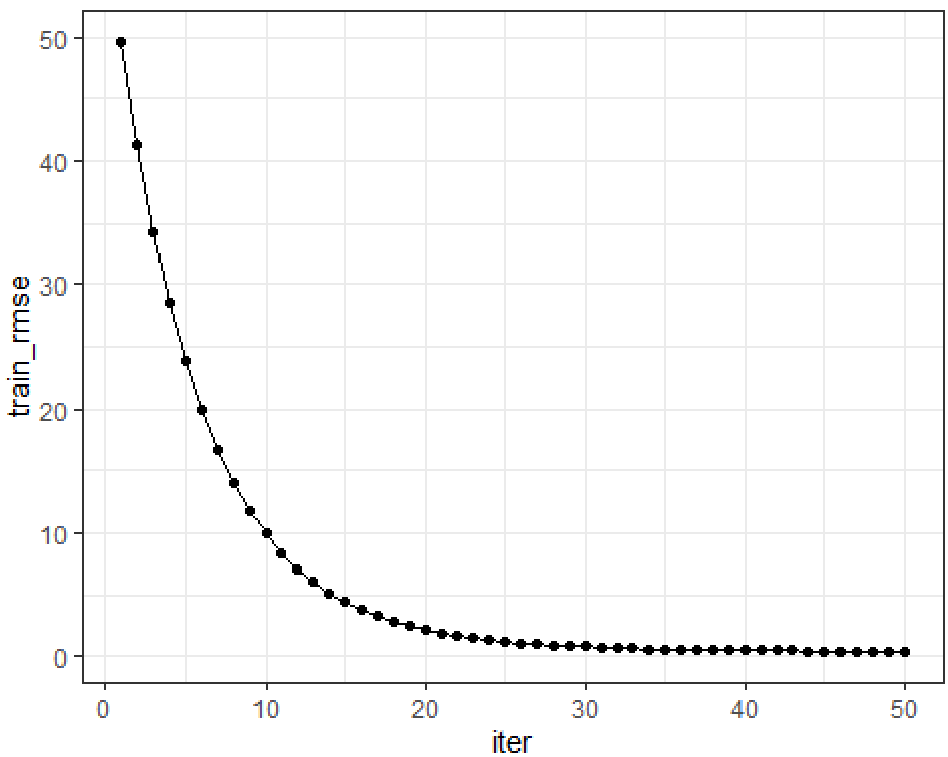
XGBoost

```
# Define learner
xgb <-makeLearner("regr.xgboost")

# Define hyperparameter space for tuning
xgbParamSpace <-makeParamSet(
  makeNumericParam("eta", lower = 0, upper = 1),
  makeNumericParam("gamma", lower = 0, upper = 5),
  makeIntegerParam("max_depth", lower = 1, upper = 20),
  makeNumericParam("min_child_weight", lower = 1, upper = 10),
  makeNumericParam("subsample", lower = 0.5, upper = 1),
  makeNumericParam("colsample_bytree", lower = 0.5, upper = 1),
  makeIntegerParam("nrounds", lower = 50, upper = 50))

# Perform hyperparameter tuning with cross-validation
tunedXgbPars <- tuneParams(xgb, task = reg_task,resampling = kFold,
                           par.set = xgbParamSpace, control = randSearch,
                           measures = list(rmse, rsq),
                           show.info = FALSE)

# View tuning results
tunedXgbPars
```

```
## Tune result:
## Op. pars: eta=0.173; gamma=0.356; max_depth=12; min_child_weight=4.6;
subsample=0.505; colsample_bytree=0.506; nrounds=50
## rmse.test.rmse=1.3078593,rsq.test.mean=0.9725991
```

An RMSE value of 1.31 is low and is good (even though not very closer to zero). The XGBoost algorithm outperforms Random Forest and the Linear Regression, but is outperformed by KNN.

```
# Set the optimal hyperparameters for the final model
tunedXgb <- setHyperPars(xgb, par.vals = tunedXgbPars$x)

# Train the final model using optimal hyperparameters
tunedXgbModel <- train(tunedXgb, reg_task)

# Extract model information
xgbModelData <- getLearnerModel(tunedXgbModel)

# Plot the model data to check if there are enough trees in the ensemble
ggplot(xgbModelData$evaluation_log, aes(iter, train_rmse)) +
  geom_line() + geom_point() + theme_bw()
```



The curve flattens after the 30th iteration and increasing the number of iterations would not have an effect in the model. This implies that there are enough trees in the ensemble.

Benchmark the KNN, Random Forest and XGBoost model-building processes

```
# Create a tuning wrapper for KNN
kknnWrapper <- makeTuneWrapper(kknn_normalized, resampling = kFold,
                               par.set = kknnParamSpace,
                               control = gridSearch,
                               measures = list(rmse, rsq))

# Create a tuning wrapper for RF
forestWrapper <- makeTuneWrapper(rf_learner, resampling = kFold,
                                 par.set = forestParamSpace,
                                 control = randSearch,
                                 measures = list(rmse, rsq))

# Create a tuning wrapper for XGB
xgbWrapper <- makeTuneWrapper(xgb, resampling = kFold,
                              par.set = xgbParamSpace,
                              control = randSearch,
                              measures = list(rmse, rsq))

# Create a list of learners
learners = list(kknnWrapper, forestWrapper, xgbWrapper)

# Use holdout cross validation for the benchmarking process
holdout <- makeResampleDesc("Holdout")

# Benchmark
bench <- benchmark(learners, reg_task, holdout, show.info = FALSE)

# View the benchmarking results
bench

##         task.id              learner.id mse.test.mean
## 1 training_data regr.kknn.preproc.tuned      1.056691
## 2 training_data regr.randomForest.tuned      4.790287
## 3 training_data      regr.xgboost.tuned      1.900932
```

According to this benchmarking results, KNN is likely to give me the best-performing model, with a mean prediction error of 1.06

Model Validation

I will use all the four models that I trained to make predictions on test data and assess how they would generalize on new, unseen data. I'll use RMSE as my performance metric.

```
# Make predictions on test data using the Linear model
lmPreds <- predict(Linear_model, newdata = test_data)$data
# Make predictions using KNN model
```

```r
knnPreds <- predict(tunedKnnModel, newdata = test_data)$data
# Make predictions using RF model
rfPreds <- predict(tunedForestModel,newdata = test_data)$data
# Make predictions using XGBoost model
xgbPreds <- predict(tunedXgbModel, newdata = test_data)$data

# Calculate test RMSE for each and every model
lm_RMSE <- mean((test_data$life_expectancy - lmPreds$response)^2) |> sqrt()
lm_RMSE

## [1] 3.417389

knn_RMSE <- mean((test_data$life_expectancy - knnPreds$response)^2) |> sqrt()
knn_RMSE

## [1] 0.6378754

rf_RMSE <- mean((test_data$life_expectancy - rfPreds$response)^2) |> sqrt()
rf_RMSE

## [1] 1.873877

xgb_RMSE <- mean((test_data$life_expectancy - xgbPreds$response)^2) |> sqrt()
xgb_RMSE

## [1] 1.194205
```

KNN outperforms all the other algorithms. A test RMSE of 0.638 is low and is good, even though not very closer to zero. On average, the KNN predictions are off by approximately 0.64 years. In other words, I would expect my predictions to be within (plus or minus) 0.64 years of true Life Expectancy.