6372: Project 1

Megan Ball, Matt Farrow, Neddy Nyatome

# Introduction

Using the World Health Organization (WHO) data compiled by Kumar Rajarshi, Deeksha Russell, and Duan Wang, we developed three different models:

* The first model was designed to be easily interpreted using linear regression.
* The second model was designed to be used as a predictive tool using linear regression.
* The third model was developed using non-parametric methods for prediction.

# Data Description

The description and context of the Life Expectancy (WHO) data set can be found [here](https://www.kaggle.com/kumarajarshi/life-expectancy-who). Data has been compiled from several different data sets into a final data set that represents health factors for 193 countries between the years of 2000-2015.

Looking at the data, there are 2,938 observations and 22 variables that cover four broad factors: immunization-related, mortality, economic, and social. various social, economic, and health-related factors. Each record in the data contains measurements for a single year within the country being measured.

# Exploratory Data Analysis

We began by plotting life expectancy into a histogram as well as a Q-Q plot (Figures 1 & 2).

# Chart, line chart Description automatically generated

Figure 1: Histogram of Life Expectancy data Figure 2: Q-Q Plot of Life Expectancy Data

As we would hope, life expectancy tends to skew towards the older side. The Q-Q plot shows some slight deviations from normality towards the edges, but after trying various transformations, the deviations from normality that are evident in the distribution did not seem severe enough to warrant a transformation and we proceeded using the original data.

Next, we began looking at correlation to narrow down our variable list before examining specific relationships (Figure 3). Based on a cut-off of > 0.9 for correlation, we removed the variable in each correlation pair that had the higher number of NA values (Figure 4). We then proceeded to look at what happens when we also remove population, since it has minimal correlation to life\_expectancy (Figure 5). In the end, we made the decision to remove under\_five\_deaths, gdp, thinness\_1\_19\_years, and population due to lack of correlation to the response variable or collinearity.

Our next task was to address the missing values in the data set (Figure 6). We then limited the scope of our analysis to not include those countries where life expectancy was missing (Figure 7). In doing that, we excluded the following countries from our scope: Cook Islands, Dominica, Marshall Islands, Monaco, Nauru, Niue, Palau, Saint Kitts and Nevis, San Marino, and Tuvalu.

Hepatitis B was now our variable with the most missing values. In looking at the relationship between Hepatitis B and Life Expectancy (Figure 8), our options with regards to the missing values were to drop them, impute them, or fill them in with 0’s. We chose different approaches based on each model.

### Interpretable Model

For our interpretable model, we made the decision to drop the hepatitis\_b variable along with the remainder of the NA’s (Figure 9). As a result of our feature engineering, we were left with only 2 records for 2015. After several looks at the data, we decided to only use the observations from the most recent four years (2011-2014).

### Linear Prediction Model

Knowing that we cannot have missing values for Ridge Regression or LASSO models, we examined the relationship of each variable that had more than 100 missing values to see which appeared to be significant.

* Hepatitis B (Figure 10)
* Total Expenditure (Figure 11)
* Alcohol (Figure 12)
* Income Composition of Resources (Figure 13)
* Schooling (Figure 14)

After reviewing the plots, we made the decision to remove Hepatitis B, total expenditure, and alcohol since the trend for those three variables was relatively flat. We then removed the remainder of the missing values from the data set before proceeding to modeling.

# Objective 1:

## Restatement of Problem

Determine the predictors that are significantly contributing to the value of life expectancy.

Model Selection

After the data was cleaned up, we went ahead and applied various model selection approaches which resulted in the creation of a data frame containing only the predictors and our response variable. We applied the following model selection algorithms:

### Interpretable Model Using Stepwise Regression

Stepwise regression in general performs multiple iterations by dropping one independent variable at the time. In each iteration, multiple models are created, and an AIC value is generated for each model. The model with the lowest AIC is retained for the next iteration. The iteration is stopped when there is no significant drop in AIC.

For the interpretable model, we applied forward and backward stepwise selection algorithms and they both returned the following variables: adult\_mortality, percentage\_expenditure, total\_expenditure, hiv\_aids, and income\_composition\_of\_resources. This means that the model with these variables had the lowest AIC.

**Stepwise Forward**

A regression model was built from the data set with predictor variables by adding predictors based on p values. At each step of the iteration, the predictors with significant p-values were added to the model. The final model included only the predictor variables: adult\_mortality, percentage\_expenditure, total\_expenditure, hiv\_aids, and income\_composition\_of\_resources.

**Stepwise Backward**

The difference between stepwise backward and stepwise forward is stepwise backward starts with all the variables in the data set and removes the variables based on the p-value. The variables are added until there are no variable to add any more. For our model, the predictor variables included were: adult\_mortality, percentage\_expenditure, total\_expenditure, hiv\_aids, and income\_composition\_of\_resources.

**Best Subsets**

We also used the Best subsets technique when we ran the stepwise regression to find and visualize our model. We visually inspected how the model was performing using metrics R-squared, adjusted R-squared, BIC and CP (Figure 15, Figure 16).

The best subsets technique visually shows that our variables (adult\_mortality, percentage\_expenditure, total\_expenditure, hiv\_aids, and income\_composition\_of\_resources) from the stepwise regression have high R-Squared, high Adjusted R-Squared, low BIC, and low CP.

### Additional Models

Since our linear model contained so many variables, we used the following models with the goal of reducing variance and ensuring that our least squares estimates are reliable when determining our best subset of predictors.

**Lasso**

After obtaining our optimal lambda, we checked for the non-zero coefficients in the model and made decisions about the variables to remove and keep before calculating for our MSE.

**Ridge Regression**

We used the Ridge regression to have our sum of squared coefficients penalized. As a result, it was revealed that we needed to add certain variables back into the model.

**Elastic NET**

We also ran Elastic NET model and compared it with Lasso and Ridge regressions.

### Cross-Validation

We applied cross-validation for all our models to ensure that the results from our train model are valid when the same model is applied to the test data.

To avoid overfitting our models, we used a cross validation technique to reserve a particular sample of dataset and made sure that we did not train the model. To measure the accuracy of our model, we used the MSE metric which is the average difference between the actual value and the predicted value of the life\_expectancy variable. This metric gave us the model’s average prediction error and our goal was to decrease this metric as much as possible to increase the accuracy of the model. We ended comparing the MSEs for our models to determine the best model.

## Checking Assumptions

Our models assume the following:

* **Independent Variables**: The value of the response variable is the result of linear combination of predictors.
* **Equal Variance**: Error variance is the same for all predictors.
* **Normal Distribution**: Distribution of errors have a normal distribution.

Looking at the fitted vs. residuals plot (Figure 19) we can see that:

* **Linearity**: At any fitted value in the plot, we can see that the mean of the residuals is roughly 0.
* **Equal Variance**: For each fitted value on the plot, we can see that the spread of the residuals is roughly the same making.
* **Normal Distribution**: Looking at the Q-Q plot, we see that the points are very close to the line. This means that errors are normally distributed. Also note that a few points are a bit far from the line but because we have such a large sample size we will not worry about these few points and will conclude that the data was sampled from a normal distribution.

## Interpretation of Regression Coefficients

We are 95% confident that the model’s intercept is between (45.41, 48.689) and the true regression coefficient’s for the predicted variables are: adult mortality (-0.016, -0.01), total expenditure (0.183, 0.359), HIV/AIDS (-1.062, -0.772), and income composition of resources (34.999, 38.894).

# Objective 2:

## Strategy

Once we had developed a linear regression model, we then looked at developing a non-parametric model for predicting the life expectancy. Although our ASE and adjusted R-squared metrics were very good for the linear regression models, there could be some question about the normality of our residuals, independence of the errors, and equal variance. By using a non-parametric model, it is no longer required that we meet those assumptions and we can proceed with as many variables as we think would be useful in our model.

As discussed in our EDA section, we used the imputed data set to run our non-parametric models, which already removed highly correlated variables and ones we suspected were not related to our response variable. For the KNN model, we used the caret package to iterate through several different K values to find the optimal K value using all of the variables we had left. We also ran a KNN model using the four variables we determined were significant from our interpretable model for comparison. We proceeded to do the same with a random forest model and our results will be discussed in the Metrics section below.

## Data Sets

There are limits to the levels of a factor that can be used in a KNN or tree model. For this reason, country was removed from the data sets used in our non-parametric models. Additionally, the model performance was tested on both an imputed data set and a data set with the NA values were removed instead of imputed.

## Metrics

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model Type | Data Set | # of Predictors | | Train ASE | Train R2 | Test ASE | Test R2 |
| KNN | Imputed | 16 | 7.437 | | 0.919 | 5.438 | 0.943 |
| KNN | Removed <NA> | 16 | 7.241 | | 0.920 | 6.175 | 0.930 |
| KNN | Imputed | 4 | 6.559 | | 0.927 | 5.302 | 0.943 |
| Tree | Imputed | 4\* | 13.7 | | - | 12.823 | - |
| Random Forest | Imputed | 16 | 3.606 | | 0.96 | 2.727 | 0.971 |
| Random Forest | Imputed | 4 | 5.855 | | 0.935 | 4.169 | 0.957 |

Table 1. Metrics for multiple non-parametric models. \*See appendix for variables used.

Based on the metrics, as seen above, the ASE and R-squared values from each model were surprisingly good. Since our R-squared was over 90% on our training data set, there was concern about over-fit, but we did not see a decrease in performance once applying to our test data set. The models were run again with the same parameters but with a different train and test split to perform an additional check against over-fit and performed similarly. This indicates that our model is not overfitting *specific to our current data set*. We could further verify whether the data overfits by running this model against a larger data set (for example, with years up to 2019 populated). The plots of our predicted output versus our actual values can be seen in figures x – z in the appendix.

## Comparison to Objective 1

Even with a non-parametric model, both KNN and random forest performed optimally when using the same four predictors as our interpretable model from the first objective. Additionally, the KNN model that used all predictors performed about the same with an imputed data set and a data set with the NA values removed. Overall, with small changes to the data set and predictors, all the nonparametric models performed similarly with ASE generally below 10 and R-squared above 90%. This is interesting to note in comparison to our purely predictive linear regression model which deemed country as an important predictor, and we were not able to use it in our non-parametric models. As noted in our analysis, we did get very high R-squared values and low ASE values which indicates there is likely some overfit in our models.

# Conclusion & Final Recommendations

# Appendix

## Figures

Chart, timeline, treemap chart

Description automatically generated

Figure 3: Correlation Matrix, original data

Chart, timeline, treemap chart

Description automatically generated

Figure 4: Correlation Matrix: excluding under 5 deaths, gdp, and thinness 1-19 years

Chart, treemap chart

Description automatically generated

Figure 5: Correlation Matrix: exluding all from Figure 2 along with population

Table

Description automatically generated Table

Description automatically generated

Figure 6: Missing Values, original data Figure 7: Missing values, removed life expectancy

Chart

Description automatically generated

Figure 8: Relationship between Hepatitis B and Life Expectancy

Table

Description automatically generated

Figure 9: Missing Data, remove Hep B & remaining NA's

Chart

Description automatically generated Chart, scatter chart

Description automatically generated

Figure 10: Life Expectancy & Hepatitis B  Figure 11: Life Expectancy & Total Expenditure

Chart, scatter chart

Description automatically generated Chart, scatter chart

Description automatically generated

Figure 12: Life Expectancy & Alcohol Figure 13: Life Expectancy & Inc. Comp. of Resources

Chart, scatter chart

Description automatically generated

Figure 14: Life Expectancy & Schooling

Shape

Description automatically generated

Figure 15: Plot of RSS, Adjusted RSq, Cp, and BIC for Best Subset Selection

A picture containing calendar

Description automatically generated

Figure 16: Variable Selection of Best Subset Selection

A picture containing table

Description automatically generated

Figure 17: Coefficients of Full Model, Forward Selection, and Backwards Selection

Chart, scatter chart

Description automatically generated

Figure 18: Comparison of Predicted Values and Life Expectancy

Diagram, schematic

Description automatically generated

Figure 19: Analysis of Residuals

Chart, scatter chart

Description automatically generated

Figure 20: Residuals from Fitting Model to the Data

# Remove country because this tree function has a maximum of 32 levels  
tree1 <- tree(life\_expectancy ~ ., data = tree\_train[, -1])  
summary(tree1)

##   
## Regression tree:  
## tree(formula = life\_expectancy ~ ., data = tree\_train[, -1])  
## Variables actually used in tree construction:  
## [1] "hiv\_aids" "income\_composition\_of\_resources"  
## [3] "adult\_mortality" "infant\_deaths"   
## Number of terminal nodes: 9   
## Residual mean deviance: 13.7 = 29990 / 2189   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -16.51000 -2.11000 -0.05483 0.00000 2.19300 16.19000

Figure 21: Output chunk for tree model showing the four variables used in the tree construction

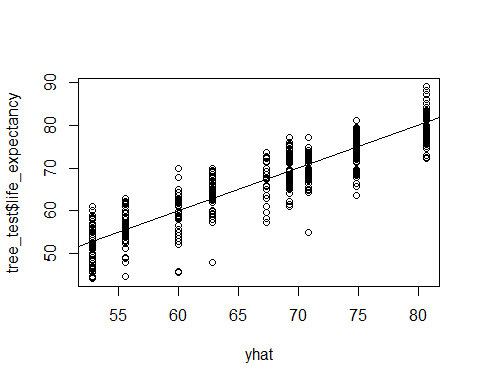


Figure 22: Model fit for our test data set using a tree regression

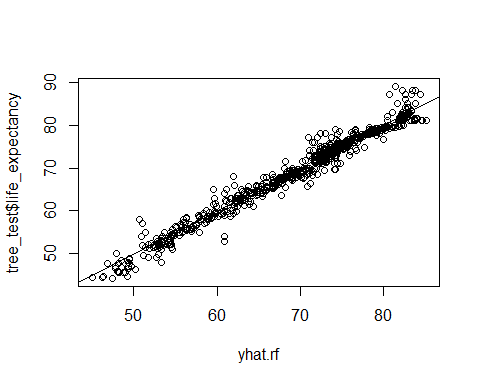


Figure 23: Model fit for our test data set using a random forest regression which used all predictors and 5 splits per node

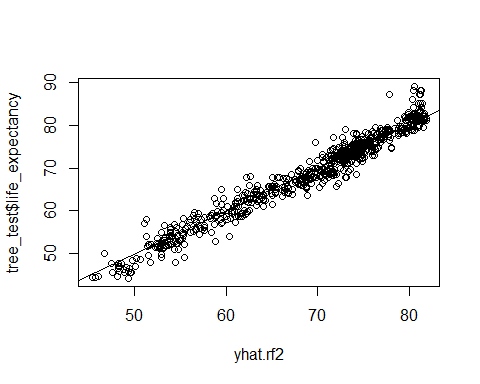


Figure 24: Model fit for our test data set using a random forest regression which used only the top four predictors and 1 split per node

## R Code

knitr::opts\_chunk$set(echo = TRUE)

**library**(tidyverse)

**library**(corrplot)

**library**(GGally)

**library**(gt)

**library**(hrbrthemes)

**library**(car)

**library**(leaps)

**library**(caret)

**library**(tree)

**library**(randomForest)

**library**(plotmo)

**library**(webshot)

*# Load data*

df <- read\_csv(here::here("data - raw", "Life Expectancy Data.csv"))

*# Clean up column names*

df\_clean <- janitor::clean\_names(df)

*# Look at the data*

glimpse(df\_clean)

df\_clean %>%

ggplot(aes(life\_expectancy)) +

geom\_histogram(fill = "steelblue", color = "black") +

labs(title = "Histogram of Life Expectancy",

x = "Age",

y = "Count") +

theme\_ipsum()

df\_clean %>%

ggplot(aes(sample = life\_expectancy)) +

geom\_qq(pch = 21, size = 3, na.rm = TRUE) +

geom\_qq\_line(color = "indianred", na.rm = TRUE) +

labs(title = "Q-Q Plot of Life Expectancy",

x = "Theoretical",

y = "Sample") +

theme\_ipsum()

*# Convert country and status to factors*

df\_clean$country <- as\_factor(df\_clean$country)

df\_clean$status <- as\_factor(df\_clean$status)

*# Run correlations and save output as images to be used in the appendix.*

ggcorr(

df\_clean,

label = TRUE,

label\_alpha = TRUE,

label\_size = 3,

layout.exp = 2,

cex = 3.5,

hjust = 1

)

*# ggsave(here::here("images", "correlation 1.png"))*

df\_clean %>%

select(-c(under\_five\_deaths, gdp, thinness\_1\_19\_years)) %>%

ggcorr(

label = TRUE,

label\_alpha = TRUE,

label\_size = 3,

layout.exp = 2,

cex = 3.5,

hjust = 1

)

*# ggsave(here::here("images", "correlation 2.png"))*

df\_clean %>%

select(-c(under\_five\_deaths, gdp, thinness\_1\_19\_years, population)) %>%

ggcorr(

label = TRUE,

label\_alpha = TRUE,

label\_size = 3,

layout.exp = 2,

cex = 3.5,

hjust = 1

)

*# ggsave(here::here("images", "correlation 3.png"))*

df\_clean <- df\_clean %>%

select(-c(under\_five\_deaths, gdp, thinness\_1\_19\_years, population))

*# Check for missing values*

tibble(variable = names(colSums(is.na(df\_clean))),

missing = colSums(is.na(df\_clean))) %>%

gt() %>%

tab\_header(title = "Missing Values in Data") %>%

*# gtsave(here::here("images", "missing-1.png")) %>%*

{.}

*# Check which countries have NA rows for life expectancy*

as\_tibble(df\_clean$country[which(is.na(df\_clean$life\_expectancy))])

*# Drop all rows where life expectancy is NA*

df\_clean <- df\_clean %>%

filter(!is.na(life\_expectancy))

*# Recheck missing value counts*

tibble(variable = names(colSums(is.na(df\_clean))),

missing = colSums(is.na(df\_clean))) %>%

gt() %>%

tab\_header(title = "Missing Values in Data") %>%

*# gtsave(here::here("images", "missing-2.png")) %>%*

{.}

*# Look at how many missing hepatitis measurements there are by country*

df\_clean %>%

group\_by(country) %>%

count(missing = is.na(hepatitis\_b)) %>%

filter(missing == TRUE) %>%

select(-missing) %>%

rename(missing = n) %>%

arrange(desc(missing))

*# Visualize the relationship to see if it looks significant*

df\_clean %>%

ggplot(aes(x = hepatitis\_b, y = life\_expectancy)) +

geom\_jitter(alpha = 0.3) +

geom\_smooth() +

labs(title = "Hepatitis B vs. Life Expectancy",

x = "Hepatitis B",

y = "Life Expectancy")

*# ggsave(here::here("images", "hepatitis-lifeexp.png"))*

*# Drop hepatitis B variable*

df\_interp <- df\_clean %>%

select(-hepatitis\_b)

*# Drop remaining rows with NA's for the interpretable model*

df\_interp <- na.omit(df\_interp)

*# Final check of missing values*

tibble(variable = names(colSums(is.na(df\_interp))),

missing = colSums(is.na(df\_interp))) %>%

gt() %>%

*# gtsave(here::here("images", "missing-3.png")) %>%*

{.}

df\_interp %>% count(year)

df\_interp <- df\_interp %>% filter(year %**in**% 2011:2014)

*# Set the maximum number of variables to consider in the model. Although the*

*# model can handle up to 20, the more we add, the less interpretable the final*

*# model will be.*

consider <- 17

*# Fit the model. We'll remove country, but keep it in the data set for*

*# interpretation.*

regfit\_full <-

regsubsets(life\_expectancy ~ .,

df\_interp[, -1],

nvmax = consider)

*# Store the regression summary*

reg\_summary <- summary(regfit\_full)

*# Look at the names of reg\_summary*

names(reg\_summary)

*# What are the R-squared values?*

reg\_summary$rsq

par(mfrow = c(2, 2))

plot(reg\_summary$rss,

xlab = "Number of Variables",

ylab = "RSS",

type = "l")

plot(reg\_summary$adjr2,

xlab = "Number of Variables",

ylab = "Adjusted RSq",

type = "l")

*# which.max(reg\_summary$adjr2)*

points(

11,

reg\_summary$adjr2[which.max(reg\_summary$adjr2)],

col = "red",

cex = 2,

pch = 20

)

plot(reg\_summary$cp,

xlab = "Number of Variables",

ylab = "Cp",

type = "l")

*# which.min(reg\_summary$cp)*

points(

9,

reg\_summary$cp[which.min(reg\_summary$cp)],

col = "red",

cex = 2,

pch = 20

)

plot(reg\_summary$bic,

xlab = "Number of Variables",

ylab = "BIC",

type = "l")

*# which.min(reg\_summary$bic)*

points(

5,

reg\_summary$bic[which.min(reg\_summary$bic)],

col = "red",

cex = 2,

pch = 20

)

plot(regfit\_full, scale = "r2")

plot(regfit\_full, scale = "adjr2")

plot(regfit\_full, scale = "Cp")

plot(regfit\_full, scale = "bic")

coef(regfit\_full, 5)

*# Forward*

regfit\_fwd <-

regsubsets(

life\_expectancy ~ .,

data = df\_interp[,-1],

nvmax = 17,

method = "forward"

)

*# summary(regfit\_fwd)*

*# Backward*

regfit\_bwd <-

regsubsets(

life\_expectancy ~ .,

data = df\_interp[,-1],

nvmax = 17,

method = "backward"

)

*# summary(regfit\_bwd)*

*# How many variables do we want to use?*

x <- 4

*# Compare coefficients*

tibble(

variables = names(coef(regfit\_full, x)),

full = round(coef(regfit\_full, x), 4),

fwd = round(coef(regfit\_fwd, x), 4),

bwd = round(coef(regfit\_bwd, x), 4)

) %>%

gt()

*# Set seed*

set.seed(123)

*# Build test and training data sets, dropping country from regfit\_best &*

*# test\_mat because I was getting a warning that 1 linear dependencies found*

train <- sample(c(TRUE, FALSE), nrow(df\_interp), rep = TRUE)

test <- (!train)

regfit\_best <- regsubsets(life\_expectancy ~ ., data = df\_interp[train, -1])

test\_mat <- model.matrix(life\_expectancy ~ ., data = df\_interp[test, -1])

*# Run a loop, and for each size `i`, extract the coefficients from `regfit\_best`*

*# for the best model of that size, multiply them into the appropriate columns of*

*# the test model matric to form the predictions, and compute the test MSE.*

val\_errors <- rep(NA, 8)

**for** (i **in** 1:8) {

coefi <- coef(regfit\_best, id = i)

pred <- test\_mat[, names(coefi)] %\*% coefi

val\_errors[i] <- mean((df\_interp$life\_expectancy[test] - pred)^2)

}

*# Find the best model*

val\_errors

coef(regfit\_best, which.min(val\_errors))

*# Write a prediction function*

predict\_regsubsets <- **function**(object, newdata, id, **...**) {

form <- as.formula(object$call[[2]])

mat <- model.matrix(form, newdata)

coefi <- coef(object, id = id)

xvars <- names(coefi)

mat[, xvars] %\*% coefi

}

*# Perform best subset selection on the full data set, and select the best model.*

regfit\_best <- regsubsets(life\_expectancy ~ ., data = df\_interp[,-1])

coef(regfit\_best, which.min(val\_errors))

k <- 10

set.seed(123)

folds <- sample(1:k, nrow(df\_interp), replace = TRUE)

cv\_errors <- matrix(NA, k, consider, dimnames = list(NULL, paste(1:consider)))

predict.regsubsets <- **function**(object, newdata, id, **...**) {

form <- as.formula(object$call[[2]])

mat <- model.matrix(form, newdata)

coefi <- coef(object, id = id)

mat[, names(coefi)] %\*% coefi

}

*# Perform cross-validation*

**for** (j **in** 1:k) {

best\_fit <-

regsubsets(life\_expectancy ~ ., data = df\_interp[folds != j, -1], nvmax = consider)

**for** (i **in** 1:15) {

pred <- predict(best\_fit, df\_interp[folds == j, -1], id = i)

cv\_errors[j, i] <-

mean((df\_interp$life\_expectancy[folds == j] - pred) ^ 2)

}

}

*# Use the apply function to average over the columns of the matrix in order to*

*# obtain a vector for which the jth element is the cross-validation error for*

*# the j-variable model.*

mean\_cv\_errors <- apply(cv\_errors, 2, mean)

mean\_cv\_errors

par(mfrow = c(1, 1))

plot(mean\_cv\_errors, type = "b")

*# Perform best subset selection on the full data set in order to obtain the*

*# variables for the final model*

reg\_best <- regsubsets(life\_expectancy ~ ., data = df\_interp[, -1])

coef(reg\_best, 4) *# the number sets the number of variables we want*

*# Build the final model using the best subset selection results*

final\_model <-

lm(

life\_expectancy ~ adult\_mortality +

total\_expenditure +

hiv\_aids +

income\_composition\_of\_resources,

data = df\_interp

)

*# Final model summary*

final\_model\_summary <- summary(final\_model)

final\_model\_summary

*# Confidence intervals*

confint(final\_model)

par(mfrow = c(1, 1))

plot(

final\_model$fitted.values,

df\_interp$life\_expectancy,

xlab = "Predicted",

ylab = "Life Expectancy"

)

lines(c(0, 90), c(0, 90), col = "red")

par(mfrow=c(2,2))

plot(final\_model)

par(mfrow = c(1, 2))

test.model <- lm(life\_expectancy ~ ., df\_interp[, -1])

plot(test.model$fitted.values,

test.model$residuals,

xlab = "Fitted Values",

ylab = "Residuals")

plot(df\_interp$life\_expectancy,

test.model$residuals,

xlab = "Life Expectancy",

ylab = "Residuals")

*# Remove all rows with an NA*

*# df\_clean <- na.omit(df\_clean)*

*# sum(is.na(df\_clean))*

*# Maximum number of variables to consider*

consider <- 4

*# Fit model*

regfit\_full <- regsubsets(life\_expectancy ~ ., df\_clean[,-1], nvmax = consider)

*# Examine regression summary*

reg\_summary <- summary(regfit\_full)

names(reg\_summary)

reg\_summary$rsq

*# Plot RSS, adjusted $R^2$ $C\_p$ and BIC for all models*

par(mfrow = c(2, 2))

plot(reg\_summary$rss, xlab = "Number of Variables", ylab = "RSS", type = "l")

plot(reg\_summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type = "l")

which.max(reg\_summary$adjr2)

points(consider, reg\_summary$adjr2[consider], col = "red", cex = 2, pch = 20)

plot(reg\_summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")

which.min(reg\_summary$cp)

points(consider, reg\_summary$cp[consider], col = "red", cex = 2, pch = 20)

which.min(reg\_summary$bic)

plot(reg\_summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")

points(consider, reg\_summary$bic[consider], col = "red", cex = 2, pch = 20)

*# Display selected variables for the best model with a given number of predictors*

plot(regfit\_full, scale = "r2")

plot(regfit\_full, scale = "adjr2")

plot(regfit\_full, scale = "Cp")

plot(regfit\_full, scale = "bic")

coef(regfit\_full, consider)

*# Forward*

regfit\_fwd <- regsubsets(life\_expectancy ~ ., data = df\_clean, method = "forward")

summary(regfit\_fwd)

*# Backward*

regfit\_bwd <- regsubsets(life\_expectancy ~ ., data = df\_clean, method = "backward")

summary(regfit\_bwd)

*# Compare coefficients*

coef(regfit\_full, consider)

coef(regfit\_fwd, consider)

coef(regfit\_bwd, consider)

*# Build test and training data sets*

set.seed(100)

index = sample(1:nrow(df\_clean), 0.7\*nrow(df\_clean))

train = df\_clean[index,] *# Create the training data*

test = df\_clean[-index,] *# Create the test data*

dim(train)

dim(test)

regfit\_best <- regsubsets(life\_expectancy ~ ., data = df\_clean[, -1])

test\_mat <- model.matrix(life\_expectancy ~ ., data = df\_clean)

*#standardize the data by scaling the numeric variables*

*# MF: commenting out gdp, but it had been previously removed*

cols = c(

'alcohol',

'hepatitis\_b',

'measles',

'bmi',

'polio',

'diphtheria',

'hiv\_aids',

*# 'gdp',*

'thinness\_5\_9\_years',

'schooling'

)

pre\_proc\_val <- preProcess(train[,cols], method = c("center", "scale"))

train[,cols] = predict(pre\_proc\_val, train[,cols])

test[,cols] = predict(pre\_proc\_val, test[,cols])

summary(train)

*#check for multicolinearity*

*#note the highly correlated variables*

*# MF: the Performance Analytics package is required for chart.Correlation. Also,*

*# the original my\_data object contains status which, as a non-numeric value*

*# appears to be giving an error. I'm going to remove both it and country and see*

*# if that fixes things.*

*# my\_data <- train[, c(1-2)]*

my\_data <- train[, -c(1:3)]

PerformanceAnalytics::chart.Correlation(my\_data, histogram=TRUE, pch=19)

*#Checking the VIF*

Auto<-train[,-1]

*# MF: the full.model call below was giving me an error that "there are aliased*

*# coeffifients in the model. I again removed country, year and status and it*

*# seems to now run.*

*# full.model<-lm(life\_expectancy~.,data=train)*

full.model<-lm(life\_expectancy~.,data=train[, -c(1:3)])

vif(full.model)

*# Build the final model using the best subset selection results*

*# MF: I commented out gdp since we'd originally excluded it from df\_clean*

final\_model <-

lm(

life\_expectancy ~ status +

alcohol +

bmi +

diphtheria +

hiv\_aids +

*# gdp +*

schooling,

data = train

)

*# Final model summary*

summary(final\_model)

**library**(glmnet)

*#Step 1 - create the evaluation metrics function*

eval\_metrics = **function**(model, df, predictions, target){

resids = df[,target] - predictions

resids2 = resids\*\*2

N = length(predictions)

r2 = as.character(round(summary(model)$r.squared, 2))

adj\_r2 = as.character(round(summary(model)$adj.r.squared, 2))

print(adj\_r2) *#Adjusted R-squared*

print(as.character(round(sqrt(sum(resids2)/N), 2))) *#RMSE*

}

*#this is out best training model that we will use for prediction*

*#Step 3 - predicting and evaluating the model on train data*

predictions = predict(final\_model, newdata = train)

eval\_metrics(final\_model, train, predictions, target = 'life\_expectancy')

*# Step 4 - predicting and evaluating the model on test data*

predictions = predict(final\_model, newdata = test)

eval\_metrics(final\_model, test, predictions, target = 'life\_expectancy')

*#lasso*

*#will use lib glmet*

*#glmnet does not work with numeric data frames*

*#Step 1 -> we will create a numeric matrix for the training*

*# MF: commenting out gdp again*

cols\_reg = c(

'life\_expectancy',

'alcohol',

'hepatitis\_b',

'measles',

'bmi',

'polio',

'diphtheria',

'hiv\_aids',

*# 'gdp',*

'thinness\_5\_9\_years',

'schooling'

)

dummies <- dummyVars(life\_expectancy ~ ., data = df\_clean[,cols\_reg])

train\_dummies = predict(dummies, newdata = train[,cols\_reg])

test\_dummies = predict(dummies, newdata = test[,cols\_reg])

print(dim(train\_dummies)); print(dim(test\_dummies))

lambdas <- 10^seq(2, -3, by = -.1)

*#create the training data matrices for x and y*

x = as.matrix(train\_dummies)

y\_train = train$life\_expectancy

x\_test = as.matrix(test\_dummies)

y\_test = test$life\_expectancy

*# Setting alpha = 1 implements lasso regression*

lasso\_reg <- cv.glmnet(x, y\_train, alpha = 1, lambda = lambdas, standardize = TRUE, nfolds = 5)

*# Best optimal lambda*

lambda\_optimal <- lasso\_reg$lambda.min

lambda\_optimal *#will give the best optimal value*

*#let's train the lasso model*

lasso\_model <- glmnet(x, y\_train, alpha = 1, lambda = lambda\_optimal, standardize = TRUE)

*#let's make predictions for both test and training*

*#also view the evaluation metrics*

*# Compute R^2 from true and predicted values*

eval\_results <- **function**(true, predicted, df) {

SSE <- sum((predicted - true)^2)

SST <- sum((true - mean(true))^2)

R\_square <- 1 - SSE / SST

RMSE = sqrt(SSE/nrow(df))

*# Model performance metrics*

data.frame(

RMSE = RMSE,

Rsquare = R\_square

)

}

*#predictions*

predictions\_train <- predict(lasso\_model, s = lambda\_best, newx = x)

eval\_results(y\_train, predictions\_train, train)

predictions\_test <- predict(lasso\_model, s = lambda\_best, newx = x\_test)

eval\_results(y\_test, predictions\_test, test)

*# Check for missing values*

tibble(variable = names(colSums(is.na(df\_clean))),

missing = colSums(is.na(df\_clean))) %>%

gt()

*#Visualize the relationship to see if it looks significant (plot again)*

*# hepatitis b*

df\_clean %>%

ggplot(aes(x = hepatitis\_b, y = life\_expectancy)) +

geom\_jitter(alpha = 0.3) +

geom\_smooth() +

labs(title = "Relationship of Hepatitis B and Life Expectancy",

x = "Hepatitis B",

y = "Life Expectancy") +

theme\_ipsum()

*# ggsave(here::here("images", "pred-mod-hepb.png"))*

*# total expenditure*

df\_clean %>%

ggplot(aes(x = total\_expenditure, y = life\_expectancy)) +

geom\_jitter(alpha = 0.3) +

geom\_smooth() +

labs(title = "Relationship of Total Expenditure and Life Expectancy",

x = "Total Expenditure",

y = "Life Expectancy") +

theme\_ipsum()

*# ggsave(here::here("images", "pred-mod-expen.png"))*

*# alcohol*

df\_clean %>%

ggplot(aes(x = alcohol, y = life\_expectancy)) +

geom\_jitter(alpha = 0.3) +

geom\_smooth() +

labs(title = "Relationship of Alcohol and Life Expectancy",

x = "Alcohol",

y = "Life Expectancy") +

theme\_ipsum()

*# ggsave(here::here("images", "pred-mod-alcohol.png"))*

*# income composition of resources*

df\_clean %>%

ggplot(aes(x = income\_composition\_of\_resources, y = life\_expectancy)) +

geom\_jitter(alpha = 0.3) +

geom\_smooth() +

labs(title = "Relationship of Income Composition of Resources\nand Life Expectancy",

x = "Income Composition of Resources",

y = "Life Expectancy") +

theme\_ipsum()

*# ggsave(here::here("images", "pred-mod-inc-comp.png"))*

*# schooling*

df\_clean %>%

ggplot(aes(x = schooling, y = life\_expectancy)) +

geom\_jitter(alpha = 0.3) +

geom\_smooth() +

labs(title = "Relationship of Schooling and Life Expectancy",

x = "Schooling",

y = "Life Expectancy") +

theme\_ipsum()

*# ggsave(here::here("images", "pred-mod-school.png"))*

*#remove hepatitis b and total\_expenditure*

df\_predict <- df\_clean %>%

select(-c(hepatitis\_b, total\_expenditure, alcohol))

*#remove the remaining NA's*

df\_predict <- na.omit(df\_predict)

*#check for NA's*

tibble(variable = names(colSums(is.na(df\_predict))),

missing = colSums(is.na(df\_predict))) %>%

gt()

*#replot after removing Na's*

*# income composition of resources*

df\_predict %>%

ggplot(aes(x = income\_composition\_of\_resources, y = life\_expectancy)) +

*# geom\_point() +*

geom\_jitter(alpha = 0.3) +

geom\_smooth() +

labs(title = "Relationship of Income Composition of Resources\nand Life Expectancy",

x = "Income Composition of Resources",

y = "Life Expectancy") +

theme\_ipsum()

*# schooling*

df\_predict %>% ggplot(aes(x = schooling, y = life\_expectancy)) +

*# geom\_point() +*

geom\_jitter(alpha = 0.3) +

geom\_smooth() +

labs(title = "Relationship of Schooling and Life Expectancy",

x = "Schooling",

y = "Life Expectancy") +

theme\_ipsum()

**library**(glmnet)

*# Ridge regression and lasso require the format 'x matrix' and 'y'. The*

*# model.matrix() function produces a matrix and automatically transforms*

*# qualitative variables into dummy variables.*

x <- model.matrix(life\_expectancy ~ ., df\_predict[,-1])[, -1]

y <- df\_predict$life\_expectancy

*# Run ridge regression*

ridge\_mod <- glmnet(x, y, alpha = 0)

dim(coef(ridge\_mod))

*# Split data into training and testing sets*

set.seed(1)

train <- sample(1:nrow(x), nrow(x)/2)

test <- (-train)

y\_test <- y[test]

*# Fit a ridge regression model on the training set, and evaluate its MSE on the*

*# test set*

ridge\_mod <- glmnet(x[train, ], y[train], alpha = 0)

ridge\_pred <- predict(ridge\_mod, s = 4, newx = x[test, ])

mean((ridge\_pred - y\_test) ^ 2)

*#[1] 17.4224*

*# The test MSE is 17.4224. If we had simply fit a model with just an intercept,*

*# we would have observed each test observation using the mean of the training*

*# observations. In that case, we could compute the test set MSE like this:*

mean((mean(y[train]) - y\_test) ^ 2)

*#[1] 89.45262*

*# We could also get the same result by fitting a ridge regression model with a*

*# very large value of ƛ.*

ridge\_pred <- predict(ridge\_mod, s = 1e10, newx = x[test, ])

mean((ridge\_pred - y\_test) ^ 2)

*#[1] 89.45262*

*# Use cross-validation to choose the tuning parameter ƛ.*

set.seed(1)

cv\_out <- cv.glmnet(x[train, ], y[train], alpha = 0)

plot(cv\_out)

bestlambda <- cv\_out$lambda.min

bestlambda

*# What is the test MSE associated with bestlambda?*

ridge\_pred <- predict(ridge\_mod, s = bestlambda, newx = x[test, ])

mean((ridge\_pred - y\_test) ^ 2)

*#[1] 16.2937*

*# Refit the ridge regression model on the full data set using the value of ƛ*

*# chosen by cross-validation*

out <- glmnet(x, y, alpha = 0)

predict(out, type = "coefficients", s = bestlambda)[1:14,]

*# Fit the lasso model*

lasso\_mod <- glmnet(x[train, ], y[train], alpha = 1)

*# Plot the lasso model*

plot\_glmnet(lasso\_mod)

*# Run cross-validation and compute the associated test error*

set.seed(1)

cv\_out <- cv.glmnet(x[train, ], y[train], alpha = 1)

plot(cv\_out)

bestlambda <- cv\_out$lambda.min

lasso\_pred <- predict(lasso\_mod, s = bestlambda, newx = x[test, ])

mean((lasso\_pred - y\_test) ^ 2)

*#[1] 16.41776*

*# Compute lasso coefficients*

out <- glmnet(x, y, alpha = 1)

lasso\_coef <- predict(out, type = "coefficients", s = bestlambda)[1:14,]

lasso\_coef[lasso\_coef != 0]

*#plot distributions*

df\_plot <- as.data.frame(df\_clean)

**for** (col **in** 5:ncol(df\_plot)) {

hist(df\_plot[,col], main=names(df\_plot[col]))

}

*# Copy the data*

df\_predict2 <- df\_clean %>%

select(-c(hepatitis\_b))

*# Impute the data set*

df\_predict2 <- df\_predict2 %>%

mutate(across(

c(polio, total\_expenditure, diphtheria),

~ replace\_na(., median(.x, na.rm = TRUE))

)) %>%

mutate(across(

c(bmi, income\_composition\_of\_resources, schooling),

~ replace\_na(., mean(.x, na.rm = TRUE))

)) %>%

mutate(across(

c(alcohol, thinness\_5\_9\_years),

~ replace\_na(., 0)

))

*#check for NA's*

tibble(variable = names(colSums(is.na(df\_predict2))),

missing = colSums(is.na(df\_predict2))) %>%

gt()

*# Ridge regression and lasso require the format 'x matrix' and 'y'. The*

*# model.matrix() function produces a matrix and automatically transforms*

*# qualitative variables into dummy variables.*

sum(is.na(df\_predict2))

x2 <- model.matrix(life\_expectancy ~ ., df\_predict2[,-1])[, -1]

y2 <- df\_predict2$life\_expectancy

*# Run ridge regression*

ridge\_mod2 <- glmnet(x2, y2, alpha = 0)

dim(coef(ridge\_mod2))

*# Split data into training and testing sets*

set.seed(1)

train2 <- sample(1:nrow(x2), nrow(x2)/2)

test2 <- (-train2)

y\_test2 <- y2[test2]

*# Fit a ridge regression model on the training set, and evaluate its MSE on the*

*# test set*

ridge\_mod2 <- glmnet(x2[train2,], y2[train2], alpha = 0)

ridge\_pred2 <- predict(ridge\_mod2, s = 4, newx = x2[test2, ])

mean((ridge\_pred2 - y\_test2) ^ 2)

*#[1] 19.39197*

*# The test MSE is 19.39197. If we had simply fit a model with just an intercept,*

*# we would have observed each test observation using the mean of the training*

*# observations. In that case, we could compute the test set MSE like this:*

mean((mean(y2[train2]) - y\_test2) ^ 2)

*#[1] 88.25497*

*# Use cross-validation to choose the tuning parameter ƛ.*

set.seed(1)

cv\_out2 <- cv.glmnet(x2[train2, ], y2[train2], alpha = 0)

plot(cv\_out2)

bestlambda2 <- cv\_out2$lambda.min

bestlambda2

*#[1] 0.6900917*

*# What is the test MSE associated with bestlambda?*

ridge\_pred2 <- predict(ridge\_mod2, s = bestlambda2, newx = x2[test2, ])

mean((ridge\_pred2 - y\_test2) ^ 2)

*#[1] 18.68608*

*# Refit the ridge regression model on the full data set using the value of ƛ*

*# chosen by cross-validation*

out2 <- glmnet(x2, y2, alpha = 0)

predict(out2, type = "coefficients", s = bestlambda2)[1:14,]

*# Fit the lasso model*

lasso\_mod2 <- glmnet(x2[train2, ], y2[train2], alpha = 1)

*# Plot the lasso model*

plot\_glmnet(lasso\_mod2)

*# Run cross-validation and compute the associated test error*

set.seed(1)

cv\_out2 <- cv.glmnet(x2[train2, ], y2[train2], alpha = 1)

plot(cv\_out2)

bestlambda2 <- cv\_out2$lambda.min

lasso\_pred2 <- predict(lasso\_mod2, s = bestlambda2, newx = x2[test2, ])

mean((lasso\_pred2 - y\_test2) ^ 2)

*#[1] 18.74681*

*# Compute lasso coefficients*

out2 <- glmnet(x2, y2, alpha = 1)

lasso\_coef2 <- predict(out2, type = "coefficients", s = bestlambda2)[1:14,]

lasso\_coef2[lasso\_coef2 != 0]

*# Fit the lasso model with data set including country*

x3 <- model.matrix(life\_expectancy ~ ., df\_predict2)[, -1]

y3 <- df\_predict2$life\_expectancy

*# Split data into training and testing sets*

set.seed(1)

train3 <- sample(1:nrow(x3), nrow(x3)/2)

test3 <- (-train3)

y\_test3 <- y3[test3]

lasso\_mod3 <- glmnet(x3[train3, ], y3[train3], alpha = 1)

*# Plot the lasso model*

plot\_glmnet(lasso\_mod3)

*# Run cross-validation and compute the associated test error*

set.seed(1)

cv\_out3 <- cv.glmnet(x3[train3, ], y3[train3], alpha = 1)

plot(cv\_out3)

bestlambda3 <- cv\_out3$lambda.min

lasso\_pred3 <- predict(lasso\_mod3, s = bestlambda3, newx = x3[test3, ])

mean((lasso\_pred3 - y\_test3) ^ 2)

*#[1] 18.74681*

*# Compute lasso coefficients*

out3 <- glmnet(x3, y3, alpha = 1)

lasso\_coef3 <- predict(out3, type = "coefficients", s = bestlambda3)[1:14,]

lasso\_coef3[lasso\_coef3 != 0]

*#build a linear model using the top 3 variables from LASSO along with country for both imputed data set and removed NA data set*

*# Build the final model using the best subset selection results on the imputed data set*

predict\_model1 <-

lm(

life\_expectancy ~

country +

income\_composition\_of\_resources +

status +

schooling +

hiv\_aids,

data = df\_predict2

)

*# Final model summary*

predict\_model1\_sum <- summary(predict\_model1)

predict\_model1\_sum

*# Get MSE*

mean(predict\_model1\_sum$residuals ^ 2)

*#[1] 4.39226*

*# Build the final model using the best subset selection results on removed NA data set*

predict\_model2 <-

lm(

life\_expectancy ~

country +

income\_composition\_of\_resources +

status +

schooling +

hiv\_aids,

data = df\_predict

)

*# Final model summary*

predict\_model2\_sum <- summary(predict\_model2)

predict\_model2\_sum

*# Get MSE*

mean(predict\_model2\_sum$residuals ^ 2)

*#[1] 4.189329*

par(mfrow = c(1, 1))

plot(

predict\_model2$fitted.values,

df\_predict$life\_expectancy,

xlab = "Predicted Life Expectancy",

ylab = "Actual Life Expectancy"

)

lines(c(0, 90), c(0, 90), col = "red")

par(mfrow=c(2,2))

plot(predict\_model2)

*#copy and rename imputed data set for KNN models*

df\_knn <- df\_predict2

*#Make new data set that does not impute values for comparison*

df\_knn2 <- df\_clean %>% select(-c(hepatitis\_b))

df\_knn2 <- drop\_na(df\_knn2)

*# Set seed*

set.seed(123)

*# Standardize the data to prep for KNN first - everything except life expectancy*

preProcValues <- preProcess(df\_knn[, -4], method = c("scale"))

df\_knn\_standard <- predict(preProcValues, df\_knn)

*# Split training/test data sets*

inTraining <-

createDataPartition(df\_knn\_standard$life\_expectancy,

p = 0.75,

list = FALSE)

knn\_train <- df\_knn\_standard[inTraining, ]

knn\_test <- df\_knn\_standard[-inTraining, ]

*# Perform same splits for data with NA's removed*

*# Standardize the data to prep for KNN first - everything except life expectancy*

preProcValues\_2 <- preProcess(df\_knn2[, -4], method = c("scale"))

df\_knn\_standard2 <- predict(preProcValues\_2, df\_knn2)

*# Split training/test data sets*

inTraining2 <-

createDataPartition(df\_knn\_standard2$life\_expectancy,

p = 0.75,

list = FALSE)

knn\_train\_na <- df\_knn\_standard2[inTraining2, ]

knn\_test\_na <- df\_knn\_standard2[-inTraining2, ]

*# Set seed*

set.seed(567)

*# Set train control: 5 repeat, 10-fold CV*

ctrl <-

trainControl(

method = "repeatedcv",

number = 10, *# 10-fold CV*

repeats = 5, *#repeat 5 times*

returnResamp = "all" *#return all metrics*

)

*# Run everything with the train control above*

knnFit <-

train(

life\_expectancy ~ .,

data = knn\_train,

method = "knn",

trControl = ctrl,

tuneLength = 10 *#run for 10 different k's*

)

knnFit

*# Check the metrics on the test set*

Predictions\_knn5 <- predict(knnFit, newdata = knn\_test)

ASE\_knn5 <- mean((Predictions\_knn5 - knn\_test$life\_expectancy)^2)

ASE\_knn5

*# Performance measurement*

postResample(knn\_test$life\_expectancy, Predictions\_knn5)

*# Plotting*

plot(knnFit, main = "knnFit Results")

plot(knnFit, metric = "Rsquared", main = "knnFit Results (R-Squared)")

plot(knnFit, metric = "MAE", main = "knnFit Results (MAE)")

*# Try another knn with a k value of less than 5*

knnFit2 <-

train(

life\_expectancy ~ .,

data = knn\_train,

method = "knn",

trControl = ctrl,

tuneGrid = expand.grid(k = c(1, 3, 5)) *#run only with k = 1, 3, 5*

)

knnFit2

*# Check the metrics on the test set*

Predictions\_knn3 <- predict(knnFit2, newdata = knn\_test)

ASE\_knn3 <- mean((Predictions\_knn3 - knn\_test$life\_expectancy)^2)

ASE\_knn3

*# Performance measurement*

postResample(knn\_test$life\_expectancy, Predictions\_knn3)

*# Plotting*

plot(knnFit2, main = "knnFit2 Results")

plot(knnFit2, metric = "Rsquared", main = "knnFit2 Results (R-Squared)")

plot(knnFit2, metric = "MAE", main = "knnFit2 Results (MAE)")

*# Run knn on data set with NA's removed*

knnFit\_na <-

train(

life\_expectancy ~ .,

data = knn\_train\_na,

method = "knn",

trControl = ctrl,

tuneLength = 10 *#run through 10 different k's*

)

knnFit\_na

*# Check the metrics on the test set*

Predictions\_knn\_na <- predict(knnFit\_na, newdata = knn\_test\_na)

ASE\_knn\_na <- mean((Predictions\_knn\_na - knn\_test\_na$life\_expectancy)^2)

ASE\_knn\_na

*# Performance measurement*

postResample(knn\_test\_na$life\_expectancy, Predictions\_knn\_na)

knnFit3 <-

train(

life\_expectancy ~ adult\_mortality +

total\_expenditure +

hiv\_aids +

income\_composition\_of\_resources,

data = knn\_train,

method = "knn",

trControl = ctrl,

tuneLength = 10

)

knnFit3

*#check metrics on test set*

Predictions\_knnfit3 <- predict(knnFit3,newdata=knn\_test)

*#performance measurement*

postResample(knn\_test$life\_expectancy,Predictions\_knnfit3)

ASE\_knn3 <- mean((Predictions\_knnfit3 - knn\_test$life\_expectancy)^2)

ASE\_knn3

plot(knnFit3)

plot(knnFit3, metric = "Rsquared")

plot(knnFit3, metric = "MAE")

*# Split training/test data sets - round 2*

set.seed(1)

inTraining\_x <-

createDataPartition(df\_knn\_standard$life\_expectancy,

p = 0.75,

list = FALSE)

knn\_train\_x <- df\_knn\_standard[inTraining\_x, ]

knn\_test\_x <- df\_knn\_standard[-inTraining\_x, ]

*# Run everything with the train control above*

knnFit\_x <-

train(

life\_expectancy ~ .,

data = knn\_train\_x,

method = "knn",

trControl = ctrl,

tuneLength = 10

)

knnFit\_x

*# Check the metrics on the test set*

Predictions\_knnFit\_x <- predict(knnFit\_x, newdata = knn\_test\_x)

ASE\_x <- mean((Predictions\_knnFit\_x - knn\_test\_x$life\_expectancy)^2)

ASE\_x

*# Performance measurement*

postResample(knn\_test\_x$life\_expectancy, Predictions\_knnFit\_x)

*# Plotting*

plot(knnFit\_x, main = "knnFit Results")

plot(knnFit\_x, metric = "Rsquared", main = "knnFit Results (R-Squared)")

plot(knnFit\_x, metric = "MAE", main = "knnFit Results (MAE)")

*# Try another knn with a k value of less than 5*

knnFit2\_x <-

train(

life\_expectancy ~ .,

data = knn\_train\_x,

method = "knn",

trControl = ctrl,

tuneGrid = expand.grid(k = c(1, 3, 5))

)

knnFit2\_x

*# Check the metrics on the test set*

Predictions\_knnFit2\_x <- predict(knnFit2\_x, newdata = knn\_test\_x)

ASE\_x <- mean((Predictions\_knnFit2\_x - knn\_test\_x$life\_expectancy)^2)

ASE\_x

*# Performance measurement*

postResample(knn\_test\_x$life\_expectancy, Predictions\_knnFit2\_x)

*# Plotting*

plot(knnFit2\_x, main = "knnFit2 Results")

plot(knnFit2\_x, metric = "Rsquared", main = "knnFit2 Results (R-Squared)")

plot(knnFit2\_x, metric = "MAE", main = "knnFit2 Results (MAE)")

knnFit3\_x <-

train(

life\_expectancy ~ adult\_mortality +

total\_expenditure +

hiv\_aids +

income\_composition\_of\_resources,

data = knn\_train\_x,

method = "knn",

trControl = ctrl,

tuneLength = 10

)

knnFit3\_x

*# Check the metrics on the test set*

Predictions\_knnFit3\_x <- predict(knnFit3\_x, newdata = knn\_test\_x)

ASE\_3x <- mean((Predictions\_knnFit3\_x - knn\_test\_x$life\_expectancy)^2)

ASE\_3x

*# Performance measurement*

postResample(knn\_test\_x$life\_expectancy, Predictions\_knnFit3\_x)

*# Plotting*

plot(knnFit3\_x, main = "knnFit2 Results")

plot(knnFit3\_x, metric = "Rsquared", main = "knnFit2 Results (R-Squared)")

plot(knnFit3\_x, metric = "MAE", main = "knnFit2 Results (MAE)")

*# Set seed*

set.seed(123)

*# Split training/test data sets*

inTraining <- createDataPartition(df\_knn$life\_expectancy, p = 0.75, list = FALSE)

tree\_train <- df\_knn[inTraining,]

tree\_test <- df\_knn[-inTraining,]

*# Remove country because this tree function has a maximum of 32 levels*

tree1 <- tree(life\_expectancy ~ ., data = tree\_train[, -1])

summary(tree1)

plot(tree1)

mean((tree1$y - tree\_train$life\_expectancy) ^ 2)

*# Check tree performance*

cv.tree1 <- cv.tree(tree1)

plot(cv.tree1$size, cv.tree1$dev, type = 'b')

*# Check the predictions*

yhat = predict(tree1, newdata = tree\_test[, -1])

plot(yhat, tree\_test$life\_expectancy)

abline(0, 1)

mean((yhat - tree\_test$life\_expectancy) ^ 2)

*# Try a random forest compared to a single tree model*

set.seed(1)

*# Remove country again*

rfFit <- randomForest(life\_expectancy ~ ., data = tree\_train[, -1])

rfFit

*# Test using all predictors for each tree*

set.seed(1)

*# Have to remove country again - using top 4 predictors*

rfFit2 <- randomForest(life\_expectancy ~ adult\_mortality +

total\_expenditure +

hiv\_aids +

income\_composition\_of\_resources, data = tree\_train[,-1])

rfFit2

*#compare y-hat for the two random forest models*

yhat.rf <- predict(rfFit, newdata = tree\_test[, -1])

yhat.rf2 <- predict(rfFit2, newdata = tree\_test[, -1])

*# Performance measurement*

postResample(tree\_test$life\_expectancy, yhat.rf)

postResample(tree\_test$life\_expectancy, yhat.rf2)

ASE\_rf <- mean((yhat.rf - tree\_test$life\_expectancy)^2)

ASE\_rf

ASE\_rf2 <- mean((yhat.rf2 - tree\_test$life\_expectancy)^2)

ASE\_rf2

plot(yhat.rf, tree\_test$life\_expectancy)

abline(0,1)

plot(yhat.rf2,tree\_test$life\_expectancy)

abline(0,1)

*# Set seed*

set.seed(1)

*# Split training/test data sets*

inTraining\_t <- createDataPartition(df\_knn$life\_expectancy, p = 0.75, list = FALSE)

tree\_train\_x <- df\_knn[inTraining\_t,]

tree\_test\_x <- df\_knn[-inTraining\_t,]

*# Try a random forest compared to a single tree model*

set.seed(567)

*# Remove country again*

rfFit\_x <- randomForest(life\_expectancy ~ ., data = tree\_train\_x[, -1])

rfFit\_x

*# Test using all predictors for each tree*

set.seed(5)

*# Have to remove country again - using top 4 predictors*

rfFit2\_x <- randomForest(life\_expectancy ~ adult\_mortality +

total\_expenditure +

hiv\_aids +

income\_composition\_of\_resources, data = tree\_train\_x[,-1])

rfFit2\_x

*#compare y-hat for the two random forest models*

yhat.rf\_x <- predict(rfFit\_x, newdata = tree\_test\_x[, -1])

yhat.rf2\_x <- predict(rfFit2\_x, newdata = tree\_test\_x[, -1])

*# Performance measurement*

postResample(tree\_test\_x$life\_expectancy, yhat.rf\_x)

postResample(tree\_test\_x$life\_expectancy, yhat.rf2\_x)

plot(yhat.rf\_x, tree\_test\_x$life\_expectancy)

abline(0,1)

plot(yhat.rf2\_x,tree\_test\_x$life\_expectancy)

abline(0,1)