Compulsory Exercise 2: Life Expectancy - The Gate into an International multi-class Classification Problem

Eirik Alexander Fagerbakke Henrik Andreas Grenersen
Halvard Emil Sand-Larsen

20 april, 2023

Abstract

We have looked into the WHO Life expectancy dataset to test different methods for predicting life expectancy and explore how well we can predict regions based on the data. To evaluate our models we will consider their performance as well as their complexity. We have looked at a variety of methods for regression and classification, and their performence for the problem at hand varied. Through our project we achieved a misclassification rate of the regions as low as 0.2, using support vector machines. For prediction, the lowest a mean square error obtained is 1.65, using a full linear regression model.

Introduction: Scope and purpose of your project

In this project, we aim to use the WHO Life expectancy dataset to determine what features affect the life expectancy (regression), and also to look into how the variables in the dataset are connected to regions (classification). The dataset contains information about 179 countries from 2000-2015, with 21 columns. We have used the data gathered in 2015, since this is the most up to date. The dataset can be found on kaggle: https://www.kaggle.com/datasets/lashagoch/life-expectancy-who-updated.

The dataset was originally gathered from the Global Health Observatory (GHO) data repository under the World Health Organization (WHO), and has been updated to fill out missing values in the original data. This was done by either taking the average over a region, or by using the closest three-year average. Countries that were missing too much data were omitted from the dataset.

Our main purpose has been to predict life expectancy, as well as to classify into regions, based on the variables in the dataset. In this process, we have also had to deal with inference, as we (for the most part) want to have an interpretable method with the most important predictors.

For life expectancy, we have first looked into linear regression and then subset selection to determine the most relevant variables. We have also used decision trees to determine life expectancy, as this yields easily interpretable results and are easy to implement.

For classification of regions, we have looked into KNN, LDA and QDA, and compared the models. Finally, we stepped away from prioritizing interpretability and looked into using SVM to classify, by using the one vs. one extension of binary SVM classification.

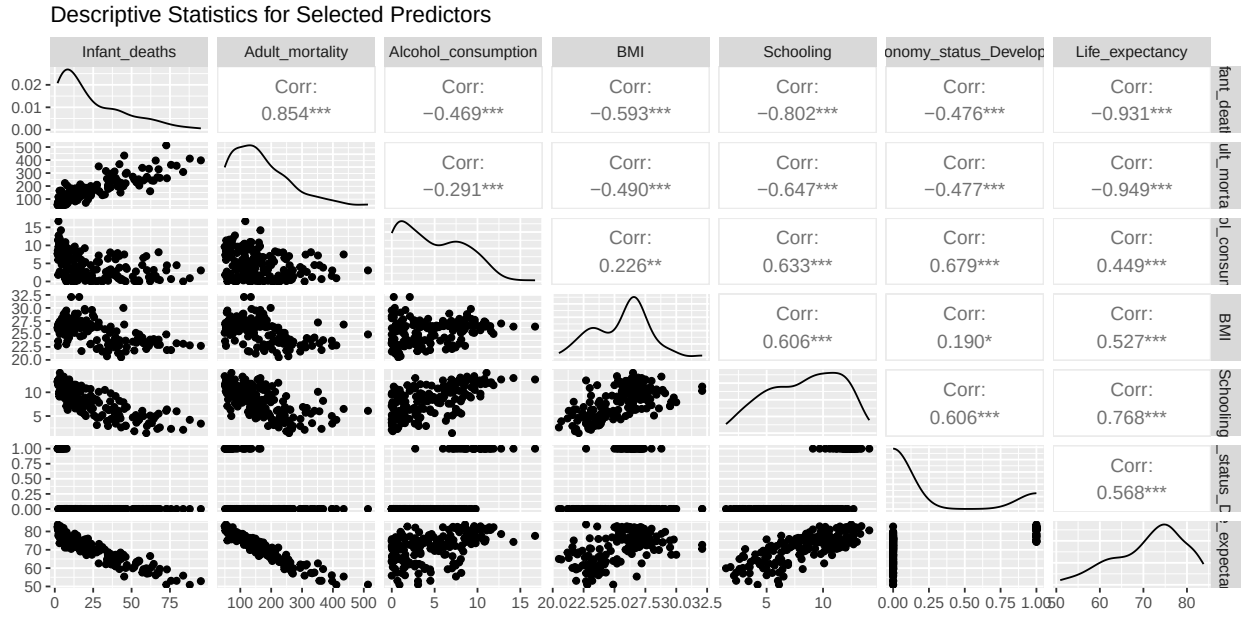
Descriptive data analysis/statistics

In order to acquaint ourselves more with our dataset we have performed some descriptive data analysis. In order to do this we have had to do data wrangling as well, and have made some different dataframes which will come in handy throughout the project.

```
# Data wrangling
whole_df <- read.csv("Life-Expectancy-Data-Updated.csv")

df <- whole_df[whole_df$Year == 2015, ]
df <- subset(df, select = -c(Economy_status_Developing, Year, Country))
```

Having prepared our data we now aim to present some interesting trends in our dataset. Because the number of possible predictors is above 20, our usual approach of using `ggpair` came up short. Therefore we have selected some of the most interesting features and plotted them together using this function below.

```
# GGpairs
gplot <- GGally::ggpairs(df[c(2, 4, 5, 8, 16, 17, 18)], title = "Descriptive Statistics for Selected Pr
print(gplot)
```

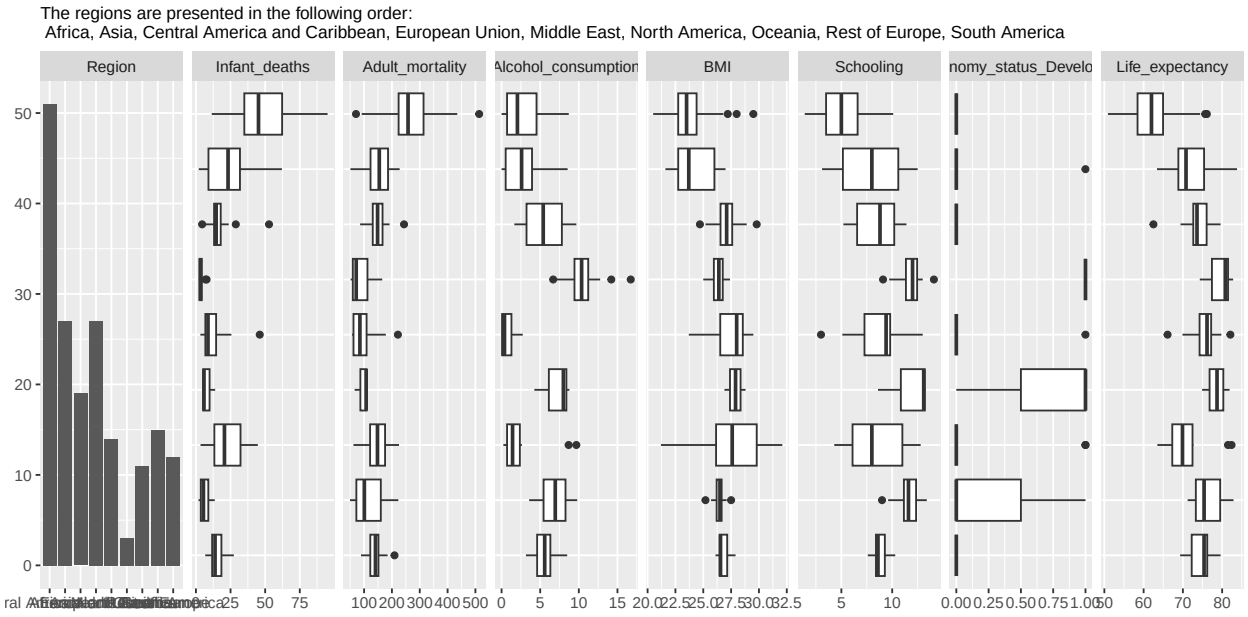


From this plot we observe that the life expectancy is strongly negatively correlated to the number of infant deaths and adult mortality in the countries. From the scatter plots of these variables we also observe that there is a linear trend among the predictors, and we will investigate this more during the section about multiple linear regression.

Another interesting point about this plot regards the predictor `Economy_status_Developed`, which is 1 if a country identifies as developed, and 0 if not. It is however important to note that this is more of an identity than a quantifiable property. As the countries themselves report what they identify as, there are not necessarily any metrics involved. This might also have some influence on this predictors predictive power. As we see from the scatter plots, there is generally a much higher variance within the class of developing countries compared to the developed countries, except for maybe alcohol consumption. There are of course a bit more developing countries in the dataset, in fact they constitute 0.7932961% of the dataset, so a higher variance might be expected to some degree.

Because we will work particularly with the region as a predictor later, we also present a plot displaying descriptive statistics for this predictor below.

```
# Regionplot
title <- paste(sort(unique(df$Region)), collapse = ", ")

gplot <- GGally::ggpairs(df[c(1, 2, 4, 5, 8, 16, 17, 18)], title = paste("The regions are presented in t
    title)) + theme(plot.title = element_text(size = 10))
gplot$nrow <- 1
gplot$yAxisLabels <- df$yAxisLabels[1]
print(gplot)
```



In the plot above we can again consider the economic status of the country, and note that there is very little variance in most regions, except for the countries classified as North American or belonging to the rest of Europe. Therefore, this predictor may not be particularly useful if one were to predict the region of a country. On the other hand we have the other predictors, which all seem to have at least some variation in all regions. Another feature that might be worth noting is that we do not have an equal distribution of the regions within our dataset, so there is a chance that some regions, such as North America, consisting of only three countries, might not appear in our training or test set and this might also cause problems for our approaches.

```
# Removing Economy_status_Developed and creating dummy variables

df <- subset(df, select = -Economy_status_Developed)

df_with_dummies <- dummy_cols(df, select_column = "Region", remove_first_dummy = TRUE)
df_with_dummies <- subset(df_with_dummies, select = -c(Region))
```

```
# Splitting into testing and training data for linear regression
set.seed(1969)

n <- nrow(df)
train_indices <- sample.int(n, 0.75 * n, replace = FALSE)
```

```
df_train_with_dummies <- df_with_dummies[train_indices, ]
df_test_with_dummies <- df_with_dummies[-train_indices, ]

df_train <- df[train_indices, ]
df_test <- df[-train_indices, ]
```

Methods

As our data now is prepared for use, we move on to predictions. Our first response is life expectancy, and we will try to predict it with linear regression.

Linear Regression and Subset Selection

Linear regression tries to estimate the predictor by adding linear terms of all the covariates and an intercept. Our first approach is to use all our covariates, which leads to a quite large model.

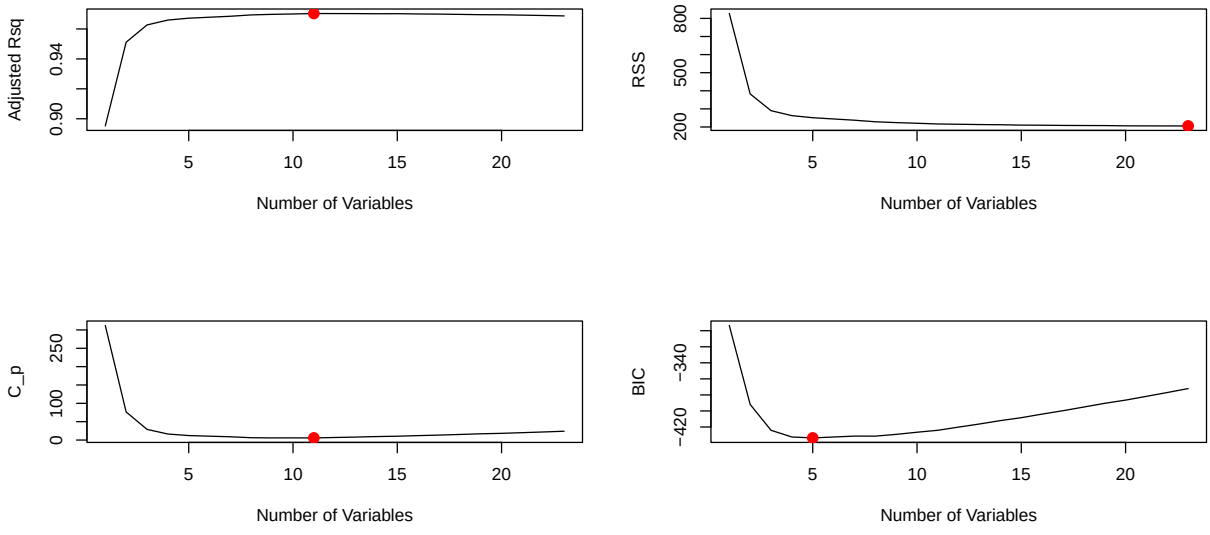
```
# Full linear regression

full_lin_mod <- lm(Life_expectancy ~ ., df_train_with_dummies)

full_lin_mod_summary <- summary(full_lin_mod)

pred_lin_mod <- predict(full_lin_mod, df_test_with_dummies)

full_lin_MSE <- mean((pred_lin_mod - df_test_with_dummies$Life_expectancy)^2)

# low p-values
alpha = 0.05
full_lin_mod_summary$coefficients[, "Pr(>|t|)"][full_lin_mod_summary$coefficients[,
    "Pr(>|t|)"] < alpha]
```

```
##                             (Intercept)                          Adult_mortality
##                            7.416720e-51                            5.091105e-24
##                          GDP_per_capita `Region_Central America and Caribbean`
##                            3.959451e-04                            2.743863e-02
##                  `Region_South America`
##                            3.986568e-02
```

From the p-values, we see that with a significance level of $\alpha = 0.05$, the variables that are deemed significant in the full linear model are the intercept, `Adult_mortality`, `GDP_per_capita`, `Region_Central America and Caribbean` and `Region_South America`. The importance of regions as predictors, seemed quite counter-intuitive to us, and led us to later investigate the relationship between the region and the other predictors further. This will be done from the section "A Change of Direction".

To try to reduce our model size, we turn to best subset selection. This method finds the subset with smallest MSE for a given number of covariates. However, we have used forward selection to save time. This no longer guarantees we find the optimal subset, as it does not consider all combinations of the predictors, but choses one at a time. We then have to pick the best of the best with a penalizing criteria. By looking at BIC, CP, RSS and adjusted R^2, we decided to use 6 covariates. We then performed linear regression with these 6 covariates, and will compare the two models further in the results section.

```r
# Best subset selection
p <- dim(df_with_dummies)

best_subset <- regsubsets(Life_expectancy ~ ., df_train_with_dummies, nvmax = p,
    method = "forward")

summary_best_subset <- summary(best_subset)

par(mfrow = c(2, 2))
plot(summary_best_subset$adjr2, xlab = "Number of Variables", ylab = "Adjusted Rsq",
    type = "l")
bsm_best_adjr2 = which.max(summary_best_subset$adjr2)
points(bsm_best_adjr2, summary_best_subset$adjr2[bsm_best_adjr2], col = "red", cex = 2,
    pch = 20)


plot(summary_best_subset$rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
bsm_best_rss = which.min(summary_best_subset$rss)
points(bsm_best_rss, summary_best_subset$rss[bsm_best_rss], col = "red", cex = 2,
    pch = 20)

plot(summary_best_subset$cp, xlab = "Number of Variables", ylab = "C_p", type = "l")
bsm_best_cp = which.min(summary_best_subset$cp)
points(bsm_best_cp, summary_best_subset$cp[bsm_best_cp], col = "red", cex = 2, pch = 20)

plot(summary_best_subset$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
bsm_best_bic = which.min(summary_best_subset$bic)
points(bsm_best_bic, summary_best_subset$bic[bsm_best_bic], col = "red", cex = 2,
    pch = 20)
```



```
# Model we actually select -
best_num_of_preds <- 6
best_preds <- names(coef(best_subset, id = best_num_of_preds))
best_preds <- best_preds[!best_preds %in% "(Intercept)"]
best_preds <- unlist(lapply(best_preds, function(t) gsub("`", "", t)))

# used for best subset selection
bsm_train <- subset(df_train_with_dummies, select = c(best_preds, "Life_expectancy"))

bsm_mod <- lm(Life_expectancy ~ ., data = bsm_train)

pred_best_lin_mod <- predict(bsm_mod, df_test_with_dummies)

best_lin_MSE <- mean((pred_best_lin_mod - df_test_with_dummies$Life_expectancy)^2)
```

Decision Trees and Random Forests

To predict life expectancy, one of the methods we have looked into is decision trees. Decision trees do not generally have good prediction performance due to variance, however they can be very useful for interpretability and are relatively easy to implement.

To choose the tree size, we will first build a large tree. We will then prune the tree according to a 10-fold cross validation.

```
set.seed(1234)
df.factors = df
df.factors$Region = as.factor(df.factors$Region)
df.factors.train = df.factors[train_indices, ]
df.factors.test = df.factors[-train_indices, ]


# start by building large tree, which will later be pruned
tree.mod = tree(Life_expectancy ~ ., data = df.factors.train, control = tree.control(nobs = n,
    mincut = 2, minsize = 4, mindev = 0.001))

K = 10

cv.tree.mod = cv.tree(tree.mod, K = K)
treesizes = cv.tree.mod$size
treedevs = cv.tree.mod$dev
tree.min = which.min(treedevs)
best = treesizes[tree.min]
```

```
use_tree_size = 7

# using 'best'
tree.best = prune.tree(tree.mod, best = use_tree_size)
tree.best.pred = predict(tree.best, newdata = df.factors.test)
tree.best.MSE = mean((tree.best.pred - df.factors.test$Life_expectancy)^2)
```

Next, we have looked into random forests. Here we select `B=500` trees, and at each cut we choose between a third of the covariates. We then take the average over all trees to get the final prediction. This leaves us with

a model that has less variance, and ensures that the trees are not too correlated, compared to in bagging. This should also be better suited for our problem than decision trees, since we have many predictors.

The disadvantage is that we get a method that is less interpretable, but we can still look at a variance plot to see what predictors are the most prevalent in the trees.

```
# random forest
B = 500  #number of trees
p = ncol(df) - 1
rf = randomForest(Life_expectancy ~ ., data = df.factors.train, mtry = p/3, ntree = B,
    importance = TRUE)
rf.pred = predict(rf, newdata = df.factors.test)

rf.MSE = mean((rf.pred - df.factors.test$Life_expectancy)^2)
```

A Change of Direction

We now want to predict the region for countries outside our training data. This is based on the assumption that countries in the same region will have similar patterns, but there will off course be abnormalities. Because of this it is unrealistic to expect perfect classification, but hopefully still an adequate precision.

KNN

To do this, we will first try KNN. Thus, there are a few things which must be decided, namely K , where K represents the number of neighboring points that affect our classification of a certain point, and a subset of our predictors. We have decided to limit this subset to two covariates because of the curse of dimensionality, as well as a fairer comparison to LDA and QDA, which we will talk more about later. To find the best subset we then use k-fold CV with a fixed $K = 10$. This is done by splitting our training data into 4 similar sized folds and then calculating the average misclassification rate when we exclude each fold and use it as a test. This error rate is calculated for every pair of covariates and we then choose the subset with the lowest error. To find our actual misclassification rate we apply KNN with the best subset and $K = 10$ to our actual test data. It is possible to try the same procedure for different K-values, but if we increase K we quickly favor the regions with more countries, while a lower K becomes more affected by outliers. To calculate the misclassification rate we make a confusion matrix and then use that the diagonal elements are predicted correctly while the off diagonals are predicted wrongly. This gives us a total error rate, but this is slightly biased since our classes have very different sizes. For instance is it more important to be accurate in Europe than in North America, just because there are more countries in Europe. As an effect of this the best subset we find might favor the larger regions and perform quite badly for the smaller region. Lastly, we performed this process with and without scaling, and the scaled data performed significantly worse. This could be because a higher weight of a covariate actually performs better than an equal weighing. Hence, we decided to perform KNN without scaling the data.

```
set.seed(1969)

df_train_kkn <- subset(df_train, select = -c(Life_expectancy))

names <- colnames(df_train_kkn)
cov1 = ""
cov2 = ""
F = 4
K = 10
for (i in 2:15) {
    # Skip number 1 because that is the region
```

```
    for (j in (i + 1):16) {
        folds <- sample(1:F, dim(df_train_kkn)[1], replace = TRUE)
        error_rate = 1
        e_temp = 0
        for (f in 1:F) {
            df_train_loc <- df_train_kkn[folds != f, ]
            df_test_loc <- df_train_kkn[folds == f, ]
            cl <- df_train_loc[, "Region"]
            KNN <- knn(subset(df_train_loc, select = c(names[i], names[j])), subset(df_test_loc,
                select = c(names[i], names[j])), cl, K)
            matr <- table(KNN, df_test_loc[, "Region"])
            e_temp = e_temp + 1 - sum(diag(matr))/length(df_train_kkn[folds == f,
                "Region"])
        }
        e_temp = e_temp/4
        if (e_temp < error_rate) {
            error_rate = e_temp
            cov1 = names[i]
            cov2 = names[j]
        }
    }
}

cl <- df_train[, "Region"]
cl_test <- df_test[, "Region"]

KNN <- knn(subset(df_train, select = c(cov1, cov2)), subset(df_test, select = c(cov1,
    cov2)), cl, 10)

matr <- table(KNN, cl_test)

knn_misclass_rate <- 1 - sum(diag(matr))/dim(df_test)[1]
```

Discriminant Analysis

Another class of classification algorithms which we have applied in this project are methods which involve discriminant analysis, both linear (LDA) and quadratic (QDA). Unlike KNN, the curse of dimensionality should not pose a problem for these methods, so a larger number of predictors may be used. However, this also makes the problem of choosing these parameters more complex. Another problem with using a high number of parameters is that we might run into problems in our algorithms because of our training sets.

An example of such a problem, which we encountered during our work, was that if we used a high number of predictors for QDA, we would get error messages that "some group is too small for 'qda' ". A possible reason for this might be that since one of the fundamental differences between LDA and QDA is that we in QDA assume class-specific covariance matrices. The number of entries in each of these matrices is equal to the number of predictors squared, and this might be very hard, if not impossible, in classes with few samples. Because the number of countries in each region in our dataset is not equally distributed, this might be especially problematic for regions with few countries, such as North America.

In order to overcome this problem we have limited ourselves to 2 parameters, which was also used in KNN. Albeit we have to choose only 2 parameters now, there are still more than `15*14` possible combinations, so we have again used k-fold cross validation in order to choose this pair of predictors. In order to do this we have first implemented two functions, where the first makes a prediction for a LDA model and a test set, and another one to calculate the misclassification rate for a pair of two predictors. In the last function, a LDA

model is built using the pair of predictors, using the fold from the cross validation, and the misclassification rate is then calculated using `lda_pred_and_error` on the left-out folds.

```
# Useful functions. It's called da because it's used on both LDA and QDA models

da_pred_and_error <- function(da_model, testing_set, show_table = FALSE) {
    da_region_pred <- predict(da_model, newdata = testing_set)$class
    da_confusion_mat <- table(da_region_pred, testing_set$Region)
    da_misclass_rate <- 1 - sum(diag(da_confusion_mat))/length(da_region_pred)
    if (show_table) {
        da_confusion_mat
    }

    return(list(da_region_pred, da_misclass_rate))
}

misc_rate <- function(pred1, pred2, fold, remaining) {
    # First we make a reduced dataframe with only the current predictors and
    # the response for the current fold
    reduced_fold <- cbind(fold$Region, fold[pred1], fold[pred2])
    colnames(reduced_fold)[1] <- "Region"

    # Doing the same for the remaining folds, so we have a 'test' set for the
    # CV
    reduced_remaining <- cbind(remaining$Region, remaining[pred1], remaining[pred2])
    colnames(reduced_remaining)[1] <- "Region"

    # Below we fit to all predictors because only the one qw want to test are
    # included in the dataframe. scale ensures that the variables have mean
    # zero and are normalized
    lda_region <- lda(Region ~ ., data = reduced_fold, scale = TRUE)

    # Below we make a prediction on the left out folds, and return the
    # misclassification rate

    misclassifcation_rate <- unlist(da_pred_and_error(lda_region, reduced_remaining)[2])
    return(misclassifcation_rate)
}
```

Then we do the actual cross validation, with 5 folds.

```
# Removal of variables we don't want to consider as predictors
col_names <- names(df_train)
col_names <- col_names[which(col_names != "Region")]
col_names <- col_names[which(col_names != "Economy_status_Developed")]
col_names <- col_names[which(col_names != "Life_expectancy")]

lowest_misclassification <- 1
best_pair <- c(0, 0)
length(col_names)
```

```
## [1] 15
```

```
# Below we iterate over all possible combinations of predictors
for (i in 1:(length(col_names) - 1)) {
    for (j in (i + 1):length(col_names)) {
        K <- 5
        misclassifications <- rep(0, K)  #Array for the misclassification rate
        # for each fold

        folds <- sample(1:K, dim(df_train)[1], replace = TRUE)

        for (k in 1:K) {
            misclassifications[k] <- misc_rate(col_names[i], col_names[j], df_train[folds ==
                k, ], df_train[folds != k, ])
        }

        mean_misclass <- mean(misclassifications)

        if (mean_misclass < lowest_misclassification) {
            lowest_misclassification <- mean_misclass
            best_pair[1] <- col_names[i]
            best_pair[2] <- col_names[j]
        }
    }
}
```

Having found our pair of predictors, we build our LDA model and make predictions in the chunk below. We standardize our variables, so that differences in scales hopefully is not too influential among our predictors. Later, we make a QDA model for the same pair of predictors, so we do not actually perform CV for feature selection for QDA. The reason for this is another, but similar to the previously presented, problem we ran into when we attempted this. When we tried 3-fold CV for instance, we again got the error message about too small groups, and this might be that because when we build our model, we now only use a third of the our training set, and we might again have too few occurrences of some regions in order to find meaningful necessary estimates.

```
# LDA for best pair of predictors from CV

lda_region <- lda(Region ~ Under_five_deaths + Thinness_five_nine_years, data = df_train,
    scale = TRUE)

pred_and_error <- da_pred_and_error(lda_region, df_test)
lda_region_pred <- pred_and_error[[1]]
lda_misclass_rate1 <- pred_and_error[[2]]

df_test_with_preds <- data.frame(Adult_mortality = df_test$Adult_mortality, Thinness_ten_nineteen_years
    Region = df_test$Region, Region_pred = lda_region_pred)

match <- df_test_with_preds$Region == df_test_with_preds$Region_pred
df_test_with_preds$match <- match

df_test_with_preds$Adult_mortality <- scale(as.numeric(df_test_with_preds$Adult_mortality))
df_test_with_preds$Thinness_ten_nineteen_years <- scale(as.numeric(df_test_with_preds$Thinness_ten_nine

lda_plot <- ggplot(df_test_with_preds, aes(x = Adult_mortality, y = Thinness_ten_nineteen_years,
    colour = Region_pred)) + geom_point(aes(shape = match)) + scale_colour_discrete(name = "Predicted R
    scale_shape_discrete(name = "Correct Prediction") + ggtitle("LDA with predictors from CV")
```

The above block is essentially repated three more times, for two QDA models and one LDA model, and the only line that changes is the first, and some details in the plotting. We have therefore omitted this code from the report, but inform which predictors were used. For the first QDA model we used - `Thinness_five_nine_years` and `Schooling`, which were the preferred predictors from the CV done on KNN and for the other we used `Thinness_ten_nineteen_years` and `Adult_mortality`. We have also fitted a LDA model which makes use of all the predictors in the same way as the way presented above. The plots `qda_plot1` and `qda_plot2` are produced along with the corresponding misclassification rates, also for the full LDA model, and these will be presented in the section regarding results.

Support Vector Machines

Finally, we look into using support vector machines to predict the regions in the dataset. This is done by using the `e1071` library, where `svm()` uses a one vs. one extension of SVM for multiclass classification. This leads to less interpretability as we have many predictors, however it might lead to a good prediction.

We have also used cross validation to determine the parameters `cost` and `gamma`. The cost parameter controls the margins and the number of support vectors, while gamma influences the shape of the decision boundary. We tried scaling the data, however this gave worse results, and we have therefore chosen to present the results with non-scaled data.

```
set.seed(1234)
svm.cv <- tune(svm, Region ~ ., data = df.factors.train, type = "C", kernel = "radial",
    ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100, 1000), gamma = c(0.001,
        0.01, 0.1, 1, 10, 100, 1000)))

cost = svm.cv$best.parameters$cost
gamma = svm.cv$best.parameters$gamma

svm.model <- svm(Region ~ ., data = df.factors.train, type = "C", kernel = "radial",
    gamma = gamma, cost = cost)

preds <- predict(svm.model, df.factors.test)
confMat <- confusionMatrix(preds, df.factors.test$Region)
confMatSummary <- summary(confMat)

svm_misclass_rate <- 1 - as.numeric(confMat$overall["Accuracy"])
```

Results and Interpretation

In this section we present the results of our projects, mainly for our regression problem regarding the life expectancy, and then for the classification problems regarding the region.

Regression Results

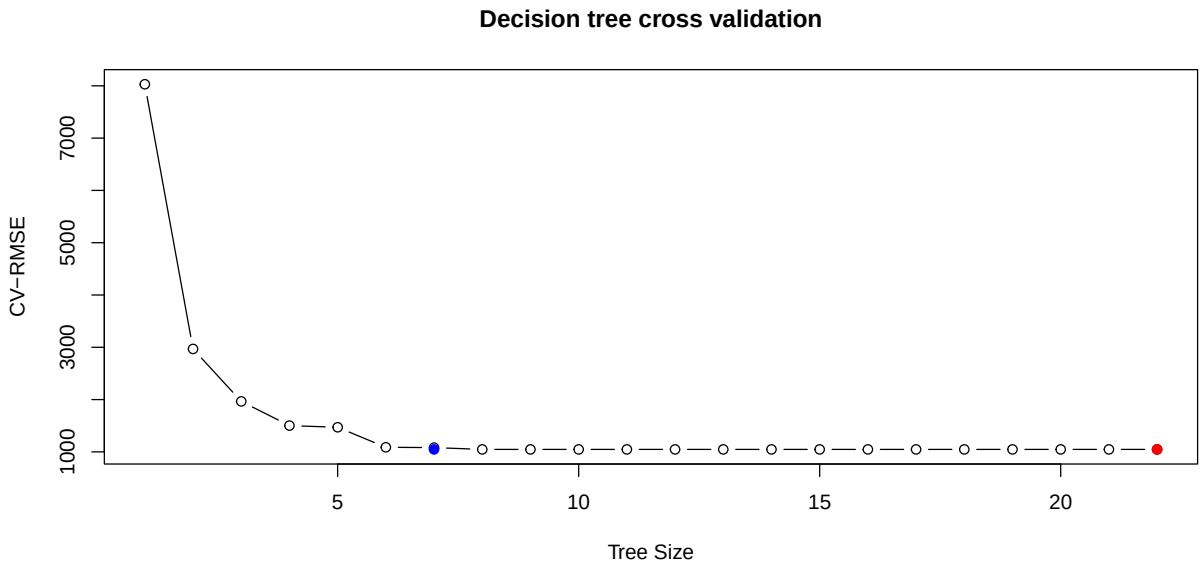
We used two different sets to perform linear regression. The full model gives a MSE of `full_lin_MSE` and the subset from best subset selection gives a MSE of `best_lin_MSE`. We see that the full model has a lower MSE, which is expected as it is the larger model, but considering that the best subset uses less than a quarter of the covariates, the increase in MSE is quite low. So the best model for predictions is the full model, but if we account for complexity, we would choose the reduced model. This is also supported by the anova test, where our F-value is relatively large, which indicates that the reduced model performs almost as well as the full model.

Model	Misclassification rate
Linear model, all predictors	1.6531088
Linear model, subset selection	1.849881
Decision tree	7.5972583
Random forest	2.3966529

Decision trees and random forests

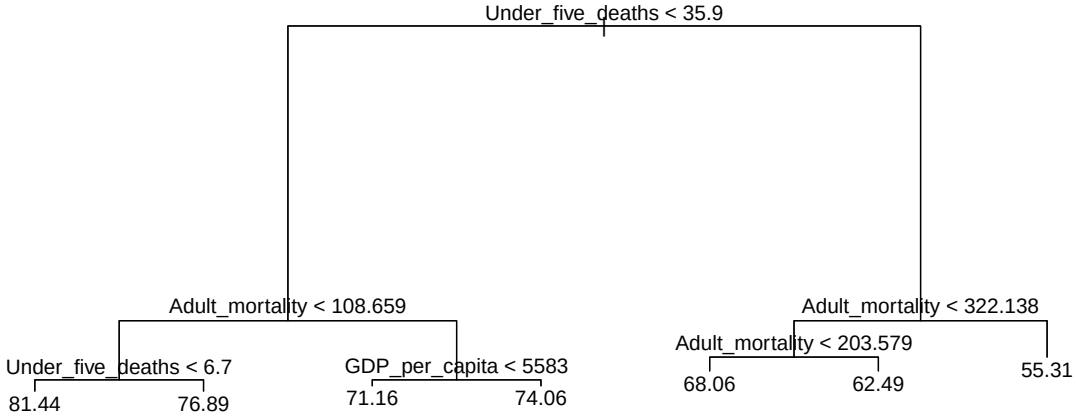
The first tree was grown by passing the arguments `mincut=2`, `minsize = 4`, `mindev=0.001`. This results in a very "bushy" tree with a lot of cuts, which was then pruned.

From the cross validation we end up with the following plot of the root-mean-square error against tree sizes:



From the plot above, we see that the tree size that yields the lowest root-mean-square error is 22, however we have chosen to use 7 as it has a RMSE value very close to 22, but yields a simpler and more interpretable method.

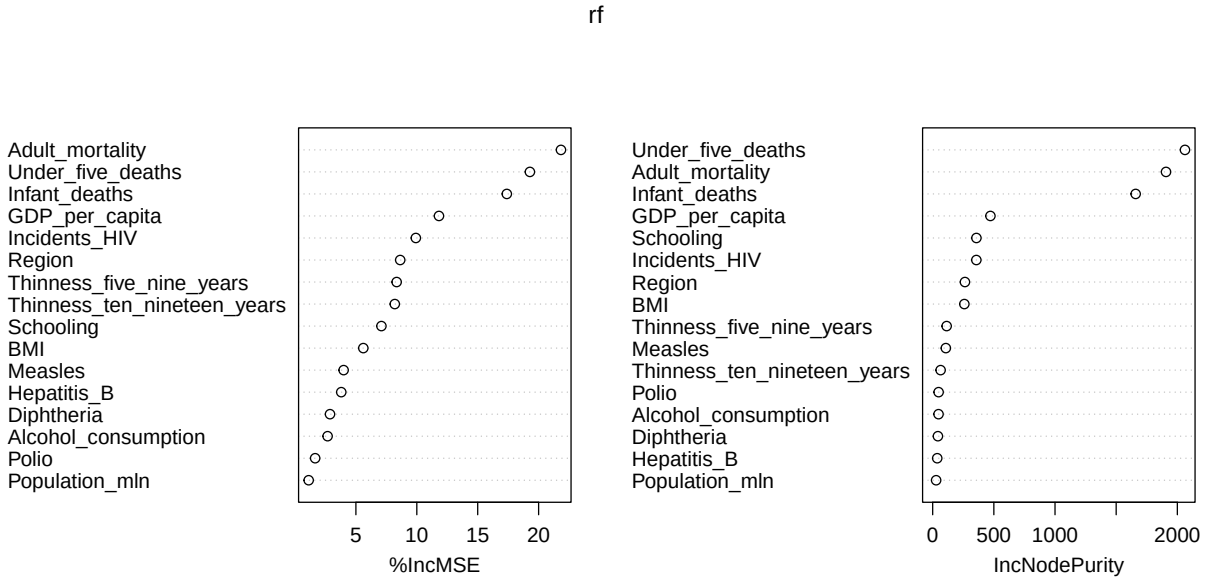
Using a tree size of 7 yields the following decision tree:



Testing the performance against the test set, we ended up with an MSE of 7.5972583. This is worse than what we achieved with the other methods, but the decision tree might still be useful for interpretability.

From the random forest with trees, we achieved an MSE of 2.3966529. This is better than the MSE we achieved from the decision tree, which is to be expected, and is more comparable with the MSE we got for the linear models.

Below is the Variance importance plot that is generated from the random forest:



From the plot of the decision tree, we can see that the variables `Under_five_deaths` and `Adult_mortality` are the most important for predicting life expectancy, according to this model. This is also reflected by the variance importance plot of the random forest. Here we again see that `Adult_mortality` and

`Under_five_deaths` are the most important predictors, according to both mean decrease accuracy and IncNodePurity.

Classification Results

Model	Misclassification rate
KNN with Thinness (5-9) and Schooling	0.4222222
QDA with Thinness (5-9) and Schooling	0.6
LDA with Thinness (10-19) and Adult Mortality	0.4222222
QDA with Thinness (10-19) and Adult Mortality	0.5111111
LDA with all predictors	0.4444444
SVM with all predictors	0.2

KNN

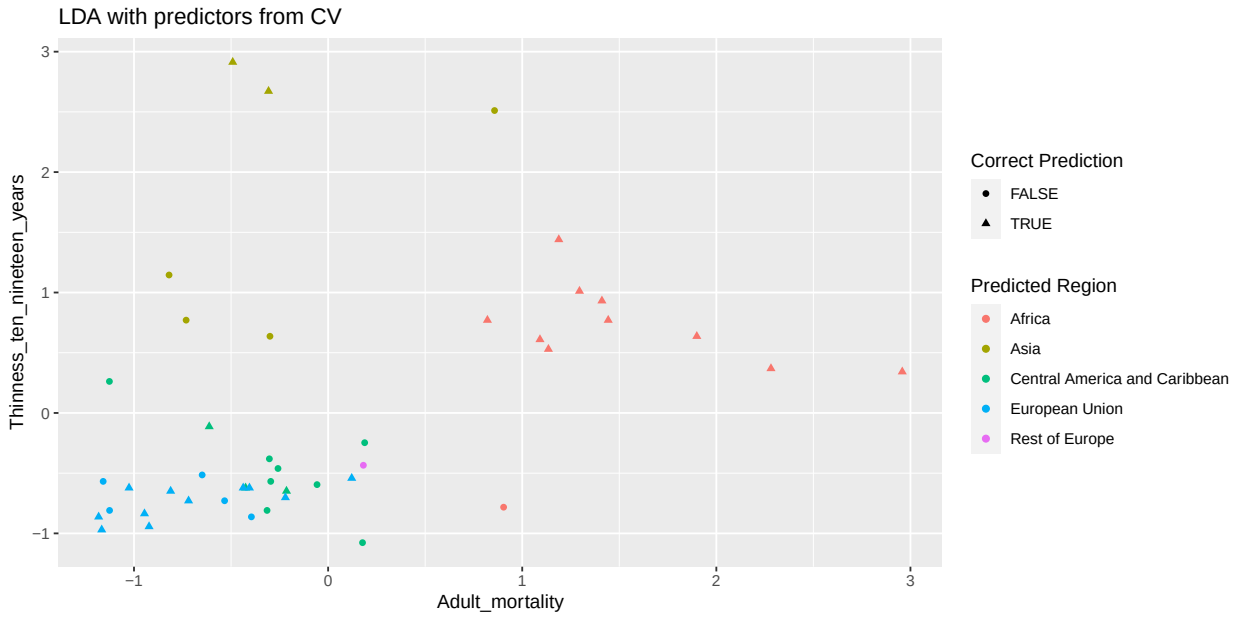
When we fix K to 10, we see that Thinness_five_nine_years and Schooling give the best predictions, according to 4-fold CV. When we apply KNN, with this subset as predictors, to our training data we obtain a misclassification rate of 42.2%. This is a somewhat high misclassification rate, but compared to other models with the same number of predictors its not bad. (Noe om at det kanskje ikke er mulig å klassifisere alle rett pga flere av regionene ligner) For instance it matches LDA and outperforms QDA, when all have 2 covariates. However, increasing the number of parameters for LDA makes it noticeably better than KNN. When comparing the models its important to remember that only looking at misclassification rate could be misleading because of the varying sizes of the regions.

Discriminant Analysis

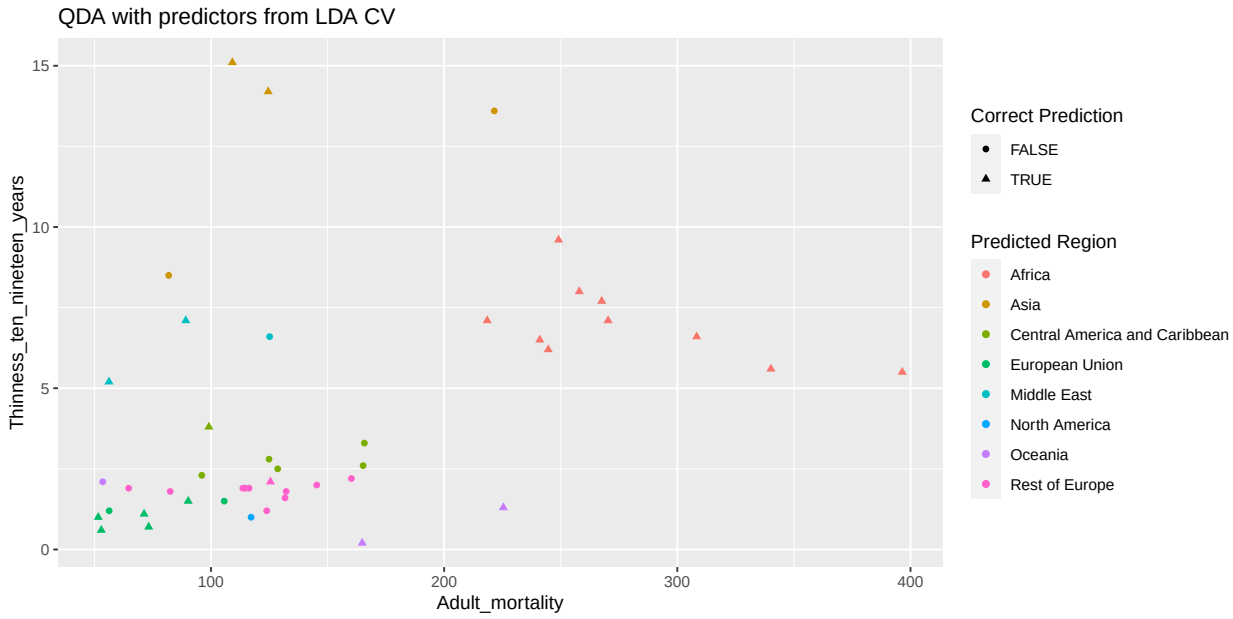
For this class of algorithms, we have generally experienced lower misclassification rates for LDA compared to QDA and and a bit lower than for KNN. A possible reason for this might be that the true decision boundaries between the different lines are somewhat linear. We also note that the LDA model with all predictors actually has a higher misclassification rate than the one with only two chosen through CV. Another disadvantage with our full model is that it is hard to visualize the decision boundaries because of the number of predictors. This illustrates how important feature selection is, and that increasing the number of predictors will not necessarily improve a model's performance, in some cases it might even lead to overfitting our data.

On the other hand our LDA model and QDA models built with predictors found through CV does not face any problems when it comes to visualization, and they can be visualized as follows.

```
print(lda_plot)
```

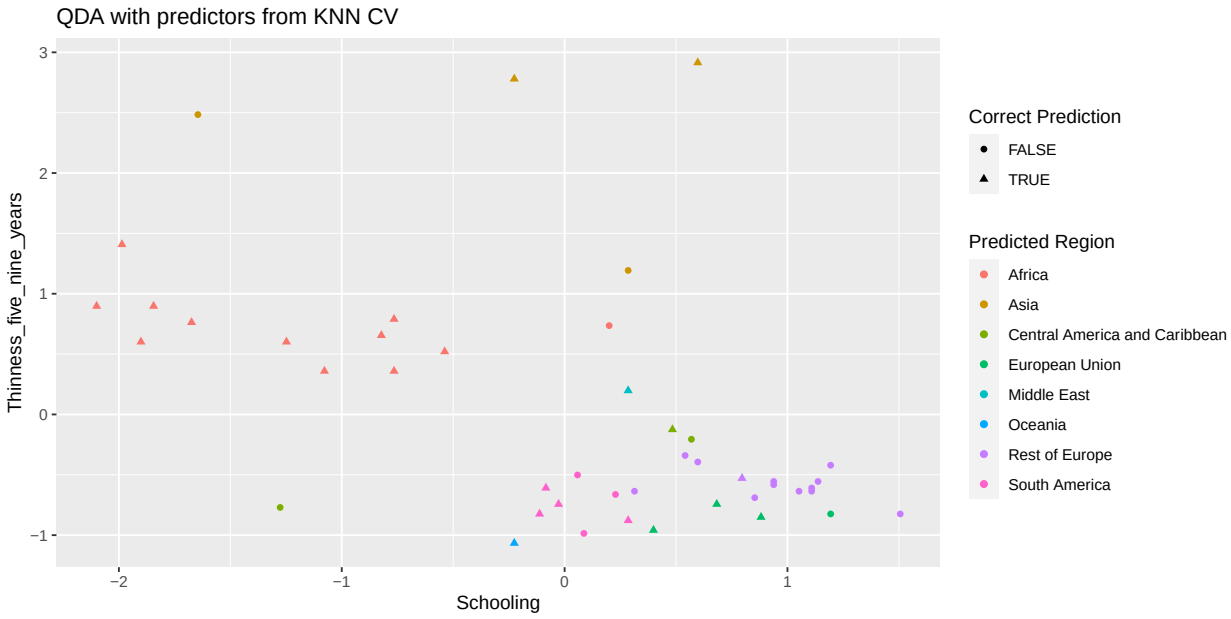


We observe that the model classifies most of the countries in Africa correctly, with one misclassifcation, and that it gets a little cluster of countries in the European Union in the bottom left corner. However, it is difficult to identify the decision boundaries, and one reason for this model's good performance might be a high prevalence of African countries both in the training and testing set. Below we consider a QDA model, built with the same pair of predictors.



We observe that this model also several African countries and countries in the EU. In fact, it also classifies two countries in the middle bottom correctly as Oceania, and this is the highest score for our discriminant analysis models. Another interesting property of this plot is that the variables are no longer scaled, but compared to the plot for the LDA model with standardized variables, we see that the placement of most points are rather similar. This can also serve as an explanation as to why standardization does not necessarily

improve our test performance, as the scales of the predictors are similar to begin with. Lastly we also consider a QDA model built with the predictors previously used in KNN.



Again, this model works quite well for African countries, but it differs from the others by the runner-up when it comes to number of correct classifications, which in this case is South American countries. This is also the model that performs the worst when it comes to misclassification rate of the DA models.

Another theory is that the performance of QDA is limited by our dataset. As previously mentioned, the combination of the number of predictors and our sample size might have prevented optimal performance. A possible approach to resolve this issue might be to group countries from different regions into new regions of larger size, but this would of course require some effort. This is because there does not necessarily exist a straightforward way to group the regions together, but one could for instance try an unsupervised method, such as clustering.

A common trait for most of the methods involving discriminant analysis is that the mortality among adults and thinness among youths, between either the ages 5 and 9 or 10 and 19, are useful predictors for the region. This is also in accordance with our descriptive data analysis, especially the box plot for Adult mortality in the upper row, which represents the distribution for African countries. Here we see that the even the 25th percentile is higher than most outliers and percentiles than in the box plots for the other regions. Therefore, it seems likely that countries with adult mortality higher than this, should be classified as African.

Support vector machines

From the cross validation, we ended up using the parameters cost=10 and gamma = 0.1. This gave a misclassification rate of 0.2, which is the lowest misclassification error discussed so far. However, we lose a lot in terms of interpretability. As we have used all predictors, we cannot draw the decision boundary.

Summary

To summarize, it is possible to predict Life_expectancy quite well with the given data set. We predicted this with a full and a reduced model, and concluded that the reduced model is most useful when considering

complexity and precision. Predicting the regions proved somewhat tougher, and most of the algorithms got a misclassification rate above 40. In general, the restriction to two covariates was likely a contributing factor, but also the inherit overlap between regions played its part. Even our best models only reached a misclassification rate of 20, but considering that we have 9 different regions, this is a decent result. Regarding inference, we have also noted that when restricting the number of predictors, the mortality of adults and thinness among youths have proved themselves as useful.