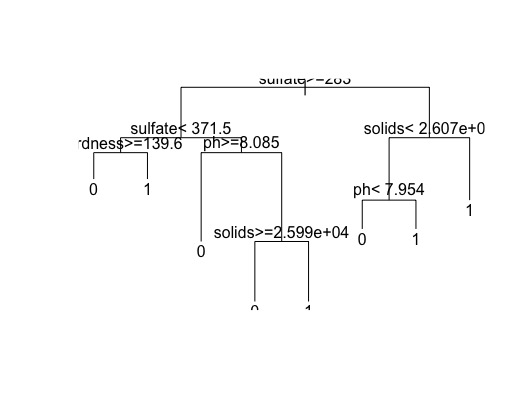
*Final Project*

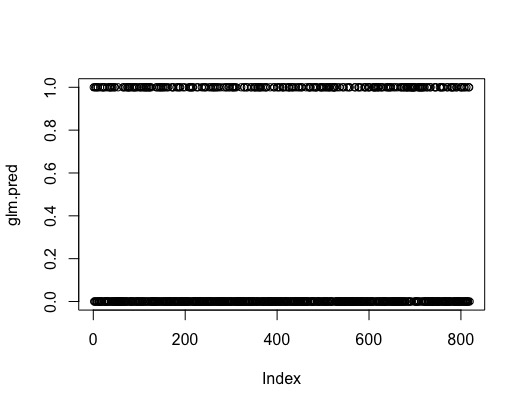
The waterData dataset contains features describing quality of water samples. The target response for this dataset is a categorical variable which labels whether a water sample is potable. The dimension of this dataset is 3276 rows by 11 columns. This dataset originally comes with 13 columns, however the last 2 columns have no data included, so these attributes can be ignored. This dataset contains various missing data points in the 'pH', 'Sulfate', and 'Trihalomethanes' features, without pattern. The features describe traits of a water sample. The features are all numerical so encoding will not be necessary. The target variable is labeled '1' for a potable sample and '0' for a non-potable sample. The ratio of '1':'0' for this data sample is 1278:1998, which simplifies to a 39:61 split. Finding features most predictive to the target will involve checking for significance values within the model that best fits this dataset. The columns are renamed to all lowercase values for brevity and the target variable is converted from an integer class to a factor for further analysis. The objective is to predict the potability of water samples.

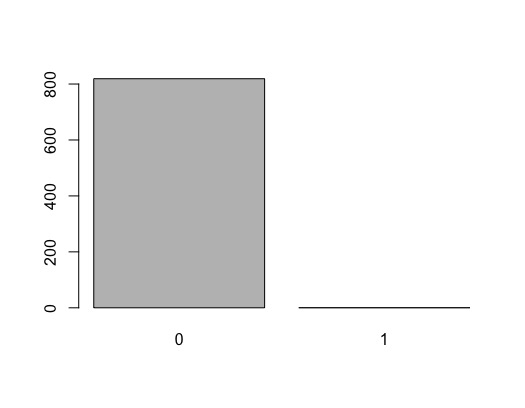
For logistic regression, the regression coefficients must be calculated. When the coefficients are calculated, we are looking for a large z-score to be able to determine if a predictor actually affects the target. This will mean finding the z-score coefficients of all of the 10 numerical variables in this dataset. Plugging in the coefficients and utilizing the base 'predict' function in R will output the predicted logistic odds of each entry. The test set is processed through the logistic regression model and these predictions for the test set will be given. We then compare the model results with the true results of the test set to be given a proportion of correct predictions. Linear discriminant analysis will take each predictor and model the response variable based on that single predictor. From there it creates a prediction in classification similar to logistic regression, which the correctness of the predictions being compared to the actual test set results. Quadratic discriminant analysis is a similar method to LDA, however, the Bayes classifier is a quadratic instead of linear. K-nearest neighbors method is highly flexible and excels when the decision boundary of the data is completely non linear. Finding the best model that fits for this data will give an indication as to how the decision boundary for this dataset looks like.

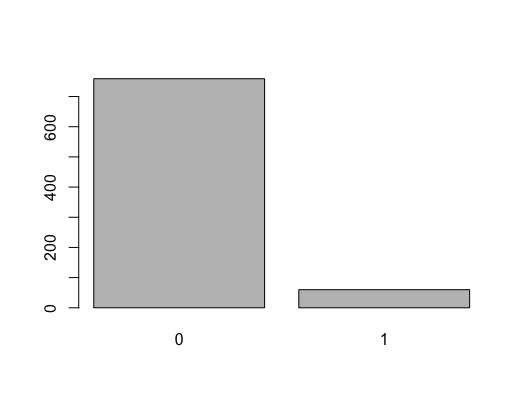
The lifeData dataset contains features which describe quality life factors that may influence the life expectancy in each country over multiple years. The target variable in this dataset is the 'life.expectancy' variable while 19 predictors for this dataset are used. The dimensions of this data is 2938 rows by 21 columns after feature engineering. The reason 19 variables are used for the bulk of the analysis is to discount the 'status' variable which is a factor, while the rest of the predictors are numeric. The 'status' variable has been recoded so that the 'Developing' value is assigned a 0 and the 'Developed' level is assigned a 1. The country value is removed from the data set due to the possibility of having too many levels within this predictor without a significant benefit in analysis. However, this action might increase co-linearity between observations because it is realistic to expect that a given country's quality of life metrics will not differ significantly from a yearly to year basis. The column names are also done in all lowercase with this dataset for brevity and predictor classes are converted to either factors or numerical, where appropriate. There is missing data throughout the predictors so MICE imputation is used with the method of regression trees to fill in the missing values of the data. Utilizing this method may present duplicates of values within missing cells, however the categorization of the regression trees helps lower the variance while providing sensible imputed data.

The analysis techniques applied to this dataset with a numerical target and mostly numerical predictors are regression trees, simple and multiple linear regression, cross-validation, ridge and lasso regression, PCR, PLS, and bagging and boosting. The objective of utilizing this handful of techniques is to explore which type of analysis builds stronger models for this particular dataset. This is primarily done by gauging the difference between the test set for the data and the predictions leveraged by each specific model through mean squared error (MSE). At a quick glance, the model displaying the least MSE would be the most promising model to research further and implement.

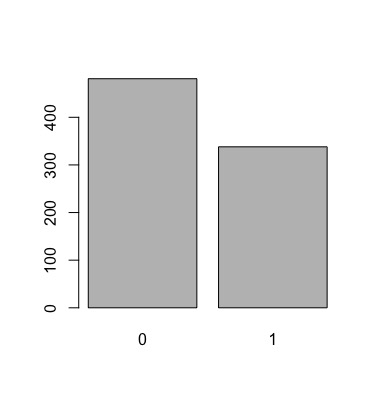
The waterData dataset uses 75% of its total observations for training and the rest for testing purposes. The first step to get an idea of how the predictors interact with the target is to make a classification tree to be able to access which predictors have a high impact on determining the degree of the target response. This classification tree sets up all of the numerical predictors and creates a tree with nodes to predict how likely an observation will fall under 0 or 1. In terms of variable importance, pH comes first followed by sulfates and solids. These three predictors have a markedly large impact as predictors than any of the other variables, based on this technique. There are 15 total nodes within this tree, with a terminal node value of 8. Comparing the tree model with the test values of potability, the model shows a near 40% MSE. This means 40% of the values predicted by the tree and placed into nodes to then further categorize among 0 and 1 are incorrect and should have been in the opposite level.

The next model moves onto finding the best subset of predictors within the nine that are provided in the dataset. Utilizing forward stepwise analysis, the model selection places the solids, organic.carbon and chloramine variables as the best predictor of the target in a multiple logistic regression model of any three variables. Using the 'glm' function for the logistic regression model, the model based on the training data when compared to the test set provides an MSE of 44.3%. An assumption is being made here that the prediction boundary between the two levels of the target is 0.4. If this boundary is to be valued at 0.5, then all of the values from the model will fall below 0.5 and would deem the model useless. The average value of the predicted likelihoods of each observation based on the model is approximately 0.4, which is why this value is used as the prediction boundary. The plot of the model itself is unremarkable, with data points falling at either 0 or 1.

Using LDA with the same three predictors above produces interesting results. The group means of each predictor within either level do not differ by a significant amount, which may imply that the variables that the forward stepwise model deemed to be the strongest among those in this dataset, may not actually be very predictive in general. Regardless, the MSE for this model comes out to 39%, which is a more encouraging figure compared to the previous models. However, this MSE is almost identical to the prior probability of level 1 of the target. The meaning behind this phenomenon is the model actually was not able to identify any level 1 observations. Thus, while it seems like the model is more reliable than the prior models, the MSE of 39% is actually fully dependent on the number of level 1 targets in the test set. In essence, the model can only identify all predictions as level 0, which gives immensely hurts the efficacy of this model. This model performs the worst so far, as it is necessary to look beyond the MSE to determine the potential of a model. In this case, the model has failed to implement a linear decision boundary.

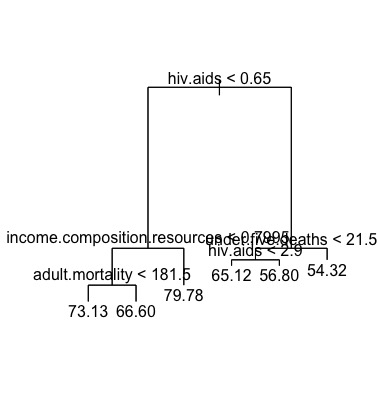
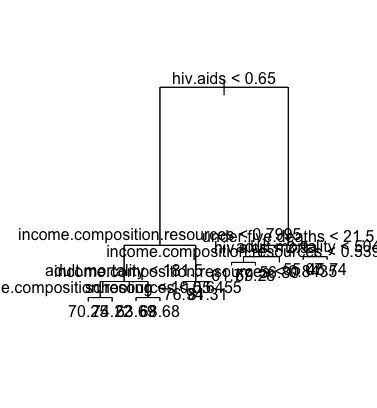
The QDA model is similar to the LDA model, but it performs better if there is a non-linear decision boundary. When the results of this model are compared to the test set, it performs better than the LDA model with an MSE of 38.8%. This figure might suggest that the prediction boundary is non-linear, yet a similar caveat applies for this model, as it may just be much better at predicting level 0 targets while failing to identify level 1 targets. This drawback is not as pronounced in QDA as it is in LDA, but could still pose a potential pitfall for model predictive analysis.

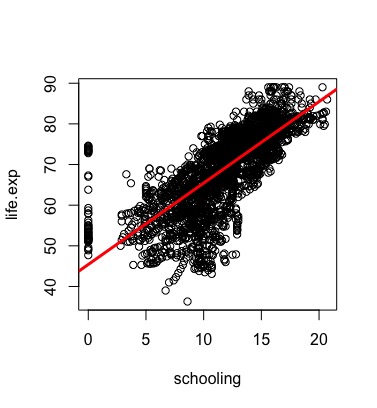
K-nearest neighbors is a model in which the k value is a parameter that can be set within the model. When k is greater, the decision boundary mimic that of an LDA. When k is of a lesser value, the more non-linear its decision boundary becomes.

Setting k = 100 gives a result very similar to the LDA, as expected. The MSE is 38.9%, which again is very similar to the prevalence of the level 1 target in the test set. The table confirms that the model identified 812 of the 819 test observations to be of level 0, while only identified 7 observations as level 1 targets.

This time setting the k value to 1 gives a more encouraging result, to a degree. The MSE for this parameter of k is 48.4%, which is not ideal, but this is a telltale sign that the model is not automatically defaulting to one level over the other. The precision of the model is still underwhelming so this model presents limited capability.

The classification tree provides the best predictive power out of the methods used for this dataset in predicting the potability of water based on the predictors given.

For the lifeData dataset, the objective is to find the best model to predict life expectancy. A regression tree would be an appropriate fit. The training set for this model is 50% of the total dataset, with the testing set being the leftover 50%. The variables used in the tree are hiv.aids, income.composition.resources, adult.mortality, schooling, and under.five.deaths. There are 11 terminal nodes assigned to this tree. The outline of this tree places great importance on the hiv.aids predictor, as it is the initial node by which observations are divided. From there, the two nodes it branches into are income.composition.resources and under.five.deaths. The former branch receives observations that fall under a threshold for hiv.aids, while the latter receives the observations that fall above the same threshold. This divide already creates a clear picture for how the observations are being processed by the tree. As expected the branch which starts with under.five.deaths ends in terminal nodes which predict a relatively short like expectancy, while the other branch buckets observations in group of general higher life expectancies. Pruning this tree presents an even more stark image of how the tree performs. The tree best performs with 11 terminal nodes, based on the cross-validation analysis, but when the tree reaches 6 terminal nodes its when the error rate levels off. A 6 terminal node pruned tree performs in such a way that all of the life expectancy values in the one branch is greater than all of the target values in the other branch. This suggests a clear partition between regression grouping and makes the hiv.aids predictor valuable as the initial node. The model performs with an MSE of 13.9%, which is encouraging.

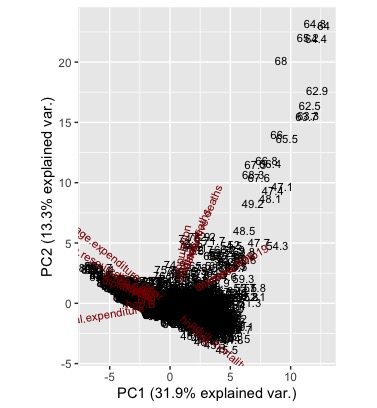
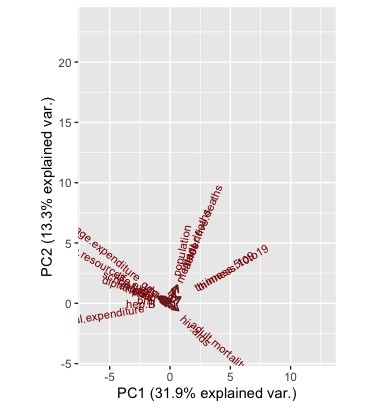
Similarly to the forward stepwise process for the waterData dataset, this technique can be used to isolate a single predictor which best performs in predicting the target over the other predictors. For this dataset, that predictor is determined to be the schooling variable. The linear model displays an intercept value of 45.41 and a predictor value of 2. This is interpreted as when the when the schooling value increases by 2 units, the life.expectancy value increases by 1 unit, which is then added to 45.41 units of the target. The plot with predictor on the x-axis and target on the y-axis confirm a fairly strong prediction value. The MSE for this model comes out to 42.3%.

There are 19 predictors to work with in this dataset, so the first task to perform a multiple linear regression is to determine how many variables would form the best performing model. Using various measures such as RSS, adjusted R squared, CP and BIC output different answers for how many predictors would work best. Here we can choose to utilize 15 variables, which is the output produced by the adjusted R squared method.

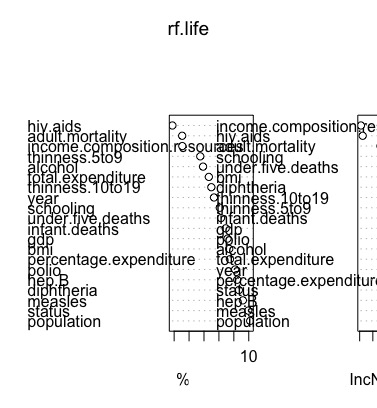
Ridge regression analysis is primarily used when the risk of co-linearity between the predictors can be a concern. The objective of ridge regression is to account for co-linearity by adding bias to a model. According to the concept of bias-variance trade-off, increasing the bias in this model should mitigate to a degree the large amount of variance that can occur between predictors that are not independent of each other. The ridge regression gives a similar output as a linear regression, but balances the coefficients of highly correlated predictors that linear regression cannot account for. The MSE for a ridge regression is highly dependent on the penalty factor lambda. We can determine the best value for lambda by using cross-validation with the 'glmnet' function on the x and y dimensions of the training set to find the lowest lambda value, which will give us the best possible MSE value for the ridge regression. The best lambda value comes out to 0.68, which leads to an MSE of 17.6%.

The lasso regression is very similar to the ridge regression, except the 'alpha' value is set to 1. This minimizes coefficients even more and leads to an MSE value of 16.9%. Of the two similar models, it seems like the lasso performs slightly better than the ridge.

The PCA for this dataset yields a 19.5% MSE. The most interesting part of this method is analyzing the components. The PCA gives this dataset 20 components, only reaching 80% accountability in variance at PC8. PC1 accounts for 32% of the variance in the target and PC2 accounts for 13% of the variance. Based on the plots, we can venture that PC1 is likely variables that developed countries are strong in, such as percentage.expenditure, schooling, and income.composition.resource. PC2 is more clearly made of predictors that developing countries would rank highly in, such as under.five.deaths and hiv.aids, which we had established with the regression tree that it is a very significant predictor for the target.

Partial least squares (PLS) works similarly to PCA. For PLS, an 80% variance is reached at PC3. And the MSE is 18.6%.

For bagging, the objective is to sample multiple subsets of the dataset with replacement. Therefore, an observation can be chosen more than once in the analysis. Bagging is composed of random forest, which is a group of trees that have their results averaged to output a result with less variance than a single tree. 500 trees are used for this method with all 19 variables being available at each split in the trees. With 500 trees, the bagging method produces an MSE of 4.7%, and with a forest of 25 trees the MSE increases to 5.0%. If the number fo predictors available for use in each split is trimmed down to 6, with a forest of 500 trees the MSE is reduces to 4.6%, which performs better than the forest with all 19 variables available for splitting. The importance plot for this iteration of random forest confirms that hiv.aids is extremely important as a predictor with the highest effect in MSE. IN terms of node purities, income.composition.resources tops hiv.aids as the more useful variable in terms of affecting variance, but %IncMSE should be given more importance. Boosting provides the lowest MSE out of the methods discussed for this dataset. Boosting provides a similar MSE with 4.4% with a random forest of 5000 trees. A random sample of data being selected and trained as the next model in the sequence adjusts its weights until a best result is reached. Boosting seems like the correct choice for the best model, both evidently and intuitively.



APPENDIX CODE

library(devtools)

library(tidyverse)

library(mice)

library(naniar)

library(leaps)

library(glmnet)

library(MASS)

library(tree)

library(rpart)

library(scales)

install\_github("vqv/ggbiplot")

library(ggbiplot)

lifeData <- read.csv("Life Expectancy Data.csv")

lifeData$Status <- recode(lifeData$Status, Developing = 0, Developed = 1)

### tidying lifeData

## cleaning column names

lifeData <- lifeData %>%

dplyr::select(-Country) %>%

dplyr::rename(

year = Year,

status = Status,

life.exp = Life.expectancy,

adult.mortality = Adult.Mortality,

alcohol = Alcohol,

hep.B = Hepatitis.B,

measles = Measles,

bmi = BMI,

polio = Polio,

total.expenditure = Total.expenditure,

diphtheria = Diphtheria,

hiv.aids = HIV.AIDS,

gdp = GDP,

population = Population,

thinness.10to19 = thinness..1.19.years,

thinness.5to9 = thinness.5.9.years,

income.composition.resources = Income.composition.of.resources,

schooling = Schooling) %>%

mutate(

status = as.factor(status),

year = as.numeric(year),

infant.deaths = as.numeric(infant.deaths),

measles = as.numeric(measles),

under.five.deaths = as.numeric(under.five.deaths)

)

## checking NAs

sapply(lifeData, function(x) sum(is.na(x)))

## MICE imputation

set.seed(25)

life.pred.mat <- quickpred(lifeData)

life.impute <- mice(lifeData, m = 5, meth = "cart", seed = 5, predictorMatrix = life.pred.mat)

lifeData2 <- complete(life.impute, 1)

lifeData <- lifeData2

sapply(lifeData, function(x) sum(is.na(x)))

## data diligence

pairs(lifeData)

summary(lifeData)

sapply(lifeData, class)

## regression tree

set.seed(25)

train.tree <- sample(1:nrow(lifeData), nrow(lifeData)/2)

tree.life <- tree(life.exp ~ ., lifeData, subset = train.tree)

summary(tree.life)

plot(tree.life)

text(tree.life, pretty = 0)

cv.life <- cv.tree(tree.life)

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

prune.life <- prune.tree(tree.life, best = 6)

plot(prune.life)

text(prune.life ,pretty = 0)

yhat <- predict(tree.life, newdata = lifeData[-train.tree,])

life.test <- lifeData[-train.tree,"life.exp"]

plot(yhat, life.test)

abline(0,1)

mean((yhat - life.test)^2)

## simple linear regression

# best subset selection for simple linear regression

regfit.fwd <- regsubsets(life.exp ~ ., data = lifeData, nvmax = 19, method = "forward")

summary(regfit.fwd)

# According to the forward selection algorithm, the variable 'schooling' is the

# best predictor to use in a simple linear regression

lm.fit <- lm(life.exp ~ schooling, data = lifeData)

lm.fit

attach(lifeData)

plot(schooling, life.exp)

abline(lm.fit, lwd = 3, col = 'red')

par(mfrow = c(2, 2))

plot(lm.fit)

# check non linearity

plot(predict(lm.fit), residuals(lm.fit))

plot(predict(lm.fit), rstudent(lm.fit))

# leverage stats

plot(hatvalues(lm.fit))

which.max(hatvalues(lm.fit))

lm.pred <- predict(lm.fit, newdata = lifeData[-train.tree,])

mean((lm.pred - life.test)^2)

## multiple linear regression

# best subset selection for multiple linear regression

regfit.full <- regsubsets(life.exp ~ ., lifeData, nvmax = 19)

summary(regfit.full)

reg.summary <- summary(regfit.full)

names(reg.summary)

reg.summary$rsq

par(mfrow=c(2,2))

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

which.min(reg.summary$rss)

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

which.max(reg.summary$adjr2)

points(15, reg.summary$adjr2[15], col="red", cex = 2, pch = 20)

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

which.min(reg.summary$cp)

points(14, reg.summary$cp[14], col = "red", cex = 2, pch = 20)

which.min(reg.summary$bic )

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

points(12, reg.summary$bic[12], 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, 15)

# best model is with 15 predictors

# Choosing Among Models Using the Validation Set Approach and Cross-Validation

set.seed(25)

train <- sample(c(TRUE,FALSE), nrow(lifeData), rep = TRUE)

test <- (!train)

regfit.best <- regsubsets(life.exp ~ ., data = lifeData[train,], nvmax = 19)

test.mat <- model.matrix(life.exp ~ ., data = lifeData[test,])

k = 10

set.seed(25)

folds <- sample(1:k, nrow(lifeData), replace = TRUE)

cv.errors <- matrix(NA, k, 19, dimnames = list(NULL, paste(1:19)))

## ridge regression

x <- model.matrix(life.exp ~ ., lifeData)[,-1]

y <- lifeData$life.exp

grid <- 10^seq(10, -2, length = 100)

ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)

dim(coef(ridge.mod))

ridge.mod$lambda[50]

coef(ridge.mod)[,50]

sqrt(sum(coef(ridge.mod)[-1,50]^2))

ridge.mod$lambda[60]

coef(ridge.mod)[,60]

sqrt(sum(coef(ridge.mod)[-1,60]^2))

predict(ridge.mod, s = 50, type = "coefficients")[1:20,]

set.seed(25)

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

test <- (-train)

y.test <- y[test]

ridge.mod <- glmnet(x[train,], y[train], alpha = 0, lambda = grid)

ridge.pred <- predict(ridge.mod, s = 4, newx = x[test,])

mean((ridge.pred-y.test)^2)

mean((mean(y[train])-y.test)^2)

ridge.pred <-predict(ridge.mod,s=1e10,newx=x[test,])

mean((ridge.pred-y.test)^2)

ridge.pred <- predict(ridge.mod,s=0,newx=x[test,],exact=T,x=x[train,],y=y[train])

mean((ridge.pred-y.test)^2)

lm(y ~ x, subset = train)

predict(ridge.mod, s = 0, exact = T, type = "coefficients", x = x[train,], y = y[train])[1:20,]

set.seed(25)

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

plot(cv.out)

bestlam <- cv.out$lambda.min

bestlam

ridge.pred <- predict(ridge.mod,s = bestlam,newx = x[test,])

mean((ridge.pred-y.test)^2)

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

predict(out,type = "coefficients",s = bestlam)[1:20,]

## lasso

lasso.mod <- glmnet(x[train,],y[train],alpha = 1,lambda = grid)

plot(lasso.mod)

set.seed(25)

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

plot(cv.out)

bestlam <- cv.out$lambda.min

lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test,])

mean((lasso.pred-y.test)^2)

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

lasso.coef <- predict(out,type = "coefficients",s = bestlam)[1:20,]

lasso.coef

lasso.coef[lasso.coef!=0]

## PCR

library(pls)

set.seed(24)

pcr.fit <- pcr(life.exp ~., data = lifeData , scale = TRUE, validation = "CV")

summary(pcr.fit)

validationplot(pcr.fit, val.type = "MSEP")

set.seed(25)

pcr.fit <- pcr(life.exp ~., data= lifeData, subset = train, scale = TRUE, validation = "CV")

validationplot(pcr.fit, val.type = "MSEP")

pcr.pred <- predict(pcr.fit, x[test,], ncomp = 7)

mean((pcr.pred-y.test)^2)

pcr.fit <- pcr(y~x, scale = TRUE, ncomp = 7)

summary(pcr.fit)

life.pca <- prcomp(lifeData[, -2], center = TRUE, scale. = TRUE)

summary(life.pca)

ggbiplot(life.pca, alpha = 0, scale = 0)

ggbiplot(life.pca, scale = 0, labels = life.exp)

## PLS

set.seed(25)

pls.fit <- plsr(life.exp ~., data = lifeData, subset = train, scale = TRUE, validation = "CV")

summary(pls.fit)

validationplot(pls.fit, val.type = "MSEP")

pls.pred <- predict(pls.fit, x[test,], ncomp = 2)

mean((pls.pred-y.test)^2)

pls.fit <- plsr(life.exp ~., data = lifeData, scale = TRUE, ncomp = 2)

summary(pls.fit)

## bagging

library(randomForest)

set.seed(25)

bag.life <- randomForest(life.exp ~ .,data = lifeData,subset = train, mtry = 19, importance = TRUE)

bag.life

yhat.bag <- predict(bag.life, newdata = lifeData[-train,])

plot(yhat.bag, life.test)

abline(0,1)

mean((yhat.bag - life.test)^2)

bag.life <- randomForest(life.exp ~ .,data = lifeData, subset = train, mtry = 19, ntree = 25)

yhat.bag <- predict(bag.life, newdata = lifeData[-train,])

mean((yhat.bag - life.test)^2)

set.seed(25)

rf.life <- randomForest(life.exp ~. ,data = lifeData, subset = train ,mtry = 6, importance = TRUE)

yhat.rf <- predict(rf.life, newdata = lifeData[-train,])

mean((yhat.rf - life.test)^2)

importance(rf.life)

varImpPlot(rf.life)

## boosting

library(gbm)

set.seed(25)

boost.life <- gbm(life.exp ~., data = lifeData[train,],distribution = "gaussian", n.trees = 5000, interaction.depth = 4)

summary(boost.life)

par(mfrow = c(1,2))

plot(boost.life, i = "hiv.aids")

plot(boost.life, i = "income.composition.resources")

yhat.boost <- predict(boost.life, newdata= lifeData[-train,], n.trees = 5000)

mean((yhat.boost - life.test)^2)

boost.life <- gbm(life.exp ~., data = lifeData[train,], distribution="gaussian", n.trees=5000, interaction.depth=4, shrinkage=0.2, verbose=F)

yhat.boost <-predict(boost.life,newdata = lifeData[-train,], n.trees = 5000)

mean((yhat.boost - life.test)^2)

library(tidyverse)

library(mice)

library(naniar)

library(leaps)

library(glmnet)

library(MASS)

library(tree)

library(rpart)

waterData <- read.csv("Drinking\_water.csv")

### tidying waterData

## cleaning column names

waterData <- waterData %>%

dplyr::rename(

hardness = Hardness,

solids = Solids,

chloramines = Chloramines,

sulfate = Sulfate,

conductivity = Conductivity,

organic.carbon = Organic\_carbon,

trihalomethanes = Trihalomethanes,

turbidity = Turbidity,

potability = Potability) %>%

mutate(

potability = as.factor(potability)) %>%

dplyr::select(c(-X, -Carcinogenics, -medical\_waste))

## handling NAs

sapply(waterData, function(x) sum(is.na(x)))

## MICE imputation

water.pred.mat <- quickpred(waterData)

water.impute <- mice(waterData, m = 5, meth = "norm", seed = 5, predictorMatrix = water.pred.mat)

waterData <- complete(water.impute, 1)

## train/test definitions

attach(waterData)

train <- sample(1:nrow(waterData), 2457)

train.water <- waterData[train, ]

test.water <- waterData[-train, ]

## classification tree

set.seed(25)

tree.water <- rpart(potability ~ ., train.water, control = rpart.control('minsplit' = 1))

summary(tree.water)

tree.water

plot(tree.water)

text(tree.water, pretty = 0)

pred.water <- predict(tree.water, test.water, type = 'class')

table.water <- table(test.water$potability, pred.water)

table.water

mean(!test.water$potability == pred.water)

## best subset selection

regfit.fwd <- regsubsets(potability ~ ., data = waterData, nvmax = 9, method = "forward")

summary(regfit.fwd)

## logistic regression

dim(test.water)

test.potable <- potability[-train]

glm.fit <- glm(potability ~ solids + organic.carbon + chloramines,

data = waterData, family = binomial, subset = train)

glm.probs <- predict(glm.fit, test.water, type = "response")

glm.pred <- rep("0", 819)

glm.pred[glm.probs > 0.4] = "1"

table(glm.pred, test.potable)

mean(glm.pred == test.potable)

plot(glm.pred)

## LDA

lda.fit <- lda(potability ~ solids + organic.carbon + chloramines,

data = waterData, subset = train)

lda.fit

lda.pred <- predict(lda.fit, test.water)

names(lda.pred)

lda.class <- lda.pred$class

table(lda.class,test.potable)

mean(lda.class == test.potable)

plot(lda.class)

sum(lda.pred$posterior[,1] >= 0.6)

sum(lda.pred$posterior[,1] < 0.6)

lda.pred$posterior[1:20,1]

lda.class[1:50]

# QDA

qda.fit <- qda(potability ~ solids + organic.carbon + chloramines,

data = waterData, subset = train)

qda.fit

qda.class <- predict(qda.fit, test.water)$class

table(qda.class, test.potable)

mean(qda.class == test.potable)

plot(qda.class)

# KNN

library(class)

train.X <- cbind(solids ,organic.carbon, chloramines)[train ,]

test.X <- cbind(solids ,organic.carbon, chloramines)[-train ,]

train.potable <- potability[train]

set.seed(25)

knn.pred <- knn(train.X, test.X, train.potable, k = 1)

table(knn.pred, test.potable)

mean(knn.pred == test.potable)

plot(knn.pred)