Comparison of classification methods applied to Zombie dataset: Tree-based methods, Logistic Regression, kNN, Linear Discriminant Analysis, Quadratic Discriminant Analysis

The data set has 200 observation points, these are people encountered randomly during a hypothetical zombie apocalypse. Using 13 predictor variables the goal of the classification task is to predict whether a person is a human or a zombie. 9 predictors are factors with 2 to 3 levels. This poses a challenge for both parametric and non-parametric methods by distributions being non-normal and observations grouped close to the origin in 9 out of 13 dimensions. The dataset was found on Kaggle and was originally posted on Datacamp as a project for logistic regression, however, an array of classification methods will be attempted here focusing on classification rather than analyzing coefficients of regression like in the original task. Since this data was designed for logistic regression specifically, it provides a peek that this classification method might perform well with this data set, but let’s see.

Below is the original description: <https://www.datacamp.com/projects/668>

It is no longer just a threat; it is now a reality! Zombies have been spotted all over the U.S. Work with your colleagues at the [Centers for Disease Control and Prevention](https://www.cdc.gov/cpr/zombie/index.htm) to identify the characteristics and supplies that seem to keep humans safe. Develop a logistic regression model that predicts the probability of becoming a zombie based on personal characteristics like age and sex, type of neighborhood people live in, and what emergency supplies are available. Practice bivariate tests like chi-squared and t-test to identify the most relevant variables for the model, develop and run the model, check model assumptions, and use the model to predict the probability of becoming a zombie for friends, family, and even yourself! To complete this Project, you should be comfortable with basic data analysis in base R and visualization in ggplot2, and be familiar with chi-squared test, t-tests, and logistic regression. The Zombies dataset was created for this Project based on emergency preparedness recommendations. It consists of 200 observations and 14 variables. The variables include personal characteristics like age and sex, zombie status, a description of the neighborhood each participant lives in, and measures of supplies.

Context News reports suggest that the impossible has become possible…zombies have appeared on the streets of the US! What should we do? The Centers for Disease Control and Prevention (CDC) zombie preparedness website recommends storing water, food, medication, tools, sanitation items, clothing, essential documents, and first aid supplies. Thankfully, we are CDC analysts and are prepared, but it may be too late for others!

Content Our team decides to identify supplies that protect people and coordinate supply distribution. A few brave data collectors volunteer to check on 200 randomly selected adults who were alive before the zombies. We have recent data for the 200 on age and sex, how many are in their household, and their rural, suburban, or urban location. Our heroic volunteers visit each home and record zombie status and preparedness. Now it’s our job to figure out which supplies are associated with safety!

Acknowledgements DataCamp

rm(list = ls())  
setwd("C:/Users/Sanzhar/OneDrive/Documents/Bay Path/Data mining/Working folder")  
library(rpart)

zom = read.csv("zombies.csv")  
my\_zom = zom[,2:14]  
head(my\_zom)

## zombie age sex rurality household water food medication tools  
## 1 Human 18 Female Rural 1 0 Food Medication No tools  
## 2 Human 18 Male Rural 3 24 Food Medication tools  
## 3 Human 18 Male Rural 4 16 Food Medication No tools  
## 4 Human 19 Male Rural 1 0 Food Medication tools  
## 5 Human 19 Male Urban 1 0 Food Medication No tools  
## 6 Human 19 Female Urban 1 0 Food Medication tools  
## firstaid sanitation clothing documents  
## 1 First aid supplies Sanitation Clothing No documents  
## 2 First aid supplies Sanitation Clothing No documents  
## 3 First aid supplies Sanitation Clothing No documents  
## 4 No first aid supplies Sanitation Clothing No documents  
## 5 First aid supplies Sanitation No clothing No documents  
## 6 First aid supplies Sanitation Clothing No documents

#Data preparation. Convert all categorical variables into factors and assign 0 to the base level

my\_zom$zombie = factor(my\_zom$zombie)  
levels(my\_zom$zombie) = c(0, 1)#human = 0; zombie = 1  
  
my\_zom$sex = as.factor(my\_zom$sex)  
levels(my\_zom$sex) = c(0, 1)#female = 0; male = 1  
  
my\_zom$rurality = as.factor(my\_zom$rurality)  
levels(my\_zom$rurality) = c(0,1,2) #rural = 0; suburban = 1, urban = 2  
  
my\_zom$food = as.factor(my\_zom$food)  
levels(my\_zom$food) = c(1, 0)#no food = 0; food = 1  
  
my\_zom$medication = as.factor(my\_zom$medication)  
levels(my\_zom$medication) = c(1, 0)#no med = 0; med = 1  
  
my\_zom$tools = as.factor(my\_zom$tools)  
levels(my\_zom$tools) = c(0, 1)#no tools = 0; tools = 1  
  
my\_zom$firstaid = as.factor(my\_zom$firstaid)  
levels(my\_zom$firstaid) = c(1, 0)#No First Aid = 0; First Aid = 1  
  
my\_zom$sanitation = as.factor(my\_zom$sanitation)  
levels(my\_zom$sanitation) = c(0, 1)#No sanitation = 0; Sanitation = 1  
  
my\_zom$clothing = as.factor(my\_zom$clothing)  
levels(my\_zom$clothing) = c(1, 0)#No clothing = 0; Clothing = 1  
  
my\_zom$documents = as.factor(my\_zom$documents)  
levels(my\_zom$documents) = c(1, 0)#No documents = 0; Documents = 1

head(my\_zom)

## zombie age sex rurality household water food medication tools firstaid  
## 1 0 18 0 0 1 0 1 1 0 1  
## 2 0 18 1 0 3 24 1 1 1 1  
## 3 0 18 1 0 4 16 1 1 0 1  
## 4 0 19 1 0 1 0 1 1 1 0  
## 5 0 19 1 2 1 0 1 1 0 1  
## 6 0 19 0 2 1 0 1 1 1 1  
## sanitation clothing documents  
## 1 1 1 0  
## 2 1 1 0  
## 3 1 1 0  
## 4 1 1 0  
## 5 1 0 0  
## 6 1 1 0

str(my\_zom)

## 'data.frame': 200 obs. of 13 variables:  
## $ zombie : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ age : int 18 18 18 19 19 19 20 20 21 21 ...  
## $ sex : Factor w/ 2 levels "0","1": 1 2 2 2 2 1 1 1 1 1 ...  
## $ rurality : Factor w/ 3 levels "0","1","2": 1 1 1 1 3 3 2 1 3 1 ...  
## $ household : int 1 3 4 1 1 1 2 2 1 2 ...  
## $ water : int 0 24 16 0 0 0 0 0 8 8 ...  
## $ food : Factor w/ 2 levels "1","0": 1 1 1 1 1 1 2 1 2 2 ...  
## $ medication: Factor w/ 2 levels "1","0": 1 1 1 1 1 1 1 2 2 2 ...  
## $ tools : Factor w/ 2 levels "0","1": 1 2 1 2 1 2 1 1 2 2 ...  
## $ firstaid : Factor w/ 2 levels "1","0": 1 1 1 2 1 1 1 2 1 1 ...  
## $ sanitation: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ clothing : Factor w/ 2 levels "1","0": 1 1 1 1 2 1 1 1 1 1 ...  
## $ documents : Factor w/ 2 levels "1","0": 2 2 2 2 2 2 2 2 1 1 ...

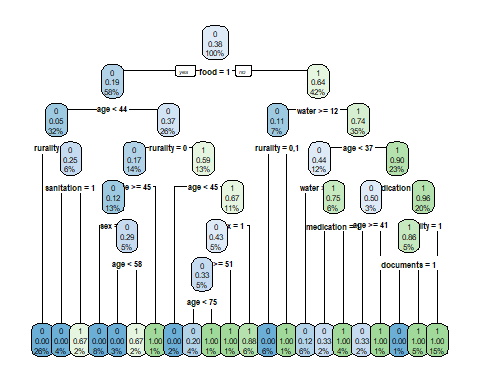
#create test and training sets

set.seed(500)  
test.indices = sort(sample(1:nrow(my\_zom), round((1/3)\*nrow(my\_zom))))  
test = my\_zom[test.indices,]  
train = my\_zom[-test.indices,]

#grow a single tree

model.control = rpart.control(minsplit = 4, xval = 5, cp =0)  
fit = rpart(zombie~., data = train, method = "class", control = model.control)

library(rpart.plot)  
rpart.plot(fit, tweak = 1.5)

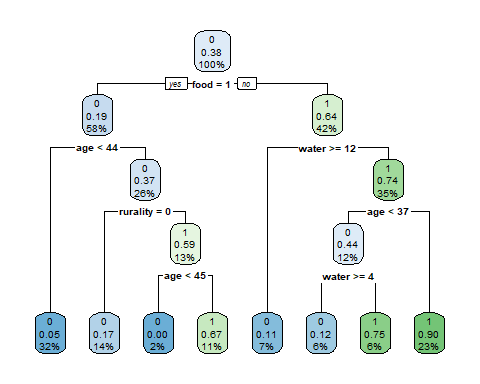


min\_cp = which.min(fit$cptable[,4])  
min\_cp

## 5   
## 5

#prune the tree

pruned\_fit = prune(fit, cp = fit$cptable[min\_cp,1] )  
rpart.plot(pruned\_fit, type =2)



#predict with a single tree

my\_pred = predict(pruned\_fit, newdata = test, type = "class")  
y\_hat = as.numeric(my\_pred)-1  
y\_true = as.numeric(test$zombie)-1  
misclass\_tree = sum(abs(y\_true - y\_hat))/length(y\_hat)  
misclass\_tree

## [1] 0.2089552

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

#Random forest

#install.packages("randomForest")  
library(randomForest)

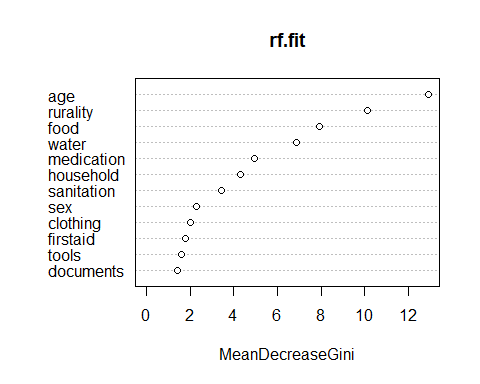
## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

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

rf.fit = randomForest(zombie~., data = train, n.tree = 100000)  
varImpPlot(rf.fit)



my\_pred\_rf = predict(rf.fit, newdata = test, type = "response")  
y\_hat\_rf = as.numeric(my\_pred\_rf)-1  
misclass\_rf = sum(abs(y\_true - y\_hat\_rf))/length(y\_hat\_rf)  
misclass\_rf

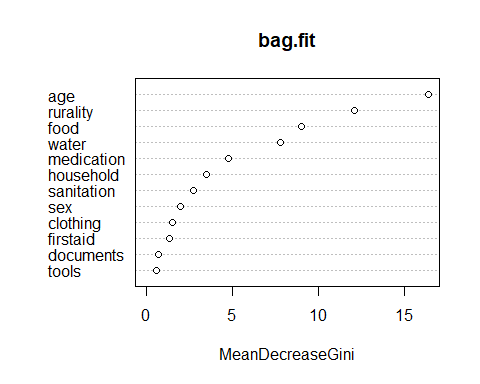
## [1] 0.1343284

#Bagging

dim(my\_zom)

## [1] 200 13

bag.fit = randomForest(zombie~., data = train, n.tree = 100000, mtry = 12)  
varImpPlot(bag.fit)



my\_pred\_bag = predict(bag.fit, newdata = test, type = "response")  
y\_hat\_bag = as.numeric(my\_pred\_bag)-1  
misclass\_bag = sum(abs(y\_true - y\_hat\_bag))/length(y\_hat\_bag)  
misclass\_bag

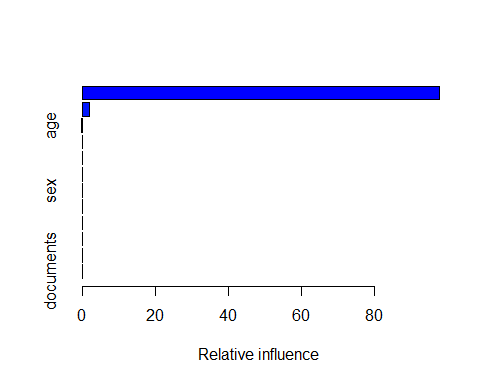
## [1] 0.1492537

#Boosting

#install.packages("gbm")  
library(gbm)

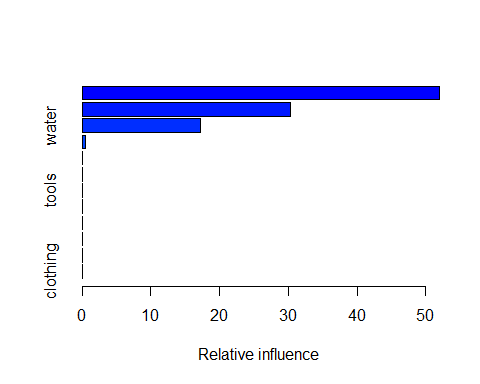
## Loaded gbm 2.1.8

boost.fit1 = gbm(zombie~.,data = train, n.trees=10000, distribution = "adaboost", shrinkage = 0.1, interaction.depth = 1)  
boost.fit2 = gbm(zombie~.,data = train, n.trees=10000, distribution = "adaboost", shrinkage = 0.6, interaction.depth = 3)  
summary(boost.fit1)



## var rel.inf  
## food food 9.786474e+01  
## rurality rurality 1.988253e+00  
## age age 1.469827e-01  
## medication medication 2.845723e-05  
## water water 4.411356e-08  
## sanitation sanitation 5.638547e-131  
## sex sex 0.000000e+00  
## household household 0.000000e+00  
## tools tools 0.000000e+00  
## firstaid firstaid 0.000000e+00  
## clothing clothing 0.000000e+00  
## documents documents 0.000000e+00

summary(boost.fit2)



## var rel.inf  
## age age 5.203216e+01  
## rurality rurality 3.030846e+01  
## water water 1.714467e+01  
## medication medication 4.401441e-01  
## sex sex 7.152418e-02  
## food food 3.034721e-03  
## tools tools 8.678514e-06  
## household household 1.046747e-43  
## sanitation sanitation 1.419076e-91  
## firstaid firstaid 2.536254e-240  
## documents documents 9.603989e-318  
## clothing clothing 0.000000e+00

#boosting seems to favor a continuous variable age and a factor with largest number of levels - rurality

y\_hat\_boost1 = predict(boost.fit1, newdata = test, n.trees = 10000, type = "response")  
misclass\_boost1 = sum(abs(y\_hat\_boost1-y\_true))/length(y\_true)  
misclass\_boost1

## [1] 0.5820896

y\_hat\_boost2 = predict(boost.fit2, newdata = test, n.trees = 10000, type = "response")  
misclass\_boost2 = sum(abs(y\_hat\_boost2-y\_true))/length(y\_true)  
misclass\_boost2

## [1] 0.5820896

#Boosting shows and unusually large misclassification error

#Logistic regression

log.fit = glm(zombie~., data = train, family = "binomial")  
summary(log.fit)

##   
## Call:  
## glm(formula = zombie ~ ., family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.97403 -0.31357 -0.04668 0.22603 2.27973   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -5.47052 2.72091 -2.011 0.044373 \*   
## age 0.08957 0.02451 3.655 0.000257 \*\*\*  
## sex1 0.75678 0.66560 1.137 0.255546   
## rurality1 1.18287 0.85239 1.388 0.165225   
## rurality2 3.33398 1.04290 3.197 0.001389 \*\*   
## household 0.29049 0.34323 0.846 0.397364   
## water -0.12864 0.06511 -1.976 0.048170 \*   
## food0 2.07628 0.75418 2.753 0.005905 \*\*   
## medication0 1.92911 0.75131 2.568 0.010239 \*   
## tools1 -0.57883 0.68501 -0.845 0.398110   
## firstaid0 -0.08110 0.66196 -0.123 0.902485   
## sanitation1 -3.60431 1.58404 -2.275 0.022882 \*   
## clothing0 -1.34434 0.73427 -1.831 0.067124 .   
## documents0 -1.19935 1.54570 -0.776 0.437791   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 177.085 on 132 degrees of freedom  
## Residual deviance: 68.691 on 119 degrees of freedom  
## AIC: 96.691  
##   
## Number of Fisher Scoring iterations: 7

my\_pred\_log = predict(log.fit, newdata = test, type ="response")  
y\_hat\_log = round(my\_pred\_log)  
  
misclass\_log = sum(abs(y\_hat\_log - y\_true))/length(y\_true)  
misclass\_log

## [1] 0.1044776

#KNN

library(class)  
knn\_train = train[,-1]  
knn\_test = test[,-1]  
cl = train[,1]  
knn.fit = knn(train, test, cl, k = 1)  
y\_hat\_knn = as.numeric(knn.fit)-1  
misclass\_knn = sum(abs(y\_hat\_knn - y\_true))/length(y\_true)  
misclass\_knn

## [1] 0.2835821

#LDA

library(MASS)  
lda.fit = lda(zombie~., data = train)  
my\_pred\_lda = predict(lda.fit, newdata = test, type = "response")  
y\_hat\_lda = as.numeric(my\_pred\_lda$class)-1  
misclass\_lda = sum(abs(y\_hat\_lda - y\_true))/length(y\_true)  
misclass\_lda

## [1] 0.08955224

#QDA

qda.fit = qda(zombie~., data = train)  
my\_pred\_qda = predict(qda.fit, newdata = test, type = "response")  
y\_hat\_qda = as.numeric(my\_pred\_qda$class)-1  
misclass\_qda = sum(abs(y\_hat\_qda - y\_true))/length(y\_true)  
misclass\_qda

## [1] 0.1343284

#let’s look at the test errors again

errors = round(rbind(misclass\_tree, misclass\_bag, misclass\_rf, misclass\_boost1, misclass\_boost2, misclass\_log, misclass\_lda, misclass\_qda, misclass\_knn),4)  
errors

## [,1]  
## misclass\_tree 0.2090  
## misclass\_bag 0.1493  
## misclass\_rf 0.1343  
## misclass\_boost1 0.5821  
## misclass\_boost2 0.5821  
## misclass\_log 0.1045  
## misclass\_lda 0.0896  
## misclass\_qda 0.1343  
## misclass\_knn 0.2836

I have examined 4 tree-based methods: a single tree, random forest, bagging and boosting. A single tree(CART) had a misclassification error of 0.20, bagging and random forest ensemble methods were able to reduce the error to 0.15 and 0.13 respectively.

Of all tree-based methods, boosting performed poorly regardless of the change in the hyper parameters, error 0.58. All the points in the test set were predicted as “Zombie”.

To contrast the tree-based methods, 3 parametric (logistic regression, LDA, QDA) and 1 non-parametric (kNN) methods were used to produce their test errors for a comparison.

Logistic regression performed quite well with an error rate 0.10. The reason for that might be in the overlapping of points and the logistic regression is known to perform well in such setting. On the other hand, we can see how KNN produced a rather large misclassification rate of 0.28. As it was suspected at the beginning of the experiment, many categorical predictors have made observations crowd near the origin.

LDA was the best method with a misclassification rate of 0.09, and its cousin QDA performed slightly worse - 0.13. This suggests that the decision boundaries between classes are rather linear than quadratic in shape.