# assignement2

Lilach Herzog & Leslie Cohen

13 5 2022

Our objective was to analyze the "wheat-seeds" dataset (where the 'Type' of the seed is the column used for labeling and classifying) using two classifier models that we built. We decided on using the KNN algorithm and the decision tree (DT) algorithm.

## preproccessing

### Upload data and libraries

```
library(gmodels)
library(C50)
library(class)

seeds <- read.csv("seeds.csv")# read data
set.seed(1234)#to initialize a pseudo random number generator.
```

### initial look at the data

We know the class 'type' is the column (label) to classify according to. we took a quick look at the data:

```
str(seeds)
```

```
## 'data.frame':    199 obs. of  8 variables:
##  $ Area           : num  15.3 14.9 14.3 13.8 16.1 ...
##  $ Perimeter      : num  14.8 14.6 14.1 13.9 15 ...
##  $ Compactness    : num  0.871 0.881 0.905 0.895 0.903 ...
##  $ Kernel.Length  : num  5.76 5.55 5.29 5.32 5.66 ...
##  $ Kernel.Width   : num  3.31 3.33 3.34 3.38 3.56 ...
##  $ Asymmetry.Coeff: num  2.22 1.02 2.7 2.26 1.35 ...
##  $ Kernel.Groove  : num  5.22 4.96 4.83 4.8 5.17 ...
##  $ Type           : int  1 1 1 1 1 1 1 1 1 1 ...
```

There are 3 types of seeds (in the column 'Type'): 1, 2 and 3.

Since the whole table is sorted according to the' type' column, and we want to work randomly, we will shuffle the whole table

In order to further understand the data and be able to work with it we used the "table" function. this function builds a contingency table of the counts at each combination of factor levels- what interests us is the Type:

```
table(seeds$Type)
```

```
##
##  1  2  3
## 66 68 65
```

We can see the Type has 66 times type 1, 68 times type 2, and 65 times type 3, about a third of each type.

Before starting on teaching the algorithm, we converted the 'Type' column to a factor, as that is what is required by the C50 package.

```
seeds$Type<-as.factor(seeds$Type)
```

Now- let's get to work!!

### split into training and test sets

```
length_of_training_set <-round(0.8*(nrow(seeds)),0) #  0.8% of the observations out of the total 199
seeds_train <- seeds[1:length_of_training_set, ]
seeds_test <- seeds[(length_of_training_set+1):nrow(seeds), ]
```

Since we know that there are about a third of each type in the data set, we want to be sure that the distribution is as we defined (about 1/3 of each type in of both training and test sets):

```
##
##         1         2         3
## 0.3081761 0.3333333 0.3584906
```

```
##
##     1     2     3
## 0.425 0.375 0.200
```

#### target factor (label) vector for classification

## Decision tree

The 8th column of the dataset is the type class variable, so we need to exclude it from the training data frame, but supply it as the target factor (label) vector for classification:

```
seeds_model <- C5.0(seeds_train[-8], seeds_train$Type)
```

#### Explanation of initial tree:

```
##
## Call:
## C5.0.default(x = seeds_train[-8], y = seeds_train$Type)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu Jun  9 17:24:00 2022
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 159 cases (8 attributes) from undefined.data
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (54/1)
## Kernel.Groove <= 5.533:
## :...Area <= 12.7:
##     :...Kernel.Groove <= 4.703: 1 (3)
##     :   Kernel.Groove > 4.703: 3 (52/1)
##     Area > 12.7:
```

```
##     :...Asymmetry.Coeff <= 4.773: 1 (43/1)
##         Asymmetry.Coeff > 4.773:
##         :...Kernel.Width <= 3.201: 3 (5)
##             Kernel.Width > 3.201: 1 (2)
##
##
## Evaluation on training data (159 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       6    3( 1.9%)   <<
##
##
##     (a)   (b)   (c)    <-classified as
##    ----  ----  ----
##      47     1     1    (a): class 1
##            53          (b): class 2
##       1          56    (c): class 3
##
##
##  Attribute usage:
##
##  100.00% Kernel.Groove
##   66.04% Area
##   31.45% Asymmetry.Coeff
##    4.40% Kernel.Width
##
##
## Time: 0.0 secs
```

The algorithm divided the dataset each iteration into two parts: it chooses a number and a column and everything that is greater than that number in that column goes into one group everything that is smaller or equal to that number in that column goes to the other group. It keeps them dividing until all the members in a group have the same type.

- The algorithm started with the column' groove', and the number 5.33. It found that all those with a groove bigger than that are of type 2 (sub-mission completed). The algorithm then went to classify the data in the other group (groove smaller than 5.533).
- It divided that group into two subgroups according to the column' area' and found that all those left that are larger than an area of 13.37 are all 1.
- The ones that are smaller or equal to in area of 13.37 and groove of 5.33 were again divided by the groove. Out of those with a groove that is bigger than 4.914, those with the asymmetry coefficient that is smaller or equal to 1.791 or a type one, and those that are larger than that art type 3.
- The seeds with a groove that is larger than 4.915 (and smaller than 5.533 ) are divided according to the perimeter. Those were the perimeter larger than 13.41 or type 1, those that are smaller or equal to 13 point .1 and larger than 13.19 are type 3 and those that are smaller than 13.19 are type one.

the percentage of error is relatively low: 1.3% Kernel.Groove and area seem to be the better attribute. To construct the tree, the model used 4 attributes: Kernel.Groove, area, Asymmetry.Coeff and Kernel.Width.

We want to evaluate our prediction

```
# apply model on test data
seeds_pred <- predict(seeds_model, seeds_test)
```

```
CrossTable(seeds_test$Type, seeds_pred, prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE, dnn = c('ac
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  40
##
##
##              | predicted type
##  actual type |         1 |         2 |         3 | Row Total |
## -------------|-----------|-----------|-----------|-----------|
##            1 |        17 |         0 |         0 |        17 |
##              |     0.425 |     0.000 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|
##            2 |         1 |        14 |         0 |        15 |
##              |     0.025 |     0.350 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|
##            3 |         2 |         0 |         6 |         8 |
##              |     0.050 |     0.000 |     0.150 |           |
## -------------|-----------|-----------|-----------|-----------|
## Column Total |        20 |        14 |         6 |        40 |
## -------------|-----------|-----------|-----------|-----------|
##
##
```

Type seed 2 and 3 are predicted correctly at 100%, there is no false positive or false negative. The type seed 1 has some false negatives,but few.

We will try to improve our results, using adaptive boosting to have a lower percentage of error. This is a process in which many decision trees are built and the trees vote on the best class for each example.

### Adaptive Boosting

We'll start with 10 trials, a number that has become the de facto standard, as research suggests that this reduces error rates on test data by about 25 percent:

```
# boosting with 10 trials (on training)

seeds_boost10 <- C5.0(seeds_train[-8], seeds_train$Type, trials = 10)

seeds_boost10
```

```
##
## Call:
## C5.0.default(x = seeds_train[-8], y = seeds_train$Type, trials = 10)
##
## Classification Tree
## Number of samples: 159
## Number of predictors: 7
```

```
##
## Number of boosting iterations: 10
## Average tree size: 5.5
##
## Non-standard options: attempt to group attributes
```

```
summary(seeds_boost10 )
```

```
##
## Call:
## C5.0.default(x = seeds_train[-8], y = seeds_train$Type, trials = 10)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu Jun  9 17:24:00 2022
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 159 cases (8 attributes) from undefined.data
##
## -----  Trial 0:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (54/1)
## Kernel.Groove <= 5.533:
## :...Area <= 12.7:
##     :...Kernel.Groove <= 4.703: 1 (3)
##     :   Kernel.Groove > 4.703: 3 (52/1)
##     Area > 12.7:
##     :...Asymmetry.Coeff <= 4.773: 1 (43/1)
##         Asymmetry.Coeff > 4.773:
##         :...Kernel.Width <= 3.201: 3 (5)
##             Kernel.Width > 3.201: 1 (2)
##
## -----  Trial 1:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Asymmetry.Coeff <= 2.04: 1 (17.5/3.8)
## :   Asymmetry.Coeff > 2.04: 2 (36.2)
## Kernel.Groove <= 5.533:
## :...Area > 13.34: 1 (26.4)
##     Area <= 13.34:
##     :...Asymmetry.Coeff <= 2.64: 1 (21.3/2.3)
##         Asymmetry.Coeff > 2.64: 3 (57.5/3.8)
##
## -----  Trial 2:  -----
##
## Decision tree:
##
## Perimeter > 15.46: 2 (40.9)
## Perimeter <= 15.46:
## :...Asymmetry.Coeff <= 1.502: 1 (14.6)
##     Asymmetry.Coeff > 1.502:
##     :...Kernel.Groove <= 4.783: 1 (5.7)
##         Kernel.Groove > 4.783:
##         :...Perimeter <= 13.53: 3 (46.9)
##             Perimeter > 13.53: 1 (50.9/8.7)
##
## -----  Trial 3:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (59.9/8.1)
## Kernel.Groove <= 5.533:
## :...Area > 13.34: 1 (15.5)
##     Area <= 13.34:
##     :...Kernel.Groove > 5.088: 3 (38.9)
##         Kernel.Groove <= 5.088:
##         :...Perimeter <= 13.53: 3 (32.4/12)
##             Perimeter > 13.53: 1 (12.3)
##
## -----  Trial 4:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (53.8/13)
## Kernel.Groove <= 5.533:
## :...Kernel.Groove <= 4.825: 1 (28.3/0.3)
##     Kernel.Groove > 4.825:
##     :...Asymmetry.Coeff <= 1.599: 1 (15.1)
##         Asymmetry.Coeff > 1.599: 3 (61.9/15.6)
##
## -----  Trial 5:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Asymmetry.Coeff <= 2.04: 1 (22.1/8.3)
## :   Asymmetry.Coeff > 2.04: 2 (24.5)
## Kernel.Groove <= 5.533:
## :...Asymmetry.Coeff > 4.286: 3 (24.7/3.7)
##     Asymmetry.Coeff <= 4.286:
##     :...Area > 12.76: 1 (38.1)
##         Area <= 12.76:
##         :...Kernel.Width > 3.074: 3 (5.1)
##             Kernel.Width <= 3.074:
##             :...Kernel.Groove <= 4.963: 1 (29.8/0.6)
##                 Kernel.Groove > 4.963: 3 (14.6/2.9)
##
## -----  Trial 6:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Kernel.Groove <= 5.877: 1 (18.2/7.5)
## :   Kernel.Groove > 5.877: 2 (33.2)
## Kernel.Groove <= 5.533:
## :...Kernel.Width <= 2.85: 3 (16.3)
##     Kernel.Width > 2.85:
##     :...Area > 13.34: 1 (36.4)
##         Area <= 13.34:
##         :...Kernel.Groove > 5.097: 3 (12.3)
##             Kernel.Groove <= 5.097:
##             :...Kernel.Width <= 3.119: 1 (38.2/2.6)
##                 Kernel.Width > 3.119: 3 (4.4/0.2)
##
## -----  Trial 7:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (62/8.2)
## Kernel.Groove <= 5.533:
## :...Area > 13.34: 1 (28)
##     Area <= 13.34:
##     :...Kernel.Groove <= 4.783: 1 (13.5)
##         Kernel.Groove > 4.783: 3 (55.5/15.7)
##
## -----  Trial 8:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (54.4/11.7)
## Kernel.Groove <= 5.533:
## :...Kernel.Width <= 2.85: 3 (9.9)
##     Kernel.Width > 2.85:
##     :...Asymmetry.Coeff > 5.335: 3 (9.7/2.3)
##         Asymmetry.Coeff <= 5.335:
##         :...Perimeter > 13.71: 1 (40.9)
##             Perimeter <= 13.71:
##             :...Kernel.Groove <= 4.963: 1 (32.3/2.5)
##                 Kernel.Groove > 4.963: 3 (11.7)
##
## -----  Trial 9:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (34.7)
## Kernel.Groove <= 5.533:
## :...Perimeter > 13.53:
##     :...Area <= 12.7: 3 (3.6)
##     :   Area > 12.7: 1 (59.1/2.2)
##     Perimeter <= 13.53:
##     :...Kernel.Groove <= 4.703: 1 (7.4)
##         Kernel.Groove > 4.703:
##         :...Asymmetry.Coeff <= 1.502: 1 (7.9)
##             Asymmetry.Coeff > 1.502: 3 (45.2)
##
##
## Evaluation on training data (159 cases):
##
```

```
## Trial        Decision Tree
## -----      ----------------
##    Size      Errors
##
##    0      6    3( 1.9%)
##    1      5   13( 8.2%)
##    2      5   15( 9.4%)
##    3      5    6( 3.8%)
##    4      4   32(20.1%)
##    5      7   12( 7.5%)
##    6      7   20(12.6%)
##    7      4    8( 5.0%)
##    8      6    4( 2.5%)
##    9      6    4( 2.5%)
## boost            1( 0.6%)   <<
##
##
##     (a)   (b)   (c)    <-classified as
##    ----  ----  ----
##      48     1          (a): class 1
##            53          (b): class 2
##                  57    (c): class 3
##
##
##  Attribute usage:
##
##  100.00% Perimeter
##  100.00% Asymmetry.Coeff
##  100.00% Kernel.Groove
##   66.04% Area
##   66.04% Kernel.Width
##
##
## Time: 0.0 secs
```

The classifier made 1 mistake for an error rate of 0.6% percent. This is an improvement over the previous training error rate before adding boosting! However, it remains to be seen whether we see a similar improvement on the test data. Let's take a look:

```
# boosting on test data
seeds_boost_pred10 <- predict(seeds_boost10, seeds_test)

CrossTable(seeds_test$Type, seeds_boost_pred10,
prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
dnn = c('actual Type', 'predicted Type'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
```

```
## Total Observations in Table:  40
##
##
##               | predicted Type
##  actual Type |         1 |         2 |         3 | Row Total |
## -------------|-----------|-----------|-----------|-----------|
##            1 |        17 |         0 |         0 |        17 |
##              |     0.425 |     0.000 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|
##            2 |         1 |        14 |         0 |        15 |
##              |     0.025 |     0.350 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|
##            3 |         1 |         0 |         7 |         8 |
##              |     0.025 |     0.000 |     0.175 |           |
## -------------|-----------|-----------|-----------|-----------|
## Column Total |        19 |        14 |         7 |        40 |
## -------------|-----------|-----------|-----------|-----------|
##
##
```

The model made 2 errors instead of 3 before the boost, on seeds of type 1. WE will try to improve the result, with more trials than 10, for example 15.

```
# boosting with 20 trials (on training)

seeds_boost20 <- C5.0(seeds_train[-8], seeds_train$Type, trials = 20)

seeds_boost20
```

```
##
## Call:
## C5.0.default(x = seeds_train[-8], y = seeds_train$Type, trials = 20)
##
## Classification Tree
## Number of samples: 159
## Number of predictors: 7
##
## Number of boosting iterations: 20
## Average tree size: 5.9
##
## Non-standard options: attempt to group attributes
```

```
summary(seeds_boost20 )
```

```
##
## Call:
## C5.0.default(x = seeds_train[-8], y = seeds_train$Type, trials = 20)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu Jun  9 17:24:00 2022
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 159 cases (8 attributes) from undefined.data
##
```

```
## -----  Trial 0:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (54/1)
## Kernel.Groove <= 5.533:
## :...Area <= 12.7:
##     :...Kernel.Groove <= 4.703: 1 (3)
##     :   Kernel.Groove > 4.703: 3 (52/1)
##     Area > 12.7:
##     :...Asymmetry.Coeff <= 4.773: 1 (43/1)
##         Asymmetry.Coeff > 4.773:
##         :...Kernel.Width <= 3.201: 3 (5)
##             Kernel.Width > 3.201: 1 (2)
##
## -----  Trial 1:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Asymmetry.Coeff <= 2.04: 1 (17.5/3.8)
## :   Asymmetry.Coeff > 2.04: 2 (36.2)
## Kernel.Groove <= 5.533:
## :...Area > 13.34: 1 (26.4)
##     Area <= 13.34:
##     :...Asymmetry.Coeff <= 2.64: 1 (21.3/2.3)
##         Asymmetry.Coeff > 2.64: 3 (57.5/3.8)
##
## -----  Trial 2:  -----
##
## Decision tree:
##
## Perimeter > 15.46: 2 (40.9)
## Perimeter <= 15.46:
## :...Asymmetry.Coeff <= 1.502: 1 (14.6)
##     Asymmetry.Coeff > 1.502:
##     :...Kernel.Groove <= 4.783: 1 (5.7)
##         Kernel.Groove > 4.783:
##         :...Perimeter <= 13.53: 3 (46.9)
##             Perimeter > 13.53: 1 (50.9/8.7)
##
## -----  Trial 3:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (59.9/8.1)
## Kernel.Groove <= 5.533:
## :...Area > 13.34: 1 (15.5)
##     Area <= 13.34:
##     :...Kernel.Groove > 5.088: 3 (38.9)
##         Kernel.Groove <= 5.088:
##         :...Perimeter <= 13.53: 3 (32.4/12)
##             Perimeter > 13.53: 1 (12.3)
##
```

```
## -----  Trial 4:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (53.8/13)
## Kernel.Groove <= 5.533:
## :...Kernel.Groove <= 4.825: 1 (28.3/0.3)
##     Kernel.Groove > 4.825:
##     :...Asymmetry.Coeff <= 1.599: 1 (15.1)
##         Asymmetry.Coeff > 1.599: 3 (61.9/15.6)
##
## -----  Trial 5:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Asymmetry.Coeff <= 2.04: 1 (22.1/8.3)
## :   Asymmetry.Coeff > 2.04: 2 (24.5)
## Kernel.Groove <= 5.533:
## :...Asymmetry.Coeff > 4.286: 3 (24.7/3.7)
##     Asymmetry.Coeff <= 4.286:
##     :...Area > 12.76: 1 (38.1)
##         Area <= 12.76:
##         :...Kernel.Width > 3.074: 3 (5.1)
##             Kernel.Width <= 3.074:
##             :...Kernel.Groove <= 4.963: 1 (29.8/0.6)
##                 Kernel.Groove > 4.963: 3 (14.6/2.9)
##
## -----  Trial 6:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Kernel.Groove <= 5.877: 1 (18.2/7.5)
## :   Kernel.Groove > 5.877: 2 (33.2)
## Kernel.Groove <= 5.533:
## :...Kernel.Width <= 2.85: 3 (16.3)
##     Kernel.Width > 2.85:
##     :...Area > 13.34: 1 (36.4)
##         Area <= 13.34:
##         :...Kernel.Groove > 5.097: 3 (12.3)
##             Kernel.Groove <= 5.097:
##             :...Kernel.Width <= 3.119: 1 (38.2/2.6)
##                 Kernel.Width > 3.119: 3 (4.4/0.2)
##
## -----  Trial 7:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (62/8.2)
## Kernel.Groove <= 5.533:
## :...Area > 13.34: 1 (28)
##     Area <= 13.34:
##     :...Kernel.Groove <= 4.783: 1 (13.5)
##         Kernel.Groove > 4.783: 3 (55.5/15.7)
##
## -----  Trial 8:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (54.4/11.7)
## Kernel.Groove <= 5.533:
## :...Kernel.Width <= 2.85: 3 (9.9)
##     Kernel.Width > 2.85:
##     :...Asymmetry.Coeff > 5.335: 3 (9.7/2.3)
##         Asymmetry.Coeff <= 5.335:
##         :...Perimeter > 13.71: 1 (40.9)
##             Perimeter <= 13.71:
##             :...Kernel.Groove <= 4.963: 1 (32.3/2.5)
##                 Kernel.Groove > 4.963: 3 (11.7)
##
## -----  Trial 9:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Kernel.Width <= 3.465: 1 (29.4/9.9)
## :   Kernel.Width > 3.465: 2 (23.4)
## Kernel.Groove <= 5.533:
## :...Perimeter > 13.53:
##     :...Area <= 12.7: 3 (3.4)
##     :   Area > 12.7: 1 (52.7/2.1)
##     Perimeter <= 13.53:
##     :...Asymmetry.Coeff <= 1.502: 1 (9.2)
##         Asymmetry.Coeff > 1.502:
##         :...Kernel.Groove <= 4.703: 1 (5.5)
##             Kernel.Groove > 4.703: 3 (35.4)
##
## -----  Trial 10:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Asymmetry.Coeff <= 2.04: 1 (23/7.9)
## :   Asymmetry.Coeff > 2.04: 2 (43.6)
## Kernel.Groove <= 5.533:
## :...Area > 13.34: 1 (19.8)
##     Area <= 13.34:
##     :...Asymmetry.Coeff > 4.157: 3 (27.2)
##         Asymmetry.Coeff <= 4.157:
##         :...Kernel.Length <= 5.073: 3 (10.3/1.2)
##             Kernel.Length > 5.073: 1 (35.1/5.9)
##
## -----  Trial 11:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (61.5/11.7)
## Kernel.Groove <= 5.533:
## :...Area > 13.34: 1 (15.4)
##     Area <= 13.34:
##     :...Kernel.Groove > 5.088: 3 (27.1)
##         Kernel.Groove <= 5.088:
##         :...Perimeter > 13.53: 1 (14.9)
##             Perimeter <= 13.53:
##             :...Kernel.Groove <= 4.703: 1 (6.1)
##                 Kernel.Groove > 4.703: 3 (34/4.5)
##
## -----  Trial 12:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Asymmetry.Coeff > 2.04: 2 (26.3)
## :   Asymmetry.Coeff <= 2.04:
## :   :...Kernel.Width <= 3.566: 1 (27.5)
## :       Kernel.Width > 3.566: 2 (12.4)
## Kernel.Groove <= 5.533:
## :...Asymmetry.Coeff <= 1.599: 1 (22.7)
##     Asymmetry.Coeff > 1.599:
##     :...Area > 13.34: 1 (11.7)
##         Area <= 13.34:
##         :...Kernel.Groove <= 4.825: 1 (6.4/0.2)
##             Kernel.Groove > 4.825: 3 (51.9/8.1)
##
## -----  Trial 13:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Asymmetry.Coeff <= 2.04: 1 (30.5/9.5)
## :   Asymmetry.Coeff > 2.04: 2 (20.1)
## Kernel.Groove <= 5.533:
## :...Kernel.Width <= 2.85: 3 (16.8)
##     Kernel.Width > 2.85:
##     :...Kernel.Groove <= 5.111: 1 (76.7/13)
##         Kernel.Groove > 5.111: 3 (14.9/3.9)
##
## -----  Trial 14:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Area <= 16.63: 1 (26.6/9.8)
## :   Area > 16.63: 2 (19.6)
## Kernel.Groove <= 5.533:
## :...Perimeter <= 13.53:
##     :...Asymmetry.Coeff <= 1.502: 1 (12.9)
##     :   Asymmetry.Coeff > 1.502: 3 (37.4/2.4)
##     Perimeter > 13.53:
##     :...Asymmetry.Coeff <= 4.773: 1 (54.5/2.4)
##         Asymmetry.Coeff > 4.773: 3 (8/3.3)
```

```
##
## -----  Trial 15:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (51.7/13.1)
## Kernel.Groove <= 5.533:
## :...Kernel.Width <= 2.85: 3 (12.4)
##     Kernel.Width > 2.85:
##     :...Area > 13.34: 1 (23.3)
##         Area <= 13.34:
##         :...Kernel.Groove > 5.097: 3 (13.1)
##             Kernel.Groove <= 5.097:
##             :...Asymmetry.Coeff > 4.286: 3 (6.5)
##                 Asymmetry.Coeff <= 4.286:
##                 :...Kernel.Width <= 3.129: 1 (48.1/5.6)
##                     Kernel.Width > 3.129: 3 (3.7/0.3)
##
## -----  Trial 16:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Asymmetry.Coeff <= 2.04: 1 (27.3/8.1)
## :   Asymmetry.Coeff > 2.04: 2 (22.1)
## Kernel.Groove <= 5.533:
## :...Area <= 12.7:
##     :...Kernel.Groove <= 4.961: 1 (16.6/4.1)
##     :   Kernel.Groove > 4.961: 3 (39.1)
##     Area > 12.7:
##     :...Perimeter <= 13.47: 3 (4.9)
##         Perimeter > 13.47: 1 (49/3.5)
##
## -----  Trial 17:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533: 2 (51.6/14.9)
## Kernel.Groove <= 5.533:
## :...Kernel.Width > 3.155: 1 (17.8)
##     Kernel.Width <= 3.155:
##     :...Asymmetry.Coeff > 4.157: 3 (32)
##         Asymmetry.Coeff <= 4.157:
##         :...Area > 12.7: 1 (19.7/2.1)
##             Area <= 12.7:
##             :...Kernel.Groove <= 4.963: 1 (12.2/2.5)
##                 Kernel.Groove > 4.963: 3 (25.7)
##
## -----  Trial 18:  -----
##
## Decision tree:
##
## Kernel.Width <= 3.155:
## :...Asymmetry.Coeff <= 1.502: 1 (5.4)
## :   Asymmetry.Coeff > 1.502:
## :   :...Kernel.Groove <= 4.703: 1 (3.9)
## :       Kernel.Groove > 4.703: 3 (81.9/12)
## Kernel.Width > 3.155:
## :...Kernel.Groove > 5.877: 2 (19.8)
##     Kernel.Groove <= 5.877:
##     :...Compactness <= 0.8658: 2 (6.9/0.7)
##         Compactness > 0.8658: 1 (41/2.8)
##
## -----  Trial 19:  -----
##
## Decision tree:
##
## Kernel.Groove > 5.533:
## :...Kernel.Width <= 3.465: 1 (29.4/10)
## :   Kernel.Width > 3.465: 2 (26.5)
## Kernel.Groove <= 5.533:
## :...Perimeter > 13.53:
##     :...Asymmetry.Coeff <= 4.286: 1 (40.8/2.2)
##     :   Asymmetry.Coeff > 4.286: 3 (10.9/3.5)
##     Perimeter <= 13.53:
##     :...Kernel.Groove <= 4.703: 1 (3.2)
##         Kernel.Groove > 4.703:
##         :...Asymmetry.Coeff <= 1.502: 1 (3.6)
##             Asymmetry.Coeff > 1.502: 3 (44.6)
##
##
## Evaluation on training data (159 cases):
##
## Trial        Decision Tree
## -----      ----------------
##    Size      Errors
##
##    0      6    3( 1.9%)
##    1      5   13( 8.2%)
##    2      5   15( 9.4%)
##    3      5    6( 3.8%)
##    4      4   32(20.1%)
##    5      7   12( 7.5%)
##    6      7   20(12.6%)
##    7      4    8( 5.0%)
##    8      6    4( 2.5%)
##    9      7   10( 6.3%)
##   10      6   20(12.6%)
##   11      6    2( 1.3%)
##   12      7    5( 3.1%)
##   13      5   35(22.0%)
##   14      6   16(10.1%)
##   15      7    5( 3.1%)
##   16      6   14( 8.8%)
##   17      6    3( 1.9%)
##   18      6   19(11.9%)
##   19      7   13( 8.2%)
## boost              0( 0.0%)   <<
```

```
##
##
##     (a)   (b)   (c)    <-classified as
##    ----  ----  ----
##      49                (a): class 1
##            53          (b): class 2
##                  57    (c): class 3
##
##
##  Attribute usage:
##
##  100.00% Area
##  100.00% Perimeter
##  100.00% Kernel.Width
##  100.00% Asymmetry.Coeff
##  100.00% Kernel.Groove
##   27.67% Compactness
##   18.87% Kernel.Length
##
##
## Time: 0.0 secs
```

```r
# boosting on test data
seeds_boost_pred20 <- predict(seeds_boost20, seeds_test)

CrossTable(seeds_test$Type, seeds_boost_pred20,
prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
dnn = c('actual Type', 'predicted Type'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  40
##
##
##              | predicted Type
##  actual Type |         1 |         2 |         3 | Row Total |
## -------------|-----------|-----------|-----------|-----------|
##            1 |        17 |         0 |         0 |        17 |
##              |     0.425 |     0.000 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|
##            2 |         1 |        14 |         0 |        15 |
##              |     0.025 |     0.350 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|
##            3 |         1 |         0 |         7 |         8 |
##              |     0.025 |     0.000 |     0.175 |           |
## -------------|-----------|-----------|-----------|-----------|
## Column Total |        19 |        14 |         7 |        40 |
## -------------|-----------|-----------|-----------|-----------|
##
##
```

With 10 trials, we obtained the best result. Indeed, choosing 20 trials in the boost does not improve the result. The error rate stay similar than with 10 trials.

### DT conclusion

then we should write our conclusion about the different predictions obtained. Compare these results to the boosted model; this version makes more mistakes overall, but the types of mistakes are very different. Where the previous models incorrectly classified a small number of defaults correctly, our weighted model has does much better in this regard. This trade resulting in a reduction of false negatives at the expense of increasing false positives may be acceptable if our cost estimates were accurate. To create our decision trees in this practice we used the C5.0 package.

# KNN Algorithm

## preproccessing

First we will work with a new table of data. Then,we will transform the values of the Type variable to a factor variable:

```r
seeds$Type <- factor(seeds$Type, levels = c("1", "2","3"), labels = c("1", "2","3"))

# percentage of each type
round(prop.table(table(seeds$Type))*100, digits = 1)
```

```
##
##    1    2    3
## 33.2 34.2 32.7
```

### Normalization

We'll have to normalize our data to make sure our classifier will work. Lets create a simple normalization function (min/max normalization):

```r
# min/max normalization function:
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}

# test our function
normalize(c(1, 2, 3, 4, 5))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```r
normalize(c(10, 20, 30, 40, 50))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

Use the normalization function on our data, and make sure it worked:

```r
seeds_norm <- as.data.frame(lapply(seeds[1:7], normalize))

summary(seeds_norm$Area)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1643  0.3626  0.4088  0.6483  1.0000
```

Since we want to use KNN, we have to make sure the range of values of each parameter is similar.

The next step is to create a training set and a test set, and to store the labels in a different vector to be used later. We usually want to divide our data into 80%, 20% test, but we can always try a ratio of 85-15 or even 90-10 and see if results improved.

```
# types of train set


samples_seed <- seeds[sample(nrow(seeds)),]
seeds_train_types <-samples_seed[1:159, 8]

# labels of test set

seeds_test_types <- samples_seed[160:199,8 ]
```

Now, we will run the KNN algorithm and We will start with K of 3, because we have 3 types of seeds

## run the KNN algorithm

Now, we will run the KNN algorithm and We will start with K of 12 (???), the square root of 159 and also an odd number, #reducing the chance of a tie vote.

```
seeds_test_pred <- knn(train = seeds_train, test = seeds_test, cl = seeds_train_types, k=3)
```

Since the knn function returns a factor vector of the predicted values, we'll compare that vector with the true labels we saved in advance. We'll do the comparison with the CrossTable function, from the gmodels package we'll load:

```
# it is not running
CrossTable(x = seeds_test_types, y = seeds_test_pred, prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  40
##
##
##                  | seeds_test_pred
## seeds_test_types |         1 |         2 |         3 | Row Total |
## -----------------|-----------|-----------|-----------|-----------|
##                1 |         6 |         5 |         2 |        13 |
##                  |     0.462 |     0.385 |     0.154 |     0.325 |
##                  |     0.400 |     0.312 |     0.222 |           |
##                  |     0.150 |     0.125 |     0.050 |           |
## -----------------|-----------|-----------|-----------|-----------|
##                2 |         5 |         6 |         2 |        13 |
##                  |     0.385 |     0.462 |     0.154 |     0.325 |
##                  |     0.333 |     0.375 |     0.222 |           |
##                  |     0.125 |     0.150 |     0.050 |           |
## -----------------|-----------|-----------|-----------|-----------|
##                3 |         4 |         5 |         5 |        14 |
##                  |     0.286 |     0.357 |     0.357 |     0.350 |
##                  |     0.267 |     0.312 |     0.556 |           |
##                  |     0.100 |     0.125 |     0.125 |           |
## -----------------|-----------|-----------|-----------|-----------|
##     Column Total |        15 |        16 |         9 |        40 |
##                  |     0.375 |     0.400 |     0.225 |           |
## -----------------|-----------|-----------|-----------|-----------|
##
##
```

```
seeds_test_pred <- knn(train = seeds_train, test = seeds_test, cl = seeds_train_types, k=9)
```

Since the knn function returns a factor vector of the predicted values, we'll compare that vector with the true labels we saved in advance. We'll do the comparison with the CrossTable function, from the gmodels package we'll load:

```
# it is not running
CrossTable(x = seeds_test_types, y = seeds_test_pred, prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  40
##
##
##                  | seeds_test_pred
## seeds_test_types |         1 |         2 |         3 | Row Total |
## -----------------|-----------|-----------|-----------|-----------|
##                1 |         7 |         3 |         3 |        13 |
##                  |     0.538 |     0.231 |     0.231 |     0.325 |
##                  |     0.438 |     0.176 |     0.429 |           |
##                  |     0.175 |     0.075 |     0.075 |           |
## -----------------|-----------|-----------|-----------|-----------|
##                2 |         3 |         9 |         1 |        13 |
##                  |     0.231 |     0.692 |     0.077 |     0.325 |
##                  |     0.188 |     0.529 |     0.143 |           |
##                  |     0.075 |     0.225 |     0.025 |           |
## -----------------|-----------|-----------|-----------|-----------|
##                3 |         6 |         5 |         3 |        14 |
##                  |     0.429 |     0.357 |     0.214 |     0.350 |
##                  |     0.375 |     0.294 |     0.429 |           |
##                  |     0.150 |     0.125 |     0.075 |           |
## -----------------|-----------|-----------|-----------|-----------|
##     Column Total |        16 |        17 |         7 |        40 |
##                  |     0.400 |     0.425 |     0.175 |           |
## -----------------|-----------|-----------|-----------|-----------|
##
##
```

```
seeds_test_pred <- knn(train = seeds_train, test = seeds_test, cl = seeds_train_types, k=18)
```

Since the knn function returns a factor vector of the predicted values, we'll compare that vector with the true labels we saved in advance. We'll do the comparison with the CrossTable function, from the gmodels package we'll load:

```
# it is not running
CrossTable(x = seeds_test_types, y = seeds_test_pred, prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  40
##
##
##                  | seeds_test_pred
## seeds_test_types |         1 |         2 |         3 | Row Total |
## -----------------|-----------|-----------|-----------|-----------|
##                1 |         3 |         7 |         3 |        13 |
##                  |     0.231 |     0.538 |     0.231 |     0.325 |
##                  |     0.375 |     0.269 |     0.500 |           |
##                  |     0.075 |     0.175 |     0.075 |           |
## -----------------|-----------|-----------|-----------|-----------|
##                2 |         3 |         9 |         1 |        13 |
##                  |     0.231 |     0.692 |     0.077 |     0.325 |
##                  |     0.375 |     0.346 |     0.167 |           |
##                  |     0.075 |     0.225 |     0.025 |           |
## -----------------|-----------|-----------|-----------|-----------|
##                3 |         2 |        10 |         2 |        14 |
##                  |     0.143 |     0.714 |     0.143 |     0.350 |
##                  |     0.250 |     0.385 |     0.333 |           |
##                  |     0.050 |     0.250 |     0.050 |           |
## -----------------|-----------|-----------|-----------|-----------|
##     Column Total |         8 |        26 |         6 |        40 |
##                  |     0.200 |     0.650 |     0.150 |           |
## -----------------|-----------|-----------|-----------|-----------|
##
##
```

Lets see if we can improve our results. First, lets try z-score standardization instead of normalization:

```
# z-score normalization
seeds_z <- as.data.frame(scale(samples_seed[-8]))
```

```r
summary(seeds_z$Area)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## -1.4825 -0.8866 -0.1674  0.0000  0.8686  2.1443
```

We'll quickly repeat the previous steps (divide the data into train and test, classify, and compare the results to the true values):

```r
seeds_train_z <- seeds_z[1:159, ]
seeds_test_z <- seeds_z[160:199, ]
seeds_train_types_z <- samples_seed[1:159,8]
seeds_test_types_z <- samples_seed[160:199, 8]
seeds_test_pred_z <- knn(train = seeds_train_z, test = seeds_test_z, cl = seeds_train_types_z, k=3)
CrossTable(x = seeds_test_types_z, y = seeds_test_pred_z, prop.chisq=FALSE)
```

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  40 
## 
## 
##                    | seeds_test_pred_z 
## seeds_test_types_z |         1 |         2 |         3 | Row Total | 
## -------------------|-----------|-----------|-----------|-----------|
##                  1 |        12 |         1 |         0 |        13 | 
##                    |     0.923 |     0.077 |     0.000 |     0.325 | 
##                    |     0.800 |     0.083 |     0.000 |           | 
##                    |     0.300 |     0.025 |     0.000 |           | 
## -------------------|-----------|-----------|-----------|-----------|
##                  2 |         2 |        11 |         0 |        13 | 
##                    |     0.154 |     0.846 |     0.000 |     0.325 | 
##                    |     0.133 |     0.917 |     0.000 |           | 
##                    |     0.050 |     0.275 |     0.000 |           | 
## -------------------|-----------|-----------|-----------|-----------|
##                  3 |         1 |         0 |        13 |        14 | 
##                    |     0.071 |     0.000 |     0.929 |     0.350 | 
##                    |     0.067 |     0.000 |     1.000 |           | 
##                    |     0.025 |     0.000 |     0.325 |           | 
## -------------------|-----------|-----------|-----------|-----------|
##       Column Total |        15 |        12 |        13 |        40 | 
##                    |     0.375 |     0.300 |     0.325 |           | 
## -------------------|-----------|-----------|-----------|-----------|
## 
## 
```

It seems to be better, the model predicted . ## KNN conclusions # total conclusions