Part I) Data Preparation

Yves Zango

There are three files :

- *anneal.data* which is the train data set
- *anneal.test* which is the test data set
- *anneal.names* which describe the train dataset

We decided to merge the train and test data sets to get a single data set called ***dataset*** which will be used in the next part of the our study.

1) Data Loading

The missing values are indicated by the character *"?"*.

```
anneal.data <- read.table("anneal.data", header = FALSE, sep = ",",
    na.strings = "?", check.names = TRUE, stringsAsFactors = FALSE,
    blank.lines.skip = TRUE)

anneal.test <- read.table("anneal.test", header = FALSE, sep = ",",
    na.strings = "?", check.names = TRUE, stringsAsFactors = FALSE,
    blank.lines.skip = TRUE)
```

2) Reformatting the features types to respect the initial description of "anneal.names"

We performed this reformatting task both training and testing data sets.

```
# 6 continuously-valued
continuous.features <- c("carbon", "hardness", "strength", "thick", "width", "len")

# 3 integer-valued
integer.features <- c("formability", "enamelability", "packing")

# ( factors) + target variable (classes)
factors.features <-c("family", "steel", "condition", "surface_qua", "bw_me",
                     "blue_bright_varn_clean", "shape", "oil", "bore", "classes")

# nominal char: factor with one level
char.features <-c("product_typ","temper_roll","non_ageing","surface_fin",
                  "bc","bf","bt","bl","m","chrom","phos","cbond","marvi",
                  "exptl","ferro","corr","lustre","jurofm","s","p")

features <-c("family", "product_typ", "steel", "carbon", "hardness", "temper_roll",
             "condition", "formability", "strength", "non_ageing", "surface_fin",
             "surface_qua", "enamelabili", "bc", "bf", "bt", "bw_me", "bl", "m",
             "chrom", "phos", "cbond", "marvi", "exptl", "ferro", "corr",
             "blue_bright_varn_clean", "lustre", "jurofm", "s", "p", "shape",
             "thick", "width", "len", "oil", "bore", "packing", "classes")
```

```
nb.var <- length(features)
for (i in seq_len(nb.var)) {
  if (is.element(features[i], continuous.features)) {
      anneal.data[, i] <- as.numeric(anneal.data[, i])
      anneal.test[, i] <- as.numeric(anneal.test[, i])
  }
  if (is.element(features[i], integer.features) |
      is.element(features[i], factors.features) |
      is.element(features[i], char.features))
  {
      anneal.data[, i] <- as.factor(anneal.data[, i])
      anneal.test[, i] <- as.factor(anneal.test[, i])
  }
}

names(anneal.data) <- features
names(anneal.test) <- features
dataset <- rbind(anneal.data, anneal.test)
classes <- dataset$classes
nb.obeservations <- nrow(dataset)
nb.features <- ncol(dataset) - 1 # without the target variable ("classes")
```

3) Missing Values Handling

We will handle the missing values by using the library missForest. Indeed, it performs a nonparametric imputation by using random forest algorithm to estimates the missing values.

However, first of all, we arbitrary decided to drop the features having less then 50 observations. The number of missing value "mv.file.description" for each feature is provided in the "anneal.names" file. However we will merge the training and testing data and calculate the number of missing values of the merged dataset and store them in the variable "missing.values".

```
mv.file.description <- c(0, 0, 70,  0, 0, 675,  271, 283,   0, 703,  790, 217, 785, 797,
                         680, 736, 609, 662, 798, 775, 791, 730, 798, 796, 772, 798,
                         793, 753, 798, 798, 798,   0   ,0, 0, 0, 740, 0,   789, 0)

missing.values <- numeric(nb.features)
for (i in seq_len(nb.features))
{
  missing.values[i] <- sum(is.na(dataset[,i]))
}

print(missing.values)
```

```
##  [1] 772   0  86   0   0 761 303 318   0 793 889 244 882 897 769 824 687
## [18] 749 898 872 891 824 898 896 868 898 892 847 898 898 898   0   0   0
## [35]   0 834   0 889
```

```
minimal.nb.observations <- 50
classes <- dataset$classes
features.useful <- features[missing.values < nb.obeservations - minimal.nb.observations ]
p <- length(features.useful) - 1 # without the target feature
features.useful <- features.useful[1:p]
```

```
dataset <- dataset[features.useful]
print(features.useful)
```

```
##  [1] "family"      "product_typ" "steel"       "carbon"      "hardness"
##  [6] "temper_roll" "condition"   "formability" "strength"    "non_ageing"
## [11] "surface_qua" "bf"          "bt"          "bw_me"       "bl"
## [16] "cbond"       "lustre"      "shape"       "thick"       "width"
## [21] "len"         "oil"         "bore"
```

4) Building and saving of the new dataset (without missing values) which will be used in the following setps.

```
### Building the new dataset which will be used for the study
dataset <- dataset[features.useful]
dataset.imp <- missForest(dataset, ntree = 500, replace = TRUE, parallelize = 'forests')
```

```
##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
##   missForest iteration 4 in progress...done!
##   missForest iteration 5 in progress...done!
##   missForest iteration 6 in progress...done!
```

```
dataset.without.na <- dataset.imp$ximp
dataset.without.na <- data.frame(dataset.without.na, classes = classes)
write.csv(dataset.without.na,"data.csv",row.names = FALSE)
```

Part II) Features Ranking

1) Data Loading

Loading of the *"data.csv"* file which results from the merging of "anneal.data" and "anneal.test" files realized in the previous section. We remind that this dataframe does no longer contains some missing values after the imputation techniques based on missForest and performed in the previous section.

```
dataset <- read.csv("data.csv", sep = ",")
features <- names(dataset)  # the names of the different features of the dataset
```

2) Features ranking

In this next section we built a function called **computeImportance** which computes the rank of the features by decreasing order of importance and using the randomForest training function. Indeed the rank of each feature is related to the mean of their importance (Mean of Decrease in the Gini Coefficient) computed over the M bootstraps. We ran this function **M** times (bootstrap technique) to get a more reliable ranking. A parallelized loop (foreach library) has been used to boost the computation. Finally, we saved the features ranking in a csv file for the next step.

```
M <- 100   # Number of bootstraps

computeImportance <- function()
{
  rf.model <- randomForest(classes ~ .,dataset, importance = TRUE, doBest=TRUE)
  importance_features <- as.data.frame(rf.model$importance)["MeanDecreaseGini"]
  return(importance_features)
}

importance_bootstraps <- foreach(m=1:M,.combine = cbind) %dopar% {computeImportance()}
importance_matrix_mean <- apply(importance_bootstraps,1,mean)

final_rank <- sort(importance_matrix_mean,
                   index.return = TRUE,
                   decreasing = TRUE)$ix
featuresRanked <- data.frame(rank=final_rank,FeatNames=features[final_rank])

#Printing the features by decreasing order of importance
print(featuresRanked)
```

```
##    rank   FeatNames
## 1     5    hardness
## 2    19       thick
## 3     3       steel
## 4    20       width
## 5     8 formability
## 6     9    strength
## 7     1      family
## 8    11 surface_qua
## 9    21         len
## 10   18       shape
## 11    4      carbon
```

```
## 12    7   condition
## 13   14       bw_me
## 14   22         oil
## 15   23        bore
## 16    2 product_typ
## 17    6 temper_roll
## 18   10  non_ageing
## 19   12          bf
## 20   13          bt
## 21   15          bl
## 22   16       cbond
## 23   17      lustre
```

```
write.csv(featuresRanked,"featuresRanking.csv",row.names = FALSE)
```

Part III) Features Selection

1) Loading of Data and Features ranking

Let us remind that the file *featuresRanking.csv* contains a dataframe where the first column is the rank of the features by a decreasing importance order and the second column the corresponding name.

```
featuresRanked <- read.csv("featuresRanking.csv", sep = ",")
fRank <- featuresRanked$rank
fNames <- as.character(featuresRanked$FeatNames)
p <- length(fRank)
dataset <- read.csv("data.csv", sep = ",")
```

2) Features selection

In this part we will first compute the error of misclassification for different subsets of features whose size will be incrementally increased. More explicitely, we start by the subset containing an unique feature which is the most important one (determined by the feature ranking process). The second subset will contain the two most important features. Then, the third subset will contain the three most important features and so on. Overall, if we denote by $\boldsymbol{p}$ the total number of features, we would have $\boldsymbol{p}$ means of misclassification error corresponding to each of the subsets.

This process is implemented in a function that we called **computeError**.

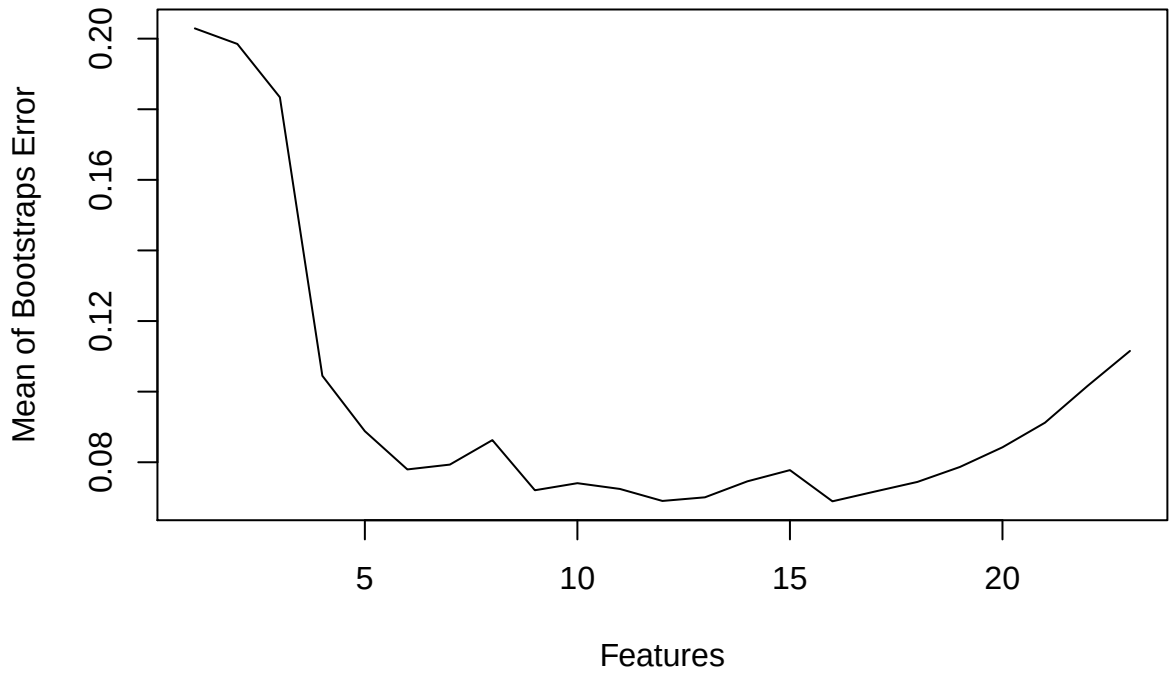
Then a bootstrap technique (M=100) is performed to get a more consistent result. Finally the mean error over the M p-dimensional errors of misclassification is computed and then plotted. Moreover we determined the index $\boldsymbol{p}^{opt}$ corresponding to the minimal value of this curve. It represents the optimal number of features to select.

```
computeError <- function(){
  # train size 2/3  and 1/3 for the test
  train.index <- sample(1:nrow(dataset),size = floor(2/3 * nrow(dataset)))
  train.data <- dataset[train.index,]
  test.data <- dataset[-train.index,]

  nbfeatures <- numeric(0)
  errorCumFeatures <- numeric(p)
  for (i in seq_len(p))
  {
    nbfeatures <- fRank[1:i]
    training <- data.frame(train.data[,nbfeatures],classes=train.data$classes)
    names(training) <- c(fNames[1:i],"classes")
    testing <- data.frame(test.data[,nbfeatures],classes=test.data$classes)
    names(testing) <- c(fNames[1:i],"classes")
    rf.model <- randomForest(classes ~ ., training, doBest=TRUE)
    Ypred.rf <- predict(rf.model, newdata = testing, type = "class", na.action = na.roughfix)
    errorCumFeatures[i] <- mean(Ypred.rf != test.data$classes)
    errorCumFeatures
  }
  return(errorCumFeatures)
}

## Bootstrap M=100
```

```
M <- 100
errorBootstrap <- foreach(m=1:M,.combine = cbind) %dopar% {computeError() }
errMean <- apply(errorBootstrap,1,mean)
```

2) Plot of the misclassification error' curve

```
## The optimal number of features and the best features selected
plot(errMean,type="l", xlab = "Features", ylab = "Mean of Bootstraps Error")
```



3) The optimal number of features and the best features to select

Overall, we can observe on the plot, that the misclassification decreases when the number of features increases until a number of features equal to $p^{opt}=16$. From this value, the curve starts increasing, which means that, an optimal number of features for our study is 16 and these features are presented hereafter. This value is not always the minimal values of the mean error over the M bootstraps, but it is the value from which the curve increases and does no longer decrease.

Hereafter, we print the list of best features to select.

```
p_opt <- 16
print(fNames[1:p_opt])
```

```
##  [1] "hardness"    "thick"       "steel"       "width"       "formability"
##  [6] "strength"    "family"      "surface_qua" "len"         "shape"
## [11] "carbon"      "condition"   "bw_me"       "oil"         "bore"
## [16] "product_typ"
```

1. Title of Database: Annealing Data

2. Source Information: donated by David Sterling and Wray Buntine.

3. Past Usage: unknown

4. Relevant Information:
   -- Explanation: I suspect this was left by Ross Quinlan in 1987 at the
      4th Machine Learning Workshop. I'd have to check with Jeff Schlimmer
      to double check this.

5. Number of Instances: 798

6. Number of Attributes: 38
   -- 6 continuously-valued
   -- 3 integer-valued
   -- 29 nominal-valued

7. Attribute Information:
    1. family: --,GB,GK,GS,TN,ZA,ZF,ZH,ZM,ZS
    2. product-type: C, H, G
    3. steel: -,R,A,U,K,M,S,W,V
    4. carbon: continuous
    5. hardness: continuous
    6. temper_rolling: -,T
    7. condition: -,S,A,X
    8. formability: -,1,2,3,4,5
    9. strength: continuous
   10. non-ageing: -,N
   11. surface-finish: P,M,-
   12. surface-quality: -,D,E,F,G
   13. enamelability: -,1,2,3,4,5
   14. bc: Y,-
   15. bf: Y,-
   16. bt: Y,-
   17. bw/me: B,M,-
   18. bl: Y,-
   19. m: Y,-
   20. chrom: C,-
   21. phos: P,-
   22. cbond: Y,-
   23. marvi: Y,-
   24. exptl: Y,-
   25. ferro: Y,-
   26. corr: Y,-
   27. blue/bright/varn/clean: B,R,V,C,-
   28. lustre: Y,-
   29. jurofm: Y,-
   30. s: Y,-
   31. p: Y,-
   32. shape: COIL, SHEET
   33. thick: continuous
   34. width: continuous
   35. len: continuous
   36. oil: -,Y,N
   37. bore: 0000,0500,0600,0760
   38. packing: -,1,2,3
   classes: 1,2,3,4,5,U

   -- The '-' values are actually 'not_applicable' values rather than
      'missing_values' (and so can be treated as legal discrete
      values rather than as showing the absence of a discrete value).

8. Missing Attribute Values: Signified with "?"
   Attribute: Number of instances missing its value:
   1 0
   2 0
   3 70

4	0
5	0
6	675
7	271
8	283
9	0
10	703
11	790
12	217
13	785
14	797
15	680
16	736
17	609
18	662
19	798
20	775
21	791
22	730
23	798
24	796
25	772
26	798
27	793
28	753
29	798
30	798
31	798
32	0
33	0
34	0
35	0
36	740
37	0
38	789
39	0

9. Distribution of Classes

Class Name:	Number of Instances:
1	8
2	88
3	608
4	0
5	60
U	34

	798