```
##############################################################################
# Project:      Credit Scoring Analysis (petit example)
# Description:  Part 2 - Feature Selection and Profiling
# Data:         CleanCreditScoring.csv
# By:           Gaston Sanchez
# url:          www.gastonsanchez.com
#
# Note:
#     Check the preprocessing steps described in
#     Part1_CredScoring_Preprocessing.R
##############################################################################

# remember to change your working directory!!! (don't use mine)
# setwd("/Users/gaston/Documents/Gaston/StatsDataMining")

# read cleaned data set
dd = read.csv("CleanCreditScoring.csv", header=TRUE, stringsAsFactors=TRUE)

# ============================================================================
# Feature selection for continuous variables
# ============================================================================

# Let's start with the feature selection for continuous vars
# We'll apply Fisher's F-test between each cont. variable
# and "Status" (the response variable)

# select data frame with continuous variables
var.cont = subset(dd, select=c(Seniority, Time, Age, Expenses,
    Income, Assets, Debt, Amount, Price, Finrat, Savings))

# number of continuous variables
ncon = ncol(var.cont)

# create empty vector to store results
pval.cont = rep(NA, ncon)

# get the p-values from the F-tests
for (i in 1:ncon) {
    pval.cont[i] = oneway.test(var.cont[,i] ~ dd$Status)$p.value
}
# add names to pval.cont
names(pval.cont) = names(var.cont)

# by ordering the continuous variables according to their
# p-values, we get a ranking of associations with Status
# What variables could be discarded?
sort(pval.cont)

# we can get some charts to see what's going on
# let's produce some barplots in a single window
par(mfrow = c(3,4), mar = c(3,3,3,3))
for (i in 1:ncon)
{
  barplot(tapply(var.cont[,i], dd$Status, mean),
          main = paste("Means by", names(pval.cont)[i]), cex.main=0.9,
          border = NA, col = c("steelblue", "skyblue"))
  abline(h = mean(var.cont[,i]), col="gray40")
  legend(0, mean(var.cont[,i]), "global mean", bty="n", text.col="gray20")
}


# ============================================================================
# Feature selection for categorical variables
# ============================================================================
```

```
# The next step is to do the feature selection for the
# categorical variables. We'll apply chi-square tests
# between each categorized variable and Status

# select data frame with categorical variables
var.cat = subset(dd, select=c(ageR, seniorityR, timeR, expensesR, incomeR,
    assetsR, debtR, amountR, priceR, finratR, savingsR, Home,
    Marital, Records, Job))

# number of categorical variables
ncat = ncol(var.cat)

# create vector to store results
pval.cat = rep(0, ncat)

# calculate p-values from chi-square tests
for (i in 1:ncat) {
    pval.cat[i] = (chisq.test(var.cat[,i], dd$Status))$p.value
}

# add names
names(pval.cat) = names(var.cat)

# order categorical variables according
# to their dependence of Status
sort(pval.cat)


# ============================================================================
# Profiling based on continuous variables
# ============================================================================
# The next stage is a little bit trickier but it is
# also a much more interesting analysis: profiling!

# For continuous variables:
# hypothesis test comparing the mean of the group with the global mean

# We need to define a function that calculates the p-value of the
# test comparing the mean of the group with the global mean
# (this will only detect positive deviations, though)
WhoGetsWhatCon <- function(who, what)
{
    # 'who-gets-what'
    # who: continuous variable (eg income)
    # what: categorical variable (eg Status)

    # how many obs in each category
    nk <- as.vector(table(what))
    # total number of categories
    n <- sum(nk)
    # get who-mean for each category in what
    xk <- tapply(who, what, mean)
    # compare mean of each group with global mean
    # txk follows a t-student distribution
    txk <- (xk - mean(who)) / (sd(who)*sqrt((n-nk)/(n*nk)))
    # p-value t-distribution
    pxk <- pt(txk, n-1, lower.tail=F)
    pxk
}

# matrix to store results
pvalk.con = matrix(NA, ncon, nlevels(factor(dd$Status)))
for (i in 1:ncon) {
  pvalk.con[i,] = WhoGetsWhatCon(var.cont[,i], dd$Status)
```

```
}
colnames(pvalk.con) = levels(factor(dd$Status))
rownames(pvalk.con) = names(var.cont)
# show me the numbers
pvalk.con

# how would you profile "bad" clients? What about "good" clients?
# (i.e. what variables help the most to profile clients?)
sort(pvalk.con[,1])
sort(pvalk.con[,2])


# ================================================================================
# Profiling based on categorical variables
# ================================================================================

# hypothesis test comparing the mean of the group
# with the global mean (Status categories)

WhoGetsWhatCat <- function(who, what)
{
    # 'who-gets-what' where:
    # who: categorical (expl)
    # what: categorical (resp)

    # table
    what_who <- table(what, who)
    # total number
    n <- sum(what_who)
    # row margin
    pk <- rowSums(what_who) / n
    # column margin
    pj <- colSums(what_who) / n
    # proportional table by rows
    # prop.table(table(who, what), margin=1)
    pf <- what_who / (n*pk)

    # z-test comparing proportions
    pjm <- matrix(data=pj, nrow=dim(pf)[1], ncol=dim(pf)[2], byrow=T)
    dpf <- pf - pjm
    dvt <- sqrt(((1-pk)/(n*pk)) %*%t (pj*(1-pj)))
    zkj <- dpf / dvt
    # zkj follows a normal distribution
    pzkj <- pnorm(zkj, lower.tail=F)
    list(rowpf=pf, vtest=zkj, pval=pzkj)
}

# create list to store results
pvalk.cat = as.list(1:ncat)
for (i in 1:ncat) {
    pvalk.cat[[i]] = WhoGetsWhatCat(var.cat[,i], dd$Status)$pval
}
names(pvalk.cat) = names(var.cat)

for (k in 1:nlevels(dd$Status)) {
    print(paste("P-values of Status:", levels(dd$Status)[k]))
    for (j in 1:ncat) {
        print(names(pvalk.cat)[j])
        print(sort(pvalk.cat[[j]][k,]))
        cat("\n")
    }
    cat(rep("=", 50), "\n\n", sep="")
}
```

```
# exploratory plots
par(ask=TRUE)
par(mfrow=c(1,3))
n = nrow(dd)
for (i in 1:ncat)
{
  rowprof <- WhoGetsWhatCat(var.cat[,i], dd$Status)$rowpf
  marg <- table(var.cat[,i]) / n
  plot(marg, type="l", ylim=c(0,0.6),
       main=paste("Prop. of pos & neg by", row.names(pval.cat)[i]))
  lines(rowprof[1,], col="blue")
  lines(rowprof[2,], col="red")
  legend("topright", c("pos","neg"), col=c("blue","red"), lty=1)
}
```