Assignment 4

Brett Burk

Thursday, March 12, 2015

Contents

```
# Set the folder where the files are contained
folder <- "C:\\Users\\Brett\\Dropbox\\CUNY\\621\\Week6\\"
# Read in the csv and only use complete cases
jd.train <- read.csv(paste0(folder, "jury-training-data(1).csv"))
jd.train <- jd.train[complete.cases(jd.train),]

jd.public <- read.csv(paste0(folder, "jury-learning-data-public(1).csv"))
jd.public <- jd.public[complete.cases(jd.public),]

jd.private <- read.csv(paste0(folder, "jury-learning-data-private(1).csv"))
jd.private <- jd.private[complete.cases(jd.private),]
```

1 I rewrote the code, but it is based off previous work.

```
entropy <- function(d){
  # Creates a table of the values and returns each value as its probability
  prob <- table(d)/sum(table(d))
  # Finds the log base 2 of the probability table and multiplies it by the inverse of the probability t
  e <- sum((-prob) * log(prob, base=2))
  return (e)
}
```

```
infogain <- function(d, a){
  # Calculates the entropy of d
  ed <- entropy(d)
  # Splits the data frame on d
  split.df <- split(d, a)
  # Returns the mean value of the entropy function after its been split
  s <- mean(sapply(1:length(split.df), function(x) entropy(split.df[x])))
  return(ed - s)
}
```

2

```
checkgain <- function(input.df, target.var){
  # Creates a data frame without the target variable
  curr.df <- input.df[-target.var]
```

```
  # Creates an array of the target variable
  curr.var <- input.df[target.var]
  # Checks the information gain
  temp <- sapply(1:ncol(curr.df), function(x) infogain(curr.df[,x], curr.var))
  best.gain.pos <- which.max(temp)
  return(as.list(c(best.gain.pos, temp)))
}
```

3

```
split.tree <- function(inc.df, var.int){
  split.col <- checkgain(inc.df, var.int)[[1]]
  split.col.name <- names(inc.df[split.col])
  curr.split <- split(inc.df, inc.df[,split.col])
  temp <- sapply(1:length(curr.split), function(x) table(curr.split[[x]][,var.int])/nrow(curr.split[[x]]
  colnames(temp) <- names(curr.split)
  for (i in 1:length(curr.split)){
    curr.split[[i]] <- curr.split[[i]][,-split.col]
  }
  return(list(split.col.name, temp, curr.split, split.col))
}

first.split <- split.tree(jd.train, 5)
```

```
build.tree <- function(curr.df, var.int, curr.tree){
  if (ncol(curr.df) < 3){
    print(c(curr.tree, names(curr.df)[1])[-1])
  }
  else {
    temp.df <- split.tree(curr.df, var.int)
    var.int <- var.int - 1
    for(i in 1:(length(temp.df[[3]]))){
      temp.tree <- c(curr.tree, names(temp.df[[3]])[i])
      build.tree(temp.df[[3]][[i]], var.int, temp.tree)
    }
  }
}

# build.tree(jd.train, 5, array())
```

```
total.branches <- list()
build.tree <- function(curr.df, var.int, curr.tree){
  if (ncol(curr.df) < 3){
    temp.df <- split.tree(curr.df, var.int)
    var.int <- var.int - 1
    for(i in 1:(length(temp.df[[3]]))){
      temp.tree <- c(curr.tree, names(temp.df[[3]])[i])
```

```
      total.branches[[length(total.branches)+1]] <<- temp.tree[-1]
      print(temp.tree[-1])
    }
  }
  else {
    temp.df <- split.tree(curr.df, var.int)
    var.int <- var.int - 1
    for(i in 1:(length(temp.df[[3]]))){
      temp.tree <- c(curr.tree, names(temp.df[[3]])[i])
      build.tree(temp.df[[3]][[i]], var.int, temp.tree)
    }
  }
}

build.tree(jd.train, 5, array())
```

4

```
## [1] "Female"      "Divorced"    "Older Adult" "Employed"
## [1] "Female"       "Divorced"     "Older Adult"  "Not Employed"
## [1] "Female"        "Divorced"      "Younger Adult" "Employed"
## [1] "Female"        "Divorced"      "Younger Adult" "Not Employed"
## [1] "Female"      "Married"     "Older Adult" "Employed"
## [1] "Female"       "Married"      "Older Adult"  "Not Employed"
## [1] "Female"        "Married"       "Younger Adult" "Employed"
## [1] "Female"        "Married"       "Younger Adult" "Not Employed"
## [1] "Female"      "Single"      "Older Adult" "Employed"
## [1] "Female"       "Single"       "Older Adult"  "Not Employed"
## [1] "Female"        "Single"        "Younger Adult" "Employed"
## [1] "Female"        "Single"        "Younger Adult" "Not Employed"
## [1] "Male"        "Divorced"    "Employed"    "Older Adult"
## [1] "Male"          "Divorced"      "Employed"      "Younger Adult"
## [1] "Male"         "Divorced"     "Not Employed" "Older Adult"
## [1] "Male"          "Divorced"      "Not Employed"  "Younger Adult"
## [1] "Male"        "Married"     "Employed"    "Older Adult"
## [1] "Male"          "Married"       "Employed"      "Younger Adult"
## [1] "Male"         "Married"      "Not Employed" "Older Adult"
## [1] "Male"          "Married"       "Not Employed"  "Younger Adult"
## [1] "Male"        "Single"      "Employed"    "Older Adult"
## [1] "Male"          "Single"        "Employed"      "Younger Adult"
## [1] "Male"         "Single"       "Not Employed" "Older Adult"
## [1] "Male"          "Single"        "Not Employed"  "Younger Adult"
```

```
for(i in 1:length(total.branches)){

  if(total.branches[[i]][1] == "Female"){
    total.branches[[i]][5] <- names(which.max(table(subset(jd.train,
                                      gender == "Female" &
                                        marital == total.branches[[i]][2] &
                                        agegroup == total.branches[[i]][3] &
                                        employment == total.branches[[i]][4])[,5])))
  }
  else{
```

```
    total.branches[[i]][5] <- names(which.max(table(subset(jd.train,
                                      gender == "Male" &
                                        marital == total.branches[[i]][2] &
                                        agegroup == total.branches[[i]][4] &
                                        employment == total.branches[[i]][3])[,5])))
  }
}
#total.branches
```

```
jd.public[,6] <- "Guilty"

jd.public[jd.public$gender == "Female" &
          jd.public$marital == "Married" &
            jd.public$agegroup == "Older Adult",6] <- "Not Guilty"
jd.public[jd.public$gender == "Male",6] <- "Not Guilty"
jd.public[jd.public$gender == "Male" &
            jd.public$marital == "Married",6] <- "Guilty"
jd.public[jd.public$gender == "Male" &
            jd.public$marital == "Married" &
            jd.public$agegroup == "Younger Adult" &
            jd.public$employment == "Not Employed",6] <- "Not Guilty"

sum(jd.public[,5] == jd.public[,6])/nrow(jd.public)
```

```
## [1] 0.6806723
```

```
##### My guesses #####
bretts.classifiers <- jd.private

bretts.classifiers[,5] <- "Guilty"
bretts.classifiers[bretts.classifiers$gender == "Female" &
                     bretts.classifiers$marital == "Married" &
                     bretts.classifiers$agegroup == "Older Adult",5] <- "Not Guilty"
bretts.classifiers[bretts.classifiers$gender == "Male",5] <- "Not Guilty"
bretts.classifiers[bretts.classifiers$gender == "Male" &
                     bretts.classifiers$marital == "Married",5] <- "Guilty"
bretts.classifiers[bretts.classifiers$gender == "Male" &
                     bretts.classifiers$marital == "Married" &
                     bretts.classifiers$agegroup == "Younger Adult" &
                     bretts.classifiers$employment == "Not Employed",5] <- "Not Guilty"

write.csv(bretts.classifiers, paste0(folder, "jd.private.classified.csv"))
```

Nearly 70% accuracy is not fantastic, and it can be tweaked a bit by removing certain branches, but an ideal model isn't required for the deliverables (and an ideal model would be much easier to create using random forests, etc. anyway)