Patient6

Anuhya B S

2022-06-09

```
#Importing libraries
library('R.matlab')
```

```
## R.matlab v3.6.2 (2018-09-26) successfully loaded. See ?R.matlab for help.
```

```
##
## Attaching package: 'R.matlab'
```

```
## The following objects are masked from 'package:base':
##
##     getOption, isOpen
```

```
library(caTools)
library(e1071)
library(class)
library(tree)
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
#Loading Data
p1 <- readMat("data-science-P6.mat")

info <- as.data.frame(p1[2])
info <- t(info)
info <- as.data.frame(info)
lab.grp <-as.data.frame(matrix(nrow=0,ncol=1))
lab.wrd <-as.data.frame(matrix(nrow=0,ncol=1))
for (i in 1:360){
  lab.grp <-  rbind(lab.grp,info$cond[[i]])
  lab.wrd <-  rbind(lab.wrd,info$word[[i]])
}

p1.data <- p1$data
voxels <-as.data.frame(matrix(nrow=0,ncol=21764))
for (i in 1:360){
  voxels <-  rbind(voxels,p1.data[[i]][[1]])
}
```

```
# Principal Component Analysis for Feature Reduction
pr.out <- prcomp(voxels)
cumsum((pr.out$sdev^2)/sum(pr.out$sdev^2))
```

```
##   [1] 0.5926955 0.6523899 0.6914647 0.7246089 0.7496722 0.7653352 0.7788271
```

```
##   [8] 0.7916351 0.8023169 0.8106088 0.8181318 0.8245763 0.8308272 0.8359027
##  [15] 0.8406575 0.8453180 0.8494096 0.8531739 0.8567337 0.8600497 0.8630310
##  [22] 0.8658205 0.8684081 0.8708275 0.8729895 0.8751192 0.8772190 0.8792573
##  [29] 0.8811670 0.8830079 0.8847103 0.8863016 0.8878009 0.8892880 0.8907363
##  [36] 0.8921611 0.8935530 0.8948612 0.8961514 0.8973902 0.8985855 0.8997468
##  [43] 0.9008760 0.9019662 0.9030101 0.9040317 0.9050292 0.9059684 0.9068979
##  [50] 0.9077942 0.9086716 0.9095296 0.9103824 0.9111887 0.9119847 0.9127614
##  [57] 0.9135265 0.9142742 0.9150146 0.9157350 0.9164400 0.9171407 0.9178315
##  [64] 0.9185056 0.9191679 0.9198183 0.9204614 0.9210953 0.9217142 0.9223148
##  [71] 0.9229064 0.9234891 0.9240670 0.9246417 0.9252041 0.9257575 0.9263094
##  [78] 0.9268435 0.9273674 0.9278889 0.9284041 0.9289179 0.9294191 0.9299132
##  [85] 0.9304051 0.9308934 0.9313756 0.9318535 0.9323272 0.9327998 0.9332692
##  [92] 0.9337298 0.9341825 0.9346332 0.9350815 0.9355241 0.9359653 0.9363994
##  [99] 0.9368302 0.9372576 0.9376796 0.9380998 0.9385149 0.9389263 0.9393354
## [106] 0.9397418 0.9401467 0.9405505 0.9409498 0.9413464 0.9417390 0.9421275
## [113] 0.9425129 0.9428936 0.9432727 0.9436500 0.9440254 0.9443989 0.9447675
## [120] 0.9451336 0.9454994 0.9458637 0.9462246 0.9465836 0.9469387 0.9472911
## [127] 0.9476393 0.9479872 0.9483339 0.9486787 0.9490225 0.9493634 0.9497026
## [134] 0.9500393 0.9503754 0.9507089 0.9510404 0.9513714 0.9517000 0.9520277
## [141] 0.9523532 0.9526774 0.9529991 0.9533184 0.9536359 0.9539528 0.9542686
## [148] 0.9545820 0.9548945 0.9552048 0.9555145 0.9558217 0.9561272 0.9564323
## [155] 0.9567358 0.9570383 0.9573395 0.9576395 0.9579380 0.9582354 0.9585321
## [162] 0.9588263 0.9591198 0.9594116 0.9597022 0.9599911 0.9602796 0.9605673
## [169] 0.9608536 0.9611394 0.9614243 0.9617084 0.9619908 0.9622724 0.9625525
## [176] 0.9628317 0.9631102 0.9633867 0.9636628 0.9639381 0.9642124 0.9644854
## [183] 0.9647579 0.9650292 0.9652998 0.9655699 0.9658386 0.9661064 0.9663738
## [190] 0.9666402 0.9669058 0.9671696 0.9674324 0.9676945 0.9679555 0.9682154
## [197] 0.9684748 0.9687326 0.9689903 0.9692473 0.9695025 0.9697567 0.9700100
## [204] 0.9702627 0.9705148 0.9707660 0.9710162 0.9712656 0.9715143 0.9717616
## [211] 0.9720085 0.9722549 0.9724990 0.9727423 0.9729854 0.9732281 0.9734701
## [218] 0.9737109 0.9739507 0.9741899 0.9744280 0.9746650 0.9749017 0.9751373
## [225] 0.9753724 0.9756069 0.9758404 0.9760733 0.9763059 0.9765377 0.9767684
## [232] 0.9769987 0.9772285 0.9774577 0.9776864 0.9779143 0.9781419 0.9783687
## [239] 0.9785938 0.9788181 0.9790424 0.9792662 0.9794892 0.9797106 0.9799317
## [246] 0.9801523 0.9803721 0.9805916 0.9808091 0.9810263 0.9812433 0.9814599
## [253] 0.9816758 0.9818911 0.9821053 0.9823190 0.9825318 0.9827440 0.9829553
## [260] 0.9831662 0.9833761 0.9835851 0.9837933 0.9840011 0.9842089 0.9844155
## [267] 0.9846216 0.9848274 0.9850319 0.9852359 0.9854392 0.9856421 0.9858444
## [274] 0.9860462 0.9862469 0.9864469 0.9866465 0.9868452 0.9870429 0.9872398
## [281] 0.9874365 0.9876323 0.9878276 0.9880223 0.9882167 0.9884103 0.9886033
## [288] 0.9887950 0.9889863 0.9891771 0.9893673 0.9895560 0.9897438 0.9899314
## [295] 0.9901181 0.9903034 0.9904883 0.9906723 0.9908559 0.9910388 0.9912212
## [302] 0.9914026 0.9915838 0.9917648 0.9919446 0.9921232 0.9923009 0.9924781
## [309] 0.9926544 0.9928303 0.9930058 0.9931793 0.9933525 0.9935246 0.9936952
## [316] 0.9938655 0.9940351 0.9942042 0.9943720 0.9945390 0.9947053 0.9948694
## [323] 0.9950320 0.9951934 0.9953548 0.9955156 0.9956741 0.9958316 0.9959872
## [330] 0.9961410 0.9962930 0.9964436 0.9965927 0.9967414 0.9968890 0.9970341
## [337] 0.9971786 0.9973209 0.9974618 0.9976007 0.9977378 0.9978745 0.9980098
## [344] 0.9981432 0.9982764 0.9984084 0.9985380 0.9986662 0.9987932 0.9989186
## [351] 0.9990424 0.9991661 0.9992894 0.9994112 0.9995315 0.9996509 0.9997686
## [358] 0.9998846 1.0000000 1.0000000
```

```
pcs <- as.data.frame(pr.out$x[,1:300])
pcs$grp <- lab.grp$V1
```

```
#pcs$wrd <- lab.wrd$V1
```

```
# Splitting data into training and test data
set.seed(100)
#sample <- sample(1:nrow(pcs), 300)
pcs.train <- pcs[1:300,]
pcs.test <- pcs[301:360,]
#pcs.train <- subset(pcs, sample == TRUE)
#pcs.test  <- subset(pcs, sample == FALSE)
pcs.train.x <- subset(pcs.train, select = -c(grp))
pcs.train.labs <- pcs.train$grp
pcs.test.x <- subset(pcs.test, select = -c(grp))
pcs.test.labs <- pcs.test$grp
```

```
# Classification Algorithms

# Naive Bayes Classifier

nb.fit <- naiveBayes(grp ~ . , data = pcs.train)
nb.class <- predict(nb.fit,pcs.test.x)
nb.class
```

```
##  [1] tool      kitchen   buildpart insect    buildpart bodypart  bodypart
##  [8] kitchen   vehicle   kitchen   animal    vegetable furniture clothing
## [15] bodypart  clothing  vegetable animal    vehicle   vegetable clothing
## [22] clothing  clothing  animal    animal    animal    vegetable insect
## [29] kitchen   buildpart furniture buildpart kitchen   manmade   vegetable
## [36] furniture animal    insect    insect    building  animal    vegetable
## [43] kitchen   building  animal    clothing  tool      clothing  kitchen
## [50] building  bodypart  buildpart manmade   buildpart kitchen   animal
## [57] building  building  vegetable animal
## 12 Levels: animal bodypart building buildpart clothing furniture ... vehicle
```

```
confusion_mat.nb = as.matrix(table(Actual_Values = pcs.test.labs, Predicted_Values = nb.class))
print(confusion_mat.nb)
```

```
##              Predicted_Values
## Actual_Values animal bodypart building buildpart clothing furniture insect
##     animal         1        0        1         1        0         0      1
##     bodypart       0        0        1         0        3         0      0
##     building       1        0        1         0        1         1      0
##     buildpart      1        0        0         0        0         0      2
##     clothing       1        0        0         1        0         2      0
##     furniture      0        1        0         0        0         0      0
##     insect         1        0        0         0        1         0      0
##     kitchen        0        1        1         1        0         0      0
##     manmade        1        1        0         1        1         0      0
##     tool           2        0        1         1        0         0      0
##     vegetable      1        1        0         1        0         0      1
##     vehicle        1        0        0         0        1         0      0
##              Predicted_Values
## Actual_Values kitchen manmade tool vegetable vehicle
##     animal          0       0    1         0       0
##     bodypart        0       0    0         1       0
##     building        1       0    0         0       0
```

```
##     buildpart       1       0    0         0       1
##     clothing        1       0    0         0       0
##     furniture       0       1    0         3       0
##     insect          0       0    1         1       1
##     kitchen         2       0    0         0       0
##     manmade         1       0    0         0       0
##     tool            0       0    0         1       0
##     vegetable       1       0    0         0       0
##     vehicle         1       1    0         1       0
```

```
print(mean(nb.class == pcs.test$grp))
```

```
## [1] 0.06666667
```

```
# KNN

knn.pred <- knn(pcs.train.x, pcs.test.x, pcs.train.labs, k=5)

confusion_mat.knn = as.matrix(table(pcs.test.labs, knn.pred))
print(confusion_mat.knn)
```

```
##              knn.pred
## pcs.test.labs animal bodypart building buildpart clothing furniture insect
##     animal         1        1        0         0        0         1      0
##     bodypart       2        0        0         0        0         1      0
##     building       1        0        0         1        0         1      0
##     buildpart      2        0        0         0        0         0      0
##     clothing       1        0        1         1        0         0      0
##     furniture      0        0        0         0        1         1      0
##     insect         0        0        0         0        0         1      0
##     kitchen        1        0        0         0        2         0      0
##     manmade        0        1        0         1        1         0      1
##     tool           2        1        0         0        1         0      0
##     vegetable      2        1        0         0        1         1      0
##     vehicle        2        0        0         0        0         0      0
##              knn.pred
## pcs.test.labs kitchen manmade tool vegetable vehicle
##     animal          0       1    0         1       0
##     bodypart        0       1    1         0       0
##     building        0       1    1         0       0
##     buildpart       0       1    1         0       1
##     clothing        0       1    1         0       0
##     furniture       0       0    1         1       1
##     insect          0       1    2         1       0
##     kitchen         0       1    1         0       0
##     manmade         0       0    1         0       0
##     tool            0       1    0         0       0
##     vegetable       0       0    0         0       0
##     vehicle         1       0    0         1       1
```

```
print(mean(knn.pred == pcs.test$grp))
```

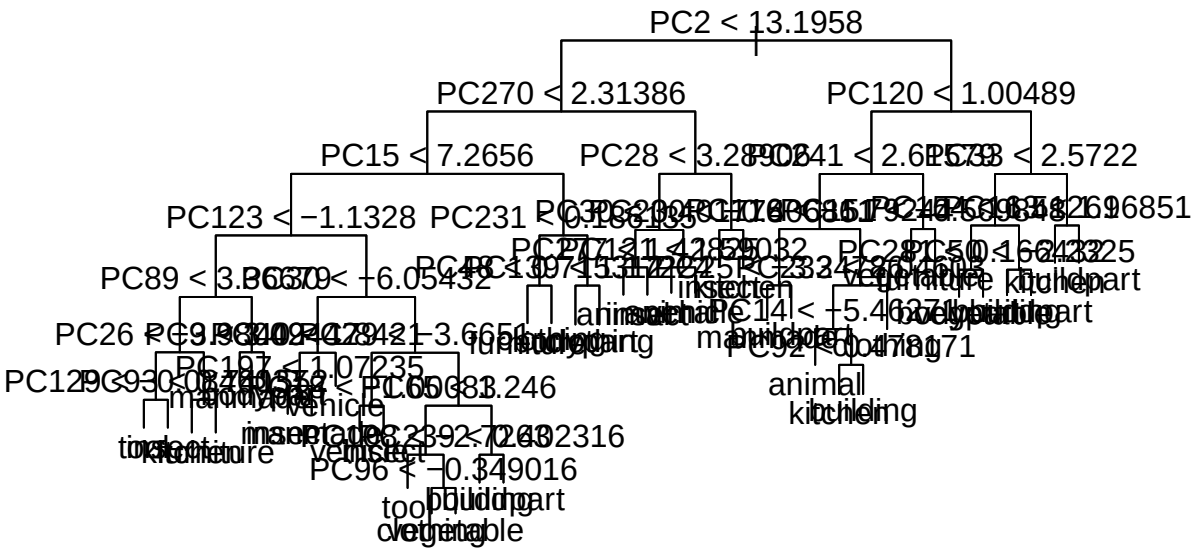
```
## [1] 0.05
```

```
# Decision Trees
set.seed(100)
tree.fit <- tree(as.factor(grp) ~ ., data = pcs.train)
```

```
summary(tree.fit)
```

```
##
## Classification tree:
## tree(formula = as.factor(grp) ~ ., data = pcs.train)
## Variables actually used in tree construction:
##  [1] "PC2"   "PC270" "PC15"  "PC123" "PC89"  "PC26"  "PC129" "PC93"  "PC9"
## [10] "PC30"  "PC40"  "PC197" "PC79"  "PC18"  "PC65"  "PC108" "PC96"  "PC239"
## [19] "PC231" "PC48"  "PC139" "PC28"  "PC277" "PC121" "PC230" "PC120" "PC241"
## [28] "PC116" "PC25"  "PC233" "PC14"  "PC92"  "PC151" "PC33"  "PC154" "PC281"
## [37] "PC50"  "PC163"
## Number of terminal nodes:  40
## Residual mean deviance:  2.023 = 526 / 260
## Misclassification error rate: 0.3833 = 115 / 300
```

```
plot(tree.fit)
text(tree.fit, pretty = 0)
```




```
tree.pred <- predict(tree.fit, newdata = pcs.test, type = "class")
tree.pred
```

```
##  [1] building  kitchen   bodypart  furniture animal    bodypart  building
##  [8] bodypart  vegetable building  vegetable clothing  kitchen   furniture
## [15] manmade   animal    building  vegetable clothing  bodypart  manmade
## [22] building  furniture kitchen   manmade   manmade   animal    clothing
## [29] clothing  bodypart  kitchen   tool      clothing  building  buildpart
## [36] kitchen   vegetable kitchen   buildpart buildpart vegetable clothing
```

```
## [43] insect    animal    vehicle   clothing  manmade   vehicle   insect
## [50] insect    furniture kitchen   animal    kitchen   manmade   kitchen
## [57] building  clothing  tool      kitchen
## 12 Levels: animal bodypart building buildpart clothing furniture ... vehicle
```

```
confusion_mat.dt = as.matrix(table(pcs.test.labs, tree.pred))
print(confusion_mat.dt)
```

```
##              tree.pred
## pcs.test.labs animal bodypart building buildpart clothing furniture insect
##     animal         0        1        0         0        2         0      0
##     bodypart       2        0        0         0        2         0      0
##     building       0        0        1         0        0         1      1
##     buildpart      0        0        0         1        1         1      0
##     clothing       1        0        1         0        0         0      0
##     furniture      0        0        2         1        1         0      0
##     insect         1        0        1         0        0         0      0
##     kitchen        0        2        0         0        0         0      1
##     manmade        0        0        1         0        0         1      1
##     tool           0        0        0         1        0         0      0
##     vegetable      0        1        1         0        1         0      0
##     vehicle        1        1        0         0        1         1      0
##              tree.pred
## pcs.test.labs kitchen manmade tool vegetable vehicle
##     animal          0       1    0         1       0
##     bodypart        0       0    0         0       1
##     building        1       0    0         1       0
##     buildpart       1       1    0         0       0
##     clothing        2       1    0         0       0
##     furniture       0       1    0         0       0
##     insect          0       1    0         2       0
##     kitchen         1       0    1         0       0
##     manmade         1       0    0         1       0
##     tool            2       0    1         0       1
##     vegetable       1       1    0         0       0
##     vehicle         1       0    0         0       0
```

```
print(mean(tree.pred == pcs.test$grp))
```

```
## [1] 0.06666667
```

```
# Random Forest

rf.fit <- randomForest(as.factor(grp) ~ ., data = pcs.train,, mtry = 80, importance = TRUE)
summary(rf.fit)
```

```
##                Length Class  Mode
## call              6   -none- call
## type              1   -none- character
## predicted       300   factor numeric
## err.rate       6500   -none- numeric
## confusion       156   -none- numeric
## votes          3600   matrix numeric
## oob.times       300   -none- numeric
## classes          12   -none- character
## importance     4200   -none- numeric
```

```
## importanceSD    3900   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y                300   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```
rf.pred <- predict(rf.fit, newdata = pcs.test, type = "class")
confusion_mat.rf = as.matrix(table(pcs.test.labs, rf.pred))
print(confusion_mat.rf)
```

```
##              rf.pred
## pcs.test.labs animal bodypart building buildpart clothing furniture insect
##     animal         0        0        0         1        0         0      0
##     bodypart       0        0        2         1        0         0      0
##     building       0        2        1         0        2         0      0
##     buildpart      0        0        0         1        0         0      0
##     clothing       1        0        1         0        1         0      1
##     furniture      0        0        0         1        0         2      0
##     insect         0        0        1         0        0         0      0
##     kitchen        0        0        0         0        0         1      0
##     manmade        1        0        2         0        0         0      0
##     tool           1        0        1         0        1         0      0
##     vegetable      2        1        0         0        0         0      0
##     vehicle        0        0        2         1        1         0      0
##              rf.pred
## pcs.test.labs kitchen manmade tool vegetable vehicle
##     animal          0       0    1         1       2
##     bodypart        0       0    0         1       1
##     building        0       0    0         0       0
##     buildpart       0       0    3         1       0
##     clothing        0       0    1         0       0
##     furniture       0       0    1         1       0
##     insect          0       1    2         0       1
##     kitchen         2       1    1         0       0
##     manmade         0       0    0         2       0
##     tool            0       0    1         0       1
##     vegetable       0       2    0         0       0
##     vehicle         0       1    0         0       0
```

```
print(mean(rf.pred == pcs.test$grp))
```

```
## [1] 0.1333333
```

```
manmade <- c("furniture", "clothing", "manmade", "tool", "kitchen", "vehicle", "building", "buildpart")
natural <- c("insect", "animal", "vegetable", "bodypart")
df_new <- within(pcs, {
 cls <- "manmade"
 cls[grp %in% manmade] <- "manmade"
 cls[grp %in% natural] <- "natural"
})
pcs$cls <- df_new$cls
```

```r
# Splitting data into training and test data
set.seed(100)
#sample <- sample(1:nrow(pcs), 300)
pcs.train <- pcs[1:300,]
pcs.test <- pcs[301:360,]
#pcs.train <- subset(pcs, sample == TRUE)
#pcs.test  <- subset(pcs, sample == FALSE)
pcs.train.x <- subset(pcs.train, select = -c(grp,cls))
pcs.train.labs <- pcs.train$cls
pcs.test.x <- subset(pcs.test, select = -c(grp,cls))
pcs.test.labs <- pcs.test$cls
```

```r
# Classification Algorithms

# Naive Bayes Classifier

nb.fit <- naiveBayes(cls ~ . , data = pcs.train)
nb.class <- predict(nb.fit,pcs.test.x)
nb.class
```

```
##  [1] manmade manmade manmade manmade manmade manmade natural manmade manmade
## [10] manmade manmade manmade manmade manmade natural manmade natural natural
## [19] manmade manmade natural manmade manmade manmade manmade manmade manmade
## [28] manmade manmade manmade manmade manmade manmade natural manmade manmade
## [37] natural natural natural manmade natural natural manmade manmade natural
## [46] manmade manmade manmade manmade manmade manmade manmade manmade manmade
## [55] manmade natural manmade manmade manmade manmade
## Levels: manmade natural
```

```r
confusion_mat.nb = as.matrix(table(Actual_Values = pcs.test.labs, Predicted_Values = nb.class))
print(confusion_mat.nb)
```

```
##              Predicted_Values
## Actual_Values manmade natural
##       manmade      32       8
##       natural      15       5
```

```r
print(mean(nb.class == pcs.test$cls))
```

```
## [1] 0.6166667
```

```r
# KNN

knn.pred <- knn(pcs.train.x, pcs.test.x, pcs.train.labs, k=9)

confusion_mat.knn = as.matrix(table(pcs.test.labs, knn.pred))
print(confusion_mat.knn)
```

```
##              knn.pred
## pcs.test.labs manmade natural
##       manmade      34       6
##       natural      16       4
```

```r
print(mean(knn.pred== pcs.test$cls))
```

```
## [1] 0.6333333
```

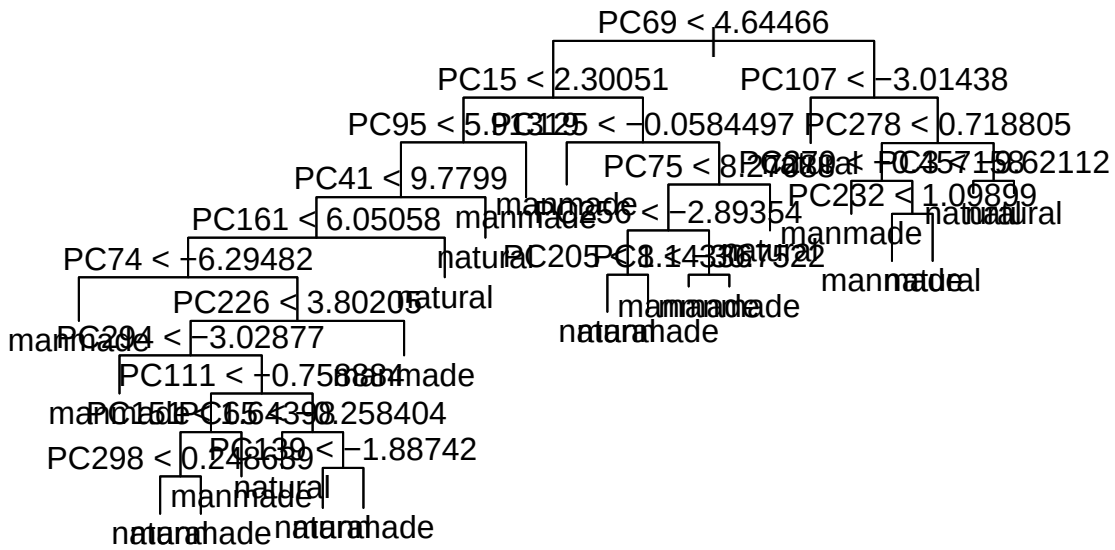
```r
# Decision Trees
set.seed(100)
tree.fit <- tree(as.factor(cls) ~ ., data = pcs.train)
```

```
## Warning in tree(as.factor(cls) ~ ., data = pcs.train): NAs introduced by
## coercion
```

```r
summary(tree.fit)
```

```
##
## Classification tree:
## tree(formula = as.factor(cls) ~ ., data = pcs.train)
## Variables actually used in tree construction:
##  [1] "PC69"  "PC15"  "PC95"  "PC41"  "PC161" "PC74"  "PC226" "PC294" "PC111"
## [10] "PC151" "PC298" "PC65"  "PC139" "PC125" "PC75"  "PC256" "PC205" "PC8"
## [19] "PC107" "PC278" "PC270" "PC232" "PC3"
## Number of terminal nodes:  24
## Residual mean deviance:  0.1998 = 55.15 / 276
## Misclassification error rate: 0.03333 = 10 / 300
```

```r
plot(tree.fit)
text(tree.fit, pretty = 0)
```




```r
tree.pred <- predict(tree.fit, newdata = pcs.test, type = "class")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```
tree.pred
```

```
##  [1] manmade natural manmade manmade manmade natural natural manmade manmade
## [10] natural manmade natural manmade natural natural natural manmade natural
## [19] natural manmade manmade manmade manmade natural manmade manmade manmade
## [28] manmade manmade manmade natural natural manmade natural natural manmade
## [37] natural manmade manmade manmade manmade natural natural manmade natural
## [46] manmade natural natural natural natural manmade manmade natural natural
## [55] natural manmade manmade manmade natural manmade
## Levels: manmade natural
```

```
confusion_mat.dt = as.matrix(table(pcs.test.labs, tree.pred))
print(confusion_mat.dt)
```

```
##              tree.pred
## pcs.test.labs manmade natural
##       manmade      19      21
##       natural      14       6
```

```
print(mean(tree.pred== pcs.test$cls))
```

```
## [1] 0.4166667
```

```
# Random Forest

rf.fit <- randomForest(as.factor(cls) ~ ., data = pcs.train,, mtry = 80, importance = TRUE)
summary(rf.fit)
```

```
##                 Length Class  Mode
## call               6   -none- call
## type               1   -none- character
## predicted        300   factor numeric
## err.rate        1500   -none- numeric
## confusion          6   -none- numeric
## votes            600   matrix numeric
## oob.times        300   -none- numeric
## classes            2   -none- character
## importance      1204   -none- numeric
## importanceSD     903   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y                300   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```
rf.pred <- predict(rf.fit, newdata = pcs.test, type = "class")
confusion_mat.rf = as.matrix(table(pcs.test.labs, rf.pred))
print(confusion_mat.rf)
```

```
##              rf.pred
## pcs.test.labs manmade natural
##       manmade      40       0
##       natural      10      10
```

```
print(mean(rf.pred== pcs.test$cls))
```

```
## [1] 0.8333333
```