Untitled

nuanjun zhao

10/21/2019

```
load("../output/feature_test.RData")
load("../output/feature_train.RData")
```

```
#the datasets after feature selection using random forest and forward
load("../output/Forward_features.RData")
load("../output/RandomForest_features.RData")
foward.index <- as.numeric(substr(forward_names,8,11))
rf.index <- as.numeric(substr(rf_names,8,11))
dat_trainf<-dat_train[foward.index]
dat_trainf$emotion_idx<-dat_train$emotion_idx
dat_trainr<-dat_train[rf.index]
dat_trainr$emotion_idx<-dat_train$emotion_idx
dat_testf<-dat_test[foward.index]
dat_testf$emotion_idx<-dat_test$emotion_idx
dat_testr<-dat_test[rf.index]
dat_testr$emotion_idx<-dat_test$emotion_idx
#the datasets after cutting off half face and using pca to reeduce dimension
load("../output/test_x.RData")
load("../output/test_y.RData")
load("../output/train_y.RData")
load("../output/train_x.RData")
test<-data.frame(test_x,emotion_idx=test_y)
train<-data.frame(train_x,emotion_idx=train_y)
```

```
# error <- as.numeric()
# for(i in 1:2){
# data.adaboost <- boosting(emotion_idx~., data=dat_train_1, mfinal=i)
# data.pred <- predict.boosting(data.adaboost,newdata = dat_test)
# error[i] <- data.pred$error
# }
# data.pred
```

Adaboost method:

```
library(adabag)
```

```
## Loading required package: rpart
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Loading required package: foreach
```

```
## Loading required package: doParallel
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
#adaboost method using random forest
B <- 100
data.adaboostr <- boosting(emotion_idx~., data=dat_trainr, mfinal=B)
data.predr <- predict.boosting(data.adaboostr,newdata = dat_testr)
data.predr1 <- predict.boosting(data.adaboostr,newdata = dat_trainr)
#training error
errorr1 <- data.predr1$error
errorr1
```

```
## [1] 0.8375
```

```
#testing error
errorr <- data.predr$error
errorr
```

```
## [1] 0.882
```

```
pred.indr <- as.numeric(rownames(data.predr$confusion))
data.predr$confusion[order(pred.indr),]
```

```
##                Observed Class
## Predicted Class  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
##               1   2  1  5  1  3  0  3  1  0  0  0  1  2  2  3  7  1  8  0
##               2   1 11  2  1  2  1  4 14 17  4  1  2  1  0  1  1  5  0  4
##               3   3  1  2  0  0  1  0  0  0  0  1  0  0  0  0  0  2  1  0
##               4   2  1  5  9  7 14  6  2  2  1  7  4  4  1  4  1  0  0  0
##               5   0  1  1  0  3  0  3  3  0  0  1  1  0  0  0  2  1  0  3
##               11  0  0  0  0  0  0  0  0  0  0  7  0  0  0  0  0  0  0  1
##               14  5  1  7  8 12  5  7  3  7 10 10 19 12 19  9 11  7  8 18
##               17  0  1  2  0  0  0  0  2  2  0  0  0  0  0  0  1  5  1  3
##               18  3  0  1  0  2  0  0  2  0  1  1  1  1  0  0  1  2  1  2
##                Observed Class
## Predicted Class 20 21 22
##               1   0  2  0
##               2   5  4  2
##               3   2  0  0
##               4   1  1  4
##               5   0  1  0
##               11  0  0  0
##               14  8 10 14
##               17  3  3  1
##               18  3  2  1
```

```
#adaboost method using forward
B <- 100
data.adaboostf <- boosting(emotion_idx~., data=dat_trainf, mfinal=B)
data.predf <- predict.boosting(data.adaboostf,newdata = dat_testf)
data.predf1 <- predict.boosting(data.adaboostf,newdata = dat_trainf)
#training error
errorf1<- data.predf1$error
errorf1
```

```
## [1] 0.8695
```

```
#testing error
errorf <- data.predf$error
```

```
errorf
```

```
## [1] 0.908
```

```
pred.indf<- as.numeric(rownames(data.predf$confusion))
data.predf$confusion[order(pred.indf),]
```

```
##                 Observed Class
## Predicted Class  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
##               1  3  0  2  1  0  3  2  0  0  0  0  0  1  2  0  4  0  2  2
##               2  1  7  1  1  6  2  8 13 10  1  2  4  0  0  2  4  7  1  7
##               3  4  5  5  4  3  1  5  5  8  3  2  4  3  3  4  3  6  6  4
##               4  0  0  0  1  0  2  1  0  0  0  0  0  0  0  0  1  0  0  0
##               5  1  0  0  0  2  0  0  0  0  0  0  0  0  0  0  1  1  2  0
##               8  1  2  1  0  5  0  1  5  0  0  0  0  0  0  0  1  1  2  0
##               11 0  0  0  1  0  1  1  0  0  0  6  0  0  0  0  0  0  0  0
##               14 6  3 16 11 13 12  5  4 10 12 18 20 16 17 11 10  8  6 18
##                 Observed Class
## Predicted Class 20 21 22
##               1  0  0  1
##               2 10  4  2
##               3  4  3  6
##               4  0  0  0
##               5  0  0  0
##               8  2  0  0
##               11 0  0  0
##               14 6 16 13
```

```
#adaboost method using half face and pca datasets
B <- 100
data.adaboost <- boosting(emotion_idx~., data=train, mfinal=B)
data.pred <- predict.boosting(data.adaboost,newdata = test)
data.pred1<- predict.boosting(data.adaboost,newdata = train)
#training error
error1 <- data.pred1$error
error1
```

```
## [1] 0.727
```

```
#testing error
error <- data.pred$error
error
```

```
## [1] 0.76
```

```
pred.ind <- as.numeric(rownames(data.pred$confusion))
data.pred$confusion[order(pred.ind),]
```

```
##                 Observed Class
## Predicted Class  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
##               1  7  0  3  2  0  0  1  0  1  2  1  1  1  4  2  1  2  0  2
##               2  0  9  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0
##               3  1  1 14  3  0  3  4  0  1  5  3  0  3  1  1  0  1  0  1
##               4  0  0  1  5  0  3  0  0  1  0  0  0  1  1  1  0  0  0  1
##               5  1  0  0  0 10  0  1  1  0  0  0  0  0  0  1  0  2  3  1
##               8  0 10  2  0  2  2  3 21  4  0  1  0  0  0  2  2  2  2  0
##               9  0  1  0  1  0  0  0  1  6  3  1  0  2  1  0  2  0  0  0
```

```
##              11  0  0  1 11  0  3  0  0  3  8  6  7  7  1  1  3  0  0  0
##              12  0  0  1  0  0  2  1  0  1  0  7  4  3  0  0  1  0  0  1
##              13  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##              14  1  0  3  2  4  2  9  0  0  4  1  4  1 17  9  1  2  1  7
##              16  5  3  4  5  0  4  3  1  1  0  1  1  1  0  2  8  0  0  2
##              17  5  1  1  0  8  2  5  2  2  0  0  0  0  1  4  2 13 11  2
##                Observed Class
## Predicted Class 20 21 22
##              1   2  1  3
##              2   0  0  0
##              3   0  4  4
##              4   0  0  0
##              5   1  1  0
##              8   7  1  1
##              9   0  4  0
##              11  4  7  6
##              12  1  0  0
##              13  0  0  0
##              14  4  5  3
##              16  2  1  1
##              17  2  3  1
```

svm method:

```
library(e1071)
```

```
#svm method using random forest
svm.modelr<-svm(emotion_idx~.,data=dat_trainr)
svm.predr<-predict(svm.modelr,dat_testr)
svm.predr1<-predict(svm.modelr,dat_trainr)
svm.predr
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   1   1   1  16   1  22  18  15   1   1   3   1  16  15   5  17   2   8
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   3   2   8   8   2   2   9  10   2   2   2   3   2   5   2  16   3   1
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  10   3  14  19   3  20   3  15  20   4  10  21  16   1  20   3  14   9
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   6  20  18   5   4   6   4  13   4   4   4   6  14   6  10  18   4  14
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   3   3   3  14  17   5  18  16  12  15   2   5   7   6   5   6   3   2
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   5   1  14   5   5   5  17   7   1  18  14  18  15  17  15  17   4   4
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##   4  15   4   4   5   4   6  14   4   8  22   5  13  10   6   8  14  14
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
##  14   2   3  12   2   7  13   7   4   9   7   3   3   6   7   4  16  22
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
##   3  16  15   3  15   3  17   8   2   8   8  20   8   8   8   8   8   9
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##  13   9   7   3   8   2   2   8   2   8   8   8   2  21   8   9   9   3
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
```

```
##  17   2   2   9   2  21   2   9   2   9   9   2  20   2  11  20  17   2
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
##   9  22   8  21   3  20  21  13  10  10  19  13  22  10  12  13   7  15
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
##  17  21  14  13  21   1  11  12  21  11  21  11  11  11  11  11  11   6
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
##   6  20  20  14  14   6   4  17   2  17  22  14   4  20   1  13  11  12
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
##  13   4  13  12   1  14  14  14  14   5  19  12  15  14  12  10  13  12
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
##  11  14  14   3  14   4   9  20   4  14   1   4  14  22  14  14  22  13
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
##  14  15  13  21  14  14  20  21  14  12  14  15  14  21  12  15  13  14
## 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
##   4  18   4  22   1  12   1  14  14  14  14   1  14  13   4  13   1  14
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
##   5  14   1  14   1  15  17  14   7  14  17  18   3  16   3  16  17  18
## 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##   3  10   1  13  16   6   3  14  14  18  14   3  18  18  16  21  21  19
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
##  17  18   8  17  17  10  21   8   8   8  17  17   3  16  14  17  20  14
## 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
##  14   2  17   5  21  17   1  21   5  18   1  12   1  18  17  17  17   1
## 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
##   1   3  11   5  16   1   3  21  19  19  10   9  21  14   9  17  10  10
## 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
##  12  13  13  17   1   3   6  14  10  14  14  10  22   3  16  14  15  14
## 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
##   3  17   3   9   3   9  17  17  17  12  21  22  17  20  10  21  18  10
## 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468
##  21  14  21  15  20  19   3   3   7   8   9  19   1  14  10  21  13   3
## 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486
##   9  22   6  14  14  20  10  14   4  21  21  17  19  10  10   1   3  20
## 487 488 489 490 491 492 493 494 495 496 497 498 499 500
##   4  14  10  13  17   4  20  22  12  18  20  13  14   9
## Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

```
#training accuracy
mean(svm.predr1==dat_trainr[,dim(dat_trainr)[2]])
```

```
## [1] 0.407
```

```
#testing accuracy
mean(svm.predr==dat_testr[,dim(dat_testr)[2]])
```

```
## [1] 0.216
```

```
#svm method using forward
svm.modelf<-svm(emotion_idx~.,data=dat_trainf)
svm.predf<-predict(svm.modelf,dat_testf)
svm.predf1<-predict(svm.modelf,dat_trainf)
svm.predf
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
##  1  1 18 16  5  1 17 15 18  1  3  3 14 16  5  3  2 17
## 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
```

```
##  20   2   7   8   2   2   2   3   2   9   2   7  22   1   2  17   3   1
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  10   3   7  20   3  17   3  16   9  16  16   3   1   1  14   3  15   8
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  18  20   7   5   4   7   4   4   4  21   4   6  14   5  10   1  13  13
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  14  10  17  14   3   5  18   5   1   5   2   5   7   6   5   6  17  20
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   5   1  17   5  15   1  17  14   1   1  14   7  14  17  14  17   4   4
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##  13   4   4   4   6   6   6  14   4  11  22   1  14  14  14  17  14  14
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
##  14   2   3  14   3  16   7   7   4   7   7   1   2   6   7   4   1   5
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
##   1  16  15   3  14  17   2   8   2   8   7   7   8   2   9   8   8   3
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##  18   9   7  18   2   2   8  17   5  17   6   9  17  20  18   9   2  18
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
##  17   9   2   9   2  21   9   9   2   9   9   8  20   2   5  11  17  13
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
##   9  14   9  21  17  20  20  13  13  10  13  13  22  10  13  11   7   3
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
##  14  21  14  13  21  13  11  12  12  11  11  11  11  11  11  11  11  14
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
##  11  20  12  13  14   4  14  14   4  14  21  14  14  10   1  13  11  12
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
##  13  12  13   4   1  14  14  14  14  14  10   6  14  14   9  16  13  12
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
##  13  14  14   3  14  21   7  10  14  14   1  12  14  12  12  14  17  12
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
##   6  17   4  22  20  12  18  21  14  13  14  15  14  21  14  14   1  14
## 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
##  14  17  13  22   1  14   1   6   1  14  14   1  18   3   3  13   1  14
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
##   5   1   1  14   1  17  17   4   4  14   3  18   3   1  18  16   2  18
## 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##  18   7   1   6  16   4  16  14  14  16   1  12  18  14   4  21  17  14
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
##  17   1  18  18  17  10  19   9   9   2  17  17   3   8  16  17  21   3
## 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
##   4   2  20   5   4   8   1  22   5   1   1  13   1   1  17  16  18  18
## 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
##  18  17  11   1  13   3   3   3  19   3  10  21  21  14   2   8  10  19
## 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
##  14  13  20   5   1  16  14  14  10   3   5  14  17   3  10  14   3  14
## 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
##  20   3   3   7   3   9   8  17  17   4   9  20  17   3  19  14   1   3
## 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468
##  21  13  21  17  20   3   3   3  13   9   8  13   1  14  10   7  14  20
## 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486
##   2   3  14  14  14  20  10   7   4  21  21  22  19  19  10   1  10  14
## 487 488 489 490 491 492 493 494 495 496 497 498 499 500
##  16  20  10   7  12  17  12  17  12  10  10  20  21  18
## Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

```
#training accuracy
mean(svm.predf1==dat_trainf[,dim(dat_trainf)[2]])
```

```
## [1] 0.397
```

```
#testing accuracy
mean(svm.predf==dat_testf[,dim(dat_testf)[2]])
```

```
## [1] 0.19
```

```
#svm method using half face and pca datasets
svm.model<-svm(emotion_idx~.,data=train)
svm.pred<-predict(svm.model,test)
svm.pred1<-predict(svm.model,train)
svm.pred
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   1   3   1   1   1   1   1   1   1   1   1   3   1   3   1   1   1  22
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   1   1   2   2   2   8   2   2   2   2   2   2  20   9   9   2   8   9
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   2   9   9   8   2   2   9   2   2   3  22  21   4   3   3   3   3   3
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   3   3   3   3   3   1   3   3  13   1   3   1   3  11  10   1  21  10
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   3   3  13   4   4   4   4   4   4   3   4   4  10  22   6   4  13   4
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##  13  11  12  13  11   4   3   1  10  19  13   1  13   4  11   5   5  18
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##   5  15  18   5   5   8  15   5  15   1  20  14   1  16  14  18   5   5
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
##  15   5  14  12   3   6  12   6   6  21   6   6   6   6  11  12   6  16
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
##  13  12  19  11  12  16   7   7  21   7   7   7   6   7  16   6   4   7
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   3  16   7   7  18   7  19   1  12  19  19   8  17  17  21   8   8   8
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
##   8   8   8   8   8   8   8   8   8   8   8  17   8   8   8  18   8   8
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
##  19  21   8   2   8   9  10   9   9   9   2   9   7   4   9   8   6   9
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
##  21   9   9   9   9   9   8   9   8  22  10  10   3  10  10  11  10  10
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
##   3  10   8  10  19  13  22  13   3   1  13  10  10  12  12  12  12  21
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
##  12  12  13  10  10  11  16  10   6   6  11   4   9  13  22  11  20  12
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
##  12  10  12  12  12  12  13  12  11  12  14  21   3  12  11  13   4   4
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
##   4  11  12   4   1  13   4  13  11  12  12   6  12  11  11  21  13  14
## 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
##  14  14  14  15  14  14  14  14  14  14  14  14  14  14  14  14  14  14
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
##  14  14  20  14  14  14  19  15  14  13  15  15  14  14  21  15  15  15
## 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
```

```
##  18  13  14  13  19  17  16  14  14   1  18  18  16  16  16  16  12   8
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
##  16  16   4  12  16  16  11  16  16   7  13  11  16  20   3  17  21  17
## 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
##  17  18  17   7  18  19  17   8  17  14  21  19  19  13  20  18   8  16
## 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
##  20  18  18   7  18  18  18   7  18  17  18  16  20   1  14  14  17  22
## 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
##  19  19   7  19  21  21  20  19  20  19  21   7  18   1  19  19  16  18
## 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
##  20   9  20  21  19  21  21  20  11  21  20  20  11   7  14  21  20  16
## 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468
##  21  22  16  17  21  21  21  21  21  22  17  21  19  19  20   7  21  20
## 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486
##  20   7  13  21  14  19   4  13  21  19  21  19   1  19  22  11   2  22
## 487 488 489 490 491 492 493 494 495 496 497 498 499 500
##  11  22  19  19  22  11  22  10  11  11   3  10  19   3
## Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

```
#training accuracy
mean(svm.pred1==train[,dim(train)[2]])
```

```
## [1] 0.8935
```

```
#testing accuracy
mean(svm.pred==test[,dim(test)[2]])
```

```
## [1] 0.454
```

Conclusion: For the features selected by random forest and forward methods, we can see the test accuracy is pretty low.