```
> library("data.table")
> ohsumed.dt <- fread("data.txt", sep = " ")
> ohsumed.dt <- ohsumed.dt[, -c(48,49), with = FALSE] # remove residual cols
> col.names <- paste(rep("C", 45), seq(1:45), sep = "")
> col.names <- c("r", "qid", col.names, "docid")
> setnames(ohsumed.dt, col.names)
> letor <- ohsumed.dt
> for (j in names(letor)) set(letor,j=j,value= gsub(".*:", "", letor[[j]]))
> setcolorder(letor, c(names(letor)[1:2], names(letor)[48], names(letor)[3:47]))
> for (j in names(letor)) set(letor,j=j,value= as.numeric(gsub(",","",letor[[j]])))
> head(letor[,c(1:10), with = FALSE])
   r qid  docid C1       C2       C3       C4       C5       C6       C7
1: 0   1 244338  5 3.465736 0.500000 0.476551 37.33056 11.43124 37.30702
2: 2   1 143821  3 2.079442 0.600000 0.546965 37.33056 11.43124 37.30702
3: 0   1 285257  2 1.386294 0.333333 0.308301 37.33056 11.43124 37.30702
4: 0   1 201684  1 0.693147 0.166667 0.154151 37.33056 11.43124 37.30702
5: 1   1  48192  0 0.000000 0.000000 0.000000 37.33056 11.43124 37.30702
6: 2   1 111457  3 2.079442 0.375000 0.353349 37.33056 11.43124 37.30702
> length(unique(letor$qid)) ## number of queries
[1] 106
> length(unique(letor$docid))
[1] 14430
> sum(duplicated(letor)) > 0
[1] FALSE
> sum(apply(as.matrix(letor), 2, function(x) sum(is.na(x)))) > 0
[1] FALSE
> library("caret")
> suppressMessages(require(caret)) ## http://topepo.github.io/caret/index.html
> highly.corelated <- caret::findCorrelation(cor(letor[,4:48, with = FALSE], use = "na.or.complet
e"),cutoff=0.8)
> letor[is.finite(rowSums(letor))]
       r qid  docid C1       C2       C3       C4       C5        C6       C7
    1: 0   1 244338  5 3.465736 0.500000 0.476551 37.33056 11.431241 37.30702
    2: 2   1 143821  3 2.079442 0.600000 0.546965 37.33056 11.431241 37.30702
    3: 0   1 285257  2 1.386294 0.333333 0.308301 37.33056 11.431241 37.30702
    4: 0   1 201684  1 0.693147 0.166667 0.154151 37.33056 11.431241 37.30702
    5: 1   1  48192  0 0.000000 0.000000 0.000000 37.33056 11.431241 37.30702
   ---
16136: 2 106 337888  0 0.000000 0.000000 0.000000 30.12974  8.809276 30.06821
16137: 0 106  36526  0 0.000000 0.000000 0.000000 30.12974  8.809276 30.06821
16138: 0 106  63379  0 0.000000 0.000000 0.000000 30.12974  8.809276 30.06821
16139: 1 106 298134  0 0.000000 0.000000 0.000000 30.12974  8.809276 30.06821
16140: 0 106 299223  0 0.000000 0.000000 0.000000 30.12974  8.809276 30.06821
            C8        C9       C10       C11       C12       C13       C14
    1: 2.006924 25.062451 14.212437 22.580680 3.117095 -36.71723 -32.63940
    2: 2.130849 15.659869 10.905215 18.428062 2.913875 -42.14207 -39.56138
    3: 1.163112  9.538162  6.073774 10.573272 2.358329 -46.53021 -46.47484
    4: 0.703238  6.121707  4.308803  6.800513 1.916998 -48.25346 -50.20441
    5: 0.000000  0.000000  0.000000  0.000000 0.000000 -52.20285 -54.28562
   ---
16136: 0.000000  0.000000  0.000000  0.000000 0.000000 -39.39546 -40.30405
16137: 0.000000  0.000000  0.000000  0.000000 0.000000 -39.39546 -40.30405
16138: 0.000000  0.000000  0.000000  0.000000 0.000000 -39.39546 -40.30405
16139: 0.000000  0.000000  0.000000  0.000000 0.000000 -39.39546 -40.30405
16140: 0.000000  0.000000  0.000000  0.000000 0.000000 -39.39546 -40.30405
            C15 C16      C17      C18      C19      C20      C21     C22
    1: -32.63940  38 9.757305 0.230303 0.223611 28.30307 9.340024 24.8098
    2: -39.56138  16 6.643790 0.320000 0.306589 28.30307 9.340024 24.8098
    3: -46.47484   8 4.682131 0.190476 0.186621 28.30307 9.340024 24.8098
    4: -50.20441  14 6.222576 0.254545 0.246105 28.30307 9.340024 24.8098
    5: -54.73754  10 5.780744 0.105263 0.104279 28.30307 9.340024 24.8098
   ---
16136: -40.95962   0 0.000000 0.000000 0.000000 22.06210 7.214033 19.2672
16137: -40.95962   0 0.000000 0.000000 0.000000 22.06210 7.214033 19.2672
16138: -40.95962   0 0.000000 0.000000 0.000000 22.06210 7.214033 19.2672
16139: -40.95962   0 0.000000 0.000000 0.000000 22.06210 7.214033 19.2672
16140: -40.95962   0 0.000000 0.000000 0.000000 22.06210 7.214033 19.2672
            C23       C24      C25      C26      C27       C28       C29
    1: 0.851304 158.07287 5.076668 30.14395 3.405984 -37.41985 -37.65477
    2: 1.048966  63.44905 4.900453 28.62004 3.354107 -36.65880 -36.65880
```

```
    3: 0.643924  29.17422 4.131035 26.03933 3.259608 -38.01927 -37.75786
    4: 0.893261  57.57792 4.681687 27.55313 3.316116 -37.39692 -37.51615
    5: 0.412808  40.85401 3.402132 26.70044 3.284680 -36.16633 -37.20667
   ---
16136: 0.000000   0.00000 0.000000  0.00000 0.000000 -45.12191 -41.70991
16137: 0.000000   0.00000 0.000000  0.00000 0.000000 -45.12191 -41.70991
16138: 0.000000   0.00000 0.000000  0.00000 0.000000 -45.12191 -41.70991
16139: 0.000000   0.00000 0.000000  0.00000 0.000000 -45.12191 -41.70991
16140: 0.000000   0.00000 0.000000  0.00000 0.000000 -45.12191 -41.70991
             C30 C31       C32      C33      C34      C35      C36      C37
    1: -36.90575  43 10.585220 0.245714 0.238503 27.70163 9.139690 23.81948
    2: -35.57552  19  7.272398 0.345455 0.330244 27.70163 9.139690 23.81948
    3: -36.73825  10  5.375278 0.208333 0.203710 27.70163 9.139690 23.81948
    4: -36.31332  15  6.356108 0.245902 0.237425 27.70163 9.139690 23.81948
    5: -35.87141  10  5.780744 0.102041 0.101116 27.70163 9.139690 23.81948
   ---
16136: -44.09045   0  0.000000 0.000000 0.000000 21.33253 7.034527 18.62712
16137: -44.09045   0  0.000000 0.000000 0.000000 21.33253 7.034527 18.62712
16138: -44.09045   0  0.000000 0.000000 0.000000 21.33253 7.034527 18.62712
16139: -44.09045   0  0.000000 0.000000 0.000000 21.33253 7.034527 18.62712
16140: -44.09045   0  0.000000 0.000000 0.000000 21.33253 7.034527 18.62712
             C38       C39      C40      C41      C42       C43       C44
    1: 0.880399 173.44652 4.842425 31.19000 3.440098 -36.62747 -36.94917
    2: 1.103254  73.79281 4.726054 28.96880 3.366219 -36.18067 -36.30400
    3: 0.677402  35.10113 3.935930 27.07577 3.298639 -37.35149 -37.29503
    4: 0.853638  61.48781 4.218229 27.00540 3.296037 -37.53975 -37.76682
    5: 0.391696  39.91445 2.970979 26.83246 3.289612 -36.32375 -37.38072
   ---
16136: 0.000000   0.00000 0.000000  0.00000 0.000000 -45.27840 -41.69412
16137: 0.000000   0.00000 0.000000  0.00000 0.000000 -45.27840 -41.69412
16138: 0.000000   0.00000 0.000000  0.00000 0.000000 -45.27840 -41.69412
16139: 0.000000   0.00000 0.000000  0.00000 0.000000 -45.27840 -41.69412
16140: 0.000000   0.00000 0.000000  0.00000 0.000000 -45.27840 -41.69412
             C45
    1: -35.82978
    2: -35.19596
    3: -35.99993
    4: -36.63202
    5: -36.06247
   ---
16136: -44.23520
16137: -44.23520
16138: -44.23520
16139: -44.23520
16140: -44.23520
> suppressMessages(require(verification)) # for arear under the ROC
Warning messages:
1: package 'verification' was built under R version 3.4.2
2: package 'fields' was built under R version 3.4.2
3: package 'spam' was built under R version 3.4.2
4: package 'dotCall64' was built under R version 3.4.2
5: package 'maps' was built under R version 3.4.2
6: package 'CircStats' was built under R version 3.4.2
7: package 'dtw' was built under R version 3.4.2
8: package 'proxy' was built under R version 3.4.2
> suppressMessages(require(ROCR))         # for performance measurements
> suppressMessages(require(randomForest)) # rf, gbm and svm are also
> suppressMessages(require(gbm))          # included in the caret.
> suppressMessages(require(e1071))        # package
> letor$r.bin <- ifelse(letor$r > 1, 1, 0)
> setcolorder(letor, c(names(letor)[1:3], names(letor)[49], names(letor)[4:48]))
> set.seed(921981)
> rIdx <- sample(length(unique(letor$qid)), length(unique(letor$qid))*0.1)
> rqid <- unique(letor$qid)[rIdx]
> Test <- as.data.frame(subset(letor, qid %in% rqid))
> Train <-  as.data.frame(subset(letor, !(qid %in% rqid))[,c(2,4:48), with = FALSE])
> qid.train <- unique(Train$qid)
> nsamples <- 4 # ideally a multiple of the number of cores
> set.seed(921980)
> rqid2 <- qid.train[sample(length(qid.train), 10, TRUE)]
```

```
> train <- subset(Train, !(qid %in% rqid2))[,2:ncol(Train)]
> suppressMessages(library(doParallel))
> suppressMessages(library(foreach))
> registerDoParallel(cores = 4)
> n = floor(length(qid.train)/nsamples)
> err.vect <- rep(NA, nsamples)
> cv.train <- foreach (i = 1:nsamples) %dopar% {
+     s = (i-1) * n+1
+     f = i*n
+     sub <- s:f
+     cv.qidtrain <-  qid.train[-sub]
+     subset(Train, qid %in% cv.qidtrain)[,2:ncol(Train)]
+ }
>
> cv.test <- foreach (i = 1:nsamples) %dopar% {
+     s = (i-1) * n+1
+     f = i*n
+     sub <- s:f
+     cv.qidtest <- qid.train[sub]
+     subset(Train, qid %in% cv.qidtest)[,2:ncol(Train)]
+ }
>
> ## The following two lines should be run to adequately tune the cost and gamma.
> ## However it takes too long in my personal computer.
> ## bestPar <- foreach() tune.svm(x = cv.train[[1]][,-1], y = as.factor(cv.train[[1]][,1]), gamm
a = 2^(-4:4), cost = 2^(2:10))
>
> ## run folds in pararell
> modelDataSVM <- foreach(i = 1:nsamples, .packages = c("e1071")) %dopar% {
+
+     svm(x = cv.train[[i]][,-1], y = as.factor(cv.train[[i]][,1]),
+ probability=TRUE, cost=100, gamma=1, na.action =na.omit)
+
+ }
>
> predictSVM <- foreach(i = 1:nsamples, .packages = c("e1071")) %dopar% {
+
+     predict(modelDataSVM[[i]], cv.test[[i]][,-1], probability=TRUE)
+
+ }
>
> ## AUC
> for (i in 1:nsamples) {
+
+     prob <- attr(predictSVM[[i]], "probabilities")[,2]
+     err.vect[i] <- roc.area(cv.test[[i]][,1], prob)$A
+     cat("AUC for fold ", i, ":", err.vect[i], "\n")
+
+ }
AUC for fold  1 : 0.6380102
AUC for fold  2 : 0.5812451
AUC for fold  3 : 0.5442167
AUC for fold  4 : 0.5676883
```