Task 2

Task 1

Introduction

For task 2, we are going to deal with a dataset containing information about 4601 webmails. We have 48 variables describing the frequency of some specific words like "remove" in each observation, 6 variables describing the frequency of some specific chars like "$" in one observation, and three variables, capital_run_length_longest, capital_run_length_average and capital_run_length_total, describing the length of the longest uninterrupted sequence of capital letters, the average length of uninterrupted sequences of capital letters, and the total number of capital letters in each observation respectively. We also have a variable called spam, which indicates whether this webmail is a spam with 0 and 1, where 1 for spam, and 0 for not spam. Here all our variables are numeric type. Our task is to use these 57 attribute variables to classify whether a webmail is spam.

Methodology

In order to validate the accuracy of our methods, we firstly divide our dataset into a train set, which contains 2500 observations, and a test set, which contains 2101 observations. We use the train set to train our models, and then apply it to the test set to validate its accuracy.

Results

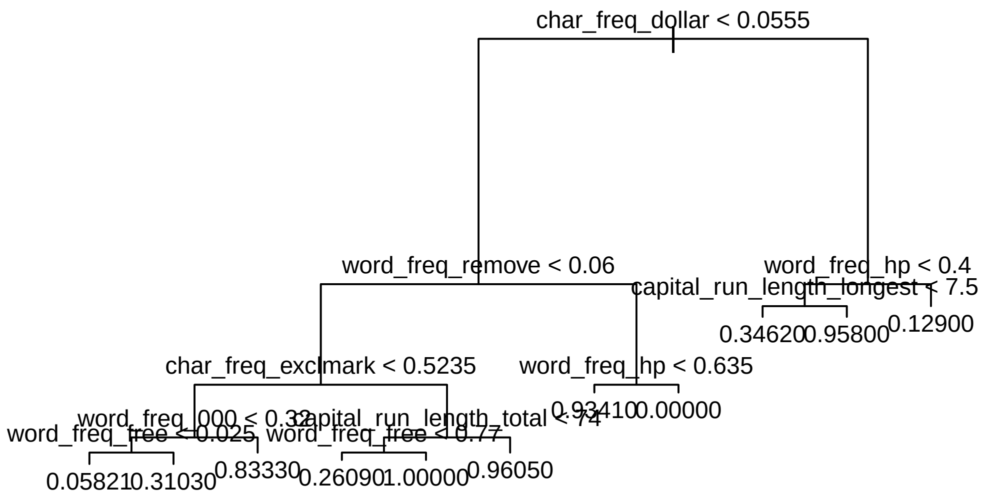
Classification Trees

In this part, we are going to discuss the results obtained by complex tree model and pruned tree model. We begin with construct a complex tree model by dividing our observation into small non-overlapping regions according to some numerical criteria. Here we split our dataset until each leaf of our classification tree contains only less then 2 observations. The method used here is recursive binary splitting.

```
    #grow complex tree using deviance as criterion
tree.mod=tree(spam~.,data.train,control=tree.control(nobs=2500,minsize=2,mincut=1),split="deviance")
summary(tree.mod)
```

Regression tree: tree(formula = spam ~ ., data = data.train, control = tree.control(nobs = 2500, minsize = 2, mincut = 1), split = "deviance") Variables actually used in tree construction: [1] "char_freq_dollar" "word_freq_remove"
[3] "char_freq_exclmark" "word_freq_000"
[5] "word_freq_free" "capital_run_length_total"
[7] "word_freq_hp" "capital_run_length_longest" Number of terminal nodes: 11 Residual mean deviance: 0.06945 = 172.9 / 2489 Distribution of residuals: Min. 1st Qu. Median Mean 3rd Qu. Max. -0.96050 -0.05821 -0.05821 0.00000 0.04203 0.94180

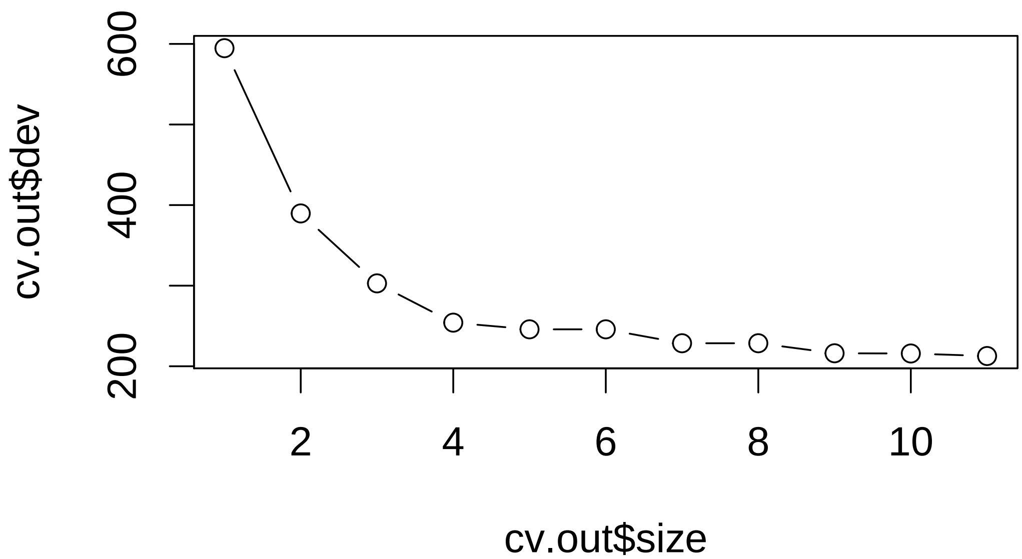
```r
#plot tree
plot(tree.mod)
text(tree.mod,pretty=0,cex=0.8)
```



We can see clearly here that criteria concerning the frequency of *"$"*, *"remove"*, *"!"*, *"hp"*, *"our"*, *"free"*, *"edu"* as well as the length of the longest uninterrupted sequence of capital letters are used for splitting. It's actually quite reasonable, because from our own experience, spam webmails are always advertisements on money related topics or education related topics, and are always filled with words in capital letters, together with exclamation symbols, to draw attention.

Since the process of recursive binary splitting may lead to overfitting, where we obtain a complex tree with a good fit on the training data, but with a poor performance on test data, we introduce the process of tree pruning. Actually, a smaller tree with fewer splits may have a lower variance at the cost of acceptable little bias. In order to decide the optimal tuning parameter which leads to both much lower variance and acceptable bias, we use cross-validation to make a selection. In fact, the least cross-validation error implies the least probability of overfitting, as it's also a train-test process.

```r
#use cross-validation to select tuning parameter for pruning the tree
set.seed(0829539)
cv.out=cv.tree(tree.mod,K=5)
par(cex=1.4)
plot(cv.out$size,cv.out$dev,type='b')
```

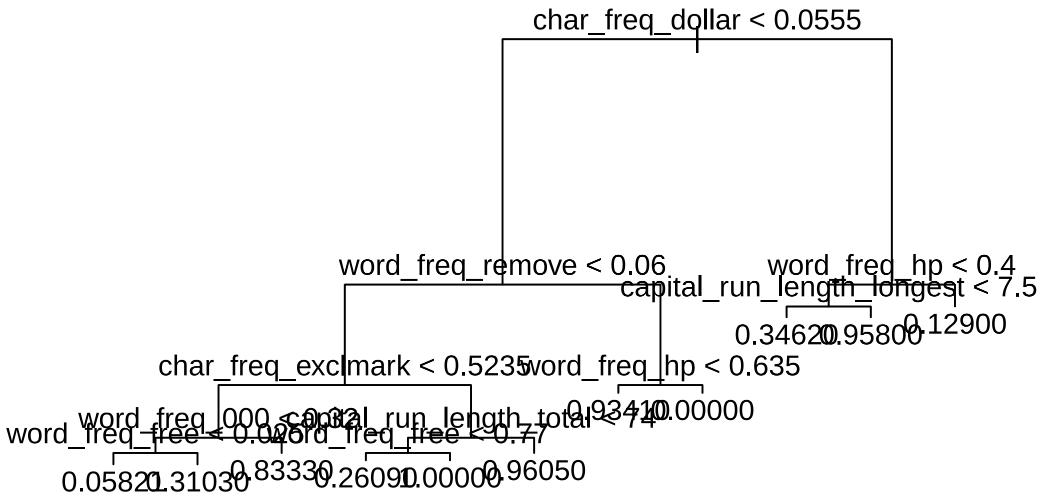


However, in this specific task, we can see that the cross-validation error is monotonously decreasing, so the optimal fold number is the original fold number. We choose best $size = 12$ here such that the pruned tree model here is exactly the same as our complex tree model.

```
#prune the tree
prune.mod=prune.tree(tree.mod,best=12)
```

```
Warning in prune.tree(tree.mod, best = 12): best is bigger than tree size
```

```
plot(prune.mod)
text(prune.mod,pretty=0)
```



Now we validate the accuracy of our classification tree model with the test data.

```
#make predictions on training and test set using the unpruned tree
#pred.train<-predict(tree.mod,newdata=data.train)
#classif.train<-ifelse(pred.train[,2]>=pred.train[,1],1,0)
#err(data.train$spam,classif.train)
```

```
#pred.test<-predict(tree.mod,newdata=data.test)
#classif.test<-ifelse(1*pred.test[,2]>=pred.test[,1],1,0)
#err(data.test$spam,classif.test)
```

```
#make predictions on training and test set using the pruned tree
#pred.train<-predict(prune.mod,newdata=data.train)
#classif.train<-ifelse(pred.train[,2]>=pred.train[,1],1,0)
#err(data.train$spam,classif.train)
#pred.test<-predict(prune.mod,newdata=data.test)
#classif.test<-ifelse(1*pred.test[,2]>=pred.test[,1],1,0)
#err(data.test$spam,classif.test)
```

We can conclude that our classification model performs very well. since the complex tree is the same as pruned tree here, we can see that the test error is just very slightly higher than the train error. There is not much overfitting here.

Different Classification Models

Now for this part, we are going to compare different classification models using two different classification scenarios.

Firstly, we consider the scenario where we have different prior probabilities and equal classification costs. We use the entire dataset to figure out the prior probability.

```
#(a)Account for different prior probabilities and equal classification costs
# (1) linear discriminant analysis
set.seed(0829539)
ldaS.out<-lda(spam~.,data=spamdata)
print(ldaS.out)
```

Call: lda(spam ~ ., data = spamdata)

Prior probabilities of groups: 0 1 0.6059552 0.3940448

Group means: word_freq_make word_freq_address word_freq_all word_freq_3d word_freq_our 0 0.0734792 0.2444656 0.2005811 0.0008859397 0.1810402 1 0.1523387 0.1646498 0.4037948 0.1646718147 0.5139548 word_freq_over word_freq_remove word_freq_internet word_freq_order 0 0.04454448 0.00938307 0.03841463 0.03804878 1 0.17487590 0.27540541 0.20814120 0.17006067 word_freq_mail word_freq_receive word_freq_will word_freq_people 0 0.1671700 0.0217109 0.5363235 0.06166428 1 0.3505074 0.1184335 0.5499724 0.14354661 word_freq_report word_freq_addresses word_freq_free word_freq_business 0 0.04240316 0.008317791 0.0735868 0.04834648 1 0.08357419 0.112079426 0.5183618 0.28750689 word_freq_email word_freq_you word_freq_credit word_freq_your word_freq_font 0 0.09729197 1.270341 0.00757891 0.4387016 0.04522597 1 0.31922780 2.264539 0.20552124 1.3803696 0.23803640 word_freq_000 word_freq_money word_freq_hp word_freq_hpl word_freq_george 0 0.007087518 0.01713773 0.89547346 0.431994261 1.265265423 1 0.247054606 0.21287921 0.01747932 0.009172642 0.001549917 word_freq_650 word_freq_lab word_freq_labs word_freq_telnet word_freq_857 0 0.19380560 0.1627941176 0.165853659 0.106032999 0.0773063128 1 0.01879757 0.0006839493 0.005968009 0.001274131 0.0005184777 word_freq_data word_freq_415 word_freq_85 word_freq_technology word_freq_1999 0 0.1509864 0.077786944 0.169454806 0.14167145 0.19774390 1 0.0145615 0.001776062 0.006927744 0.02951462 0.04346939 word_freq_parts word_freq_pm word_freq_direct word_freq_cs word_freq_meeting 0 0.018723099 0.12167862 0.08311693 0.0720265423 0.216807747 1 0.004710425 0.01242692 0.03671815 0.0000551572 0.002443464 word_freq_original word_freq_project word_freq_re word_freq_edu 0 0.070581062 0.126635581 0.4157604 0.28718436 1 0.008450083 0.006243795 0.1250910 0.01472697 word_freq_table word_freq_conference char_freq_dotcomma char_freq_parenthesis 0 0.008192253 0.051226686 0.05028085 0.1585782 1 0.001218974 0.002101489 0.02057308 0.1089702 char_freq_bracket char_freq_exclmark char_freq_dollar char_freq_pound 0 0.022683644 0.1099835 0.01164849 0.02171306 1 0.008198566 0.5137126 0.17447821 0.07887700 capital_run_length_average capital_run_length_longest 0 2.377301 18.21449 1 9.519165 104.39327 capital_run_length_total 0 161.4709 1 470.6194

Coefficients of linear discriminants: LD1 word_freq_make -0.2053433845 word_freq_address -0.0496520077 word_freq_all 0.1618979041 word_freq_3d 0.0491205095 word_freq_our 0.3470862316 word_freq_over 0.4898352934 word_freq_remove 0.8776953914 word_freq_internet 0.3874021379 word_freq_order 0.2987224576 word_freq_mail 0.0621045827 word_freq_receive 0.2343512301 word_freq_will -0.1148308781 word_freq_people 0.0490659059 word_freq_report 0.0200317976 word_freq_addresses 0.0763541263 word_freq_free 0.3093868784 word_freq_business 0.2131611128 word_freq_email 0.2283383726 word_freq_you 0.0582547287 word_freq_credit 0.2544047910 word_freq_your 0.2171922810 word_freq_font 0.1845200015 word_freq_000 0.7204900016 word_freq_money 0.3746324145 word_freq_hp -0.0955222618 word_freq_hpl -0.0891499158 word_freq_george -0.0502923959 word_freq_650 0.0164352762 word_freq_lab -0.0307054282 word_freq_labs -0.2141201057 word_freq_telnet -0.0960127853 word_freq_857 0.0260981607 word_freq_data -0.1730466665 word_freq_415 0.2107954203 word_freq_85 -0.1284707897 word_freq_technology 0.1091451554 word_freq_1999 -0.1368948926 word_freq_parts -0.2202625065 word_freq_pm -0.0814242179 word_freq_direct 0.1680076688 word_freq_cs -0.0344749731 word_freq_meeting -0.1522032232 word_freq_original -0.2606575435 word_freq_project -0.1334635603 word_freq_re -0.1453064935 word_freq_edu -0.1558616802 word_freq_table -0.8044795076 word_freq_conference -0.2399825033 char_freq_dotcomma -0.5774724823 char_freq_parenthesis -0.2471396406 char_freq_bracket -0.2433979819 char_freq_exclmark 0.2804998615 char_freq_dollar 0.9611097972 char_freq_pound 0.1141464080 capital_run_length_average 0.0009590191 capital_run_length_longest 0.0002751450 capital_run_length_total 0.0003291749

So, we take $P(isspam) = 0.394$, $P(notspam) = 0.606$ as our prior probability, and we can verify that their sum equals 1.

The four classification models we are going to compare is *1) Linear Discriminant Analysis*, *2) Bagging*, *3) Random Forests* and *4) Gradient Boosting*. Since the dimension here is very high and the mathematical assumption of our dataset is very weak, we do not suppose that Linear Discriminant Analysis without Principal Component analysis here will have very good performance. We just take it as a baseline. The three other methods are all methods used to improve the classification tree model, as tree models are always with high variance, thus high probability to cause overfitting. The results are listed as below:

```
lda.out<-lda(spam~.,prior=c(0.606,0.394),data=data.train)
print(lda.out)
```

1) Linear Discriminant Analysis Call: lda(spam ~ ., data = data.train, prior = c(0.606, 0.394))

Prior probabilities of groups: 0 1 0.606 0.394

Group means: word_freq_make word_freq_address word_freq_all word_freq_3d word_freq_our 0 0.07138179 0.2409365 0.1957826 0.0008185986 0.1976228 1 0.16452210 0.1668859 0.4086125 0.2194655704 0.5017266 word_freq_over word_freq_remove word_freq_internet word_freq_order 0 0.04830386 0.01184676 0.04060249 0.03678454 1 0.18016444 0.26947585 0.22677287 0.18334018 word_freq_mail word_freq_receive word_freq_will word_freq_people 0 0.1605894 0.01955468 0.5285658 0.0591814 1 0.3366393 0.11578623 0.5660329 0.1439363 word_freq_report word_freq_addresses word_freq_free word_freq_business 0 0.04299280 0.007976424 0.08952849 0.04613621 1 0.08830421 0.115005139 0.50793422 0.29245632 word_freq_email word_freq_you word_freq_credit word_freq_your word_freq_font 0 0.09777996 1.264460 0.01058939 0.4250753 0.04102816 1 0.32486125 2.270051 0.21265159 1.4180781 0.25781089 word_freq_000 word_freq_money word_freq_hp word_freq_hpl word_freq_george 0 0.007013752 0.01527832 0.83885396 0.406293386 1.2639030779 1 0.267255910 0.21683453 0.01283659 0.008263104 0.0007605344 word_freq_650 word_freq_lab word_freq_labs word_freq_telnet word_freq_857 0 0.188559267 0.1680091683 0.155671251 0.0889325475 0.0642567125 1 0.008499486 0.0003699897 0.008756423 0.0004110997 0.0004830421 word_freq_data word_freq_415 word_freq_85 word_freq_technology word_freq_1999 0 0.14044532 0.065108055 0.166981009 0.14369352 0.19031434 1 0.01560123 0.002487153 0.004491264 0.03008222 0.03874615 word_freq_parts word_freq_pm word_freq_direct word_freq_cs word_freq_meeting 0 0.02419122 0.14044532 0.07253438 0.0751604453 0.241263916 1 0.00340185 0.01218911 0.03709147 0.0001027749 0.001963001 word_freq_original word_freq_project word_freq_re word_freq_edu 0 0.063837590 0.125166994 0.4432678 0.28961362 1 0.007985612 0.007389517 0.1329702 0.01497431 word_freq_table word_freq_conference char_freq_dotcomma char_freq_parenthesis 0 0.007557302 0.048408644 0.04621480 0.1614165 1 0.001377184 0.001593011 0.02189414 0.1096372 char_freq_bracket char_freq_exclmark char_freq_dollar char_freq_pound 0 0.02596333 0.1050550 0.01189915 0.02047086 1 0.01060740 0.5402107 0.17836691 0.08214491 capital_run_length_average capital_run_length_longest 0 2.418952 18.21218 1 10.737508 111.23947 capital_run_length_total 0 157.9594 1 477.7636

Coefficients of linear discriminants: LD1 word_freq_make -0.1738052540 word_freq_address -0.0448853873 word_freq_all 0.1720206608 word_freq_3d 0.0497191935 word_freq_our 0.3303970841 word_freq_over 0.4013297768 word_freq_remove 0.8727936217 word_freq_internet 0.3774681175 word_freq_order 0.3390530980 word_freq_mail 0.0685041162 word_freq_receive 0.3077592196 word_freq_will -0.0977968068 word_freq_people 0.2062596269 word_freq_report 0.0389851753 word_freq_addresses 0.1009990985 word_freq_free 0.2412547225 word_freq_business 0.1109536015 word_freq_email 0.1843676049 word_freq_you 0.0637683171 word_freq_credit 0.3089379962 word_freq_your 0.2454535652 word_freq_font 0.1791886211 word_freq_000 0.7480713382 word_freq_money 0.3317635774 word_freq_hp -0.1059717939 word_freq_hpl -0.0675864649 word_freq_george -0.0481129633 word_freq_650 -0.0720968303 word_freq_lab -0.0101309921 word_freq_labs -0.1239777451 word_freq_telnet -0.2256495114 word_freq_857 -0.0300165160 word_freq_data -0.2163059464 word_freq_415 0.3795931122 word_freq_85 -0.0587161725 word_freq_technology 0.1116499532 word_freq_1999 -0.1577812210 word_freq_parts -0.2238978915 word_freq_pm -0.0350372753 word_freq_direct 0.1363150120 word_freq_cs 0.0091338456 word_freq_meeting -0.1592545975 word_freq_original -0.2687481023 word_freq_project -0.1210036081 word_freq_re -0.1313502850 word_freq_edu -0.1589663324 word_freq_table -0.7164016098 word_freq_conference -0.2623493565 char_freq_dotcomma -0.6216710832 char_freq_parenthesis -0.3637728541 char_freq_bracket -0.2686608836 char_freq_exclmark 0.3970285708 char_freq_dollar 0.8798682727 char_freq_pound 0.1801713789 capital_run_length_average 0.0007269859 capital_run_length_longest 0.0002708059 capital_run_length_total 0.0003282127

Here we can see that the three attributes highly correlated with spam webmails—the frequency of "$", "!" and "remove", are exactly the same as what we have got from our classification tree model. What is interesting here is we can also see that the frequency of "table", ";" and "parenthesis" shows strong negative correlation with spam webmails.

```r
pred.train<-predict(lda.out,data.train)
err(data.train$spam,pred.train$class)
```

```
    predicted
```

observed 0 1 0 1460 67 1 202 771 [1] 0.1076

```r
pred.test<-predict(lda.out,data.test)
err(data.test$spam,pred.test$class)
```

```
    predicted
```

observed 0 1 0 1205 56 1 195 645 [1] 0.1194669

```r
# (2) bagging
set.seed(0829539)
bag.mod=randomForest(spam~.,data=data.train,classwt=c(0.606,0.394),mtry=57,ntree=5000,importance=TRUE)
```

2) Bagging

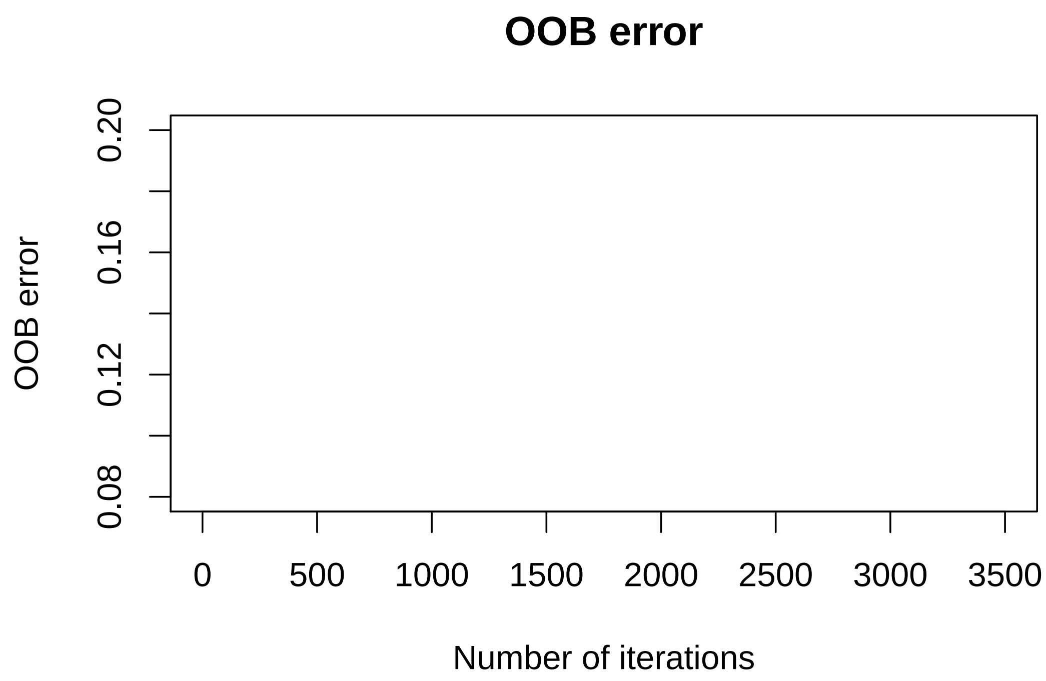
```
Warning in randomForest.default(m, y, ...): The response has five or fewer
unique values. Are you sure you want to do regression?
```

```r
bag.mod
```

Call: randomForest(formula = spam ~ ., data = data.train, classwt = c(0.606, 0.394), mtry = 57, ntree = 5000, importance = TRUE) Type of random forest: regression Number of trees: 5000 No. of variables tried at each split: 57

```
      Mean of squared residuals: 0.05103299
                % Var explained: 78.53
```

```
#plot oob error
par(cex=1.2)
plot(1:3500,bag.mod$err.rate[1:3500,1],xlab="Number of iterations",ylab="OOB error",ylim=c(0.08,0.2),pc
lines(1:3500,bag.mod$err.rate[1:3500,1],col="red")
```




```
#plot variable importance
importance(bag.mod,plot=TRUE)
```

%IncMSE IncNodePurity

word_freq_make 29.238001 1.43694535 word_freq_address 35.425096 1.36654325 word_freq_all 34.444832 1.56949725 word_freq_3d 39.480311 1.27201578 word_freq_our 92.541652 9.40583404 word_freq_over 56.510031 2.62433852 word_freq_remove 239.889704 77.81934337 word_freq_internet 58.039504 3.78580332 word_freq_order 43.288150 1.72174626 word_freq_mail 40.166404 3.38175085 word_freq_receive 34.848904 2.50026111 word_freq_will 44.327874 3.49020667 word_freq_people 25.061372 1.00148740 word_freq_report 29.112454 1.39151730 word_freq_addresses 19.252189 0.16267556 word_freq_free 144.314882 20.62843111 word_freq_business 68.909895 4.46474605 word_freq_email 45.032836 3.43741522 word_freq_you 62.763783 8.94593128 word_freq_credit 35.059070 1.22669108 word_freq_your 92.695828 9.29638301 word_freq_font 55.093978 1.69572789 word_freq_000 88.445922 7.61249271 word_freq_money 71.856474 7.62438668 word_freq_hp 201.461853 29.83623245 word_freq_hpl 42.770024 1.32662718 word_freq_george 177.087408 8.54149539 word_freq_650 32.151251 0.79129005 word_freq_lab 6.381817 0.12404326 word_freq_labs 31.280593 0.95982160 word_freq_telnet 36.691582 0.39188751 word_freq_857 11.639281 0.09456832 word_freq_data 30.122861 1.66917018 word_freq_415 11.562723 0.09732713 word_freq_85 21.363234 0.39851721 word_freq_technology 17.846752 0.93144708 word_freq_1999 48.721609 1.53924981 word_freq_parts 19.179823 0.70897506 word_freq_pm 42.719767 1.56804568 word_freq_direct 10.475127 0.20655720 word_freq_cs 8.036261 0.16783509 word_freq_meeting 121.877608 4.79688763 word_freq_original 7.613654 0.08293994 word_freq_project 21.964169 0.60144962 word_freq_re 57.172110 4.21329753 word_freq_edu 166.933081 9.81686435 word_freq_table 1.789255 0.12910522 word_freq_conference 61.707547 1.04695593 char_freq_dotcomma 37.801419 2.59914263 char_freq_parenthesis 53.010009 4.14389190 char_freq_bracket 23.143077 0.86573798 char_freq_exclmark 244.958151 77.02105463 char_freq_dollar 184.530127 192.74915824 char_freq_pound 29.862777 0.86822198 capital_run_length_average 90.700431 13.30348032 capital_run_length_longest 147.022471 22.31552976 capital_run_length_total 82.513585 21.39809315

```
varImpPlot(bag.mod,type=2,cex=1.2)
```