Assignment 2: Classification for the Detection of Opinion Spam

Anouk van der Lee (6620590), Shu Zhao (6833519), Fleur Petit (5583837)

21, October, 2020

Contents

Dataset for training, validation and test

Dataset split

```r
# 1 fold = 80 samples
# Fold 1-4 for training = 1:320 from true + 1:320 from false
index.train <- c(c(1:320), 400+c(1:320))

# Training document-term matrix
train.dtm <- DocumentTermMatrix(reviews.all[index.train])
# Test document-term matrix
test.dtm <- DocumentTermMatrix(reviews.all[-index.train],
                               list(dictionary = dimnames(train.dtm)[[2]]))

# Training document-term matrix for bigrams
BigramTokenizer <- function (x) {
  unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)
}

train.dtm.bigram <- DocumentTermMatrix(reviews.all[index.train],
                                       control = list(tokenize = BigramTokenizer))
```

```
# Test document-term matrix for bigrams
test.dtm.bigram <- DocumentTermMatrix(reviews.all[-index.train],
                                      list(dictionary = dimnames(train.dtm.bigram)[[2]]))
```

Feature selection

```
# Remove terms that occur in less than 5% of the documents
# Training document-term matrix
train.dtm.freq <- removeSparseTerms(train.dtm, 0.95)
# Test document-term matrix
test.dtm.freq <- DocumentTermMatrix(reviews.all[-index.train],
                                    list(dictionary = dimnames(train.dtm.freq)[[2]]))

# Training document-term matrix for bigrams
train.dtm.bigram.freq <- removeSparseTerms(train.dtm.bigram, 0.99)
# Test document-term matrix for bigrams
test.dtm.bigram.freq <- DocumentTermMatrix(reviews.all[-index.train],
                                           list(dictionary = dimnames(train.dtm.bigram.freq)[[2]]))

# Combine unigrams and bigrams
# Training document-term matrix for unigrams and bigrams
train.dtm.bigram.freq <- cbind(train.dtm.freq, train.dtm.bigram.freq)
# Test document-term matrix for unigrams and bigrams
test.dtm.bigram.freq <- cbind(test.dtm.freq, test.dtm.bigram.freq)
```

```
# Select top-n mutual-information terms from total vocabularies
index.top300 <- calculate.topn(train.dtm, topn = c(1:300))
train.dtm.top300 <- train.dtm[, index.top300]
test.dtm.top300 <- test.dtm[, index.top300]

# Select top-n mutual-information terms from frequent terms
index.freq.top200 <- calculate.topn(train.dtm.freq, topn = c(1:200))
train.dtm.freq.top200 <- train.dtm[, index.freq.top200]
test.dtm.freq.top200 <- test.dtm[, index.freq.top200]
```

Classifiers

Multinomial naive Bayes

```
# 4-fold cross validation
reviews.mnb.pred <- c()
reviews.mnb.actual <- c()

# hyper-parameter: feature selection
# the best option: only mutual information to select top-300 terms out of 6900 total terms
train.dtm.mnb <- train.dtm.top300
```

```r
for(i in 1:4) {
  # Validation fold
  val.start <- (i - 1) * 80 + 1
  val.end <- val.start + 80 - 1
  val.range <- c(c(val.start:val.end), 320+c(val.start:val.end))

  # Training fold
  train.range <- c(c(1:320)[-val.range], 320+c(1:320)[-val.range])

  # Train the model with priors and conditional probabilities
  reviews.mnb <- train.mnb(as.matrix(train.dtm.mnb)[train.range,], labels[index.train][train.range])
  # Make predictions
  reviews.mnb.pred <- c(reviews.mnb.pred, predict.mnb(reviews.mnb, as.matrix(train.dtm.mnb)[val.range,]
  reviews.mnb.actual <- c(reviews.mnb.actual, labels[index.train][val.range])
}

# Confusion matrix
conf.mat.mnb <- table(reviews.mnb.actual, reviews.mnb.pred, dnn = c("actual", "predicted"))
perf.mnb <-performance(conf.mat.mnb)

perf.mnb %>% kable()
```

metric	value
recall	0.9531250
miss-rate	0.0468750
fall-out	0.1218750
selectivity	0.8781250
prevalence	0.5000000
precision	0.8866279
false omission rate	0.0506757
pos likelihood ratio	7.8205128
neg likelihood ratio	0.0533808
accuracy	0.9156250
false discovery rate	0.1133721
neg predictive value	0.9493243
diagnostic odds ratio	146.5042735
F1	0.9186747

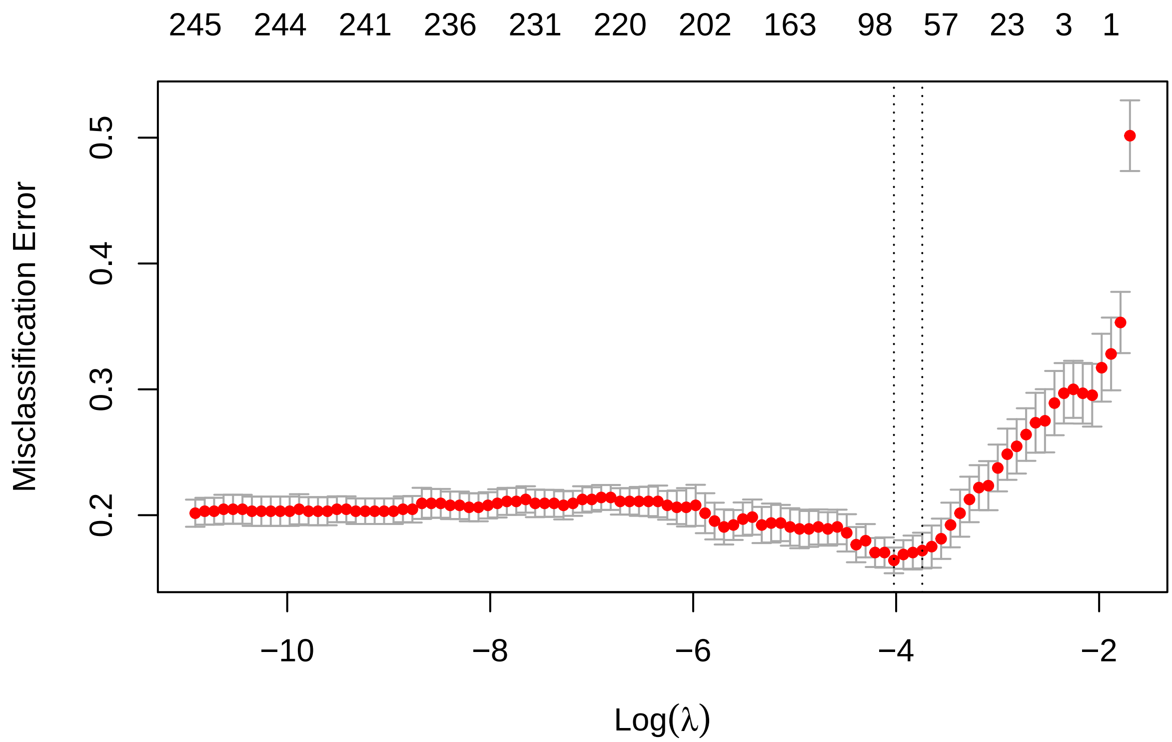
```r
# 4-fold cross validation
reviews.mnb.bigram.pred <- c()
reviews.mnb.bigram.actual <- c()

# hyper-parameter: feature selection
# the best option: only mutual information to select top-300 terms out of 6900 total terms
train.dtm.mnb.bigram <- train.dtm.bigram.freq

for(i in 1:4) {
  # Validation fold
  val.start <- (i - 1) * 80 + 1
  val.end <- val.start + 80 - 1
  val.range <- c(c(val.start:val.end), 320+c(val.start:val.end))
```

```
  # Training fold
  train.range <- c(c(1:320)[-val.range], 320+c(1:320)[-val.range])

  # Train the model with priors and conditional probabilities
  reviews.mnb <- train.mnb(as.matrix(train.dtm.mnb.bigram)[train.range,], labels[index.train][train.rang
  # Make predictions
  reviews.mnb.bigram.pred <- c(reviews.mnb.bigram.pred,
                               predict.mnb(reviews.mnb, as.matrix(train.dtm.mnb.bigram)[val.range,]))
  reviews.mnb.bigram.actual <- c(reviews.mnb.bigram.actual, labels[index.train][val.range])
}

# Confusion matrix
conf.mat.mnb.bigram <- table(reviews.mnb.bigram.actual, reviews.mnb.bigram.pred,
                             dnn = c("actual", "predicted"))
perf.mnb.bigram <-performance(conf.mat.mnb.bigram)

perf.mnb.bigram %>% kable()
```

metric	value
recall	0.8000000
miss-rate	0.2000000
fall-out	0.1468750
selectivity	0.8531250
prevalence	0.5000000
precision	0.8448845
false omission rate	0.1899110
pos likelihood ratio	5.4468085
neg likelihood ratio	0.2344322
accuracy	0.8265625
false discovery rate	0.1551155
neg predictive value	0.8100890
diagnostic odds ratio	23.2340426
F1	0.8218299

Regularized logistic regression

```
# hyper-parameter: feature selection
# the best option: ???
train.dtm.glmet <- train.dtm.freq
test.dtm.glmet <- test.dtm.freq

# Logistic regression with lasso penalty
reviews.glmnet <- cv.glmnet(as.matrix(train.dtm.glmet), labels[index.train],
                            family="binomial", type.measure="class")
# Cross-validation on lambda
plot(reviews.glmnet)
```

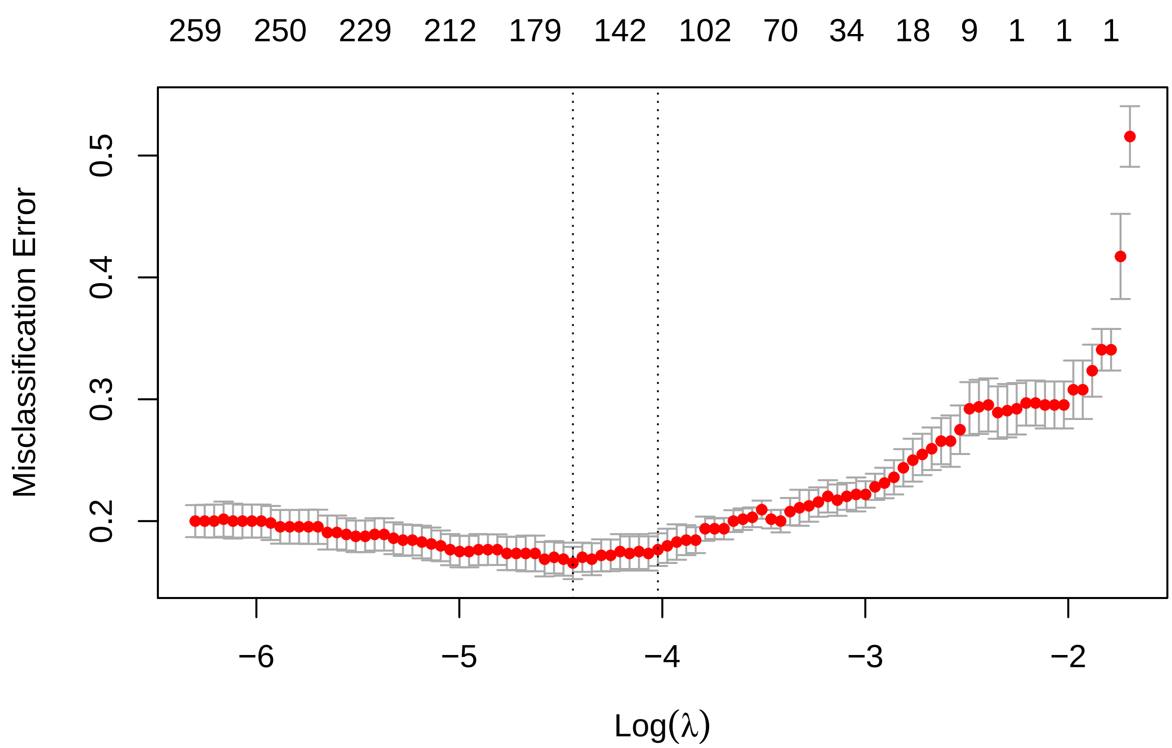



```r
# coef(reviews.glmnet, s="lambda.1se")

# Make predictions on the test set
reviews.logreg.pred <- predict(reviews.glmnet, newx=as.matrix(test.dtm.glmet),
                               s="lambda.1se", type="class")

# Confusion matrix
conf.mat.logreg <- table(labels[-index.train], reviews.logreg.pred, dnn = c("actual", "predicted"))

perf.logreg <-performance(conf.mat.logreg)
perf.logreg %>% kable()
```

metric	value
recall	0.7125000
miss-rate	0.2875000
fall-out	0.1000000
selectivity	0.9000000
prevalence	0.5000000
precision	0.8769231
false omission rate	0.2421053
pos likelihood ratio	7.1250000
neg likelihood ratio	0.3194444
accuracy	0.8062500
false discovery rate	0.1230769
neg predictive value	0.7578947

metric	value
diagnostic odds ratio	22.3043478
F1	0.7862069

```r
# hyper-parameter: feature selection
# the best option: ???
train.dtm.glmet.bigram <- train.dtm.bigram.freq
test.dtm.glmet.bigram <- test.dtm.bigram.freq

# Logistic regression with lasso penalty
reviews.glmnet.bigram <- cv.glmnet(as.matrix(train.dtm.glmet.bigram), labels[index.train],
                                   family="binomial", type.measure="class")
# Cross-validation on lambda
plot(reviews.glmnet.bigram)
```



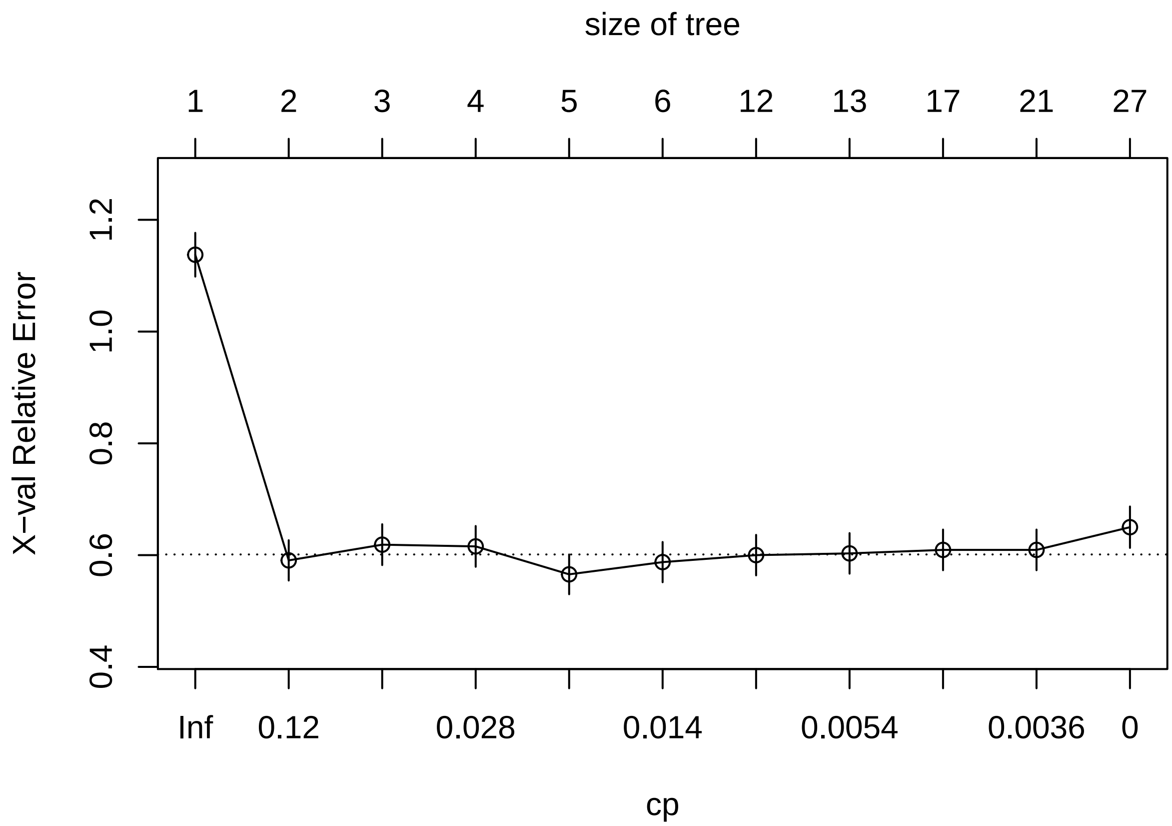

```r
# coef(reviews.glmnet.bigram, s="lambda.1se")

# Make predictions on the test set
reviews.logreg.bigram.pred <- predict(reviews.glmnet.bigram, newx=as.matrix(test.dtm.glmet.bigram),
                                      s="lambda.1se", type="class")

# Confusion matrix
conf.mat.logreg.bigram <- table(labels[-index.train], reviews.logreg.bigram.pred,
                                dnn = c("actual", "predicted"))
```

```r
perf.logreg.bigram <-performance(conf.mat.logreg.bigram)
perf.logreg.bigram %>% kable()
```

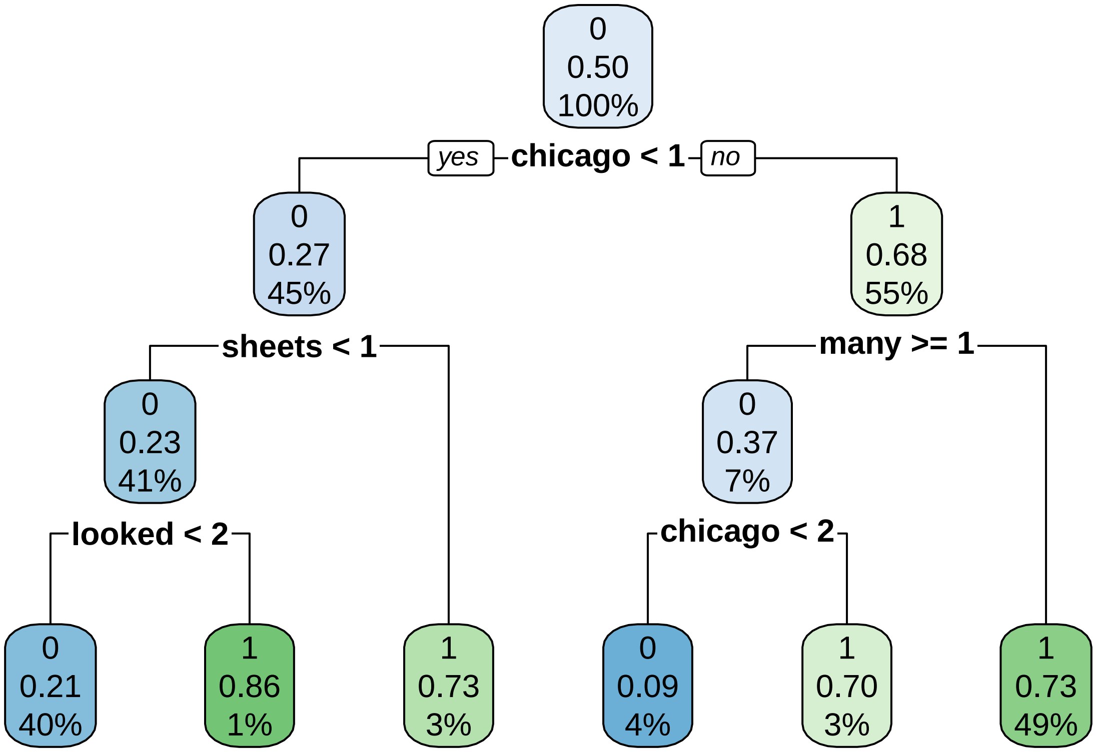
metric	value
recall	0.6250000
miss-rate	0.3750000
fall-out	0.0875000
selectivity	0.9125000
prevalence	0.5000000
precision	0.8771930
false omission rate	0.2912621
pos likelihood ratio	7.1428571
neg likelihood ratio	0.4109589
accuracy	0.7687500
false discovery rate	0.1228070
neg predictive value	0.7087379
diagnostic odds ratio	17.3809524
F1	0.7299270

Classification trees

```r
# hyper-parameter: feature selection
# the best option: ???
train.dtm.rpart <- train.dtm.freq
test.dtm.rpart <- test.dtm.freq

# Grow the tree
reviews.rpart <- rpart(label ~ .,
                       data = data.frame(as.matrix(train.dtm.rpart), label = labels[index.train]),
                       cp = 0,
                       method = "class")

# Plot cv-error of pruning sequence
plotcp(reviews.rpart)
```




```
# Tree with lowest cv error
reviews.rpart.pruned <- prune(reviews.rpart, cp=0.014)
# Plot the tree
rpart.plot(reviews.rpart.pruned)
```

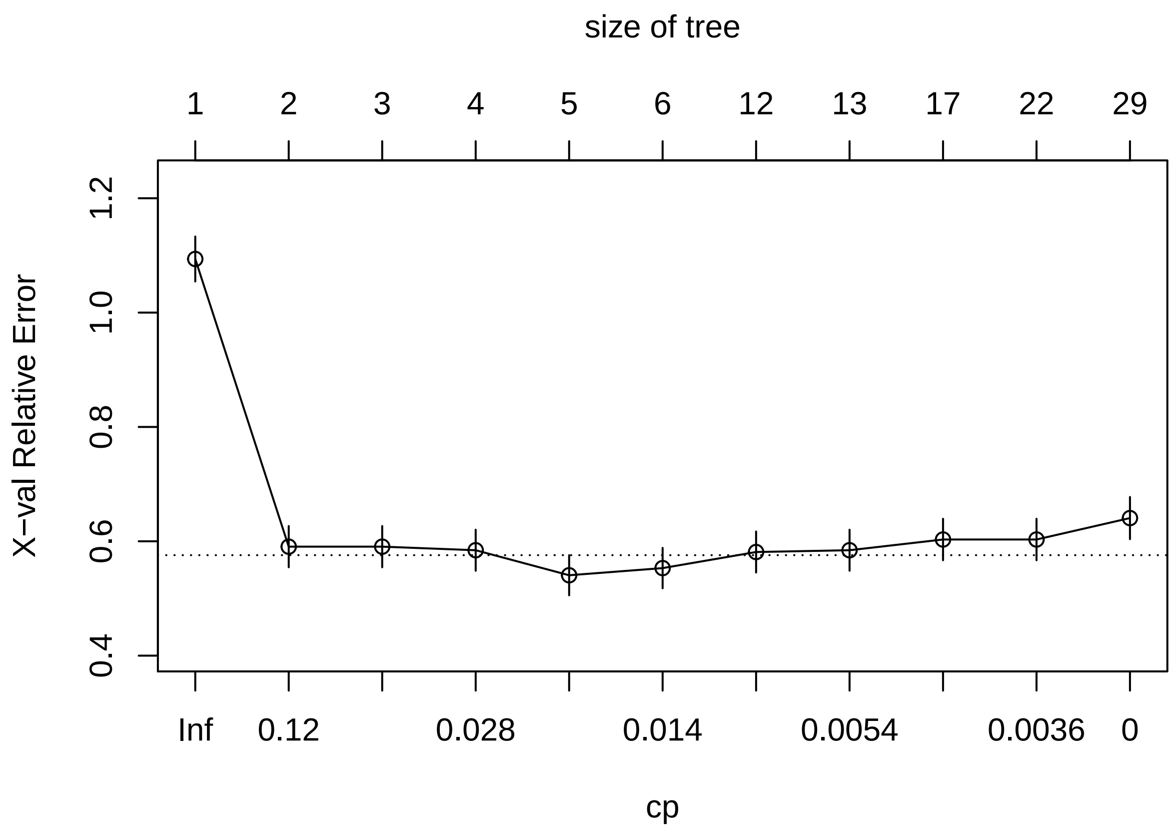



```r
# Make predictions on the test set
reviews.rpart.pred <- predict(reviews.rpart.pruned,
                              newdata = data.frame(as.matrix(test.dtm.rpart)),
                              type = "class")

# Confusion matrix
conf.mat.rpart <- table(labels[-index.train], reviews.rpart.pred, dnn = c("actual", "predicted"))

perf.rpart <-performance(conf.mat.rpart)
perf.rpart %>% kable()
```

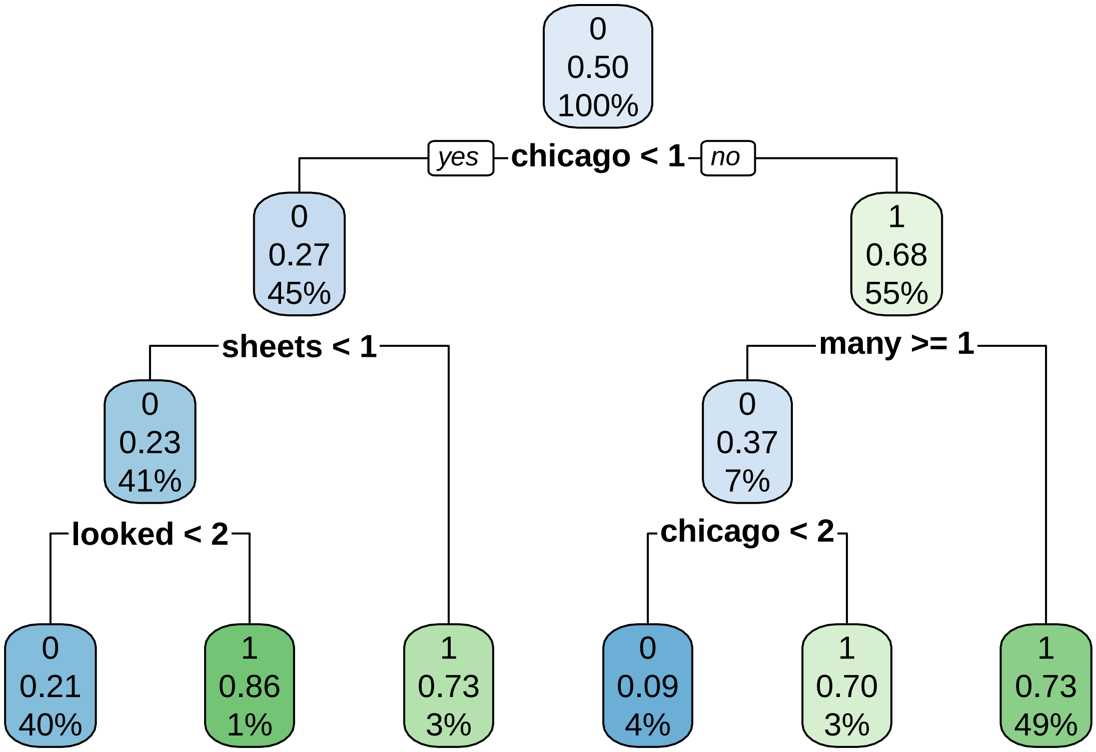
metric	value
recall	0.6000000
miss-rate	0.4000000
fall-out	0.2875000
selectivity	0.7125000
prevalence	0.5000000
precision	0.6760563
false omission rate	0.3595506
pos likelihood ratio	2.0869565
neg likelihood ratio	0.5614035
accuracy	0.6562500
false discovery rate	0.3239437
neg predictive value	0.6404494
diagnostic odds ratio	3.7173913

metric	value
F1	0.6357616

```r
# hyper-parameter: feature selection
# the best option: ???
train.dtm.rpart.bigram <- train.dtm.bigram.freq
test.dtm.rpart.bigram <- test.dtm.bigram.freq

# Grow the tree
reviews.rpart.bigram <- rpart(label ~ .,
                              data = data.frame(as.matrix(train.dtm.rpart.bigram),
                                                label = labels[index.train]),
                              cp = 0, method = "class")

# Plot cv-error of pruning sequence
plotcp(reviews.rpart.bigram)
```




```r
# Tree with lowest cv error
reviews.rpart.bigram.pruned <- prune(reviews.rpart.bigram, cp=0.014)
# Plot the tree
rpart.plot(reviews.rpart.bigram.pruned)
```




```r
# Make predictions on the test set
reviews.rpart.bigram.pred <- predict(reviews.rpart.bigram.pruned,
                                    newdata = data.frame(as.matrix(test.dtm.rpart.bigram)),
                                    type = "class")

# Confusion matrix
conf.mat.rpart.bigram <- table(labels[-index.train], reviews.rpart.bigram.pred,
                              dnn = c("actual", "predicted"))

perf.rpart.bigram <-performance(conf.mat.rpart.bigram)
perf.rpart.bigram %>% kable()
```

metric	value
recall	0.6000000
miss-rate	0.4000000
fall-out	0.2875000
selectivity	0.7125000
prevalence	0.5000000
precision	0.6760563
false omission rate	0.3595506
pos likelihood ratio	2.0869565
neg likelihood ratio	0.5614035
accuracy	0.6562500
false discovery rate	0.3239437
neg predictive value	0.6404494

metric	value
diagnostic odds ratio	3.7173913
F1	0.6357616

Random forests

```
# hyper-parameter: feature selection
# the best option: ???
train.dtm.rf <- train.dtm.freq
test.dtm.rf <- test.dtm.freq

# Train random forest with default settings: 500 trees and mtry = 17
reviews.rf <- randomForest(as.factor(label) ~ .,
                           data = data.frame(as.matrix(train.dtm.rf), label = labels[index.train]))
# Make predictions
reviews.rf.pred <- predict(reviews.rf,
                           newdata = data.frame(as.matrix(test.dtm.rf), label = labels[-index.train]))

# Confusion matrix
conf.mat.rf <- table(labels[-index.train], reviews.rf.pred, dnn = c("actual", "predicted"))

perf.rf <-performance(conf.mat.rf)
perf.rf %>% kable()
```

metric	value
recall	0.8000000
miss-rate	0.2000000
fall-out	0.1750000
selectivity	0.8250000
prevalence	0.5000000
precision	0.8205128
false omission rate	0.1951220
pos likelihood ratio	4.5714286
neg likelihood ratio	0.2424242
accuracy	0.8125000
false discovery rate	0.1794872
neg predictive value	0.8048780
diagnostic odds ratio	18.8571429
F1	0.8101266

```
# hyper-parameter: feature selection
# the best option: ???
train.dtm.rf.bigram <- train.dtm.bigram.freq
test.dtm.rf.bigram <- test.dtm.bigram.freq

# Train random forest with default settings: 500 trees and mtry = 17
reviews.rf.bigram <- randomForest(as.factor(label) ~ .,
                                  data = data.frame(as.matrix(train.dtm.rf.bigram),
                                                    label = labels[index.train]))
```

```r
# Make predictions
reviews.rf.bigram.pred <- predict(reviews.rf.bigram,
                                  newdata = data.frame(as.matrix(test.dtm.rf.bigram),
                                                       label = labels[-index.train]))

# Confusion matrix
conf.mat.rf.bigram <- table(labels[-index.train], reviews.rf.bigram.pred, dnn = c("actual", "predicted"

perf.rf.bigram <-performance(conf.mat.rf.bigram)
perf.rf.bigram %>% kable()
```

metric	value
recall	0.6625000
miss-rate	0.3375000
fall-out	0.1250000
selectivity	0.8750000
prevalence	0.5000000
precision	0.8412698
false omission rate	0.2783505
pos likelihood ratio	5.3000000
neg likelihood ratio	0.3857143
accuracy	0.7687500
false discovery rate	0.1587302
neg predictive value	0.7216495
diagnostic odds ratio	13.7407407
F1	0.7412587

Hyper-parameters

Questions

1. For a single classification tree, the impurity reduction is not equal to mutual information?
2. the words found in feature selection (e.g. frequqency or mutual information) = the words found in classification tree by impurity reduction = the words found in logistic regression by coef(s="lambda.1se") (e.g. as.matrix(coef(reviews.glmnet, s="lambda.1se"))[,1]['chicago'])?
3. sum(as.matrix(coef(reviews.glmnet, s="lambda.1se")) != 0) is 52, so top50 is better?