# Assignment 2: Classification for the Detection of Opinion Spam

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

*20, October, 2020*

## Contents



## Dataset for training, validation and test

### Dataset split

```
# 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

## 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

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 | 0.8437500 |
| miss-rate | 0.1562500 |
| fall-out | 0.2218750 |
| selectivity | 0.7781250 |
| prevalence | 0.5000000 |
| precision | 0.7917889 |
| false omission rate | 0.1672241 |
| pos likelihood ratio | 3.8028169 |

| metric | value |
|---|---|
| neg likelihood ratio | 0.2008032 |
| accuracy | 0.8109375 |
| false discovery rate | 0.2082111 |
| neg predictive value | 0.8327759 |
| diagnostic odds ratio | 18.9380282 |
| F1 | 0.8169440 |

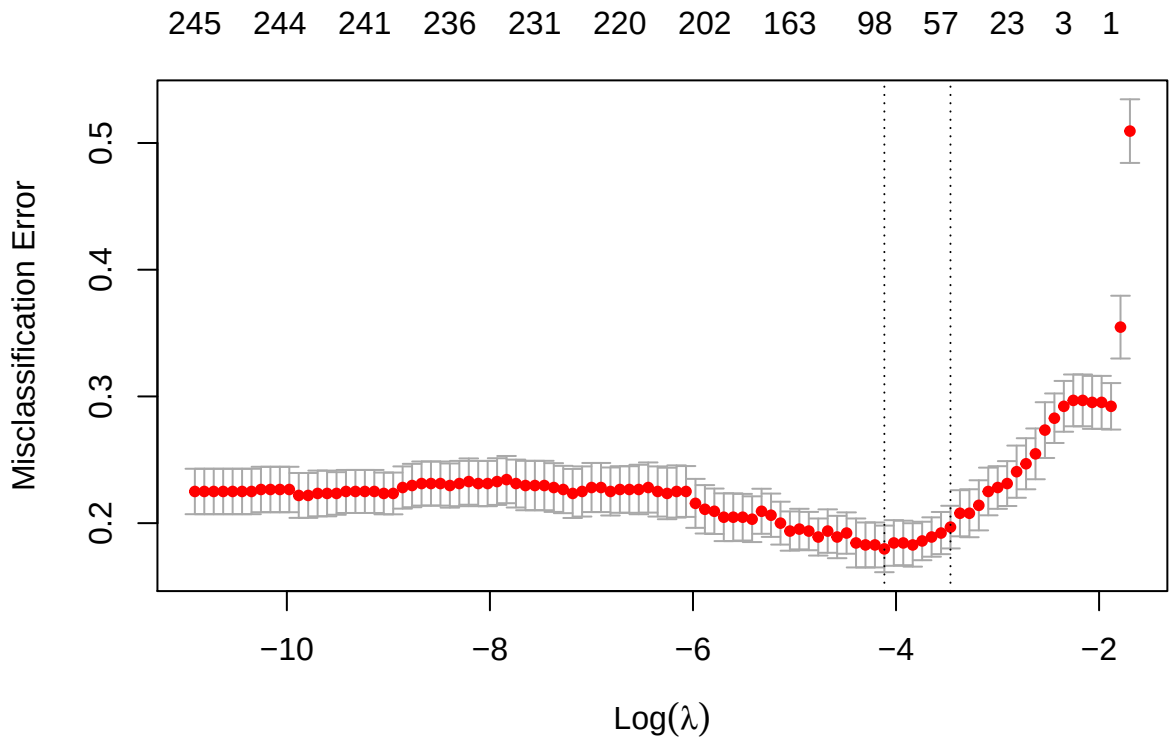
```
# 4-fold cross validation
reviews.mnb.pred <- c()
reviews.mnb.actual <- c()

# 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.bigram

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 | 0.8000000 |
| 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 |

| metric | value |
|---|---|
| F1 | 0.8218299 |

### Regularized logistic regression

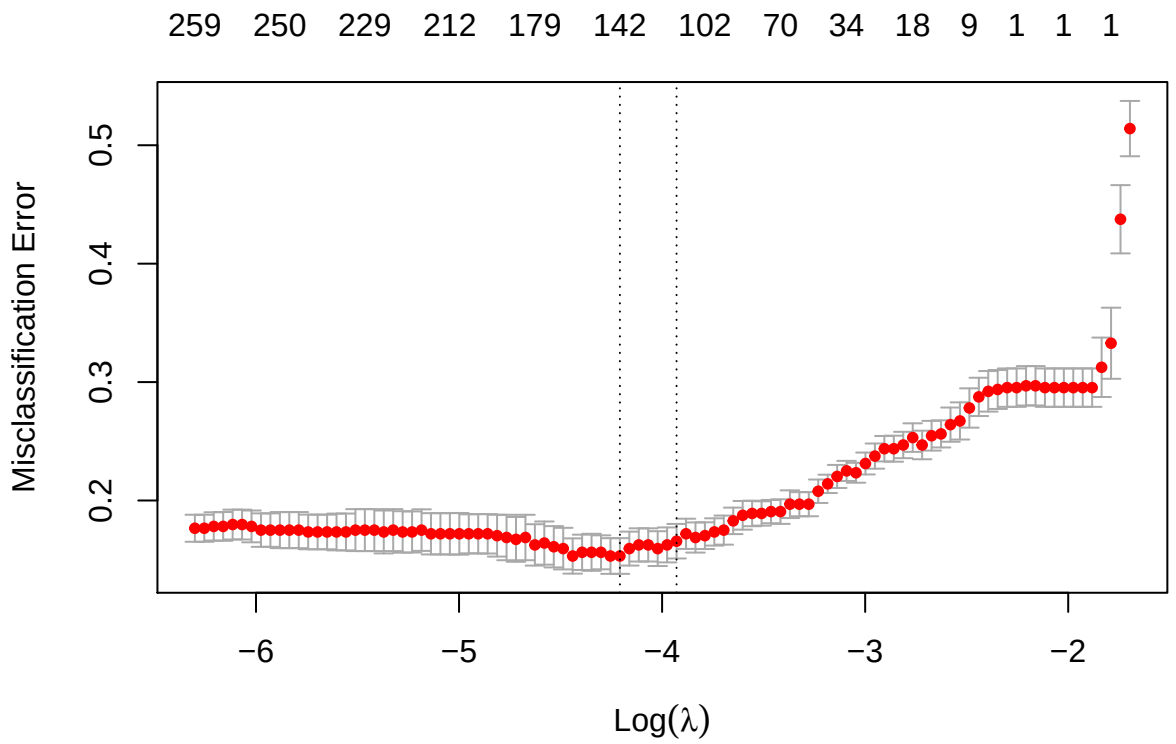
```
# Logistic regression with lasso penalty
reviews.glmnet <- cv.glmnet(as.matrix(train.dtm), labels[index.train],
                            family="binomial", type.measure="class")
# Cross-validation on lambda
plot(reviews.glmnet)
```




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

# Make predictions on the test set
reviews.logreg.pred <- predict(reviews.glmnet, newx=as.matrix(test.dtm),
                               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 | 0.6625000 |
| miss-rate | 0.3375000 |
| fall-out | 0.1125000 |
| selectivity | 0.8875000 |
| prevalence | 0.5000000 |
| precision | 0.8548387 |
| false omission rate | 0.2755102 |
| pos likelihood ratio | 5.8888889 |
| neg likelihood ratio | 0.3802817 |
| accuracy | 0.7750000 |
| false discovery rate | 0.1451613 |
| neg predictive value | 0.7244898 |
| diagnostic odds ratio | 15.4855967 |
| F1 | 0.7464789 |

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



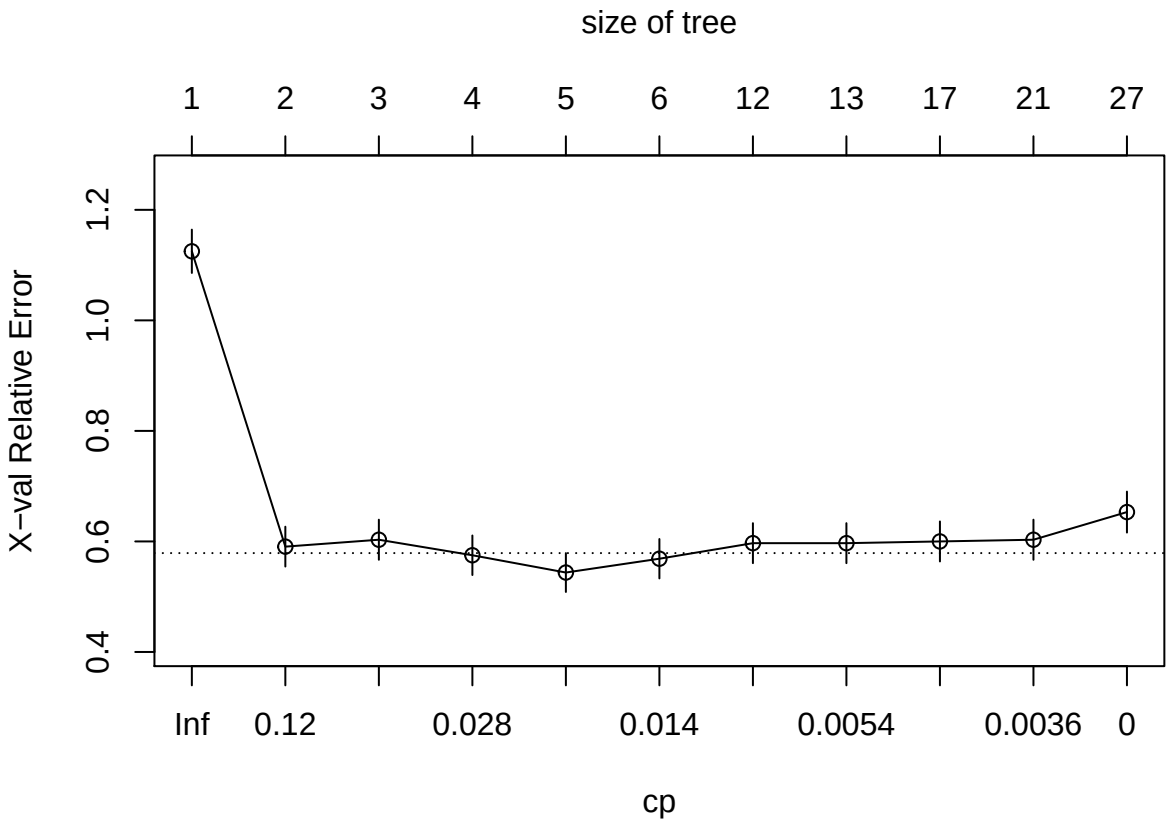

```
# coef(reviews.glmnet.bigram, s="lambda.1se")
```

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

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

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

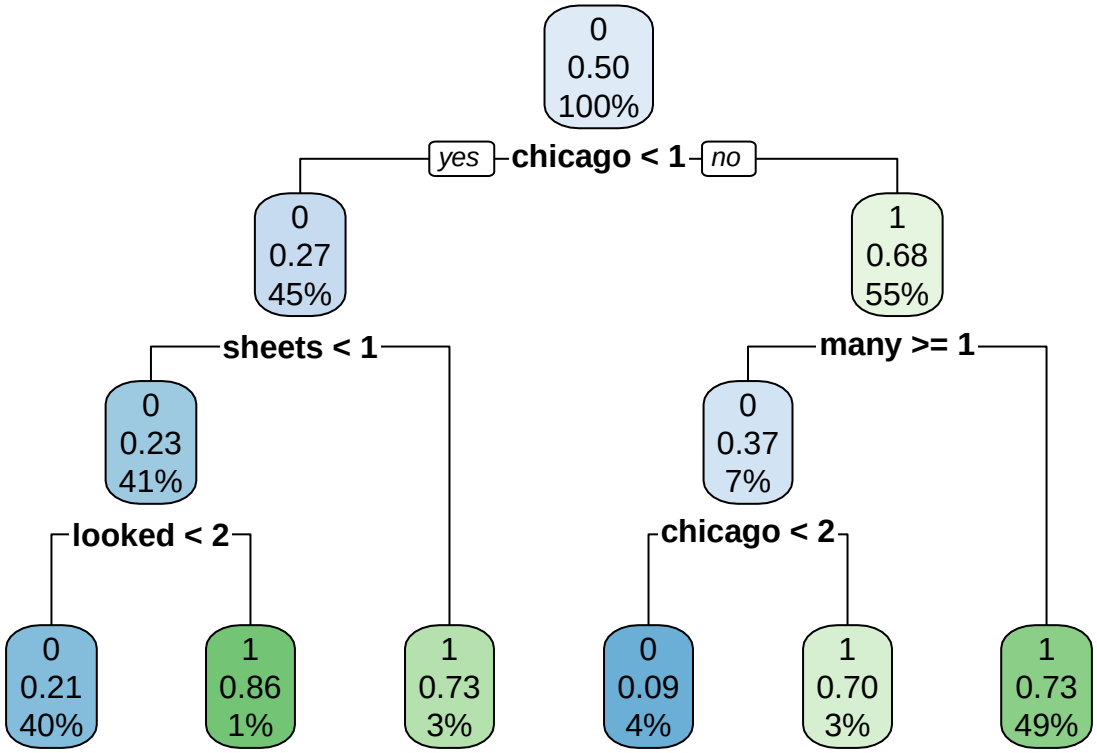
| metric | value |
|---|---|
| recall | 0.6375000 |
| miss-rate | 0.3625000 |
| fall-out | 0.0875000 |
| selectivity | 0.9125000 |
| prevalence | 0.5000000 |
| precision | 0.8793103 |
| false omission rate | 0.2843137 |
| pos likelihood ratio | 7.2857143 |
| neg likelihood ratio | 0.3972603 |
| accuracy | 0.7750000 |
| false discovery rate | 0.1206897 |
| neg predictive value | 0.7156863 |
| diagnostic odds ratio | 18.3399015 |
| F1 | 0.7391304 |

## Classification trees

```
# Grow the tree
reviews.rpart <- rpart(label ~ .,
                       data = data.frame(as.matrix(train.dtm), 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)
```




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

### Random forests

```
# Train random forest with default settings: 500 trees and mtry = 17
reviews.rf <- randomForest(as.factor(label) ~ .,
                           data = data.frame(as.matrix(train.dtm), label = labels[index.train]))
# Make predictions
reviews.rf.pred <- predict(reviews.rf,
                           newdata = data.frame(as.matrix(test.dtm), 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 | 0.7250000 |
| miss-rate | 0.2750000 |
| fall-out | 0.2000000 |
| selectivity | 0.8000000 |
| prevalence | 0.5000000 |
| precision | 0.7837838 |
| false omission rate | 0.2558140 |
| pos likelihood ratio | 3.6250000 |
| neg likelihood ratio | 0.3437500 |
| accuracy | 0.7625000 |
| false discovery rate | 0.2162162 |
| neg predictive value | 0.7441860 |
| diagnostic odds ratio | 10.5454545 |
| F1 | 0.7532468 |

## 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?