516 Case 2

Kun Qian

2/2/2021

Context

Wikipedia is a free online encyclopedia that anyone can edit and contribute to. It's the single biggest encyclopedia on the internet available in many languages. On the English language version alone, there are currently 4.3 million pages and about 130,000 edits per day.

One of the consequences of being open and editable by anyone is that some people vandalize pages. This can be in the form of removing content, adding promotion or inappropriate content, or changing the meaning of an article. The purpose of this project is to build a model to predict vandalism for the page 'Language'. Data is the version history of this page.

Preparation

```
# load data
wiki <- read.csv('Wikipedia.csv')
# Take a look at the data
head(wiki)
```

```
##   Vandal Minor LoggedIn HTTP NumWordsAdded NumWordsRemoved
## 1      0     1        1    1            96                0
## 2      0     1        1    0             3                1
## 3      0     0        1    0             0                4
## 4      0     1        0    0            10               92
## 5      0     1        1    1            94               10
## 6      0     0        1    0             4                4
```

A.Data Exploration

i. How many cases of vandalism were detected in the history of this page?

```
sum(wiki$Vandal)
```

```
## [1] 1815
```

```
sum(wiki$Vandal)/nrow(wiki)
```

```
## [1] 0.4682663
```

1815 cases of vandalism were detected. In total, 46.8% of total edits were classified as vandalism.

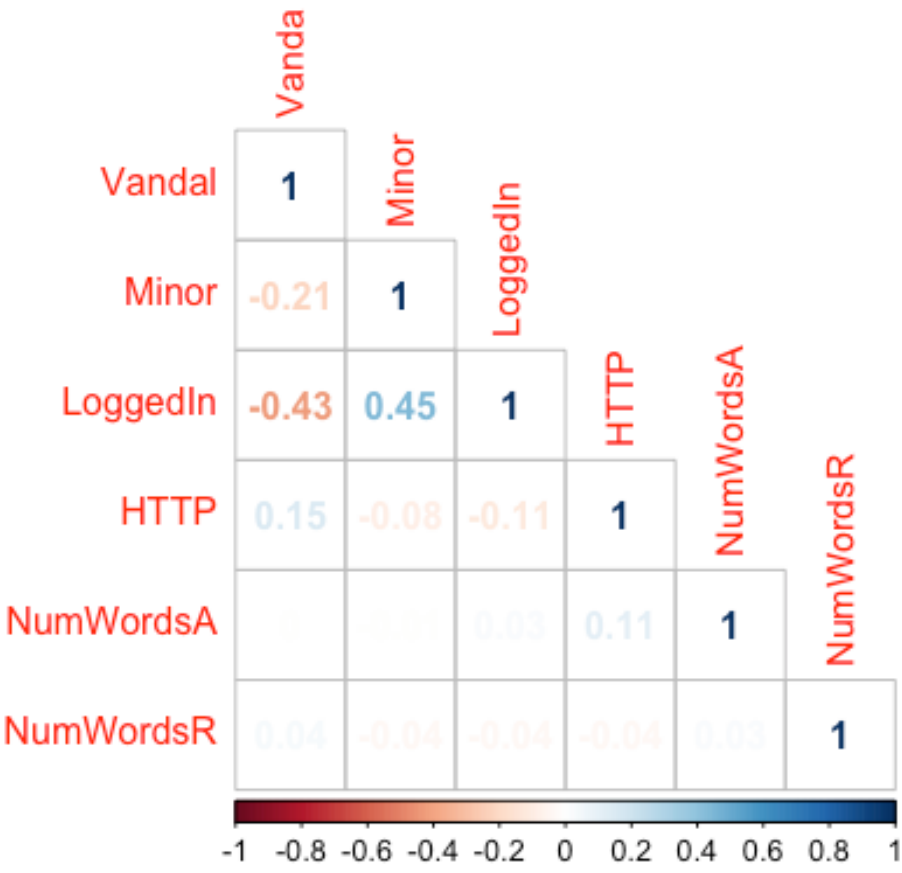
ii. What is the average number of words that were added? What is the average number of words that were removed?

```
mean(wiki$NumWordsAdded)

## [1] 4.050052

mean(wiki$NumWordsRemoved)

## [1] 3.5129
```

Average number of words that were added: 4.05 words; Average number of words that were removed: 3.5 words.

iii. Which variable is the most correlated with teh variable Vandal?

```
library(corrplot)
colnames(wiki)[5:6] <- c("NumWordsA", "NumWordsR")
corrplot(cor(wiki), type='lower', method='number', tl.srt=90)
```



Whether the editor logged in or not is most correlated to Vandal, negatively. The correlation is -0.43.

B. Split the data, and report accuracy of the baseline model

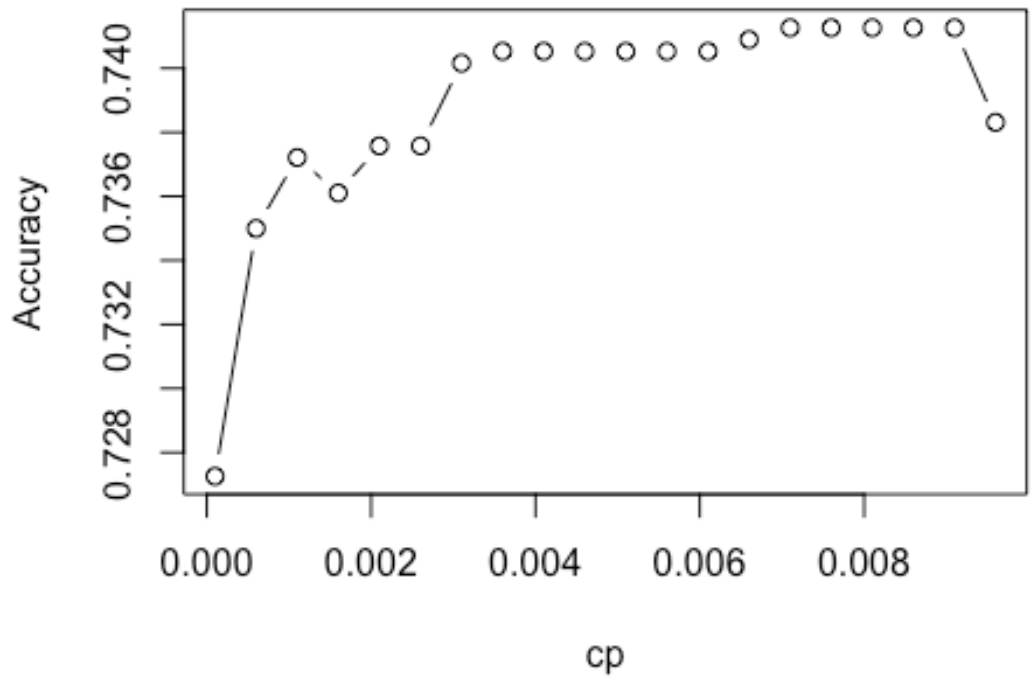
```
# load the library for random splitting
library(caTools)

# set the random seed
set.seed(1234)

# create a boolean vector that randomly assign 70% of the original data with
'TRUE', while maintaining the relative Vandal ratio in both 'True' and
'False' groups
split <- sample.split(wiki$Vandal, SplitRatio = 0.7)

# filter and store the train data
wiki_train <- wiki[split==TRUE,]
wiki_test <- wiki[split==FALSE,]

# get the accuracy of the baseline model on the testing set
sum(wiki_test$Vandal==0)/length(wiki_test$Vandal)
```

```
## [1] 0.5313844
```

The accuracy of always predicting 'not vandalism' has an accuracy rate of 53.1%.

C. Build a CART model to predict Vandal, using all other variables as independent variables.

```
# load the package of recursive partitioning
library(rpart)
library(rpart.plot)

# First, fit with a raw tree model
wiki.tree <- rpart(Vandal~., wiki_train, method='class')

# then, use cross validation to tune the model and pick the best complexity
parameter (cp)
library(caret)
library(e1071)
# use 10 folds for cross-validation
numFolds <- trainControl(method='cv', number=10)
# hand pick a sequence of possible cp values
cpGrid = expand.grid(.cp=seq(0.0001, 0.01, 0.0005))
# Use the train function to try each of the cp value on cross-validation sets
t <- train(as.factor(Vandal)~., wiki_train, method='rpart',
trControl=numFolds, tuneGrid=cpGrid)
# plot the relationship between cp and accuracy
plot(t$results$cp, t$results$Accuracy, type='b', ylab="Accuracy", xlab="cp")
```




```
# print the cp value with highest accuracy
t$bestTune$cp

## [1] 0.0091

# then, build a better tree model with the selected cp value
wiki.tree.CV <- rpart(Vandal~., wiki_train, method='class', cp=t$bestTune$cp,
minbucket=25)

# construct a function to report evaluation metrics
getAccuracy <- function(model) {
  # use the tree model to make prediction on the test data
  pred <- predict(model, wiki_test, type='class')
  # report confusion matrix
  cm <- table(wiki_test$Vandal, pred)
  # report accuracy rate
  accuracy <- sum(diag(cm))/sum(cm)
  # also take a look at precision, recall, and f1 score
  precision <- posPredValue(as.factor(pred), as.factor(wiki_test$Vandal),
positive="1")
  recall <- sensitivity(as.factor(pred), as.factor(wiki_test$Vandal),
positive="1")
  F1 <- (2 * precision * recall) / (precision + recall)
```

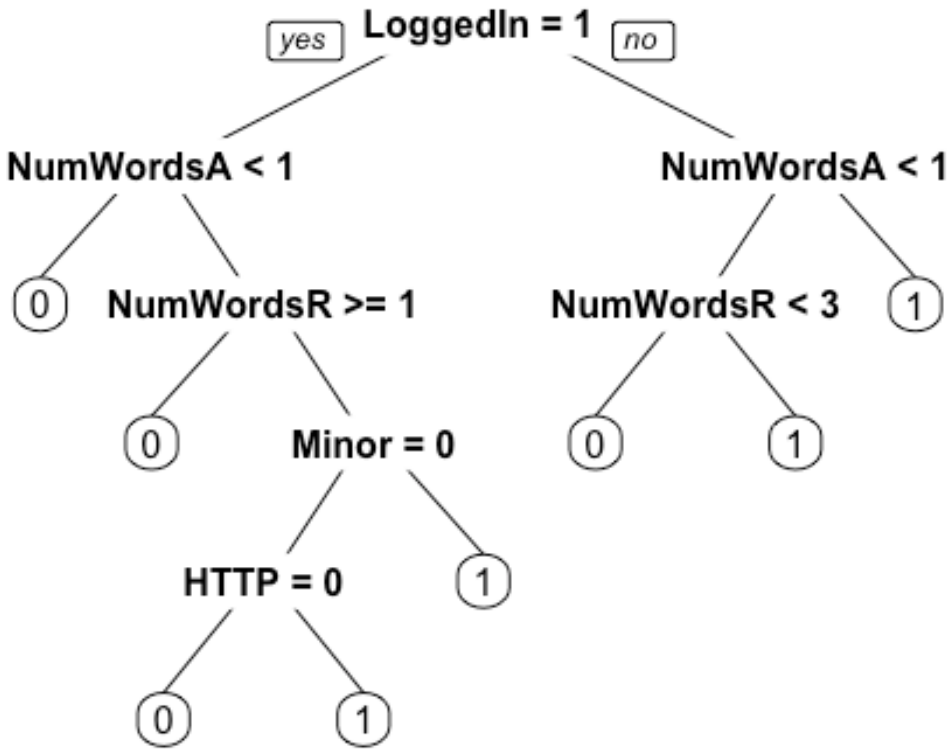
```
  # construct a named list to return the result
  result <- c(accuracy, precision, recall, F1)
  names(result) <- c("accuracy", "precision", "recall", "F1")
  return(result)
}

# Get the evaluation metrics for the raw model and tuned model
library(knitr)
kable(getAccuracy(wiki.tree), digits=3, col.names = "Raw Tree")
```

	Raw Tree
accuracy	0.733
precision	0.890
recall	0.492
F1	0.634

```
kable(getAccuracy(wiki.tree.CV), digits=3, col.names = "Tuned Tree")
```

	Tuned Tree
accuracy	0.743
precision	0.827
recall	0.571
F1	0.675

```
# plot the tuned tree
prp(wiki.tree.CV)
```



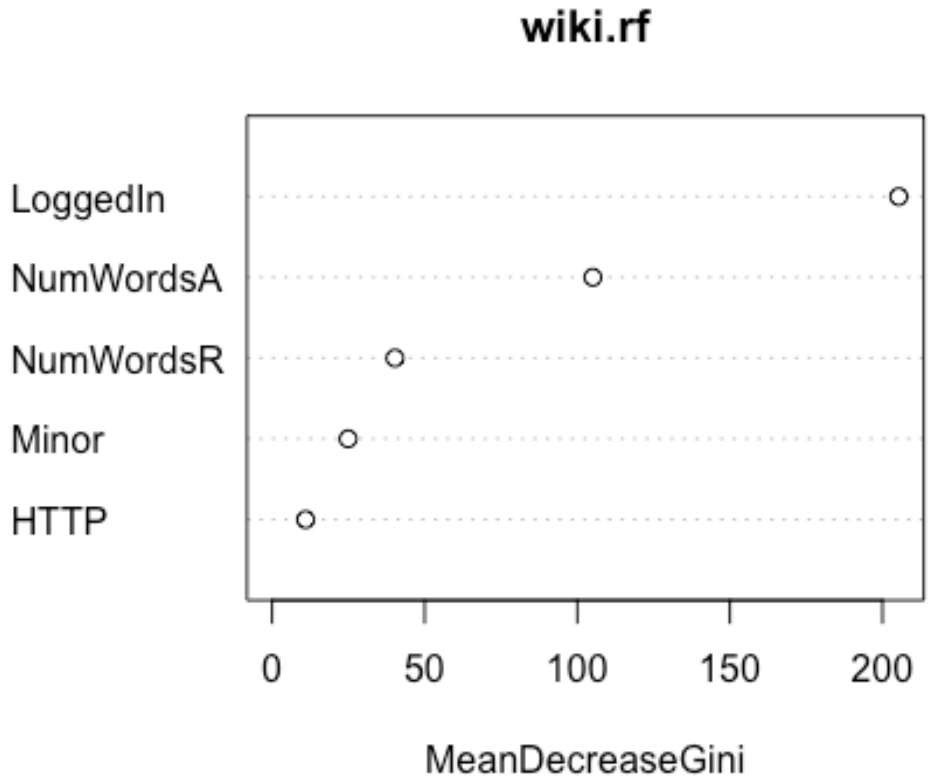
- The raw tree has an accuracy rate of 73.3%; The tuned tree improves accuracy to 74.3%. We also see an increase of 0.04 in F1 score.
- The accuracy of the tuned tree model is much higher than the baseline model
- All variables are used in the tuned model. The most significant variable seems to be 'Logged-In'. The second most significant variable is "NumWordsAdded".

D. Build a random forest model to predict Vandal

```
library(randomForest)
# build the model
wiki.rf <- randomForest(as.factor(Vandal)~., data=wiki_train, nodesize=25,
ntree=1000)
# make prediction
wiki.rf.pred <- predict(wiki.rf, wiki_test)
# report the evaluation metrics
getAccuracy(wiki.rf)
```

```
##  accuracy precision    recall        F1
## 0.7437661 0.8310992 0.5688073 0.6753813
```

```r
# plot feature importance
varImpPlot(wiki.rf)
```



The random forest model improves the accuracy and F1 score a little bit from the CART model. The most significant feature remains to be 'Looged-In'.

E. Evaluation Quesitions

i. Do you think the model you built could be useful to Wikipedia to detect vandalism? Why or why not?

Yes. The CART has an accuracy of 75.3%, meaning it can correctly predict vandalism at about 74.3% of the time. It also has a high precision, meaning that out of all cases it predict as 'vandalism', there's a 82.7% of chance that the case is indeed vandalism. Despite there will still be undetected vandalism cases due to a relatively low recall, the algorithm will be effective enough to deploy and to automatically detect, report, and prevent vandalism.

ii. If you could collect more data about the edits, what variables would you want? Why?

- More user behavioral data such as 'past activities', 'number of total edits', 'time spent on the site', 'time spent on editting' and etc. Since 'Logged-In' is the most significant variable, it seems like information about the editor is helpful to determine vandalism. Intuitively, first-time users with low interaction and commitment with Wikipedia is more likely to commit vandalism. If we can get more information on who the editor is and what has he done in Wikipedia before, we might improve the model.

- If the editor add words, what are the words s/he added. This variable directly look into the content of the editing. We could use something similar to a spam detection model to classify the likelihood that the words added is ads or spam.

iii. The data used in this problem was collected from the page about Language. Do you think this model would easily extend to other pages? Why or why not?

The variables used in the current model is quite generic and should be able to also apply to other pages. However, it's necessary to retrain the model on a more diverse data sets consist of multiple pages. This is because wiki pages can be about very different topics. The corresponding vandalism on different pages might also be different. For example, a page of a politician is susceptible to fake news and personal assault; a page of a commercial company is more likely to be damaged by competitors' ads and angry customers. In this case one possible solution is to train this generic model on all kinds of pages' data; Another way is to categorize all pages into categories such as 'people', 'history', 'science', 'business', and etc, and build separate models based on the different traits of each category.