Natural Language Processing for E-Commerce Trademark Infringement Detection

Zhiyu Lin

Synthesis

E-commerse platforms such as Taobao.com contain lots of trademark infringement acts, which could lead to consumer confusion and mistrust. Because the number of such infringements is extremely large, our goal here is to utilize machine learning and natural language processing to detect trademark infringements. Assuming that authentic branded items and fake ones are different in product quality, sales service and price, and that such differences could be reflected in consumer reviews, we aim to build different review profiles for authentic and fake items. Here we decided to use VANS sneakers as an example.

We first used web crawler and downloaded consumer comments from more than 12,000 VANS listings on Tianmao and Taobao. We will use Python Jieba dictionary to separate the consumer comments into phrase features and count the frequency of appearances for each feature. We will also select the important features, perform some exploratory analysis, and apply several models such as lasso selection, decision trees and random forest with cross validation. Our goal is to achieve a high classification accuracy.

Data Generation

```
library(readxl)
library(plyr)
library(dplyr)
```

Generate word frequency files in Python

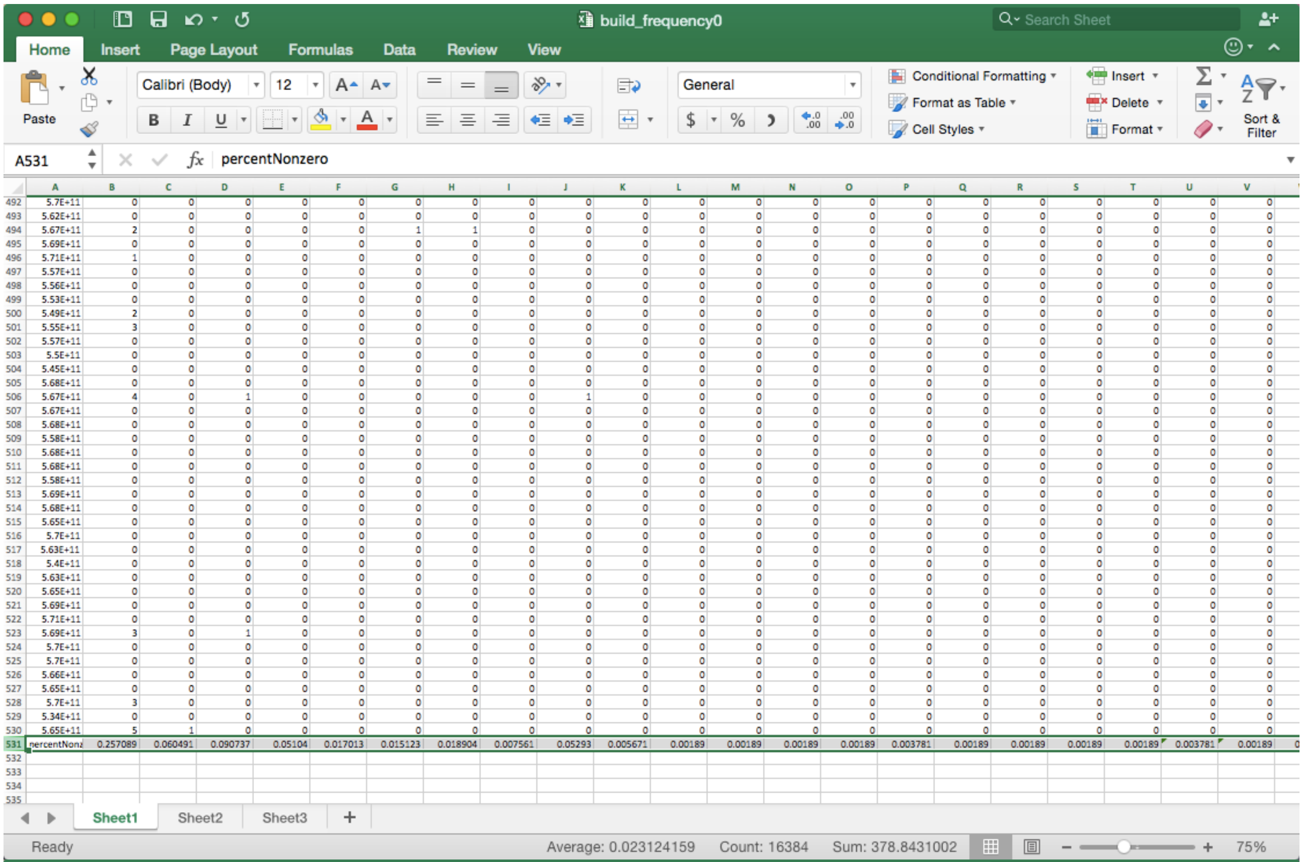
This step is done previously through Jieba dictionary in Python. Code and crawler data credit to Duo Cao and can be found in the Google Drive.

Select features

The build_frequency files contain features on Tianmao (build_frequency.xlsx) and Taobao (build_frequency0.xlsx). We know that all Tianmao listings contain real VANS items and all Taobao listings contain fake ones. In these files, each row represents an item ID and each column a feature word or phrase extracted by Jieba dictionary.

Looking vertically, Tianmao frequency file has 392 items and Taobao 529 items. Looking horizontally, both files contain more than 16000 features, which include “good” features that are informative of whether an item is real or fake and “bad” features that are random noises such as sentence conjectures.

We want to first narrow down the features by selecting the item IDs that contain 50+ comments. Their IDs are extracted through Excel Pivot Tables and are listed in forms of row numbers down below.

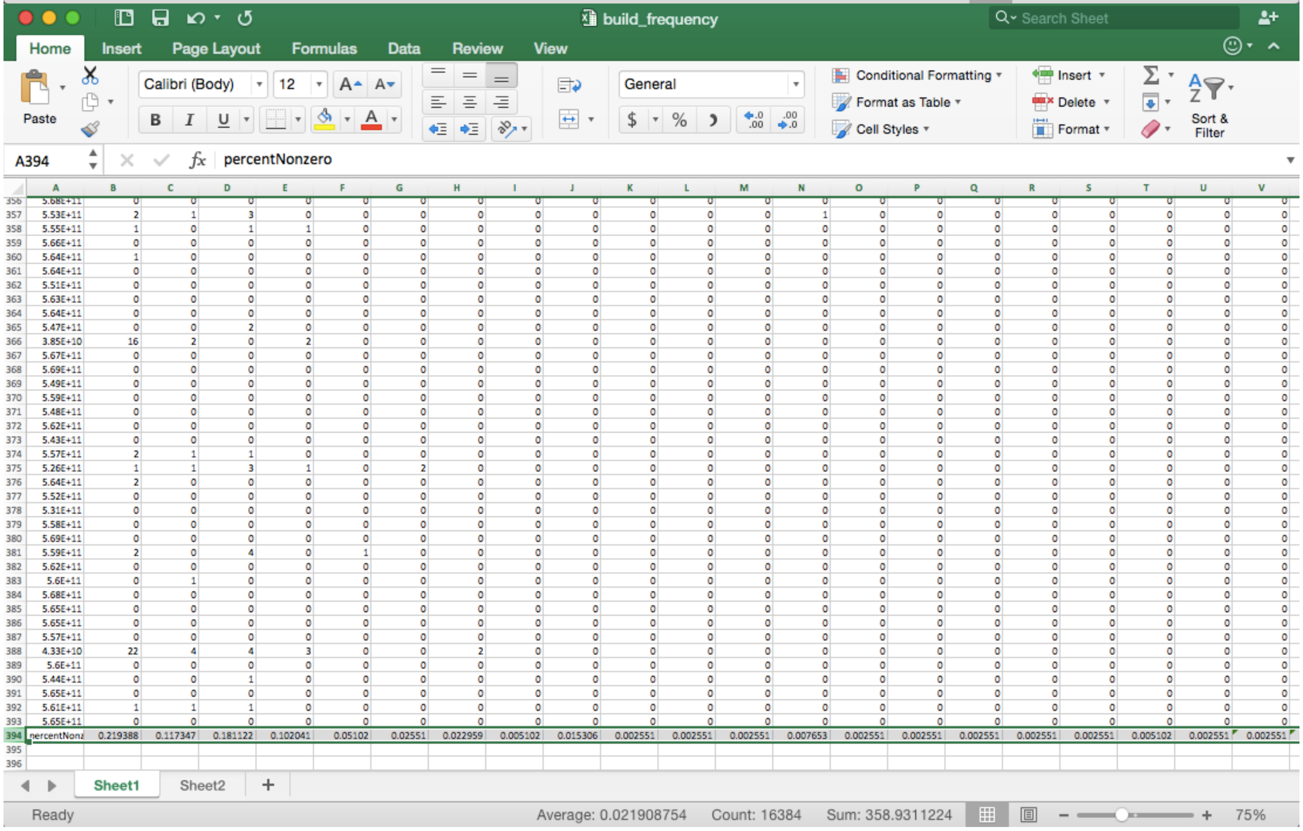


A531 percentNonzero

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
492	5.7E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
493	5.62E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
494	5.67E+11	2	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
495	5.69E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
496	5.71E+11	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
497	5.57E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
498	5.56E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
499	5.53E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
500	5.49E+11	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
501	5.55E+11	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
502	5.57E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
503	5.5E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
504	5.45E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
505	5.68E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
506	5.67E+11	4	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
507	5.67E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
508	5.68E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
509	5.58E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
510	5.68E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
511	5.68E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
512	5.58E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
513	5.69E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
514	5.68E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
515	5.65E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
516	5.7E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
517	5.63E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
518	5.4E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
519	5.63E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
520	5.65E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
521	5.69E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
522	5.71E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
523	5.69E+11	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
524	5.7E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
525	5.7E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
526	5.66E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
527	5.65E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
528	5.7E+11	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
529	5.34E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
530	5.65E+11	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
531	percentNonz	0.257089	0.060491	0.090737	0.05104	0.017013	0.015123	0.018904	0.007561	0.05293	0.005671	0.00189	0.00189	0.00189	0.00189	0.003781	0.00189	0.00189	0.00189	0.00189	0.003781	0.00189
532																						
533																						
534																						
535																						

Sheet1 Sheet2 Sheet3 +

Ready Average: 0.023124159 Count: 16384 Sum: 378.8431002 75%



A394 percentNonzero

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
356	5.68E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
357	5.53E+11	2	1	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
358	5.55E+11	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
359	5.66E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
360	5.64E+11	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
361	5.64E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
362	5.51E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
363	5.63E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
364	5.64E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
365	5.47E+11	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
366	3.85E+10	16	2	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
367	5.67E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
368	5.69E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
369	5.49E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
370	5.59E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
371	5.48E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
372	5.62E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
373	5.43E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
374	5.57E+11	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
375	5.26E+11	1	1	3	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
376	5.64E+11	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
377	5.52E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
378	5.31E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
379	5.58E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
380	5.69E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
381	5.59E+11	2	0	4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
382	5.62E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
383	5.6E+11	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
384	5.68E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
385	5.65E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
386	5.65E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
387	5.57E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
388	4.33E+10	22	4	4	3	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
389	5.6E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
390	5.44E+11	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
391	5.65E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
392	5.61E+11	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
393	5.65E+11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
394	percentNonz	0.219388	0.117347	0.181122	0.102041	0.05102	0.02551	0.022959	0.005102	0.015306	0.002551	0.002551	0.002551	0.007653	0.002551	0.002551	0.002551	0.002551	0.002551	0.005102	0.002551	0.002551
395																						
396																						

Sheet1 Sheet2 +

Ready Average: 0.021908754 Count: 16384 Sum: 358.9311224 75%

```
tmX <- read_excel("/Users/Zhiyu666/Desktop/Legaltech project/build_frequency.xlsx", s
heet = "Sheet1")
tmY <- tmX[c(104,237,9,214,332,34,259,126,12,365,150,356,24,330,380,103,246,100,261,3
83,271,142,122,115,8,346,343,317,374,89,264,267,151,357,316,182,269,233,299,208,173,5
3,29,355,102,22,167,175,219,387,155,319,310,354,346,27,302,240,179,141,30,291,78,71,9
1,68,88,10,87,186,218,260,105,391,124,195,38,326,170,234,129,222,127,364,119,118,293,
389,50,378,217,75,101,169,369,348,230,265,157,7,287,136,99,5,117,353,371,344,156,336,
212,23,111,375,35,154,304,97,5,185,370,55,96,220,206,284,250,273,203,54,110,3,93,292,
312,386),]

tbX <- read_excel("/Users/Zhiyu666/Desktop/Legaltech project/build_frequency0.xlsx",
sheet = "Sheet1")
tbY <- tbX[c(301,438,131,251,141,34,236,393,24,529,257,448,365,38,386,136,110,134,84,
10,219,411,493,505,384,1,377,197,58,452,174,429,175,118,151,156,28,12,64,271,410,478,
241,308,440,177,415,409,6,522,171,333,404,317,466,94,189,191,150,318,446,25,247,237,2
39,210,462,481,86,176,3,500,265,188,202,242,124,267,469,299,364,120,21,464,109,184,34
0,292,467,370,108,233,22,73,213,309,509,352,357,57,128,264,182,137,11,428,507,399,472
,329,187,488,62,215,216,15,102,17,66,284,527,103,200,154,69,388,252,142,303,162,92,27
5,60,347,421,217,19,207,379,375,149,414,450,31,4,320,93,164,523,476,98,26,506,355,293
,165,477,419,20,461,249,470,82,445,238,83,283,155,113,483,59,486,157,431,433,441,97,2
23,266,95,459,168,499,326,53,51,159,42,127,422,153,5,161,201,296,418,104,228,248,443,
129,513,290,274,342,302,516,172,325,391,205,41,278,412,148),]
```

Now that we have narrowed down the number of items, we want to reduce the number of features as well. The function below sort by the percent of nonzero entries in the build frequency files, and only keeps the top 250 most frequent features for Tianmao and Taobao build frequency files.

```
selectFeatures <- function(myData,myData50){
  lastRow <- nrow(myData)
  percent <- as.data.frame(myData[c(lastRow),])
  percentKeep <- sort(percent, decreasing = TRUE)[1:250]
  colNum <- match(percentKeep,as.data.frame(myData[lastRow,]))
  myFeatures <- myData50[,colNum]
  write.table(myFeatures, "/Users/Zhiyu666/Desktop/Legaltech project/tb100Features.tx
t", sep = "\t")  # change the name to tm100Features for tmX and tmY
}

selectFeatures(tmX,tmY)
selectFeatures(tbX,tbY)
```

I saved the selected feature files in .txt format for convenience. Manually change them into .xlsx format.

Now that we have selected the items and features, we need to stack them on top of each other. Our complete data should be a collection of all item IDs and features selected in Tianmao and Taobao feature files. If feature A is selected for an item in Tianmao but not in Taobao, then its frequencies for Taobao should be listed as zero, and vice versa.

More over, we want to add two columns to the left. One is the item sequencing number (ID), and the other is their y-variable (TF), 1 for Tianmao and 0 for Taobao.

In order to do this, we want to merge tm100Features and tb100Features by stacking the outer left on top of outer right. Again export the file as completeData.txt and converge the text file into excel.

```
tm100 <- read_excel("/Users/Zhiyu666/Desktop/Legaltech project/tm100Features.xlsx", s
heet = "Sheet3")
tb100 <- read_excel("/Users/Zhiyu666/Desktop/Legaltech project/tb100Features.xlsx", s
heet = "Sheet2")
left <- join(tm100,tb100,type="left")
```

```
## Joining by: ID, TF, 很, 好, 不错, 也, 鞋子, 还, 没有, 买, 舒服, 就, 不, 挺, 可以, 大, 填
写, 质量, 物流, 满意, 上, 脚, 真的, 会, 小, 但是, 不过, 很快, 没, 快, 合适, 码, 超级, 和, 特别,
起来, 一点, 来, 一, 不是, 一直, 东西, 再, 大小, 一双, 以后, 一个, 一次, 哦, 后, 做工, 啦, 刚,
下次, 宝贝, 挺舒服, 卖家, 包装, 刚好, 万斯, 不会, 一下, 值得, 便宜, 一天, 帮, 两天, 帅气, 哒
```

```
right <- join(tm100,tb100,type="right")
```

```
## Joining by: ID, TF, 很, 好, 不错, 也, 鞋子, 还, 没有, 买, 舒服, 就, 不, 挺, 可以, 大, 填
写, 质量, 物流, 满意, 上, 脚, 真的, 会, 小, 但是, 不过, 很快, 没, 快, 合适, 码, 超级, 和, 特别,
起来, 一点, 来, 一, 不是, 一直, 东西, 再, 大小, 一双, 以后, 一个, 一次, 哦, 后, 做工, 啦, 刚,
下次, 宝贝, 挺舒服, 卖家, 包装, 刚好, 万斯, 不会, 一下, 值得, 便宜, 一天, 帮, 两天, 帅气, 哒
```

```
total <- merge(left,right,all=TRUE)
write.table(total, "/Users/Zhiyu666/Desktop/completeData.txt", sep = "\t")
```

Similarly, we could lower the cut percentage and select even less features. Here, I generated two files: completeData.xlsx and completeData50.xlsx, which contain 180 and 62 features, respectively. We could use either one depending on how narrow we want the feature set to be.

Exploratory Analysis

```
library(tidyr)
library(ggplot2)
library(gridExtra)
library(ISLR)
library(glmnet)
library(tibble)
```

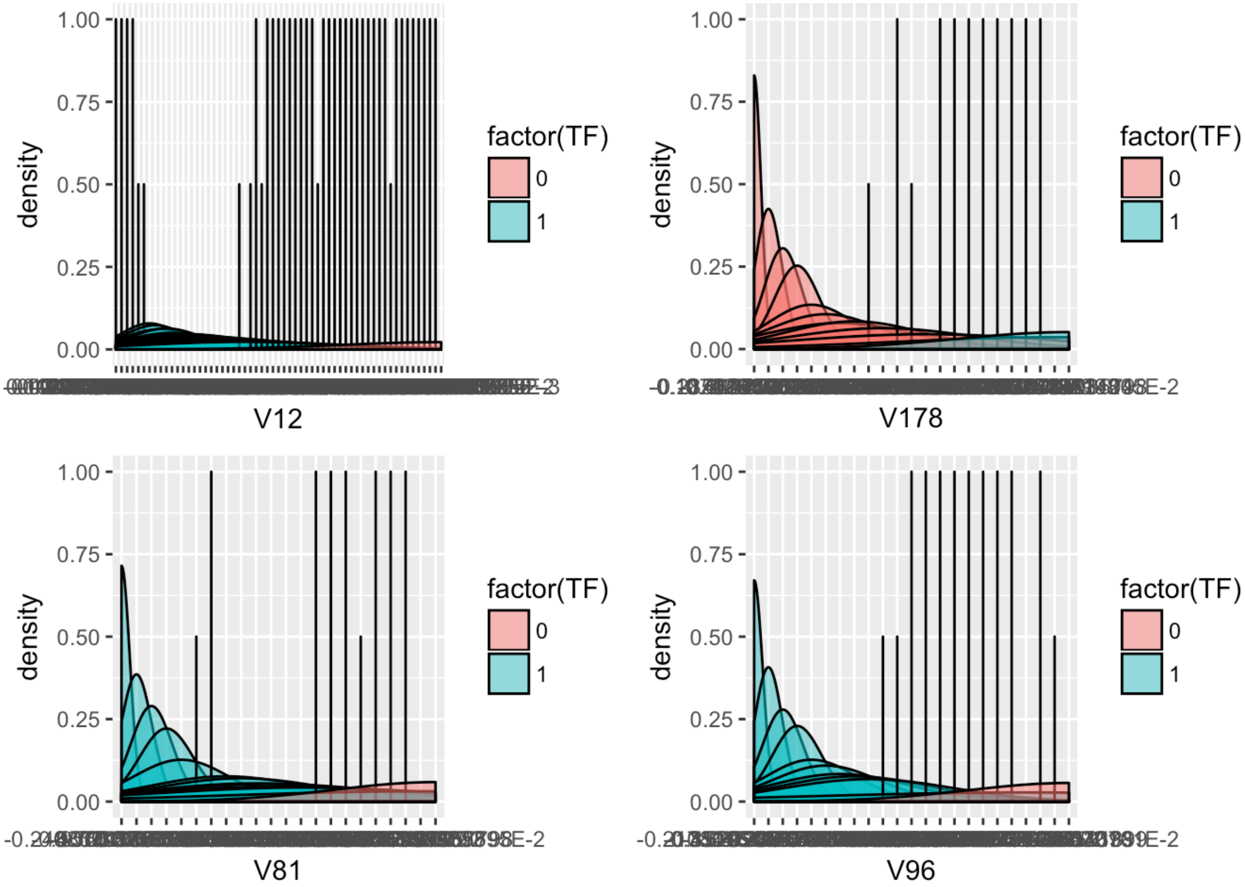
Density plots

First, going back to the big idea, we are interested in building different feature profiles for a real VANS item and a fake one listed online. Let's look at a few distribution plots.

1-D density plots

```
myData <- read_excel("/Users/Zhiyu666/Desktop/Legaltech project/completeData.xlsx", s
heet = "Sheet4")
myDatabinary <- read_excel("/Users/Zhiyu666/Desktop/Legaltech project/completeData.xl
sx", sheet = "Sheet5")
myData50 <- read_excel("/Users/Zhiyu666/Desktop/Legaltech project/completeData50.xls
x", sheet = "Sheet2")
myData50binary <- read_excel("/Users/Zhiyu666/Desktop/Legaltech project/completeData5
0.xlsx", sheet = "Sheet3")

d1 <- myData %>% ggplot(aes(x=V12,fill=factor(TF))) + geom_density(alpha=.5) #V12 is
 "正品"
d2 <- myData %>% ggplot(aes(x=V178, fill=factor(TF))) + geom_density(alpha=.5) #V178
 is "孩子"
d3 <- myData %>% ggplot(aes(x=V81, fill=factor(TF))) + geom_density(alpha=.5) #V81 is
 "一般"
d4 <- myData %>% ggplot(aes(x=V96, fill=factor(TF))) + geom_density(alpha=.5) #V96 is
 "半码"
grid.arrange(d1,d2,d3,d4, nrow = 2)
```



I randomly selected 4 features and wanted to see whether the real VANS items have different distributions than the fake ones. It turns out they do. In the plots above, real VANS items are blue and fake ones are red. The verticle axis represents density and the horizontal access represents frequency. For example, in the first plot, we could say that real VANS items contain the feature “正品” a lot more frequent than fake VANS items. In the second plot, we could say that the real VANS items contain the feature “孩子” a lot less than the fake ones.

2-D density plots

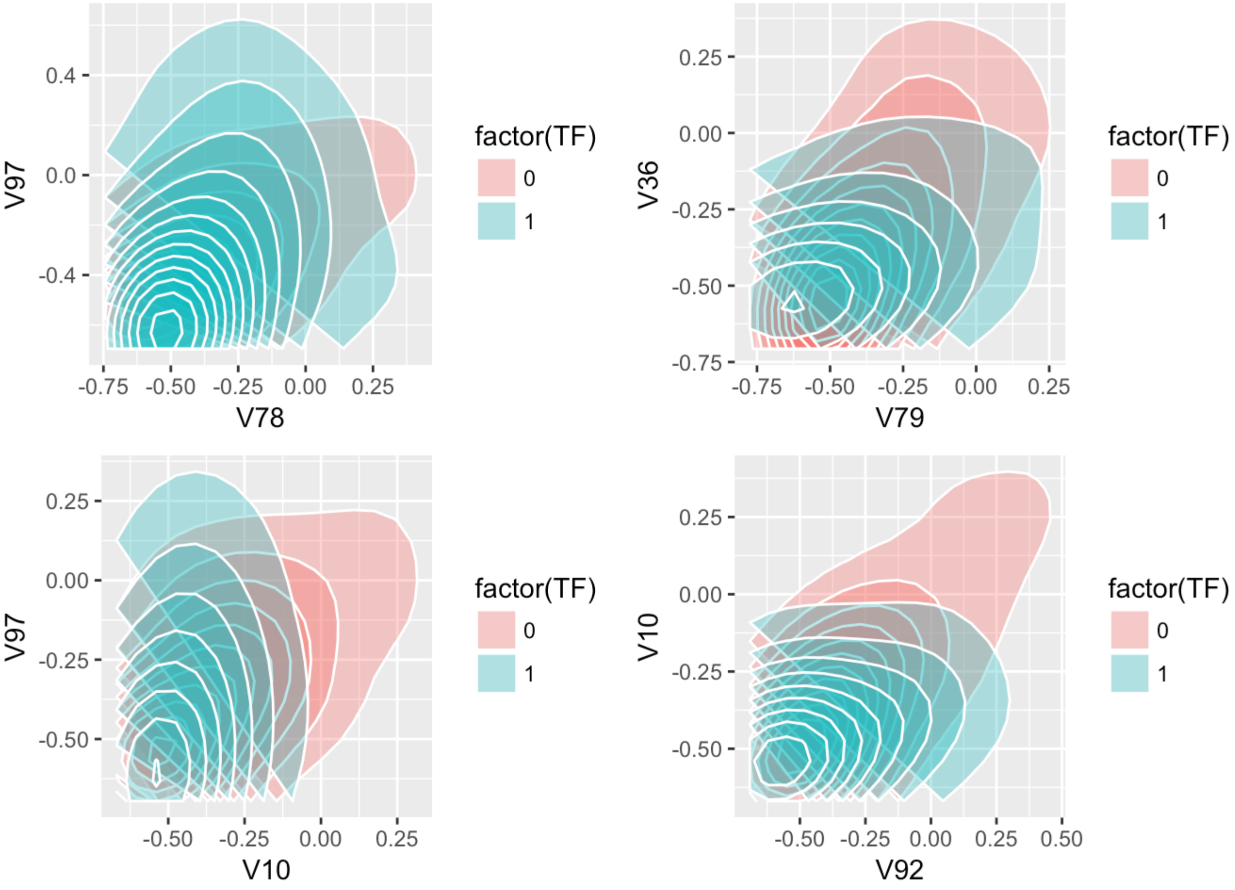
```
p1 <- ggplot(myData, aes(x=V78, y=V97) ) +
  stat_density_2d(aes(fill = factor(TF)), geom = "polygon", colour="white",alpha=0.35
)  # x = "卖家", y = "便宜"

p2 <- ggplot(myData, aes(x=V79, y=V36) ) +
  stat_density_2d(aes(fill = factor(TF)), geom = "polygon", colour="white",alpha=0.35
)  # x = "包装", y = "快"

p3 <- ggplot(myData, aes(x=V10, y=V97) ) +
  stat_density_2d(aes(fill = factor(TF)), geom = "polygon", colour="white",alpha=0.35
)  # x = "舒服" Y = "便宜"

p4 <- ggplot(myData, aes(x=V92, y=V10) ) +
  stat_density_2d(aes(fill = factor(TF)), geom = "polygon", colour="white",alpha=0.35
) # x = "值得", y= "舒服"

grid.arrange(p1,p2,p3,p4, nrow = 2)
```



Blue represents authentic VANS sneakers and red represents the fake ones. From these plots we could also see that the distribution profiles for authentic VANS sneakers look significantly different than the fake ones.

Lasso feature selection

Lasso selection is one of the most common methods to select a subset of predictors to enhance prediction accuracy. Here we want to use lasso selection to pick the most important predictors to determine the authenticity of a VANS item listed online.

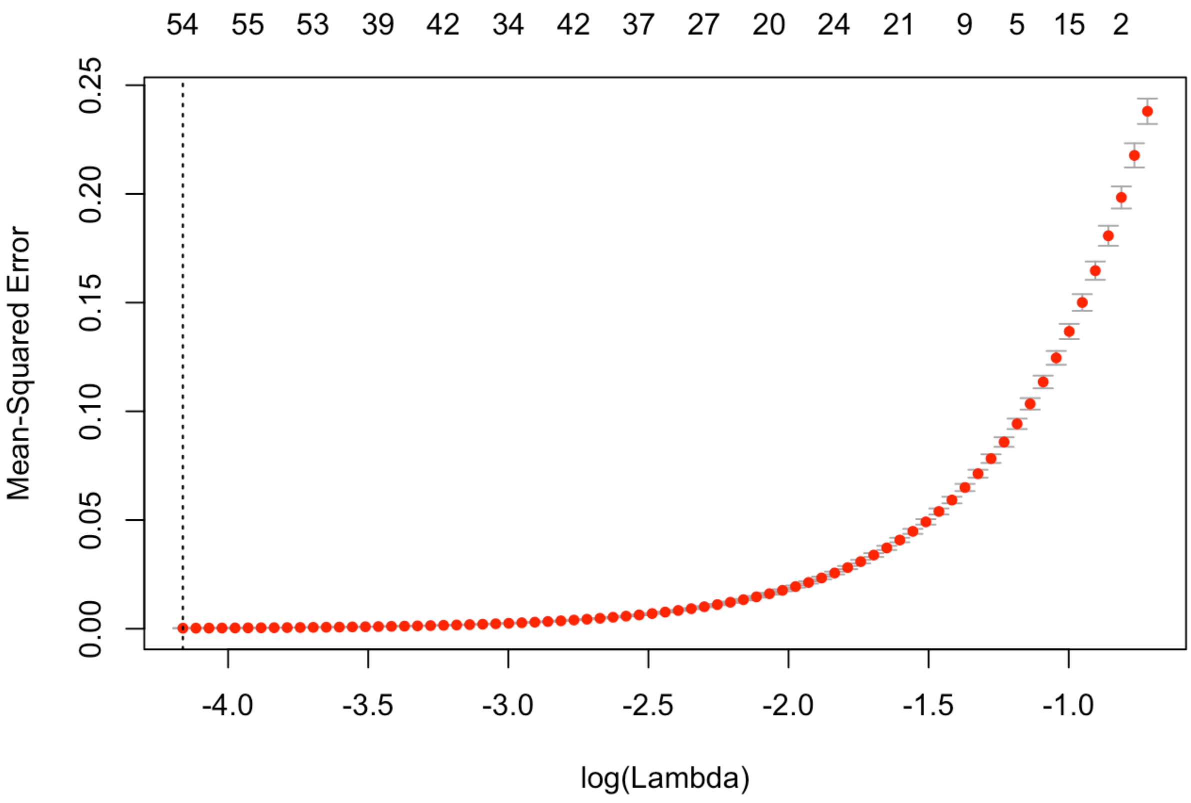
```
x <- model.matrix(TF ~., data=myData)[,-c(1:2)]
y <- as.factor(myData$TF)
lambdas <- c(0,10,100,1000)
lasso1 <- glmnet(x = x, y = y, alpha = 1, family = "binomial", lambda = lambdas)
#coef(lasso1) # too long for display
head(coef(lasso1))
```

```
## 6 x 4 sparse Matrix of class "dgCMatrix"
##                     s0         s1         s2            s3
## (Intercept) -0.4579831 -0.4579831 -0.4579831 -7.557217e+00
## V1           .          .          .          2.216291e-05
## V2           .          .          .          2.530113e-04
## V3           .          .          .         -4.326308e-05
## V4           .          .          .         -5.752584e-04
## V5           .          .          .         -4.003083e-04
```

From the coefficient output we could see that all explanatory variables that we manually selected by the percent of non-zero comments before are “good” indicators of the authenticity. Lasso could not narrow down to just a few. This might suggest that our data-mining process is a bit overdone and that we could try this out again on the original word frequency dataset later.

To add cross-validation to the lasso selection, we could see that the penalty factor lambda does not vary much, affirming our finding that the dataset might be too clean.

```
cv.lasso.mod <- cv.glmnet(x = x, y = myData$TF, alpha = 1)
plot(cv.lasso.mod)
```



```
(best.lambda <- cv.lasso.mod$lambda.min)
```

```
## [1] 0.01558591
```

We could see from the cross validation lasso plot that lasso selection fails to produce a decreasing MSPE with the increasing Log(lambda). There might be two reasons. First, a large amount of predictive variables are significant. Second, these predictive variables might be highly correlated.

Models

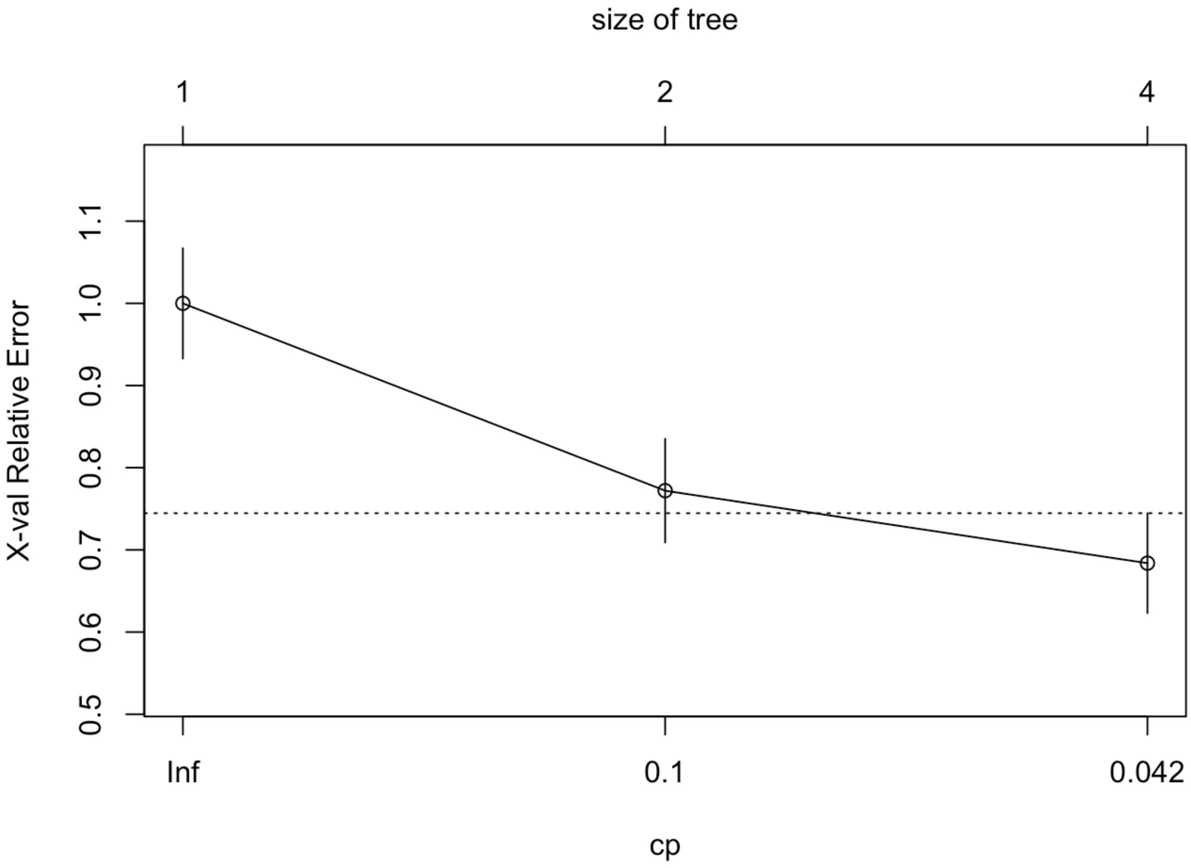
Since we have a specific response variable (TF) and this response variable is binary, we will use decision tree and random forest as our main model.

```
library(rpart)
library(forcats)
library(rpart.plot)
library(randomForest)
library(gbm)
```

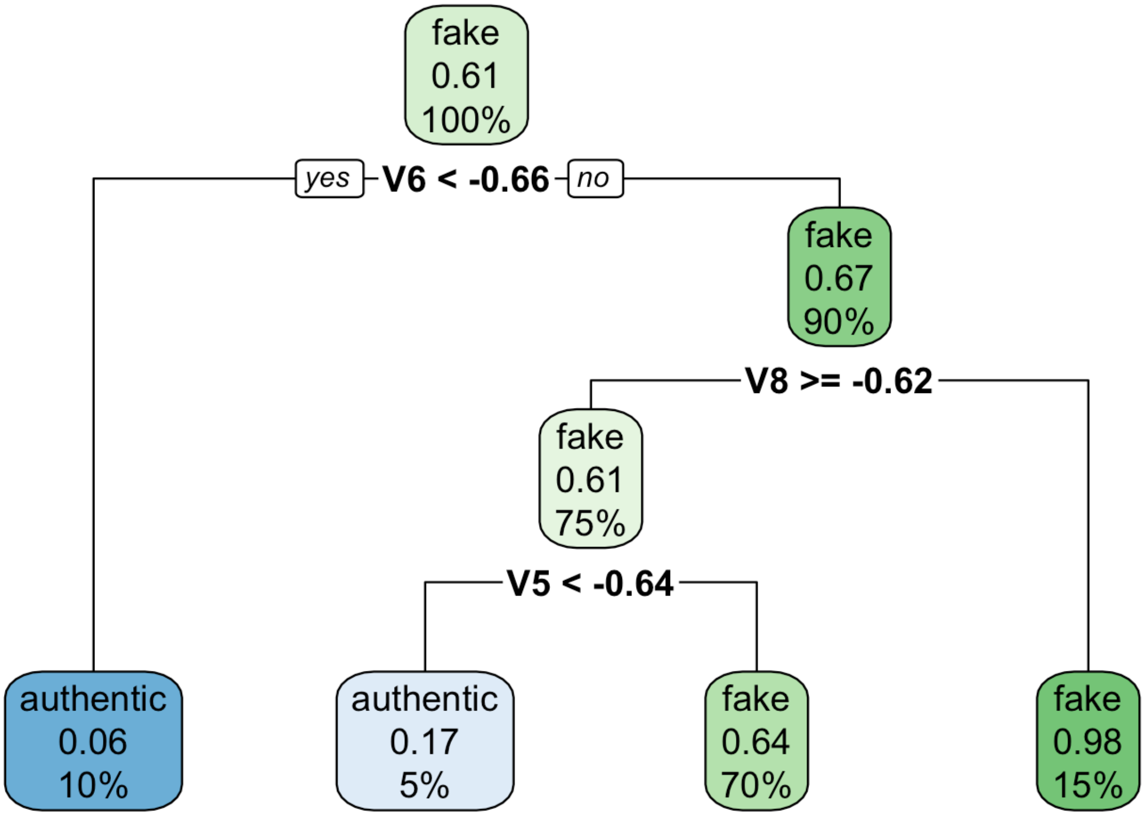
Model 1: tree_0

First of all, we want to try out a simple decision tree with a couple random predictive variables. I pick “质量”, “满意” and “小”, which are represented as V5, V6 and V8 respectively.

```
tree_0 <- rpart(TF ~ V8+V5+V6, myData50binary, cp=0.04)
plotcp(tree_0)
```



```
rpart.plot(tree_0,tweak = 1.2)
```



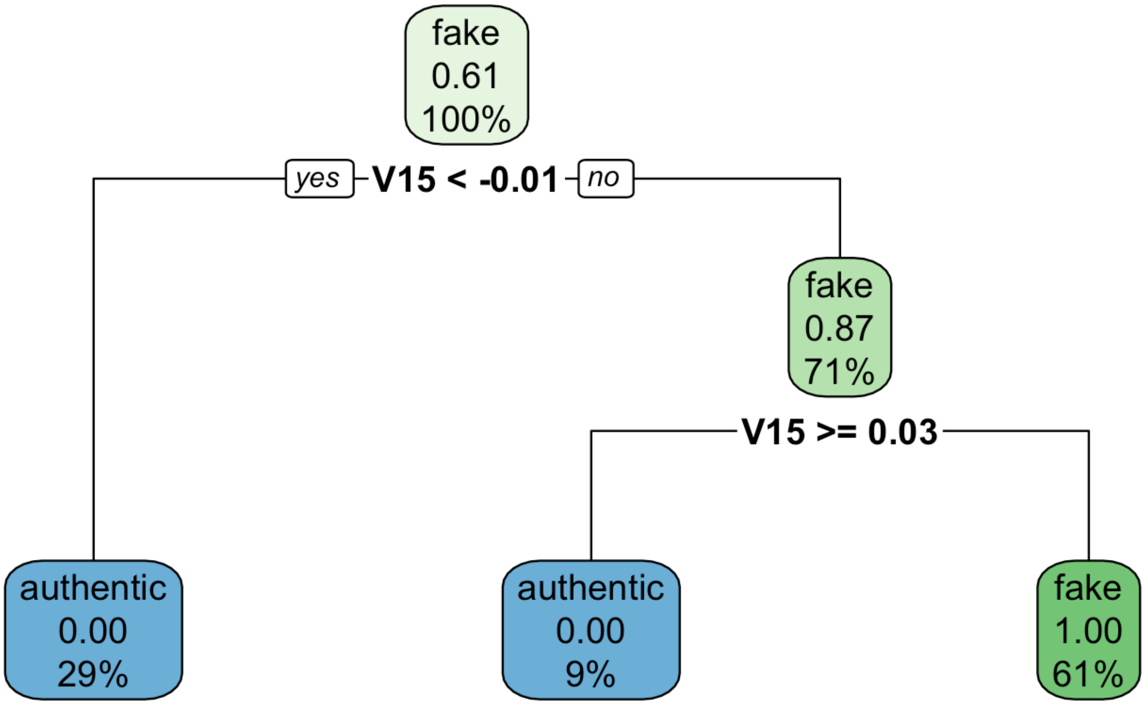
Each node shows the predicted class (authentic or fake), the predicted probability of being fake and the percentage of observations in the node. For example, the root node has 100% of the observations with an authenticity rate at 0.39 .

To classify an observation’s authenticity according to this plot, we first look at its V6 (“满意”) value. If an observation’s V6 score is lower than -0.66, then it is 94% likely to be authentic. However, if an observation’s V6 score is equal to or larger than -0.66 then we will look into its V8 (“小”) value. If its V8 score is smaller than -0.62, then it is 98% likely to be fake. However, if an observation’s V8 score is larger than or equal to -0.62, we will look at its V5 (“质量”) value. If its V5 is smaller than -0.64, it is 83% likely to be authentic, otherwise it is 64% likely to be fake.

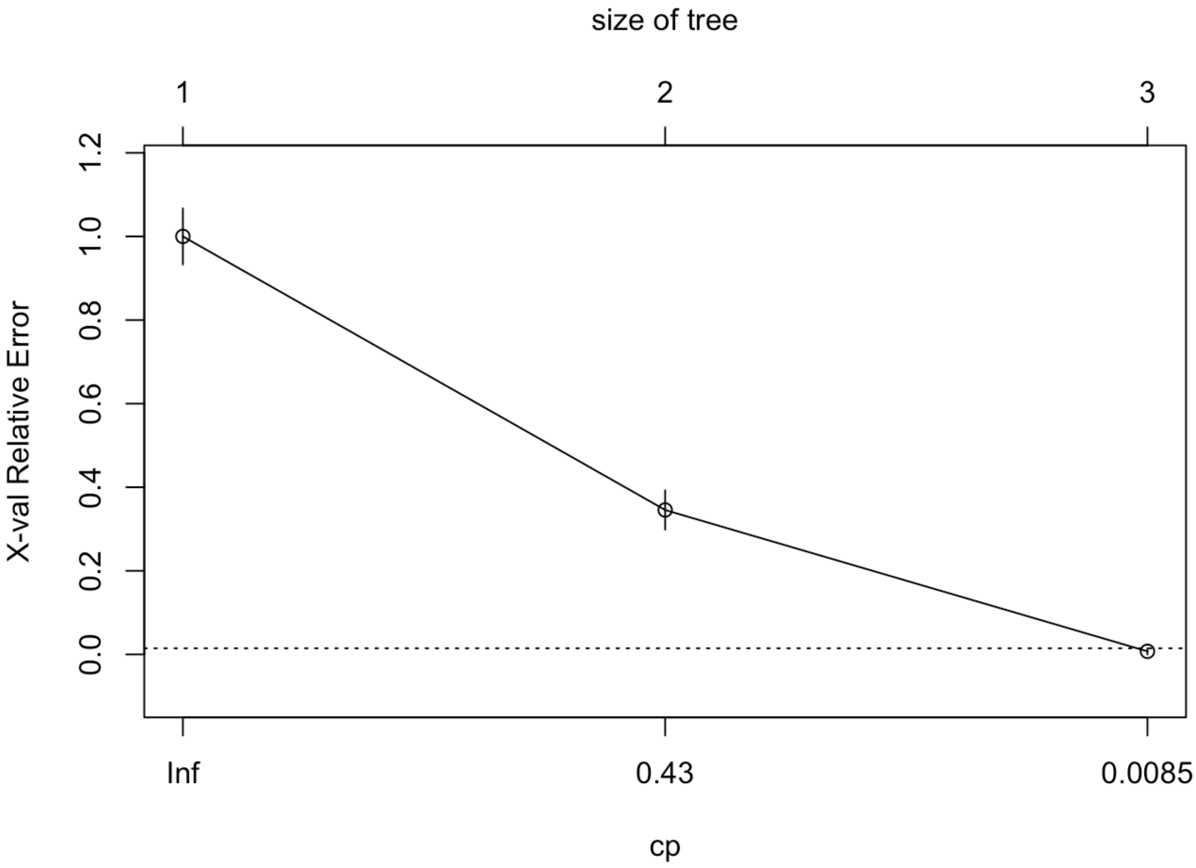
Model 2: tree_big

Now we will incorporate all the explanatory variables and let decision tree pick the important ones.

```
tree_big <- rpart(TF ~ ., myData50binary, cp=0.0003)
rpart.plot(tree_big,tweak = 1.2)
```



```
plotcp(tree_big)
```

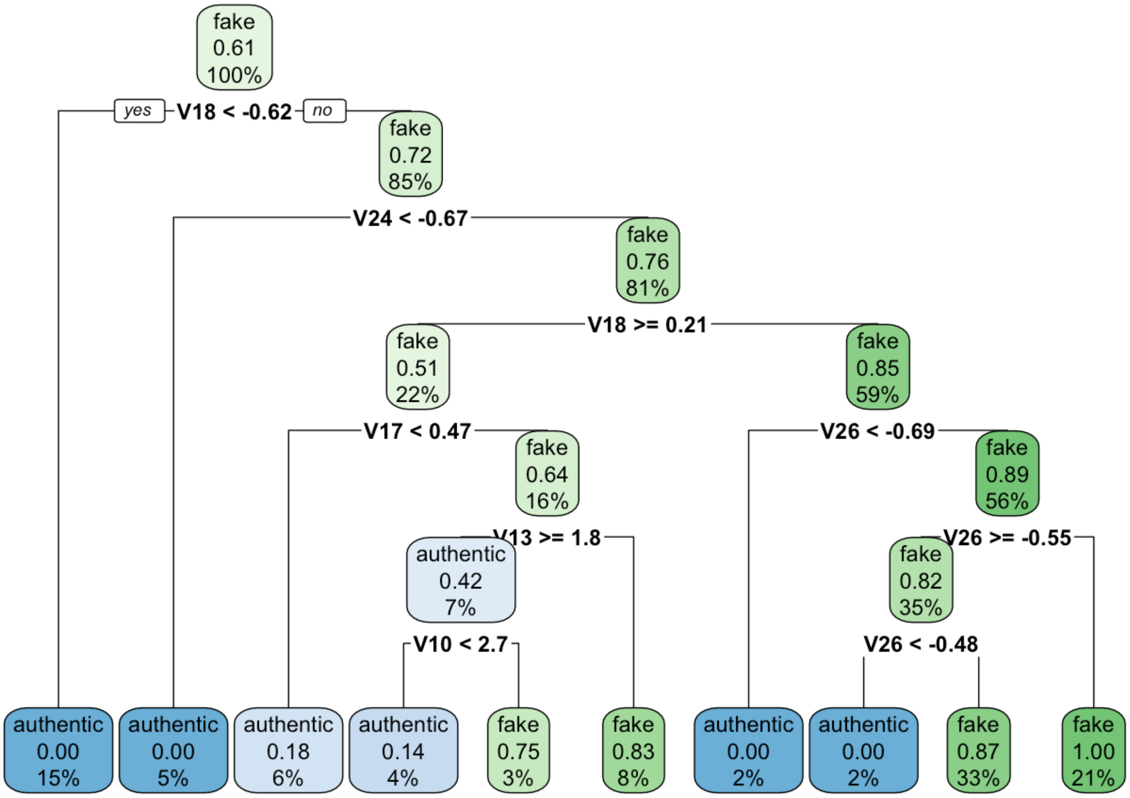


It turns out, V15 (“帅”) is the only important factor. As long as the V15 score is below -0.01 or above 0.03, we can be 100% sure that the item is authentic.

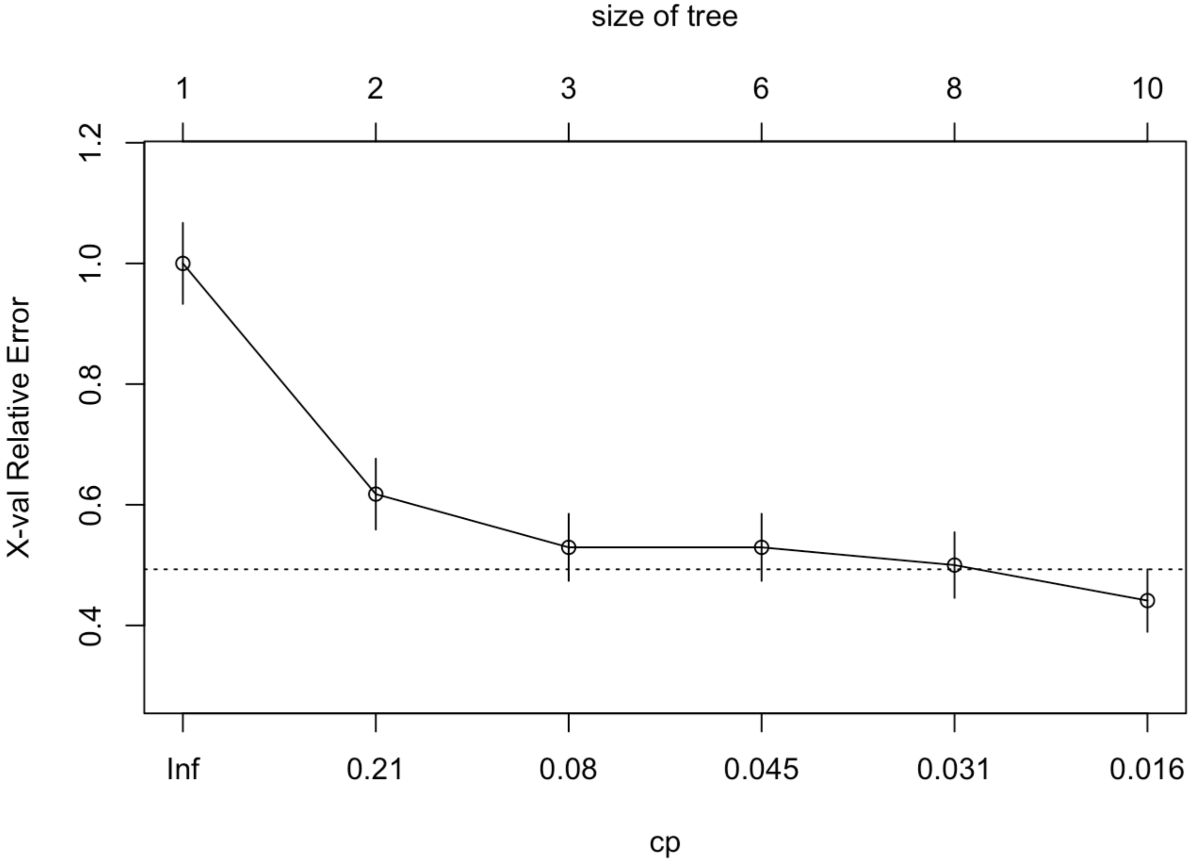
However, the reason we have this conclusion is that the way we manipulated our dataset makes it incorporate lots of factors that are so distinctive that only authentic items have nonzero values or vice versa. Therefore, the tree_big model is not informative enough and we should only keep the decision tree features as the ones that have nonzero values for both authentic and fake items.

Model 3: tree_1 and tree_1_prune

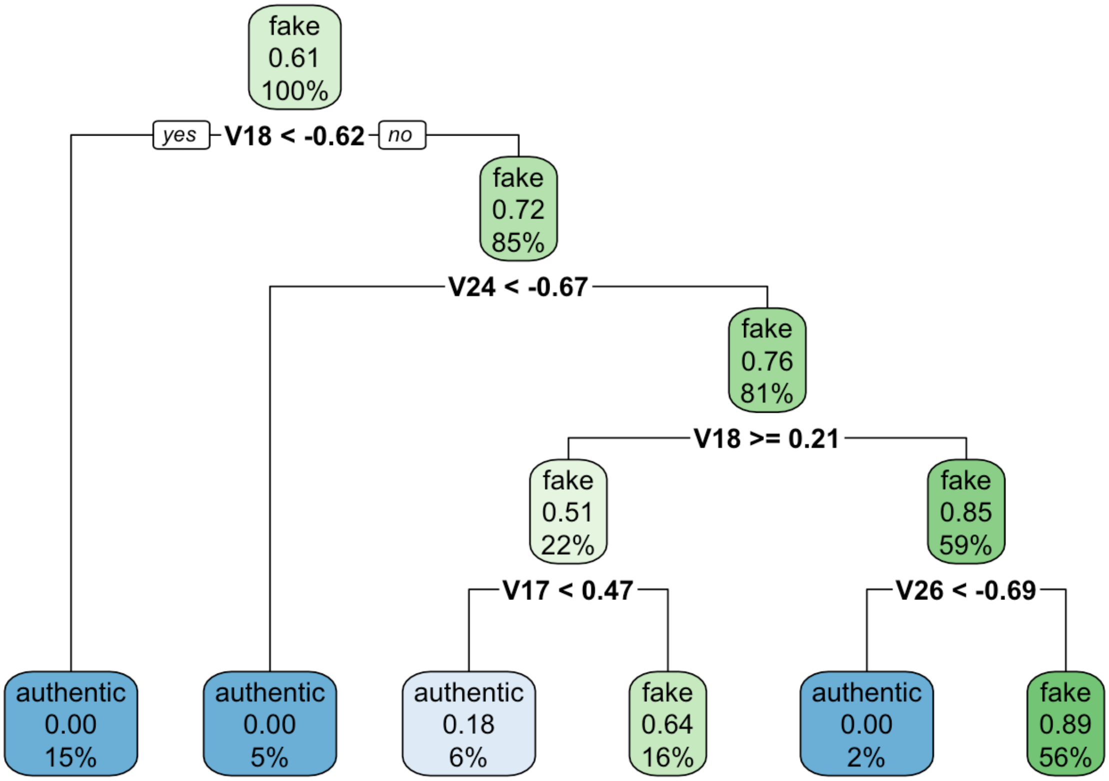
```
tree_1 <- rpart(TF ~ V1+V3+V5+V6+V7+V8+V9+V10+V13+V14+V17+V18+V20+V24+V26+V30, myData
50binary)
rpart.plot(tree_1,tweak = 1.2)
```



```
plotcp(tree_1)
```



```
tree_1_prune <- rpart(TF ~ V1+V3+V5+V6+V7+V8+V9+V10+V13+V14+V17+V18+V20+V24+V26+V30,
myData50binary, cp = 0.045)
rpart.plot(tree_1_prune,tweak = 0.9)
```



The model tree_1 incorporates all nonzero features. Roughly speaking, from the complexity plot we can see that the relative error is optimized when cp=0.08. Therefore, we will prune the tree by controlling cp. The tree_1_prune model shows that when we take V18 (“包装”), V24 (“便宜”), V17 (“卖家”) and V26 (“一天”) into consideration, our prediction accuracy achieves the highest.

Model 4: random forest with cross-validation

Setting up cross-validation groups

First, we will set up training and testing groups.

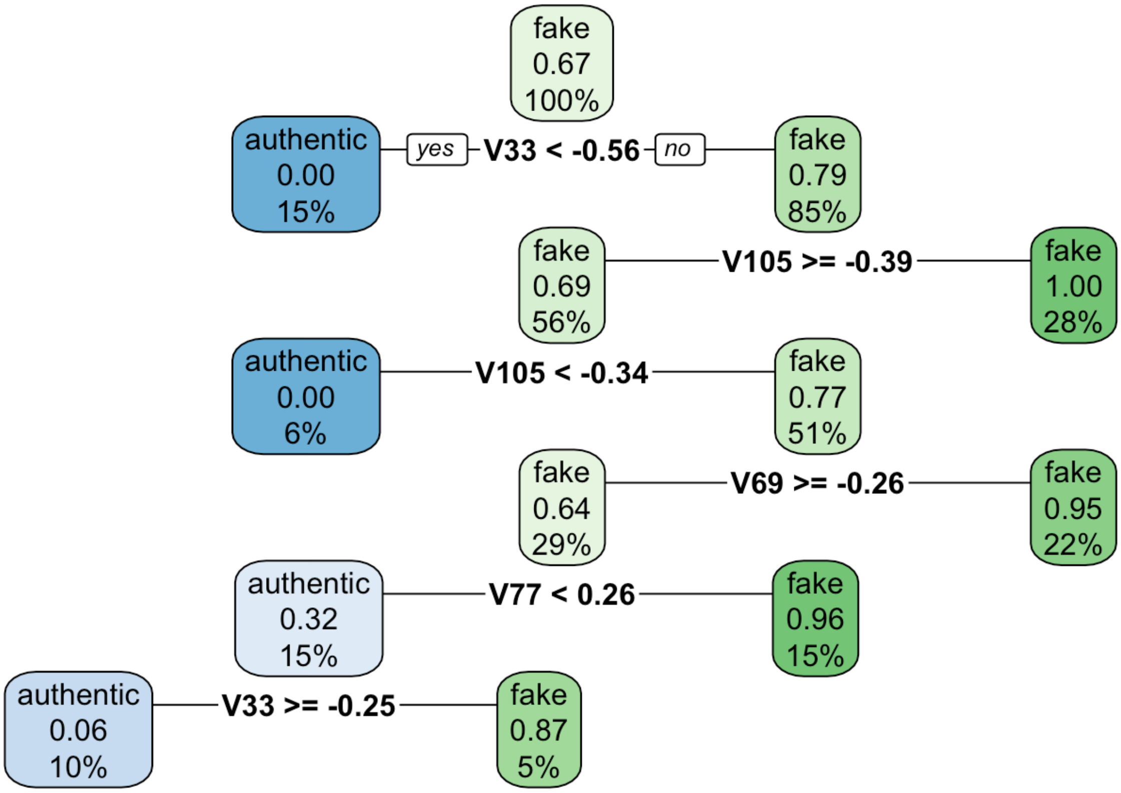
```
myData_trtest <- myDatabinary %>%
  mutate(grp = sample(c("train", "test"), size=n(),prob=c(.5,.5), replace=TRUE),
         TF = factor(TF))

train <- myData_trtest %>% filter(grp == "train") %>% select(-grp)
test <- myData_trtest %>% filter(grp == "test") %>% select(-grp)
```

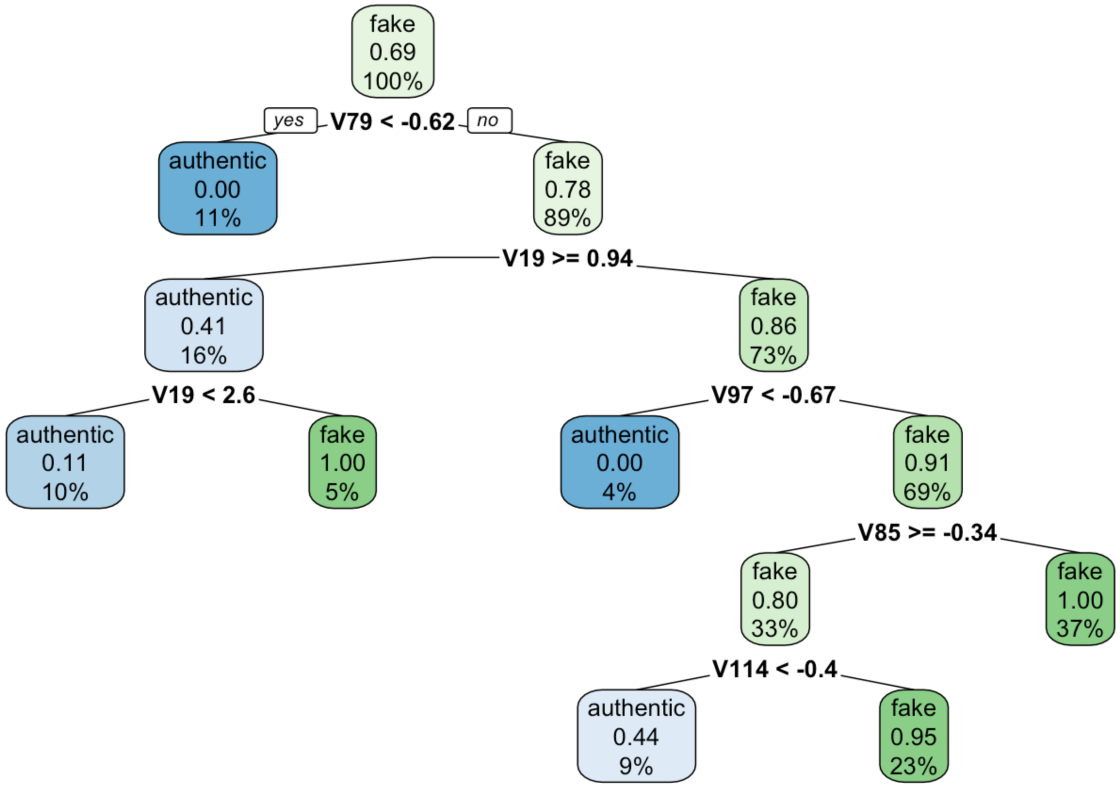
Example of how bagging works

To indicate how random forests work, I am setting three different seeds and presenting the predicted classification along with the seeds.

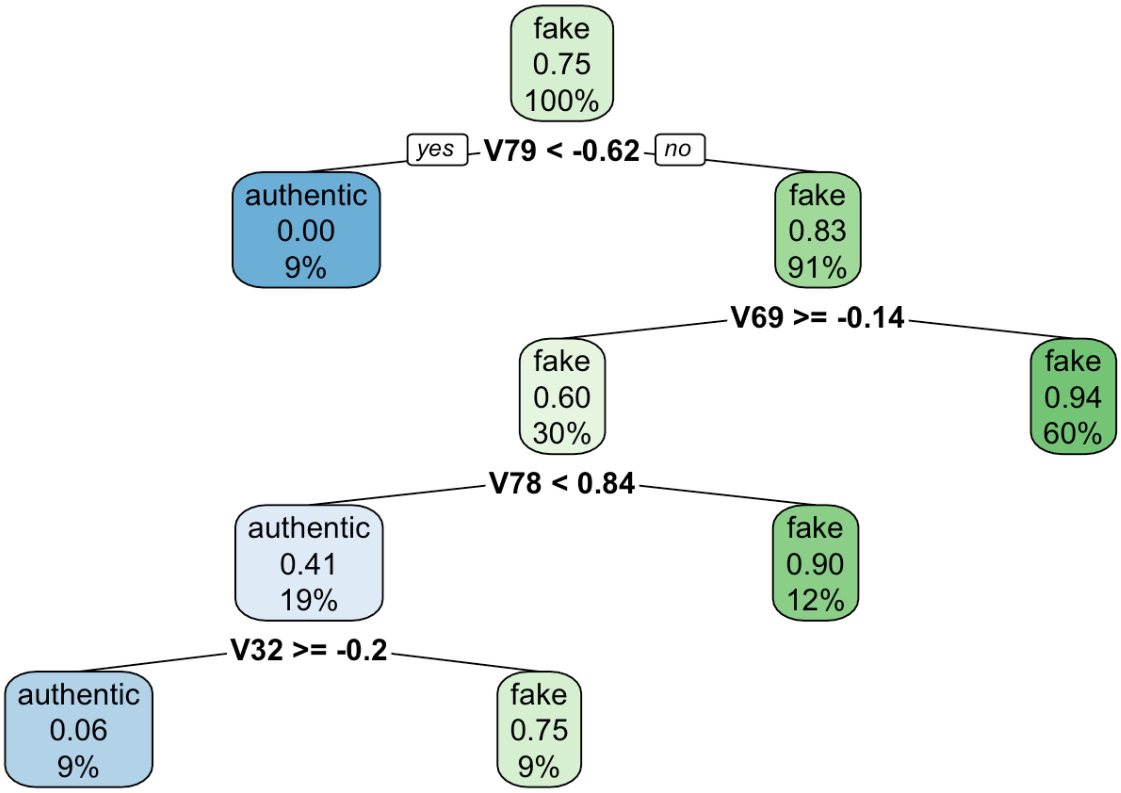
```
set.seed(1)
# Below we are selecting all the nonzero variables.
tree_boot1 <-rpart(TF ~ V1+V2+V3+V4+V5+V7+V8+V9+V10+V11+V13+V14+V16+V18+V19+V21+V24+V
26+V27+V28+V29+V30+V31+V32+V33+V34+V35+V36+V38+V40+V44+V45+V46+V47+V49+V52+V54+V55+V5
7+V58+V59+V60+V61+V63+V65+V66+V67+V68+V69+V71+V73+V74+V76+V77+V78+V79+V84+V85+V86+V87
+V92+V97+V102+V105+V108+V111+V114, data=train %>% sample_frac(size = 1, replace = TRU
E))
rpart.plot(tree_boot1, fallen.leaves = FALSE)
```



```
set.seed(2)
tree_boot2 <-rpart(TF ~ V1+V2+V3+V4+V5+V7+V8+V9+V10+V11+V13+V14+V16+V18+V19+V21+V24+V
26+V27+V28+V29+V30+V31+V32+V33+V34+V35+V36+V38+V40+V44+V45+V46+V47+V49+V52+V54+V55+V5
7+V58+V59+V60+V61+V63+V65+V66+V67+V68+V69+V71+V73+V74+V76+V77+V78+V79+V84+V85+V86+V87
+V92+V97+V102+V105+V108+V111+V114, data=train %>% sample_frac(size = 1, replace = TRU
E))
rpart.plot(tree_boot2, fallen.leaves = FALSE)
```



```
set.seed(3)
tree_boot3 <-rpart(TF ~ V1+V2+V3+V4+V5+V7+V8+V9+V10+V11+V13+V14+V16+V18+V19+V21+V24+V
26+V27+V28+V29+V30+V31+V32+V33+V34+V35+V36+V38+V40+V44+V45+V46+V47+V49+V52+V54+V55+V5
7+V58+V59+V60+V61+V63+V65+V66+V67+V68+V69+V71+V73+V74+V76+V77+V78+V79+V84+V85+V86+V87
+V92+V97+V102+V105+V108+V111+V114, data=train %>% sample_frac(size = 1, replace = TRU
E))
rpart.plot(tree_boot3, fallen.leaves = FALSE)
```



```
test %>% select(TF) %>%
  mutate(p1=predict(tree_boot1, test, type = "class"),
         p2=predict(tree_boot2, test, type = "class"),
         p3=predict(tree_boot3, test, type = "class")) %>%
  slice(1:100)
```

```
## # A tibble: 100 x 4
##    TF        p1    p2        p3
##    <fct>     <fct> <fct>     <fct>
##  1 fake      fake  authentic fake
##  2 fake      fake  fake      fake
##  3 authentic fake  authentic fake
##  4 authentic fake  fake      fake
##  5 authentic fake  authentic fake
##  6 authentic fake  fake      fake
##  7 fake      fake  fake      fake
##  8 authentic fake  authentic fake
##  9 fake      fake  fake      fake
## 10 fake      fake  fake      fake
## # ... with 90 more rows
```

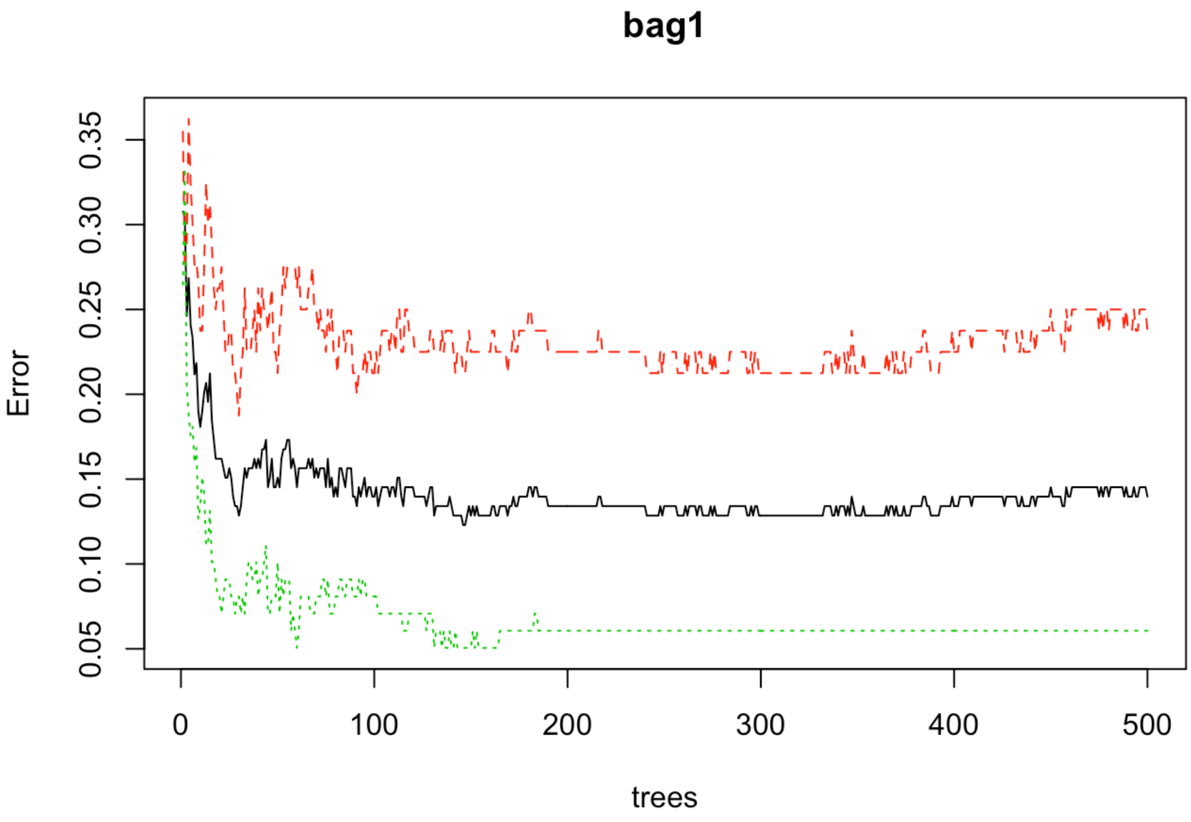
Random Forest

As shown in the first row of the table above, when p1 shows fake, p2 shows fake and p3 shows authentic, our predicted value is fake. In the second row, when all three seeds shows authentic, the predicted value will be authentic.

```
bag1 <- randomForest(TF ~ V1+V2+V3+V4+V5+V7+V8+V9+V10+V11+V13+V14+V16+V18+V19+V21+V24
+V26+V27+V28+V29+V30+V31+V32+V33+V34+V35+V36+V38+V40+V44+V45+V46+V47+V49+V52+V54+V55+
V57+V58+V59+V60+V61+V63+V65+V66+V67+V68+V69+V71+V73+V74+V76+V77+V78+V79+V84+V85+V86+V
87+V92+V97+V102+V105+V108+V111+V114, data=test, importance = TRUE)
bag1
```

```
##
## Call:
##  randomForest(formula = TF ~ V1 + V2 + V3 + V4 + V5 + V7 + V8 +      V9 + V10 + V1
1 + V13 + V14 + V16 + V18 + V19 + V21 + V24 +      V26 + V27 + V28 + V29 + V30 + V31
+ V32 + V33 + V34 + V35 +      V36 + V38 + V40 + V44 + V45 + V46 + V47 + V49 + V52 +
V54 +      V55 + V57 + V58 + V59 + V60 + V61 + V63 + V65 + V66 + V67 +      V68 + V69
+ V71 + V73 + V74 + V76 + V77 + V78 + V79 + V84 +      V85 + V86 + V87 + V92 + V97 +
V102 + V105 + V108 + V111 +      V114, data = test, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 13.97%
## Confusion matrix:
##           authentic fake class.error
## authentic        61   19  0.23750000
## fake              6   93  0.06060606
```

```
plot(bag1)
```



From the bag1 summary we could see that our overall misclassification rate is 12.64%. The false positive rate is 0.28571429 and false negative rate is 0.03603604. The plot is a visualization of the error rate with respect to the number of trees used. We used 500 trees here.

Variable Importance

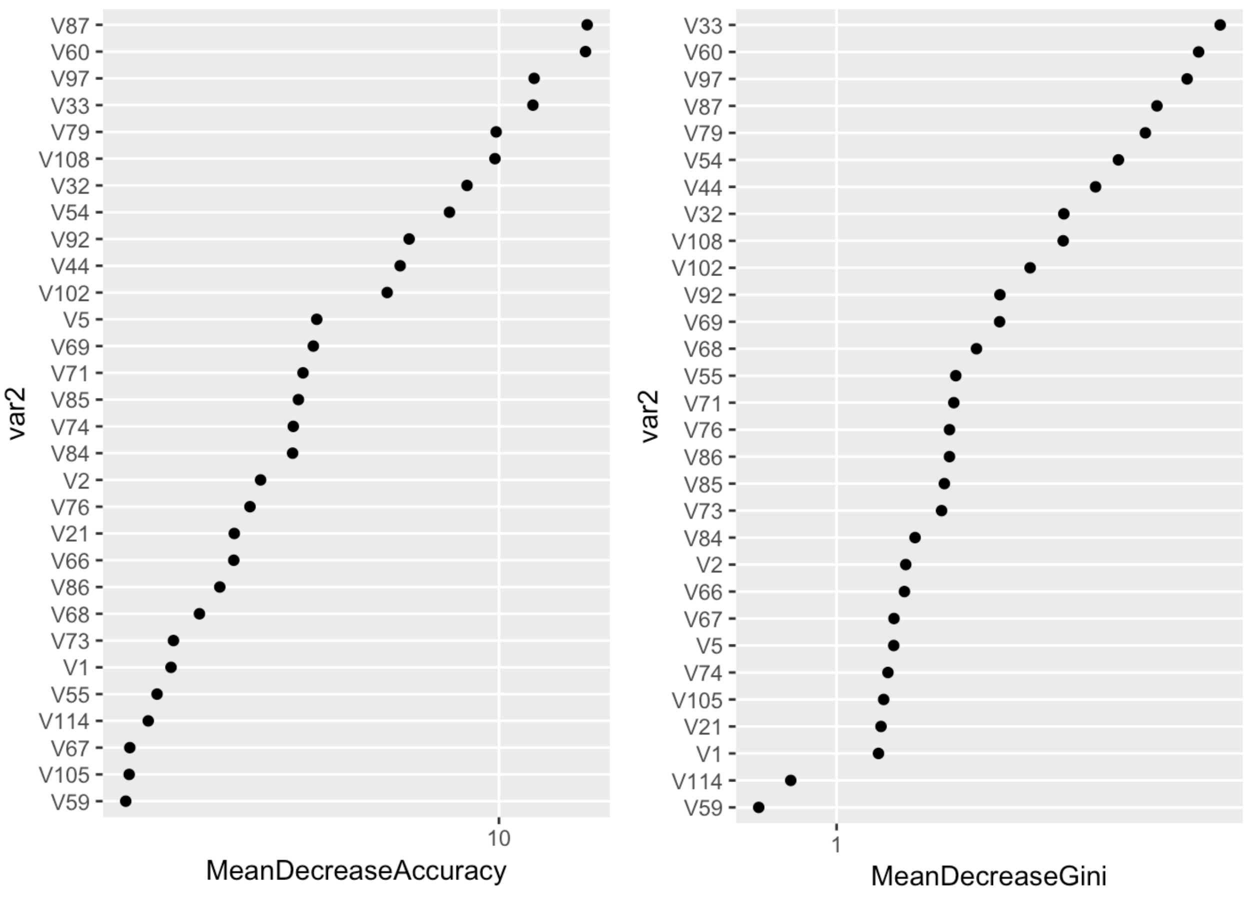
To interpret what are the most important features to make accurate classifications, we will take a look at these two factors: Mean Decrease Accuracy and Mean Decrease Gini.

```
imp_meas <-
  as.tibble(importance(bag1)) %>%
  mutate(var = row.names(importance(bag1))) %>%
  select(var, MeanDecreaseAccuracy, MeanDecreaseGini) %>%
  arrange(desc(MeanDecreaseAccuracy))

imp1 <- imp_meas %>%
  mutate(var2 = fct_reorder(var, MeanDecreaseAccuracy)) %>%
  slice(1:30) %>%
  ggplot(aes(x=MeanDecreaseAccuracy, y=var2)) +
  geom_point() +
  scale_x_log10()

imp2 <- imp_meas %>%
  mutate(var2 = fct_reorder(var, MeanDecreaseGini)) %>%
  slice(1:30) %>%
  ggplot(aes(x=MeanDecreaseGini, y=var2)) +
  geom_point() +
  scale_x_log10()

grid.arrange(imp1, imp2, nrow=1)
```



The variables on top are the most important ones, which are “两天”, “包装”, “大小”, “便宜”, “值得”, “做工”, “下次”,“满意”, “万斯” and so on. What this indicates is that customer comments concerning the item’s delivery, package, sizing, price, craft, expected next time purchase, general satisfaction and the direct translation of the brand could all be good predictors of whether an branded item is authentic or fake.

Conclusion and Future Research

Our big idea is to use word frequency counts in customer reviews as predictors to generate profiles for authentic and fake branded items sold online, assuming that their frequency distribution profiles will reflect some differences. In this case, we used VANS sneakers sold on Tianmao and Taobao as an example.

We first generated the word frequency counts through Jieba dictionary in Python. Then, we manually selected items with at least 50 comments and feature words with the most nonzero entries. We also built 4 “myData” files depending on the complexity (50+ features or 100+ features) and response variable type (binary or continuous) for later use.

In exploratory analysis, we saw that authentic VANS sneakers do have different distributions than fake ones, as indicated in the 1-dimensional and 2-dimensional density plots. Moreover, we tested lasso selection but it failed, suggesting that our variables might be highly correlated or that there is a large amount of significant indicators.

In the modeling section, we tried 3 decision trees and a random forest. The trees are shown to be a great model for this dataset and we could manipulate the complexity level with the tradeoff between interpretability and predicability. The randomforest returns a 12.64% misclassification rate, which is pretty low and indicates good reliability of the selected factors.

Last but not the least, we looked at the variable importance by the bagging Gini index. Variables such as “”, “” and “” are shown to be good indicators.

Overall, using natural language processing, or specifically word frequency distribution analysis on consumer reviews, we are able to predict with 87% confidence whether a listed VANS sneakers on Taobao.com is authentic or fake. Future research should focus on analyzing the feature correlation and sentiment analysis.