1) Practice questions on Text Analytics, Clustering in R

a) `emails ← read.csv("emails.csv", stringsAsFactors = FALSE)`

`str(emails)`
`nrow(emails)`

5728 — Total no. of emails

`table(emails$spam)`

0	1
4360	1368

Spam emails

b) `strwrap(emails[1,1])`
`strwrap(emails[1000,1])`

} Same start "Subject:"

c) Yes - since the number of times a word appears might be different in spam and ham email messages.
For example, a long email might have the word subject occur more often & this might be indicative of ham emails.

d) `max(nchar(emails$text))` — 43952

e) `which.min(nchar(emails$text))` — 1992

`strwrap(emails$text[1992])`

"Subject: fyi"

f) library(tm)

corpus ← Corpus(VectorSource(emails$text))

corpus ← tm_map(corpus, tolower)

~~corpus ← tm_map(corpus, removeText)~~

corpus ← tm_map(corpus, removePunctuation)

corpus ← tm_map(corpus, removeWords, stopwords("english"))

corpus ← tm_map(corpus, StemDocument)

dtm ← DocumentTermMatrix(corpus)

str(dtm) 28687 Terms

g) spdtm ← removeSparseTerms(dtm, 0.95)

str(spdtm) 330 terms

h) emailsSparse ← as.data.frame(as.matrix(spdtm))

colnames(emailsSparse) ← make.names(colnames(emailsSparse))

which.max(colSums(emailsSparse))

Enron

i) emailsSparse $ spam ← emails $ spam

Sort(colSums(subset(emailsSparse, spam == 0)))

hou	will	vinc	subject	ect	enron
5569	6802	8531	8625	11417	13388

These words appear atleast 5000 times in the dataset.

j) Sort(colSums(subset(emailsSparse, spam == 1)))

Compani	span	will		subject
1065	1368	1450		1577

k) The frequencies of these most common words are likely to help differentiate between spam and ham.
For example "enron" appears very often in ham as compared to spam.

l) The models we build are personalized and would need to be further tested before being used as a spam filter for another person.

m)

```
emailsSparse$spam <- as.factor(emailsSparse$spam
library(caTools)
set.seed(123)
spl <- sample.split(emailsSparse$spam, 0.7)
train <- subset(emailsSparse, spl==TRUE)
test <- subset(emailsSparse, spl==FALSE)

SpamLog <- glm(Spam~., data=train,
               family="binomial")

library(rpart)
set.seed(123)
SpamCART <- rpart(Spam~., data=train)

library(randomForest)
set.seed(123)
SpamRF <- randomForest(Spam~., data=train)

predictLog <- predict(spamLog, newdata=train,
                      type="response"
predictCART <- predict(SpamCART, newdata=train)
predictRF <- predict(SpamRF, newdata=train,
                     type="prob"
predictCART <- predictCART[,2]
predictRF <- predictRF[,2]
```

table (predictLog < 0.0001) 3046

FALSE	TRUE
964	3046

table (predictLog > 0.9999) 954

FALSE	TRUE
3056	954

table (predictLog ≥ 0.0001 & predictdog ≤ 0.9999) 10

FALSE	TRUE
4000	10

n) None of the variables are significant at the $p = 0.05$ level

Note that there was also trouble for the logistic regression model to converge in this example. Summary(spamdog)

```
library(rpart.plot)
```

o) plot(spamCART)

The words 'vine' and 'enron' appear at the top of the CART model.

The words 'thou' and 'kaminski' do not appear.

p) table (predictLog ≥ 0.5, train $spam)

	0	1
FALSE	3052	4
TRUE	0	954

Accuracy = $\frac{3052+954}{3052+954+4}$

= 0.9990025

library (ROCR)

predictLog1 ← prediction (predictLog, train $spam)

perfLog1 ← performance (predictLog1, measure = "auc

AUC on training set = 0.9999959

q) table (predictCART ≥ 0.5, train $spam)

	0	1
FALSE	2885	64
TRUE	167	894

Accuracy = 0.94

r) predictCART1 ← prediction (predictCART, train $spam)

perfCART1 ← performance (predictCART1, measure = "auc"

AUC on training set = 0.969

8) table (predict RF >= 0.5, train $spam)

	0	1
FALSE	3046	0
TRAIN	6	958

Accuracy = 0.998503

t) predictRF1 <- prediction (predictRF, train$spam)

perfRF1 <- performance (predictRF1) measure = "auc

AUC on training set = 0.9999959

u) In this model, logistic regression and random forest have the best performance.

v) predictLogtest <- predict (spamLog, newdata = test
type = "response")

predictCARTtest <- predict (spamCART,
newdata = test)

predictRFtest <- predict (spamRF, newdata = test
type = "prob")

predictCARTtest <- predictCARTtest[,2]

predictRFtest <- predictRFtest[,2]

table (predictLog test >= 0.5, test $ span)

	0	1
FALSE	1257	34
TRUE	51	376

Accuracy = $\frac{1257 + 376}{1257 + 376 + 34 + 51}$

= 0.9505239

w) predictLog2 ← prediction (predictLogtest, test$span)

perfLog2 ← performance (predictlog2, measure="auc")

Test set AUC = 0.9627517

table (predict CART test >= 0.5, test $ span)

	0	1
FALSE	1228	24
TRUE	80	386

Accuracy = 0.93

predict CART 2 ← Prediction (predictCART test, test $ span)

perf CART 2 ← performance (predictCART2, measure = "auc")

Test set AUC = 0.96

table (predict RF test >= 0.5, test $ span)

	0	1
FALSE	1290	29
TRUE	18	388

Accuracy = 0.97

predict RF2 ← prediction (predict RFtest, test $ spam)

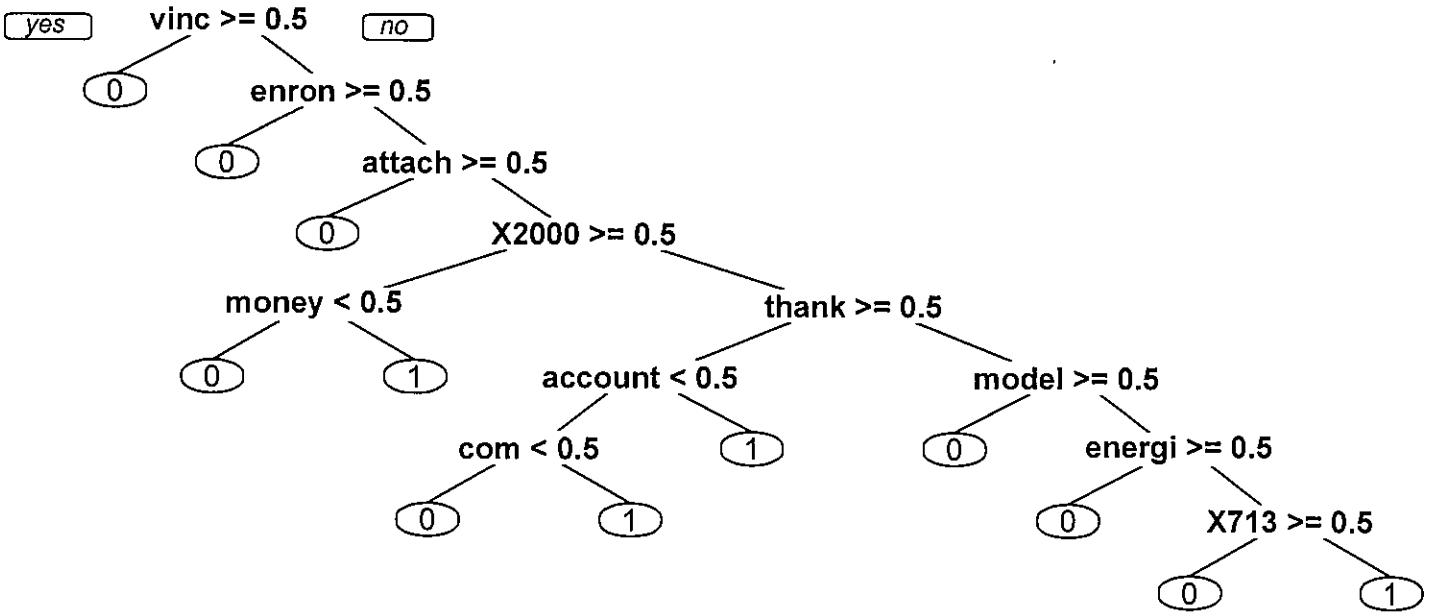
perfRF2 ← performance (predict RF2, measure = "auc");

Test set AUC = 0.99 [illegible] 6

X) The random forest has the most impressive performance in the test set both in terms of accuracy and AUC.

Y) Logistic regression had an almost perfect fit on the training set but not as good performance on the test set.

On the other hand CART and random forest models have similar accuracies in the training and test sets.


yes
vinc >= 0.5
no
0
enron >= 0.5
0
attach >= 0.5
0
X2000 >= 0.5
money < 0.5
thank >= 0.5
0
1
account < 0.5
model >= 0.5
com < 0.5
1
0
energi >= 0.5
0
1
0
X713 >= 0.5
0
1

2)

a) Stocks ← read.csv("Stocks Cluster.csv")

Str(stocks)

A total of 11580 observations

b) table(stocks $Positive Dec)/nrow(stocks)

0	1
0.453886	0.546114

54.6% of the observations have positive returns in December.

c) Cor(stocks[,1:11])

Sort(cor(stocks[,1:11]))

Dropping the variances of 1, the maximum correlation is 0.191672. between October and November.

d) ColMeans(stocks)

Largest average return in April

Smallest average return in September.

e)

```
Set.Seed(144)
Spl <- sample.split(stocks$PositiveDec, SplitRatio
                                        = 0.7
StocksTrain <- subset(stocks, Spl == TRUE)
StocksTest <- subset(stocks, Spl == FALSE)
StocksModel <- glm(PositiveDec ~ .,
                   data = StocksTrain, family
                   = "binomial")
StocksPredict <- predict(StocksModel,
                   newdata = StocksTrain, type =
                   "response")
table(StocksPredict >= 0.5, StocksTrain$PositiveDec)
```

	0	1
FALSE	990	787
POSITIVE	2689	3640

$$\text{Accuracy} = \frac{990 + 3640}{990 + 3640 + 2689 + 787} = 0.571$$

f)

StocksPredict test ← predict (stocksModel, newdata = stocksTest, type = "response")

table (stocks Predict test ≥ 0.5, stocksTest $ Positive Dec)

	0	1
FALSE	417	344
TRUE	1160	1553

Accuracy = 0.567

g) table (stocksTrain $Positive Dec)

0	1
3679	4427

Most common outcome is 1 in the training set

table (stocks Test $ Positive Dec)

0	1
1577	1897

Accuracy = 0.5460

h)

limitedTrain ← stocksTrain
limited Train $ Positive Dec ← NULL
limitedTest ← stocksTest
limited Test $ Positive Dec ← NULL

We remove the dependent variable in the clustering since needing to know the dependent variable to assign an Observation to a cluster defeats the purpose of the methodology.

i) library (caret)

preproc ← preProcess (limitedTrain)

normTrain ← predict (preproc, limitedTrain)

normTest ← predict (preproc, limitedTest)

colMeans (normTrain)

It is easy to see that mean of ReturnJan variable is 0 (as normalized)

colMeans (normTest)

-0.0004185

j) The mean of ReturnJan is much closer to 0 in normTrain than in normTest

Since the distribution of ReturnJan is different in the training and testing set.

k) set.seed (144)

km ← kmeans (normTrain, centers = 3)

table (km$cluster)

1	2	3
3157	4696	253

Cluster 2 has largest number of observations

install.packages("flexclust")

l) library(flexclust)

km.kcca ← as.kcca(km, normTrain)

clusterTrain ← predict(km.kcca)

clusterTest ← predict(km.kcca, newdata = normTest)

table(clusterTest)

1	2	3
1298	2080	96

Cluster 2 has 2080 observations in it.

m) stocksTrain1 ← subset(stocksTrain, clusterTrain == 1)

stocksTrain2 ← subset(stocksTrain, clusterTrain == 2)

stocksTrain3 ← subset(stocksTrain, clusterTrain == 3)

stocksTest1 ← subset(stocksTest, clusterTest == 1)

stocksTest2 ← subset(stocksTest, clusterTest == 2)

stocksTest3 ← subset(stocksTest, clusterTest == 3)

mean(stocksTrain1$PositiveDec) 0.6024

mean(stocksTrain2$PositiveDec) 0.5140

mean(stocksTrain3$PositiveDec) 0.4387

stocksTrain1 has the observations with highest average value of dependent variable.

n)

```
StocksModel1 <- glm(PositiveDec ~ . , data =
                StocksTrain1, family = "binomial")

StocksModel2 <- glm(PositiveDec ~ . , data =
                StocksTrain2, family = "binomial")

StocksModel3 <- glm(PositiveDec ~ . , data =
                StocksTrain3, family = "binomial")
```

Variables that have atleast a positive sign and a negative sign in one of the three models are:

ReturnJan, ReturnFeb, ReturnMar, ReturnJune, ReturnAug, ReturnOct,

o)

```
PredictTest1 <- predict(stocksModel1, newdata =
                StocksTest1, type = "response")

PredictTest2 <- predict(stocksModel2, newdata =
                StocksTest2, type = "response")

PredictTest3 <- predict(stocksModel3, newdata =
                StocksTest3, type = "response")
```

```
table(PredictTest1 > 0.5, StocksTest1$PositiveDec)
```

	0	1
FALSE	30	23
TRUE	471	774

Accuracy = 0.6191

table (Predict Test2 >= 0.5, Stocks Test 2 $ Positive Dec)

	0	1
FALSE	388	309
TRUE	626	757

Accuracy = 0.55

table (PredictTest 3 >= 0.5, Stocks Test 3 $ Positive Dec)

	0	1
FALSE	249	21
TRUE	13	13

Accuracy = 0.63

P) All Predictions ← C(PredictTest1, Predict test 2, PredictTest 3)

All Outcomes ← C (Stocks Test 1 $ Positive Dec, Stocks Test 2 $ Positive Dec, Stocks Test 3 $ Positive Dec)

table (All Predictions >= 0.5, All_Outcomes)

	0	1
FALSE	467	353
TRUE	1700	1594

Accuracy = 0.578

3)

a) citi ← read.csv ("citibike.csv")

unique (citi $ start duration)

nlevels (unique $ start duration)

unique (citi $ end duration)

nlevels (unique $ end duration)

There are a total of 329 bike stations in the dataset

b) tapply (citi $ duration, citi $ day, mean)

From the result, we see that the average trip duration is maximum on Saturdays with 894.26 se

c) max (table (citi $ starttime))

which.max (table (citi $ starttime))

Max of 65684 trips start at 18 (6 pm).

min (table (citi $ starttime))

which.min (table (citi $ starttime))

Min of 1151 trips start at 4 (4 am)

d) table(citi$gender)

1	2
510826	156912

Proportion of bikes used by female users $= \frac{156912}{510826 + 156912}$

$= 0.234$

$= 23.4\%$

e) citi$Mon <- as.integer(citi$day == "Mon")

f) Summary(citi)

The trip duration number in seconds is very high & would be expected to dominate in distance calculations.

g)
```
citi1 <- citi
citi1$tripduration <- scale(citi1$tripduration)
citi1$age <- scale(citi1$age)
citi1$gender <- scale(citi1$gender)
citi1$starttime <- scale(citi1$starttime)
citi1$Mon <- scale(citi1$Mon)
:
citi1$Sun <- scale(citi1$Sun)
```

Max (citi1 $ tripduration) 402.9s

h) The data has 667738 observations. These are a lot of bikerides & it will be hard for hierarchical clustering to handle computing the distances between all these observations (C).

i) set.seed(1)

Cluster1 <- kmeans(citi1[,c("tripduration", "gender", "age", "starthour", "Mon", ... "Sun")], centers = 10)

min (cluster1 $ size) 18148

Max (cluster1 $ size) 107185

j) cluster1 $ centers

Cluster 5 seems to be a good representation of older riders driving on Saturday

k) Cluster 10 is a reasonable cluster that describes this

l) Different results

m) Identical results

n) citi1$weekday <- 1 - as.integer(citi1$Sat==1 | citi1$Sun==1)

(Use citi here)

citi1$weekday <- scale(citi1$weekday)

Set.seed(100)

cluster2 <- kmeans(citi1[,c("tripduration", "gender", "age", "starthrs", "weekday")], centers=10)

cluster2$centers

Cluster 1 is a good cluster to describe longer trips taken by older female users on weekdays.

o) Cluster 9 is a good description of short trips taken by younger male users early on weekdays.