PeerReview_8

Minxin Cheng

Problem 1

1. Read in data file

```
teens <- read.csv("snsdata.csv")
```

2. Check and clean the data set

2.1 Check age column

```
# check the structure
str(teens)
```

```
## 'data.frame':    30000 obs. of  40 variables:
##  $ gradyear    : int  2006 2006 2006 2006 2006 2006 2006 2006 2006 2006 ...
##  $ gender      : Factor w/ 2 levels "F","M": 2 1 2 1 NA 1 1 2 1 1 ...
##  $ age         : num  19 18.8 18.3 18.9 19 ...
##  $ friends     : int  7 0 69 0 10 142 72 17 52 39 ...
##  $ basketball  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ football    : int  0 1 1 0 0 0 0 0 0 0 ...
##  $ soccer      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ softball    : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ volleyball  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ swimming    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ cheerleading: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ baseball    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ tennis      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ sports      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ cute        : int  0 1 0 1 0 0 0 0 0 1 ...
##  $ sex         : int  0 0 0 0 1 1 0 2 0 0 ...
##  $ sexy        : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ hot         : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ kissed      : int  0 0 0 0 5 0 0 0 0 0 ...
##  $ dance       : int  1 0 0 0 1 0 0 0 0 0 ...
##  $ band        : int  0 0 2 0 1 0 1 0 0 0 ...
##  $ marching    : int  0 0 0 0 0 1 1 0 0 0 ...
##  $ music       : int  0 2 1 0 3 2 0 1 0 1 ...
##  $ rock        : int  0 2 0 1 0 0 0 1 0 1 ...
##  $ god         : int  0 1 0 0 1 0 0 0 0 6 ...
##  $ church      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ jesus       : int  0 0 0 0 0 0 0 0 0 2 ...
##  $ bible       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hair        : int  0 6 0 0 1 0 0 0 0 1 ...
##  $ dress       : int  0 4 0 0 0 1 0 0 0 0 ...
```

```
##  $ blonde      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ mall        : int  0 1 0 0 0 0 2 0 0 0 ...
##  $ shopping    : int  0 0 0 0 2 1 0 0 0 1 ...
##  $ clothes     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hollister   : int  0 0 0 0 0 0 2 0 0 0 ...
##  $ abercrombie : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ die         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ death       : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ drunk       : int  0 0 0 0 1 1 0 0 0 0 ...
##  $ drugs       : int  0 0 0 0 1 0 0 0 0 0 ...
```

```
# summary age column
summary(teens$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   3.086  16.312  17.287  17.994  18.259 106.927    5086
```

It doesn't make sense that age from 3 to age 106 are all in high school, therefore, only keep the age range 13 to 20, which is a reasonable range for high school student, all the others are set as missing data.

2.2 Clean age column

```
teens$age <- ifelse(teens$age >= 13 & teens$age < 20, teens$age, NA)
summary(teens$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   13.03   16.30   17.27   17.25   18.22   20.00    5523
```

Now the there is even more missing data.

A easy solution for missing values is to exclude any case with a missing value, however, that is not always a good way to deal with the data, by which will lose a large portions of the data. The subset left from the excluding will make the dataset very small, or even worse that is systematically different or non-representative of the full population.

As age is a numeric variable, we will use a different strategy known as imputation, it fills in the missing data with a guess as to the true value. First can try the mean value.

```
mean(teens$age)
```

```
## [1] NA
```

However, it returns NA. It's due to that the mean is undefined for a vector containing missing data, can further try to remove the missing values.

```
mean(teens$age, na.rm = TRUE)
```

```
## [1] 17.25243
```

It is indicating that the average student age in this dataset is 17 years old, but we need the average age for each graduation year.

```r
# calculate mean age for each graduation year
aggregate(data = teens, age ~ gradyear, mean, na.rm = TRUE)
```

```
##   gradyear      age
## 1     2006 18.65586
## 2     2007 17.70617
## 3     2008 16.76770
## 4     2009 15.81957
```

```r
# same function, but returns a result with a equal length to the original data
ave_age <- ave(teens$age, teens$gradyear, FUN = function(x) mean(x, na.rm = TRUE))
```

Impute these means onto the missing values

```r
teens$age <- ifelse(is.na(teens$age), ave_age, teens$age)
summary(teens$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13.03   16.28   17.24   17.24   18.21   20.00
```

2.3 Check and clean gender column

```r
# check the missing values in gender column
table(teens$gender, useNA = "ifany")
```

```
##
##     F     M  <NA>
## 22054  5222  2724
```

For gender (as a categorical variable), an alternative way is to create dummy coding for unknown gender

```r
# create dummy codes
teens$female <- ifelse(teens$gender == "F" & !is.na(teens$gender), 1, 0)
teens$no_gender <- ifelse(is.na(teens$gender), 1, 0)
# compare the dommy codes with oritinal data
table(teens$gender, useNA = "ifany")
```

```
##
##     F     M  <NA>
## 22054  5222  2724
```

```r
table(teens$female, useNA = "ifany")
```

```
##
##     0     1
##  7946 22054
```

```
table(teens$no_gender, useNA = "ifany")
```

```
##
##     0     1
## 27276  2724
```

3. Train a model

kmeans() function requires a data frame containing only numeric data and a parameter specifying the desired number of clusters.

3.1 Prepare the dataset

```
# only consider 36 features
interests <- teens[5:40]
# perform z-score transformation to the dataset
interests_z <- as.data.frame(lapply(interests, scale))
```

3.2 Train the model

```
set.seed(2345)
teen_clusters <- kmeans(interests_z, 5)
```

Obtain the size of kmeans() clusters

```
teen_clusters$size
```

```
## [1]  1038   601  4066  2696 21599
```

We can see here that the smallest cluster has 600 teenagers and the largest cluster has 21,514, indicating a large gap between the number of people in the largest and smallest clusters. We don't know if this is due to a real problem, or it is caused by the initial k-means cluster centers. We then examine the coordinates of the cluster centroids.

```
teen_clusters$centers
```

```
##     basketball    football      soccer   softball  volleyball    swimming
## 1  0.362160730  0.37985213  0.13734997  0.1272107  0.09247518  0.26180286
## 2 -0.094426312  0.06691768 -0.09956009 -0.0379725 -0.07286202  0.04578401
## 3  0.003980104  0.09524062  0.05342109 -0.0496864 -0.01459648  0.32944934
## 4  1.372334818  1.19570343  0.55621097  1.1304527  1.07177211  0.08513210
## 5 -0.186822093 -0.18729427 -0.08331351 -0.1368072 -0.13344819 -0.08650052
##   cheerleading    baseball      tennis      sports         cute
## 1    0.2159945  0.25312305  0.11991682  0.77040675  0.475265034
## 2   -0.1070370 -0.11182941  0.04027335 -0.10638613 -0.027044898
## 3    0.5142451 -0.04933628  0.06703386 -0.05435093  0.796948359
## 4    0.0400367  1.09279737  0.13887184  1.08316097 -0.005291962
```

```
## 5   -0.1092056 -0.13616893 -0.03683671 -0.15903307 -0.171452198
##            sex         sexy          hot      kissed        dance
## 1  2.043945661  0.547956598  0.314845390  3.02610259  0.455501275
## 2 -0.042725567 -0.027913348 -0.035027022 -0.04581067  0.050772118
## 3 -0.003156716  0.266741598  0.623263396 -0.01284964  0.650572336
## 4 -0.033193640  0.003036966  0.009046774 -0.08755418 -0.001993853
## 5 -0.092301138 -0.076149916 -0.132614350 -0.13080557 -0.145524147
##          band   marching       music       rock         god     church
## 1  0.39009330 -0.0105463  1.21014015  1.2014998  0.41743650  0.1621804
## 2  4.09723438  5.2196105  0.51624366  0.1865286  0.09706027  0.0675347
## 3 -0.03301257 -0.1131486  0.24527495  0.1166274  0.32867738  0.5195729
## 4 -0.07317758 -0.1039509  0.07102323  0.1565155  0.04902918  0.1320602
## 5 -0.11740538 -0.1104553 -0.12755935 -0.1044230 -0.09075500 -0.1239664
##         jesus       bible        hair      dress      blonde         mall
## 1  0.12698409  0.07464400  2.59053048  0.5312082  0.36322464  0.622896285
## 2  0.05333966  0.05836708 -0.05146837  0.0492724 -0.01238629 -0.087713363
## 3  0.26142784  0.23946855  0.35590025  0.5837827  0.03301526  0.808620531
## 4  0.01776986  0.01719220  0.01714820 -0.0653358  0.03690938 -0.004723697
## 5 -0.05901846 -0.05243708 -0.19220150 -0.1286412 -0.02793327 -0.179127117
##      shopping      clothes   hollister abercrombie          die
## 1  0.27607550  1.245121599  0.31525537   0.4131560  1.712160983
## 2 -0.03710273 -0.004395251 -0.16788599  -0.1413652  0.008941101
## 3  1.07073115  0.616207360  0.85951603   0.7935060  0.062399295
## 4  0.03497875  0.016201064 -0.08381546  -0.0861708 -0.067312427
## 5 -0.21816580 -0.177738408 -0.16182051  -0.1545430 -0.085876102
##         death       drunk       drugs
## 1  0.94713629  1.83371069  2.73878856
## 2  0.05464759 -0.08699556 -0.06414588
## 3  0.12642222  0.03594162 -0.05888141
## 4 -0.01611162 -0.06891763 -0.08795059
## 5 -0.06882571 -0.08386703 -0.10777278
```

The rows of the output refer to the five clusters, while the numbers across each row indicate the cluster's average value for the interest listed at the top of the column. From the table above, we can notice some patterns such as for cluster 4, almost all the sports-related values are positive, cluster 1 has obvious higher values in kissed, music, rock, clothes, and die. These are all making up assumptions about that clusters. Meanwhile, cluster 5 has almost all negative values across all variables, might indicate they created profiles on the website but never posted any interests.

4. Improve model performance

```
# add cluster information back to the original dataset
teens$cluster <- teen_clusters$cluster
```

Then examine how the cluster assignment relates to individual characteristics.

```
# take first five cases as an example, check the relationship between cluster
# and gender, age, and friends
teens[1:5, c("cluster", "gender", "age", "friends")]
```

```
##   cluster gender    age friends
```

```
## 1       5      M 18.982       7
## 2       3      F 18.801       0
## 3       5      M 18.335      69
## 4       5      F 18.875       0
## 5       1   <NA> 18.995      10
```

```
# check average age in each cluster
aggregate(data = teens, age ~ cluster, mean)
```

```
##   cluster      age
## 1       1 17.09319
## 2       2 17.38488
## 3       3 17.03773
## 4       4 17.03759
## 5       5 17.30265
```

```
# check gender proportion in each cluster
aggregate(data = teens, female ~ cluster, mean)
```

```
##   cluster    female
## 1       1 0.8025048
## 2       2 0.7237937
## 3       3 0.8866208
## 4       4 0.6984421
## 5       5 0.7082735
```

```
# check number of friends in each cluster
aggregate(data = teens, friends ~ cluster, mean)
```

```
##   cluster  friends
## 1       1 30.66570
## 2       2 32.79368
## 3       3 38.54575
## 4       4 35.91728
## 5       5 27.79221
```

From the summary table, cluster 3 and cluster 1 has the most female students, refer back to the cluster centeroid table above, cluster 3 has relatively low value in all sports-related values and has relatively high values on hair, dress, dance, hot, and clothes-related variables, which makes sense if we think in a stereotypical way. Also as we mentioned before, cluster 4 has higher values in all sports-related variables and meanwhile has the lowest female proportion. The associations among group membership, gender, and number of friendes suggests that the clusters can be useful predictors.

Problem 2

Question 1

What are some of the key differences between SVM and Random Forest for classification? When is each algorithm appropriate and preferable? Provide examples.

Support Vector Machine model creates a feature space, which is a finite-dimensional vector space, each dimension of which represents a "feature" of a particular object. Then when an unseen case comes in, SVM assigns this new case into a particular dimension/category. Meanwhile, SVMs are not restricted to being linear classifier, they also employ the "kernel" technique that makes the model more flexible by introducing various types of non-linear decision boundaries.

Random forest is a tree-based method that employs ensembling. It consists of a number of decision trees each trained on a random bootstrapped sample of both the observations and the features. In this way, each tree is approximately decorrelated. Then at inference time, a consensus vote is taken for classification or a mean for regression.

Random Forest is intrinsically suited for multiclass problems, while SVM is intrinsically two-class. Random Forest works well with a mixture of numerical and categorical features, even when they are on various scales but they have a tendency to overfit. It works faster and easier to be interpreted. SVM maximizes the "margin" and thus relies on the concept of "distance" between different points, which also means essential scaling is needed at the preprocessing steps. For a classification problem, Random Forest returns the probability of belonging to a class. SVM returns the distance to the boundary, which is the closest distance to the boundary, you still need to convert it to probability. In general, SVM perform better on linear dependencies.

Question 2

Why might it be preferable to include fewer predictors over many?

1) It might cause redundancy and irrelevance There is a higher chance that there are hidden relationships between some of the variables, which leads to redundancy unless you identify redundant and non-redundant variables in the early phase. It is also likely that not all predictor variables are having a considerable impact on the dependent variables. You should make sure that the set of predictor variables you select does not have any irrelevant ones or give them lower significance.
2) It might end up with overfitting The data models with a large number of predictors (also referred to complex models) often encounter overfitting problem, in which case the data model performs great on training data, but performs poorly on test data.
3) It affects productivity Training error decreases as the complexity increases but the error of test data starts increasing after some point because the model does not generalize to different data sets anymore. We want models to apply not just to the exact set but to the general population from which the training data came.
4) It reduces understandability Models with fewer predictors are more interpretable.

Question 3

You are asked to provide R-Squared for a kNN regression model. How would you respond to that request?

R-squared is for data with a numerical range, but classification by kNN assigns class labels that are just labels/factors. kNN are simple classifiers that are used when data shows local structure but not global structure. The key parameter to tune is 'k'. It is used in a supervised setting, typical quality assessment consists in splitting up the data in training and test sets (n-fold cross-validation) and providing precision, recall, and F-score etc. to evaluate its performance as a classifier.

Question 4

How can you determine which features to include when building a multiple regression model?

Some common methodes used include:

1) Stepwise regression: in which we start fitting the model with each individual predictor and see which one has the lowest p-value. Then pick that variable and then fit the model using two variable one which we already selected in the previous step and taking one by one all remaining ones. Then select the one which has the lowest p-value again while maintain the significance of the impact of selected variable. Repeat these steps until we get a combination whose p-value is less than the threshold of 0.05.

2) Forward selection: similar to stepwise regression but in which we keep adding the features. We do not delete the already added feature. in every iteration, we add only those features which increase the overall model fit.

3) Backward slimination: we first include all predictors and in subsequent steps, keep on removing the one which has the highest p-value. After