Practice Problems 8

Jordan Lian

4/04/2021

```
library(tidyverse)
```

Problem 1 (60 Points)

Build an R Notebook of the social networking service example in the textbook on pages 296 to 310. Show each step and add appropriate documentation.

Step 1 - collecting data

For this analysis, I used a dataset representing a random sample of 30,000 U.S. high school students who had profiles on a well-known SNS in 2006. When the data was collected, the SNS was a popular web destination for US teenagers. Therefore, it is reasonable to assume that the profiles represent a fairly wide cross section of American adolescents in 2006.

Step 2 - exploring and preparing the data

```
# Load dataset, and explore
teens <- read.csv("snsdata.csv")
table(teens$gender)
```

```
##
##     F     M
## 22054  5222
```

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

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

```
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
```

```
# Change data to work with ages 13-20 only
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
```

1. Data preparation - dummy coding missing values

```
# Dummy code gender values
teens$female <- ifelse(teens$gender == "F" & !is.na(teens$gender), 1, 0)
teens$no_gender <- ifelse(is.na(teens$gender), 1, 0)

# Get tables
table(teens$gender, useNA = "ifany")
```

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

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

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

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

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

2. Data preparation - imputing the missing values

```
# Get the mean age, and get a table
mean(teens$age, na.rm = TRUE)
```

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

```
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
```

```
# Impute values with ifelse statement
ave_age <- ave(teens$age, teens$gradyear, FUN = function(x) mean(x, na.rm = TRUE))
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
```

Step 3 - training a model on the data

```
# Scale the data
interests <- teens[5:40]
interests_z <- as.data.frame(lapply(interests, scale))

# Get random sample
set.seed(2345)
teen_clusters <- kmeans(interests_z, 5)
```

Step 4 - evaluating model performance

```
# Get size and centers
teen_clusters$size
```

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

```
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          sex
## 1    0.2159945  0.25312305  0.11991682  0.77040675  0.475265034  2.043945661
## 2   -0.1070370 -0.11182941  0.04027335 -0.10638613 -0.027044898 -0.042725567
## 3    0.5142451 -0.04933628  0.06703386 -0.05435093  0.796948359 -0.003156716
## 4    0.0400367  1.09279737  0.13887184  1.08316097 -0.005291962 -0.033193640
## 5   -0.1092056 -0.13616893 -0.03683671 -0.15903307 -0.171452198 -0.092301138
##           sexy          hot      kissed        dance        band   marching
## 1  0.547956598  0.314845390  3.02610259  0.455501275  0.39009330 -0.0105463
## 2 -0.027913348 -0.035027022 -0.04581067  0.050772118  4.09723438  5.2196105
## 3  0.266741598  0.623263396 -0.01284964  0.650572336 -0.03301257 -0.1131486
## 4  0.003036966  0.009046774 -0.08755418 -0.001993853 -0.07317758 -0.1039509
## 5 -0.076149916 -0.132614350 -0.13080557 -0.145524147 -0.11740538 -0.1104553
##         music       rock         god     church       jesus       bible
## 1  1.21014015  1.2014998  0.41743650  0.1621804  0.12698409  0.07464400
## 2  0.51624366  0.1865286  0.09706027  0.0675347  0.05333966  0.05836708
```

```
## 3  0.24527495  0.1166274  0.32867738  0.5195729  0.26142784  0.23946855
## 4  0.07102323  0.1565155  0.04902918  0.1320602  0.01776986  0.01719220
## 5 -0.12755935 -0.1044230 -0.09075500 -0.1239664 -0.05901846 -0.05243708
##          hair      dress      blonde         mall    shopping      clothes
## 1  2.59053048  0.5312082  0.36322464  0.622896285  0.27607550  1.245121599
## 2 -0.05146837  0.0492724 -0.01238629 -0.087713363 -0.03710273 -0.004395251
## 3  0.35590025  0.5837827  0.03301526  0.808620531  1.07073115  0.616207360
## 4  0.01714820 -0.0653358  0.03690938 -0.004723697  0.03497875  0.016201064
## 5 -0.19220150 -0.1286412 -0.02793327 -0.179127117 -0.21816580 -0.177738408
##     hollister abercrombie          die       death       drunk       drugs
## 1  0.31525537   0.4131560  1.712160983  0.94713629  1.83371069  2.73878856
## 2 -0.16788599  -0.1413652  0.008941101  0.05464759 -0.08699556 -0.06414588
## 3  0.85951603   0.7935060  0.062399295  0.12642222  0.03594162 -0.05888141
## 4 -0.08381546  -0.0861708 -0.067312427 -0.01611162 -0.06891763 -0.08795059
## 5 -0.16182051  -0.1545430 -0.085876102 -0.06882571 -0.08386703 -0.10777278
```

Step 5 - improving model performance

```
teens$cluster <- teen_clusters$cluster
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
```

```
# Aggregate data
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
```

```
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
```

```
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
```

Problem 2 (40 Points)

Provide 100-300 word answers to each of the following interview questions:

1. (10 Points) What are some of the key differences between SVM and Random Forest for classification? When is each algorithm appropriate and preferable? Provide examples.

Random Forest is suited for multi-class problems, while SVM is suited for two-class problems. If you were to solve a multi-class problem with SVM, you would have to reduce it to multiple binary classification problems.

Random Forest works well with a mix of numerical and categorical features. SVM maximizes the "margin" and thus relies on the concept of "distance" between different points. You can decide is the distance is meaningful. As a result, one-hot encoding for categorical features is necessary. Additionally, min-max or other scaling methods is highly recommended during data pre-processing.

If you have data with n points and m features, you need to construct an $n \times n$ matrix by calculating n^2 dot products (computational complexity) for SVM. Therefore, as a rule of thumb, SVM is not really scalable beyond 10^5 points. A large number of features (homogeneous features with meaningful distance, is generally not a problem. A good example would be a pixel with an image.

For a classification problem, Random Forest gives you probability of belonging to class. SVM gives you distance to the boundary, you still need to convert it to probability somehow if you need probability. Regardless, SVM generally performs better than Random Forest. SVM gives you "support vectors", which are points in each class closest to the boundary between classes. Those vectors can be interpreted in different ways.

2. (10 Points) Why might it be preferable to include fewer predictors over many?

When you add irrelevant features, it increases the model's tendency to over-fit because those features introduce more noise, while adding nothing to the model besides complication. When two variables are correlated, they might be harder to interpret when using regression. Dimensionality can also be a curse as well as a blessing. Additionally, it takes more time to include more variables in your model and to get that data. As a result, is a computational and a labor cost that is incurred with adding more predictors. When thinking of these types of problems, it is always to start and think simple so others can understand your work.

3. (10 Points) You are asked to provide R-Squared for a kNN regression model. How would you respond to that request?

R^2 is a cursory measure of goodness of fit of a linear model to the data and it is used in regression analysis. It is popular when deciding between linear and non-linear regression models. kNN for classification has different evaluation metrics than regression. You can look at 'accuracy', 'true-positive', 'false-positive', etc (TP, FP, TN, FN), 'precision', 'recall', 'F1 score', etc. for evaluating the performance of a kNN regression model. Essentially, R^2 is not a good metric for kNN, and there are other models where R^2 is useful. Instead, you can look at accuracy, ROC, mean Average Precision, or F1-score. These metrics are all different, although accuracy is a good metric to start with.

4. (10 Points) How can you determine which features to include when building a multiple regression model?

You should not rely on the (in)significance of the coefficients. You should first pick the criterion that describes your prediction needs best (e.g. misclassification rate, AUC of ROC, some form of these with weights,...)

For each model of interest, evaluate this criterion. This can be done with a validation set through cross validation (typically tenfold), or whatever other options your criterion of interest allows. You also should probably pick the most parsimonious model (least variables) within one standard error of the best value. If you have time to do some extensive data processing, you can try out different features, and see which combination produces the highest accuracy values. However, this is usually unfeasible because it takes too much time and effort. So to decide which features you should incorporate, then you should use forward/backward modeling or penalized/Lasso regression. For R, the glmnet package does the job for this kind of work.