DA5030.A8.Parpattedar

Shruti Parpattedar

April 1, 2019

Problem 1

Step 1 - Collecting data

Downloading and loading the SNS dataset.

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

Step 2 - Exploring and preparing the data

Exploring the dataset. NAs where found in Gender and Age attributes.

The age attribute was altered such that only those in the range of 13 and 20 retained their values, the others were changed to NA.

Dummy coding the gender column into a column with 1 if female and 0 otherwise. no_gender column is 1 is the gender value is NA else 0.

Replacing the NA values in Age with the mean grouped based on the grad year.

```
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 ...
```

```
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
```

```
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
```

```
# Data preparation - dummy coding missing values
teens$female <- ifelse(teens$gender == "F" & !is.na(teens$gender), 1, 0)
teens$no_gender <- ifelse(is.na(teens$gender), 1, 0)
table(teens$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
```

```
# Data preparation - imputing the missing values
mean(teens$age)
```

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

```
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
```

```
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

Taking the attributes with only numeric values and applying z-scale to them.

Applying the kmeans function with 5 centers to the scaled dataset.

```
library(stats)
interests <- teens[5:40]
interests_z <- as.data.frame(lapply(interests, scale))
set.seed(2345)
teen_clusters <- kmeans(interests_z, 5)
```

Step 4 – Evaluating model performance

Displaying the cluster size and centers

```
teen_clusters$size
```

```
## [1]   871   600  5981  1034 21514
```

```
teen_clusters$centers
```

```
##    basketball   football      soccer    softball  volleyball    swimming
## 1  0.16001227  0.2364174  0.10385512  0.07232021  0.18897158  0.23970234
## 2 -0.09195886  0.0652625 -0.09932124 -0.01739428 -0.06219308  0.03339844
## 3  0.52755083  0.4873480  0.29778605  0.37178877  0.37986175  0.29628671
## 4  0.34081039  0.3593965  0.12722250  0.16384661  0.11032200  0.26943332
## 5 -0.16695523 -0.1641499 -0.09033520 -0.11367669 -0.11682181 -0.10595448
##   cheerleading    baseball      tennis      sports        cute
## 1    0.3931445  0.02993479  0.13532387  0.10257837  0.37884271
## 2   -0.1101103 -0.11487510  0.04062204 -0.09899231 -0.03265037
## 3    0.3303485  0.35231971  0.14057808  0.32967130  0.54442929
## 4    0.1856664  0.27527088  0.10980958  0.79711920  0.47866008
## 5   -0.1136077 -0.10918483 -0.05097057 -0.13135334 -0.18878627
##            sex        sexy         hot      kissed        dance         band
## 1  0.020042068  0.11740551  0.41389104  0.06787768  0.22780899 -0.10257102
## 2 -0.042486141 -0.04329091 -0.03812345 -0.04554933  0.04573186  4.06726666
## 3  0.002913623  0.24040196  0.38551819 -0.03356121  0.45662534 -0.02120728
## 4  2.028471066  0.51266080  0.31708549  2.97973077  0.45535061  0.38053621
## 5 -0.097928345 -0.09501817 -0.13810894 -0.13535855 -0.15932739 -0.12167214
##      marching      music        rock         god      church       jesus
## 1 -0.10942590  0.1378306  0.05905951  0.03651755 -0.00709374  0.01458533
## 2  5.25757242  0.4981238  0.15963917  0.09283620  0.06414651  0.04801941
## 3 -0.10880541  0.2844999  0.21436936  0.35014919  0.53739806  0.27843424
## 4 -0.02014608  1.1367885  1.21013948  0.41679142  0.16627797  0.12988313
## 5 -0.11098063 -0.1532006 -0.12460034 -0.12144246 -0.15889274 -0.08557822
##         bible         hair       dress      blonde         mall    shopping
## 1 -0.03692278  0.43807926  0.14905267  0.06137340  0.60368108  0.79806891
## 2  0.05863810 -0.04484083  0.07201611 -0.01146396 -0.08724304 -0.03865318
## 3  0.22990963  0.23612853  0.39407628  0.03471458  0.48318495  0.66327838
## 4  0.08478769  2.55623737  0.53852195  0.36134138  0.62256686  0.27101815
## 5 -0.06813159 -0.20498730 -0.14348036 -0.02918252 -0.18625656 -0.22865236
##         clothes  hollister abercrombie          die       death
## 1  0.5651537331  4.1521844  3.96493810  0.043475966  0.09857501
## 2 -0.0003526292 -0.1678300 -0.14129577  0.009447317  0.05135888
## 3  0.3759725120 -0.0553846 -0.07417839  0.037989066  0.11972190
## 4  1.2306917174  0.1610784  0.26324494  1.712181870  0.93631312
## 5 -0.1865419798 -0.1557662 -0.14861104 -0.094875180 -0.08370729
##          drunk       drugs
## 1  0.035614771  0.03443294
## 2 -0.086773220 -0.06878491
## 3 -0.009688746 -0.05973769
## 4  1.897388200  2.73326605
## 5 -0.087520105 -0.11423381
```

Step 5 – Improving model performance

Setting the cluster number to each value in the dataset.

Applying the aggregate function to age, female and friends column to calculate the mean grouped by the clusters.

```r
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       4   <NA> 18.995      10
```

```r
aggregate(data = teens, age ~ cluster, mean)
```

```
##   cluster      age
## 1       1 16.86497
## 2       2 17.39037
## 3       3 17.07656
## 4       4 17.11957
## 5       5 17.29849
```

```
aggregate(data = teens, female ~ cluster, mean)
```

```
##   cluster    female
## 1       1 0.8381171
## 2       2 0.7250000
## 3       3 0.8378198
## 4       4 0.8027079
## 5       5 0.6994515
```

```
aggregate(data = teens, friends ~ cluster, mean)
```

```
##   cluster  friends
## 1       1 41.43054
## 2       2 32.57333
## 3       3 37.16185
## 4       4 30.50290
## 5       5 27.70052
```

Problem 2

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

SVM is essentially suited for two-class problems; for multiclass problems, it will need to be reduced to multiple binary classification problems. On the other hand, Random Forest is essentially suited for multiclass problems.

"Margin" is maximized in SVM, thus making it rely on the concept of "distance" between different points. Random forest works well with a mix of both categorical and numerical features.

For example, for a classification problem, SVM gives you distance to the boundary, it will still be required to convert it to a probability. Conversely, Random Forest directly gives you probability of belonging to a class.

For problems where SVM can be applied, it generally works better than Random Forests.

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

Using a higher number of predictors increases the complexity of the model. The training error decreases as this complexity increases, but this leads to problems like overfitting.

The aim of a model is to be trained on the training dataset but be generalized enough to be applicable to the dataset from which the training dataset was taken. So, if higher number of predictors are considered the model gives rise to higher error rates when used with real data.

Thus, it makes more sense to use fewer but more significant predictors than using more predictors just to cover all the data. Reducing the number of predictors also reduces the problem of overfitting.

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

kNN regression model is an algorithm which stores all the given cases and uses them to predict the numerical target based on a similarity measure.

R-squared is a measure of goodness of fit of a linear model to the data and depicts what percent of variance in the data has been represented by the model. It is used for regression analysis.

R-squared can be provided for kNN regression since numerical values are being predicted. But, for classification using kNN there are better measures for goodness of fit. Accuracy or ROC or mean average precision are some of the evaluation metrics that could be used for kNN classification instead of R-squared.

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

There are multiple ways to determine which features to include in a multiple regression model; such as forward and backward fitting using either p-value, Adjusted R-Squared, or AIC.

Initially start with the features with the highest correlation with the dependent variable and apply regression function to it. A summary of this model will result in a table containing the p-values for each of the features included in the model. It also indicates the statistically significant features for that model. Using this information, the least significant features can be eliminated to be left with only the significant features. This is known as stepwise backward elimination.

Using p-values is one of the values of backward elimination. We can also use the step() function which fits a model based on the AIC values of each feature.