Stefan Maciolek

11/19/2019

Data Mining Midterm

METHODS:

Data were obtained from the geno dataset that consisted of a categorical response indicating respondents' number of relatives with Alzheimer's, with 3 levels, and categorical predictors based on the respondents' SNPs. Various methods of model fitting methods were tested – first the model was fitted with each method on a training set, and then it was evaluated with a testing set. 57 cases were chosen as the training data set, and 15 as the testing data set. It was decided to partition the data set by case rather than SNP, since there were dimensionality issues when trying to fit training and testing sets with different numbers of SNPs (due to there being different numbers of predictors). The partitioning was done by creating a random uniformly distributed dummy variable, sorting by this variable, and keeping the bottom 57 cases for the training set and the top 15 for the testing set after sorting. After the predictor and response data sets were partitioned into training and testing sets, the response was coded as a factor in order to ensure that R did not treat it as a continuous variable.

The following methods were used: Support Vector Machine (SVM), Random Forest, Naïve Bayes, GLM Net, and Elastic Net Penalized SVM. In each case, the model was fit on the training set, and then the testing set was run through the R prediction function using each model. The number classified in each category was output and was compared to the actual response frequency. The number of cases misclassified was used as the criterion for comparing the effectiveness of the models.

In SVM, usually there are 2 response classes, however, the R svm function from the e1071 package was able to handle this response with 3 classes. For the random forest method, the randomforest function from the package with the same name was used, and the number of trees was set to 500. Likewise, the naïve bayes was found using that function from the e1071 package. For the GLM net, a multinomial model was specified for the categorical output, and when predicting the testing set, the minimum lambda was chosen. Finally, the svmfs function in R that was used to find the penalized SVM model in R required a dichotomous response. In order to use this function, the response was recoded so that a -1 indicated a lack of relatives with Alzheimer's, and 1 indicated 1 or more relatives having the disease. In addition, the predicted cases had to be rounded to either -1 or 1.

RESULTS:

0	**1**	**2**
4	9	2

Table 1: Actual Frequencies for cases in the testing set

SVM:			Random Forest			Naïve Bayes			GLM Net			Penalized SMV	
0	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	**-1**	**1**
0	9	6	2	8	5	4	9	2	6	4	5	3	12

Table 2: Frequencies of predicted cases in the testing set by method

Method	Number Misclassified
SVM	4
Random Forest	3
Naïve Bayes	0
GLM Net	5
Penalized SVM	1

Table 3: Number of cases misclassified by method

DISCUSSION

From table 3, it can be seen that the Naïve Bayes method was the best in this study, with none of the testing data points being misclassified. Penalized SVM was a close second, however, it is important to note that responses of 1 and 2 were recoded to just be 1, therefore any misclassifications between these two categories are lost. Nonetheless, these two methods were relatively effective, as compared to all of the other methods that misclassified at least 20% of the data. Given, however, that the penalized SVM incurred a loss of resolution in the response, the Naïve Bayes would be the only recommended method for this type of analysis in the future.

It is important to note that while the Naïve Bayes method was found to be the most effective method for this data set, this is not necessarily true for every data set of this type. Furthermore, given the large number of parameters and options available in each method, there are a myriad of possibilities to try, of which only a few were practical to test in this analysis. Therefore, caution is advised with generalizing this conclusion.

APPENDIX – CODE:

```
install.packages("glmnet")
install.packages("penalizedSVM")
install.packages("e1071")
install.packages("randomForest")
install.packages("dplyr")
library("glmnet")
library("penalizedSVM")
library("e1071")
library("randomForest")
library("dplyr")
```

```
#the data set location will have to be updated
data <- load("E:\\588\\Midterm\\geno.R")
x0 <- t(x0)
x0 <- as.data.frame(x0)

#seperating data into testing and training sets
set.seed(145003903)
rando <- runif(72)
x0 <- cbind(x0,rando)
x0 <- x0[order(rando),]
y <- cbind(y, rando)
y <- y[order(rando),]
ytrain <- y[1:57,]
ytrain <- ytrain[,-2]
ytrain <- factor(ytrain)
ytest <- y[58:72,]
ytest <- ytest[,-2]
ytest <- factor(ytest)

train <- x0[1:57,]
train <- train[,-36255]
test <- x0[58:72,]
test <- test[,-36255]

#SVM:
svm <- svm(train, ytrain)
svmpredicted <- predict(svm, newdata = test)
table(svmpredicted)

#random forest:
rF <- randomForest(train, ytrain, test, ytest, ntree = 500,)
table(rF$test$predicted)

#naive Bayes
nB <- naiveBayes(train, ytrain)
nbpredicted <- predict(nB,newdata = test)
table(nbpredicted)

#GLM net:
glm <- glmnet(as.matrix(train), ytrain, family = "multinomial")
glmpredicted <- predict(glm,newx = as.matrix(test),s=min(glm$lambda),type = "class")
table(glmpredicted)
```

```
#penalized SVM:
ytest <- recode(ytest, "0" = "-1", "2" = "1")
ytrain <- recode(ytrain, "0" = "-1", "2" = "1")
psvm <- svmfs(as.matrix(train), y=ytrain, fs.method = "scad+L2",maxevals = 5)
psvmpredicted <- predict.penSVM(psvm, newdata = test)
psvmround <- rep(0,15)
for (i in 1:15){
  if (psvmpredicted$fitted[i] < 0) {psvmround[i] = -1}
  else if (psvmpredicted$fitted[i] > 0) {psvmround[i] = 1}
}
table(psvmround)
```