Stat 218 - Model Comparison Exercise

Rommel Bartolome

April 12, 2019

a) Divide your dataset into training and test sets. Call them "train" and "test." Train should contain 750 observations; test should contain 250 observations.

We divide the dataset to train and test set:

```
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------------------
## v tibble  2.0.1     v purrr   0.3.2
## v tidyr   0.8.2     v dplyr   0.7.8
## v readr   1.3.1     v stringr 1.3.1
## v tibble  2.0.1     v forcats 0.3.0
## Warning: package 'purrr' was built under R version 3.5.3
## -- Conflicts ------------------------------------------------------------------------------------ tid
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(e1071)
set.seed(1)
data(GermanCredit)

sample <- sample.int(n = nrow(GermanCredit),
                     size = floor(.75*nrow(GermanCredit)),
                     replace = F)
train <- GermanCredit[sample, ]
test  <- GermanCredit[-sample, ]
```

b) Create ten folds in your training set, each containing 75 observations. Call them fold1, fold2, fold3, . . , fold10.

We create the indexes of the ten folds in the training dataset. We will use these values later.

```
folds <- sample(1:750, 750)
```

c) Perform the fitting and assessment steps ten times. At step i, train your model using everything but foldi; use foldi to assess performance. For example, for the first iteration of this procedure, train the models using a dataset that excludes observations in fold1. Assess the models using the observations in fold1.

Fitting

i.) Fit a logistic regression model and perform model building/variable selection. (Note the variables that you have selected.)

We then fit a logistic regression model and perform model building/variable selection. We also show that significant variables.

```
logistic <- c()
for.ttest <- c()
for (x in seq(1, 10, 1)){
    i <- x + (x-1)*74
    dat <- train[-folds[i:(i+74)], ]
    model <- glm(formula = Class ~ ., data = dat, family = 'binomial')
    summary <- summary(model)
    sig_vars <- rownames(summary$coefficients)[which(summary$coefficients[,4]<= 0.05)]
    sig_vars <- sig_vars[sig_vars != '(Intercept)']
    formula <- paste0("Class ~ ", paste(sig_vars, collapse = ' + '))
    model_trunc <- glm(formula = formula(formula),
                       data = train[folds[i:(i+74)], ], family = 'binomial')
    log_pred <- predict(model_trunc, data = train[folds[i:(i+74)], ], type = 'response')

    accuracy <- sum(diag(table(log_pred > 0.5, train[folds[i:(i+74)], ]$Class))
                 /nrow(train[folds[i:(i+74)], ]))

    logistic <- c(logistic, c(x, accuracy, formula))
    for.ttest <- c(for.ttest, c(log_pred > 0.5))
}

logistic_array <- array(unlist(logistic), dim=c(3, 10)) %>% aperm() %>% data.frame()
names(logistic_array) <- c("fold", "accuracy", "formula")
logistic_array[,c(1,2)]
```

```
##    fold          accuracy
## 1     1 0.853333333333333
## 2     2 0.853333333333333
## 3     3 0.813333333333333
## 4     4 0.826666666666667
## 5     5 0.746666666666667
## 6     6 0.826666666666667
## 7     7 0.826666666666667
## 8     8              0.72
## 9     9 0.773333333333333
## 10   10              0.84
```

We also show the significant variables:

```r
logistic_array[,c(1,3)]
```

```
##    fold
## 1     1
## 2     2
## 3     3
## 4     4
## 5     5
## 6     6
## 7     7
## 8     8
## 9     9
## 10   10
##
## 1                  Class ~ Duration + InstallmentRatePercentage + Age + CheckingAccountStatus.lt.0 +
## 2            Class ~ Duration + InstallmentRatePercentage + NumberExistingCredits + CheckingAccountSt
## 3                                                                                         Class ~ Du
## 4  Class ~ Duration + InstallmentRatePercentage + NumberExistingCredits + CheckingAccountStatus.lt.0
## 5   Class ~ Duration + InstallmentRatePercentage + Age + NumberExistingCredits + CheckingAccountStatu
## 6                      Class ~ Duration + InstallmentRatePercentage + CheckingAccountStatus.lt.0 + Cl
## 7                      Class ~ Duration + InstallmentRatePercentage + CheckingAccountStatus.lt.0 + Cl
## 8                                                                                                Clas
## 9                                                                                         Class ~ Du
## 10              Class ~ Duration + NumberExistingCredits + CheckingAccountStatus.lt.0 + CheckingAcco
```

ii.) Build a support vector machine using the radial kernel. Tune by setting gamma = c(0.1, 0.5, 1, 2) - let use leave cost at its default value. (Note the final settings after performing tuning. You may use a subset of the predictors, but make sure to justify your choice.)

We also build a support vector machine using the radial kernel. We tune by setting gamma = c(0.1, 0.5, 1, 2):

```r
svm <- c()
for.ttestsvm <- c()
for (x in seq(1, 10, 1)){
    i <- x + (x-1)*74
    dat <- train[-folds[i:(i+74)], ]
    svm.model <- tune(e1071::svm, Class ~ ., data = dat, kernel = "radial",
                      ranges = list(gamma = c(0.1, 0.5, 1, 2)))
    svm_preds <- predict(svm.model$best.model,
                         newdata=train[folds[i:(i+74)], ], type="response")
    accuracy <- sum(diag(table(svm_preds, train[folds[i:(i+74)], ]$Class))) /
      nrow(train[folds[i:(i+74)], ])
    svm <- c(svm, c(x, accuracy))
    for.ttestsvm <- c(for.ttestsvm, svm_preds == 2)
}

svm_array <- array(unlist(svm), dim=c(2, 10)) %>% aperm() %>% data.frame()
names(svm_array) <- c("fold", "accuracy")
svm_array
```

```
##    fold  accuracy
## 1     1 0.7466667
## 2     2 0.6533333
## 3     3 0.6933333
## 4     4 0.6666667
```

```
## 5      5 0.7066667
## 6      6 0.7066667
## 7      7 0.7333333
## 8      8 0.6933333
## 9      9 0.7200000
## 10    10 0.7200000
```

Here we used all the the subset of predictors.

Assessment

We compute for the proportion of correctly-classified observations, rounded off up to four decimal places:

```r
assessment <- cbind(logistic_array, svm_array)[,-c(3,4)]
names(assessment) <- c("Excluded Fold",
                       "% Correct - Logistic Regression",
                       "% Correct - Radial SVM")
assessment[,2] <- assessment[,2] %>% as.character %>% as.numeric
assessment[,c(2,3)] <- format(round(assessment[,c(2,3)], 4))
assessment
```

```
##    Excluded Fold % Correct - Logistic Regression % Correct - Radial SVM
## 1              1                          0.8533                 0.7467
## 2              2                          0.8533                 0.6533
## 3              3                          0.8133                 0.6933
## 4              4                          0.8267                 0.6667
## 5              5                          0.7467                 0.7067
## 6              6                          0.8267                 0.7067
## 7              7                          0.8267                 0.7333
## 8              8                          0.7200                 0.6933
## 9              9                          0.7733                 0.7200
## 10            10                          0.8400                 0.7200
```

d) Perform a paired t-test to test if there is a difference between the proportion of correctly classified observations for the two methods. In the t.test function in R, do not forget to set paired=TRUE. Note the result (p-value) and interpret briefly.

```r
for.ttest_l <- array(unlist(for.ttest), dim=c(10, 75)) %>% aperm() %>% data.frame()
for.ttest_s <- array(unlist(for.ttestsvm), dim=c(10, 75)) %>% aperm() %>% data.frame()
t.test(for.ttest, for.ttestsvm, paired = T)
```

```
##
##  Paired t-test
##
## data:  for.ttest and for.ttestsvm
## t = 48.701, df = 749, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.7293647 0.7906353
## sample estimates:
## mean of the differences
##                    0.76
```

Here, we see that there is a difference in the between the proportion of correctly classified observations for the two methods. The p-value is small, which indicates strong evidence against the null hypothesis, so you reject the null hypothesis (there is no difference).