Dylan Patel

Aver Mckay

Jianqi Liu

Logistic regression

3/31/2020

spam Dataset

```
library(kernlab)
data('spam')
tibble::as.tibble(spam)
## # A tibble: 4,601 x 58
##     make address   all num3d   our  over remove internet order  mail
##    <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>    <dbl> <dbl> <dbl>
##  1 0        0.64  0.64     0  0.32  0      0         0    0     0
##  2 0.21     0.28  0.5      0  0.14  0.28   0.21      0.07 0     0.94
##  3 0.06     0     0.71     0  1.23  0.19   0.19      0.12 0.64  0.25
##  4 0        0     0        0  0.63  0      0.31      0.63 0.31  0.63
##  5 0        0     0        0  0.63  0      0.31      0.63 0.31  0.63
##  6 0        0     0        0  1.85  0      0         1.85 0     0
##  7 0        0     0        0  1.92  0      0         0    0     0.64
##  8 0        0     0        0  1.88  0      0         1.88 0     0
##  9 0.15     0     0.46     0  0.61  0      0.3       0    0.92  0.76
## 10 0.06     0.12  0.77     0  0.19  0.32   0.38      0    0.06  0
## # … with 4,591 more rows, and 48 more variables: receive <dbl>,
## #   will <dbl>, people <dbl>, report <dbl>, addresses <dbl>, free <dbl>,
## #   business <dbl>, email <dbl>, you <dbl>, credit <dbl>, your <dbl>,
## #   font <dbl>, num000 <dbl>, money <dbl>, hp <dbl>, hpl <dbl>,
## #   george <dbl>, num650 <dbl>, lab <dbl>, labs <dbl>, telnet <dbl>,
## #   num857 <dbl>, data <dbl>, num415 <dbl>, num85 <dbl>, technology <dbl>,
```

```
## #   num1999 <dbl>, parts <dbl>, pm <dbl>, direct <dbl>, cs <dbl>,
## #   meeting <dbl>, original <dbl>, project <dbl>, re <dbl>, edu <dbl>,
## #   table <dbl>, conference <dbl>, charSemicolon <dbl>,
## #   charRoundbracket <dbl>, charSquarebracket <dbl>,
## #   charExclamation <dbl>, charDollar <dbl>, charHash <dbl>,
## #   capitalAve <dbl>, capitalLong <dbl>, capitalTotal <dbl>, type <fct>
is.factor(spam$type)
## [1] TRUE
levels(spam$type)
## [1] "nonspam" "spam"
set.seed(42)
# spam_idx = sample(nrow(spam), round(nrow(spam) / 2))
spam_idx = sample(nrow(spam), 1000)
spam_trn = spam[spam_idx, ]
spam_tst = spam[-spam_idx, ]
fit_caps = glm(type ~ capitalTotal,
               data = spam_trn, family = binomial)
fit_selected = glm(type ~ edu + money + capitalTotal + charDollar,
                   data = spam_trn, family = binomial)
fit_additive = glm(type ~ .,
                   data = spam_trn, family = binomial)
fit_over = glm(type ~ capitalTotal * (.),
               data = spam_trn, family = binomial, maxit = 50)
# training misclassification rate
mean(ifelse(predict(fit_caps) > 0, "spam", "nonspam") != spam_trn$type)#0.34
## [1] 0.339
mean(ifelse(predict(fit_selected) > 0, "spam", "nonspam") != spam_trn$type)#0.212
## [1] 0.224
mean(ifelse(predict(fit_additive) > 0, "spam", "nonspam") != spam_trn$type)#0.0644
## [1] 0.066
mean(ifelse(predict(fit_over) > 0, "spam", "nonspam") != spam_trn$type)#0.063
## [1] 0.136
library(boot)
set.seed(1)
cv.glm(spam_trn, fit_caps, K = 5)$delta[1]#0.2134
## [1] 0.2166961
cv.glm(spam_trn, fit_selected, K = 5)$delta[1]#0.1522
```

```
## [1] 0.1587043
cv.glm(spam_trn, fit_additive, K = 5)$delta[1]#0.0784
## [1] 0.08684467
cv.glm(spam_trn, fit_over, K = 5)$delta[1]#0.108
## [1] 0.14
```

Exercise 1(part1) 1. Execute the code above. Based on the results, rank the models from "most underfit" to "most overfit". Most underfit: fit_cap > fit_additive > fit_selected >fit_over 2.Re-run the code above with 100 folds and a different seed. Does your conclusion change?

```
set.seed(2)
cv.glm(spam_trn, fit_caps, K = 100)$delta[1]#0.2138123
## [1] 0.2168058
cv.glm(spam_trn, fit_selected, K = 100)$delta[1]#0.153601
## [1] 0.1588852
cv.glm(spam_trn, fit_additive, K = 100)$delta[1]#0.06699597
## [1] 0.08098914
cv.glm(spam_trn, fit_over, K =  100)$delta[1]#0.09655727
## [1] 0.138
```

The result of the cv.glm under a different random seed and 100 folds is similar to the result before. Thus my conclusion does not change.

```
make_conf_mat = function(predicted, actual) {
  table(predicted = predicted, actual = actual)
}
spam_tst_pred = ifelse(predict(fit_additive, spam_tst) > 0,
                       "spam",
                       "nonspam")
spam_tst_pred = ifelse(predict(fit_additive, spam_tst, type = "response") > 0.5,
                       "spam",
                       "nonspam")
(conf_mat_50 = make_conf_mat(predicted = spam_tst_pred, actual = spam_tst$type))
##          actual
## predicted nonspam spam
##   nonspam    2057  157
##   spam        127 1260
table(spam_tst$type) / nrow(spam_tst)
##
##   nonspam      spam
```

```
## 0.6064982 0.3935018
```

Exercise 1(part2) 3.Generate four confusion matrices for each of the four models fit in Part 1.

```
#fit_caps
spam_tst_pred = ifelse(predict(fit_caps, spam_tst) > 0,
                       "spam",
                       "nonspam")
spam_tst_pred = ifelse(predict(fit_caps, spam_tst, type = "response") > 0.5,
                       "spam",
                       "nonspam")
(conf_mat_50 = make_conf_mat(predicted = spam_tst_pred, actual = spam_tst$type))
##          actual
## predicted nonspam spam
##   nonspam    2022 1066
##   spam        162  351
cat('accuracy:',sum(diag(conf_mat_50))/sum(conf_mat_50))
## accuracy: 0.6589836
#take spam as positive event
cat('sens:',conf_mat_50[2,2]/(conf_mat_50[2,2]+conf_mat_50[1,2]))
## sens: 0.2477064
cat('spec:',conf_mat_50[1,1]/(conf_mat_50[1,1]+conf_mat_50[2,1]))
## spec: 0.9258242
#fit_selected
spam_tst_pred = ifelse(predict(fit_selected, spam_tst) > 0,
                       "spam",
                       "nonspam")
spam_tst_pred = ifelse(predict(fit_selected, spam_tst, type = "response") > 0.5,
                       "spam",
                       "nonspam")
(conf_mat_50 = make_conf_mat(predicted = spam_tst_pred, actual = spam_tst$type))
##          actual
## predicted nonspam spam
##   nonspam    2073  615
##   spam        111  802
cat('accuracy:',sum(diag(conf_mat_50))/sum(conf_mat_50))
## accuracy: 0.7983893
#take spam as positive event
```

```
cat('sens:',conf_mat_50[2,2]/(conf_mat_50[2,2]+conf_mat_50[1,2]))
## sens: 0.5659845
cat('spec:',conf_mat_50[1,1]/(conf_mat_50[1,1]+conf_mat_50[2,1]))
## spec: 0.9491758
#fit_additive
spam_tst_pred = ifelse(predict(fit_additive, spam_tst) > 0,
                       "spam",
                       "nonspam")
spam_tst_pred = ifelse(predict(fit_additive, spam_tst, type = "response") > 0.5,
                       "spam",
                       "nonspam")
(conf_mat_50 = make_conf_mat(predicted = spam_tst_pred, actual = spam_tst$type))
##          actual
## predicted nonspam spam
##   nonspam    2057  157
##   spam        127 1260
cat('accuracy:',sum(diag(conf_mat_50))/sum(conf_mat_50))
## accuracy: 0.921133
#take spam as positive event
cat('sens:',conf_mat_50[2,2]/(conf_mat_50[2,2]+conf_mat_50[1,2]))
## sens: 0.8892025
cat('spec:',conf_mat_50[1,1]/(conf_mat_50[1,1]+conf_mat_50[2,1]))
## spec: 0.9418498
#fit_over
spam_tst_pred = ifelse(predict(fit_over, spam_tst) > 0,
                       "spam",
                       "nonspam")
spam_tst_pred = ifelse(predict(fit_over, spam_tst, type = "response") > 0.5,
                       "spam",
                       "nonspam")
(conf_mat_50 = make_conf_mat(predicted = spam_tst_pred, actual = spam_tst$type))
##          actual
## predicted nonspam spam
##   nonspam    1725  103
##   spam        459 1314
cat('accuracy:',sum(diag(conf_mat_50))/sum(conf_mat_50))
## accuracy: 0.8439322
```

```
#take spam as positive event
cat('sens:',conf_mat_50[2,2]/(conf_mat_50[2,2]+conf_mat_50[1,2]))
## sens: 0.9273112
cat('spec:',conf_mat_50[1,1]/(conf_mat_50[1,1]+conf_mat_50[2,1]))
## spec: 0.7898352
```

4.Which is the best model? Write 2 paragraphs justifying your decision. You must mention (a) the overall accuracy of each model; and (b) whether some errors are better or worse than others, and you must use the terms specificity and sensitivity. For (b) think carefully… misclassified email is a pain in the butt for users!

Answer: Fit_additive is the best model. The overall accuracy of this model is 0.9172, which is the highest among all of the models. Taken the spam as a positive event(which we show interest in), I calculate the sensitivity and specifity of those models. The sens and spec of the additive model is 0.886136 and 0.9373571. Although the value of sens of the additive model is a little bit smaller than that of the fit over model, the additive model performs best overall obviously.

Exercise 2 1.Use the bank data and create a train / test split.

```
bank <- read.csv('~/Desktop/Logistic_Regression/bank.csv')
head(bank)
##   age         job marital education default balance housing loan  contact
## 1  30  unemployed married   primary      no    1787      no   no cellular
## 2  33    services married secondary      no    4789     yes  yes cellular
## 3  35  management  single  tertiary      no    1350     yes   no cellular
## 4  30  management married  tertiary      no    1476     yes  yes  unknown
## 5  59 blue-collar married secondary      no       0     yes   no  unknown
## 6  35  management  single  tertiary      no     747      no   no cellular
##   day month duration campaign previous  y
## 1  19   oct       79        1        0 no
## 2  11   may      220        1        4 no
## 3  16   apr      185        1        1 no
## 4   3   jun      199        4        0 no
## 5   5   may      226        1        0 no
## 6  23   feb      141        2        3 no
set.seed(1)
bank_idx = sample(nrow(bank), 500)
bank_trn = bank[bank_idx, ]
bank_tst = bank[-bank_idx, ]
```

2.Run any logistic regression you like with 10-fold cross-validation in order to predict the yes/no variable (y).

```
fit_additive = glm(y ~ age+job+marital+education,
                   data = bank_trn, family = binomial)
cv.glm(bank_trn, fit_additive, K = 10)$delta[1]#0.09757089
## [1] 0.119989
bank_tst_pred = ifelse(predict(fit_additive, bank_tst) > 0,
                       "yes",
                       "no")
bank_tst_pred = ifelse(predict(fit_additive, bank_tst, type = "response") > 0.5,
                       "yes",
                       "no")
(conf_mat_50 = make_conf_mat(predicted = bank_tst_pred, actual = bank_tst$y))
##          actual
## predicted   no  yes
##       no  3528  433
##       yes   40   20
```

3.Discuss the interpretation of the coefficients in your model. That is, you must write at least one sentence for each of the coefficients which describes how it is related to the response. You may use transformations of variables if you like. FAKE EXAMPLE: age has a positive coefficient, which means that older individuals are more likely to have y = yes.

```
fit_additive
##
## Call:  glm(formula = y ~ age + job + marital + education, family = binomial,
##     data = bank_trn)
##
## Coefficients:
##        (Intercept)                 age      jobblue-collar
##           -4.36054             0.03890             0.19681
##    jobentrepreneur        jobhousemaid       jobmanagement
##            0.03157             1.12706            -0.33651
##         jobretired    jobself-employed         jobservices
##            1.46044             0.19595            -0.14179
##         jobstudent       jobtechnician       jobunemployed
##            0.92351            -0.08626             0.88351
##         jobunknown      maritalmarried       maritalsingle
##            1.02369            -0.21827             0.69460
## educationsecondary   educationtertiary    educationunknown
```

```
##           0.53199            0.81014            0.22157
##
## Degrees of Freedom: 499 Total (i.e. Null);  482 Residual
## Null Deviance:       397.6
## Residual Deviance: 362.4     AIC: 398.4
```

Age has a positive coefficient, which means that older individuals are more likely to have y = yes. Job is a multi-level variable. For the level of jobblue-collar,jobmanagement,jobself-employed,jobservices,jobtechnician,jobunemployed, they have a negative coefficient, which means that lower value of those variables tends to have y=yes. For the variable of marital and education, they have a positive coefficient, which means that higher value of those variables tends to have y=yes.

4.Create a confusion matrix of your preferred model, evaluated against your test data.

```
#fit_additive
fit_additive = glm(y ~ age+job+marital+education,
                   data = bank_trn, family = binomial)
cv.glm(bank_trn, fit_additive, K = 10)$delta[1]#0.09757089
## [1] 0.1173813
bank_tst_pred = ifelse(predict(fit_additive, bank_tst) > 0,
                       "yes",
                       "no")
bank_tst_pred = ifelse(predict(fit_additive, bank_tst, type = "response") > 0.5,
                       "yes",
                       "no")
(conf_mat_50 = make_conf_mat(predicted = bank_tst_pred, actual = bank_tst$y))
##          actual
## predicted   no  yes
##       no  3528  433
##       yes   40   20
#fit_all
fit_all = glm(y ~ .,
                   data = bank_trn, family = binomial)
cv.glm(bank_trn, fit_all, K = 10)$delta[1]#0.09757089
## [1] 0.110288
bank_tst_pred = ifelse(predict(fit_all, bank_tst) > 0,
                       "yes",
                       "no")
bank_tst_pred = ifelse(predict(fit_all, bank_tst, type = "response") > 0.5,
```

```
                         "yes",
                         "no")
(conf_mat_50 = make_conf_mat(predicted = bank_tst_pred, actual = bank_tst$y))
##            actual
## predicted    no  yes
##        no  3409  294
##        yes  159  159
```

I prefer the model of fit_all.