Lab 4 Report

Exercise 1 (Part 1):

1. Use the 5-fold cross-validation to each of the four models fit and calculate the misclassification rate. The R code is shown as below:

```
library(boot)

set.seed(1)
cv.glm(spam_trn, fit_caps, K = 5)$delta[1]
set.seed(1)
cv.glm(spam_trn, fit_selected, K = 5)$delta[1]
set.seed(1)
cv.glm(spam_trn, fit_additive, K = 5)$delta[1]
set.seed(1)
cv.glm(spam_trn, fit_over, K = 5)$delta[1]
```

And the misclassification rates of the four models are shown as below:

```
> set.seed(1)
> cv.glm(spam_trn, fit_caps, K = 5)$delta[1]
[1] 0.2138392
> set.seed(1)
> cv.glm(spam_trn, fit_selected, K = 5)$delta[1]
[1] 0.1530491
> set.seed(1)
> cv.glm(spam_trn, fit_additive, K = 5)$delta[1]
[1] 0.07356258
> set.seed(1)
> cv.glm(spam_trn, fit_over, K = 5)$delta[1]
[1] 0.135
```

When the misclassification rate is higher, the model is more underfit. And when the misclassification rate is lower, the model is more overfit.

Thus, rank the above four models from most underfit to most overfit, the result is:

fit_caps, fit_selected, fit_over, fit_additive

2. Use a different seed with 7250 and re-run the code with 100 folds. The R code is shown as below:

```
set.seed(7250)
cv.glm(spam_trn, fit_caps, K = 100)$delta[1]
set.seed(7250)
cv.glm(spam_trn, fit_selected, K = 100)$delta[1]
set.seed(7250)
cv.glm(spam_trn, fit_additive, K = 100)$delta[1]
set.seed(7250)
cv.glm(spam_trn, fit_over, K = 100)$delta[1]
```

And the misclassification rates of the four models are shown as below:

```
> set.seed(7250)
> cv.glm(spam_trn, fit_caps, K = 100)$delta[1]
[1] 0.2138285
> set.seed(7250)
> cv.glm(spam_trn, fit_selected, K = 100)$delta[1]
[1] 0.1530884
> set.seed(7250)
> cv.glm(spam_trn, fit_additive, K = 100)$delta[1]
[1] 0.06803901
> set.seed(7250)
> cv.glm(spam_trn, fit_over, K = 100)$delta[1]
[1] 0.1126211
```

Thus, rank the above four models from most underfit to most overfit, the result is:

fit_caps, fit_selected, fit_over, fit_additive

The conclusion in question 2 is the same with the conclusion in question 1. Thus, the conclusion doesn't change.

Exercise 1 (Part 2):

3. For the **fit_caps** model fit, the confusion matrix is:

```
> conf_mat_caps
          actual
predicted nonspam spam
  nonspam    2004 1016
  spam        183  398
```

The four confusion matrices are shown as below:

```
> #####fit_caps: four confusion
> #accuracy
> mean(spam_caps_pred==spam_tst$type)
[1] 0.6670369
> #Prev
> (conf_mat_caps[1,2]+conf_mat_caps[2,2])/nrow(spam_tst)
[1] 0.3926687
> #Sensitivity
> conf_mat_caps[2,2]/(conf_mat_caps[2,2]+conf_mat_caps[1,2])
[1] 0.281471
> #Specificity
> conf_mat_caps[1,1]/(conf_mat_caps[1,1]+conf_mat_caps[2,1])
[1] 0.9163237
```

Thus, for the **fit_caps** model, accuracy is 0.6670369, Prev is 0.3926687, Sensitivity is 0.281471, and Specificity is 0.9163237.

For the **fit_selected** model fit, the confusion matrix is:

```
> conf_mat_selected
          actual
predicted nonspam spam
  nonspam    2050  599
  spam        137  815
```

The four confusion matrices are shown as below:

```
> #####fit_selected: four confusion
> #accuracy
> mean(spam_selected_pred==spam_tst$type)
[1] 0.7956123
> #Prev
> (conf_mat_selected[1,2]+conf_mat_selected[2,2])/nrow(spam_tst)
[1] 0.3926687
> #Sensitivity
> conf_mat_selected[2,2]/(conf_mat_selected[2,2]+conf_mat_selected[1,2])
[1] 0.5763791
> #Specificity
> conf_mat_selected[1,1]/(conf_mat_selected[1,1]+conf_mat_selected[2,1])
[1] 0.9373571
```

Thus, for the **fit_selected** model, accuracy is 0.7956123, Prev is 0.3926687, Sensitivity is 0.5763791, and Specificity is 0.9373571.

For the **fit_additive** model fit, the confusion matrix is:

```
> conf_mat_additive
         actual
predicted nonspam spam
  nonspam    2050  161
  spam        137 1253
```

The four confusion matrices are shown as below:

```
> #####fit_additive: four confusion
> #accuracy
> mean(spam_additive_pred==spam_tst$type)
[1] 0.9172452
> #Prev
> (conf_mat_additive[1,2]+conf_mat_additive[2,2])/nrow(spam_tst)
[1] 0.3926687
> #Sensitivity
> conf_mat_additive[2,2]/(conf_mat_additive[2,2]+conf_mat_additive[1,2])
[1] 0.8861386
> #Specificity
> conf_mat_additive[1,1]/(conf_mat_additive[1,1]+conf_mat_additive[2,1])
[1] 0.9373571
```

Thus, for the **fit_additive** model, accuracy is 0.9172452, Prev is 0.3926687, Sensitivity is 0.8861386, and Specificity is 0.9373571.

For the **fit_over** model fit, the confusion matrix is:

```
> conf_mat_over
         actual
predicted nonspam spam
  nonspam    1979  153
  spam        208 1261
```

The four confusion matrices are shown as below:

```
> #####fit_over: four confusion
> #accuracy
> mean(spam_over_pred==spam_tst$type)
[1] 0.8997501
> #Prev
> (conf_mat_over[1,2]+conf_mat_over[2,2])/nrow(spam_tst)
[1] 0.3926687
> #Sensitivity
> conf_mat_over[2,2]/(conf_mat_over[2,2]+conf_mat_over[1,2])
[1] 0.8917963
> #Specificity
> conf_mat_over[1,1]/(conf_mat_over[1,1]+conf_mat_over[2,1])
[1] 0.9048925
```

Thus, for the **fit_over** model, accuracy is 0.8997501, Prev is 0.3926687, Sensitivity is 0.8917963, and Specificity is 0.9048925.

4. The **fit_additive** model is the best model.

   For the overall accuracy, the higher the accuracy is, the better the model is. Among the four models, the fit_additive model has the highest overall accuracy with a value of 0.9172452.

   For the sensitivity and specificity, the higher the sensitivity is, the better the model is, which means in all the actual spam emails, more are predicted by the model. And the higher the specificity is, the better the model is, which means in all the actual non-spam emails, more are predicted by the model. Among the four models, the fit_additive model has the highest specificity with a value of 0.9373571. Although the sensitivity of fit_additive model is not the highest, but the value 0.8861386 is close to the highest value 0.8917963 of fit_over model.

   Combine all the analysis above, it can be concluded that the fit_additive model is the best model.

Exercise 2:

1. Use the following code to split bank data to a train dataset and a test dataset. Use 50% of all the dataset as train dataset and the other 50% as test dataset.

```
#question 1
set.seed(42)
bank_idx = sample(nrow(bank), round(nrow(bank) / 2))
bank_trn = bank[bank_idx, ]
bank_tst = bank[-bank_idx, ]
```

2. Choose the 5 variables: **job**, **contact**, **month**, **duration**, **campaign** to run logistic regression with 10-fold cross-validation in order to predict the yes/no variable y. The R code is shown as below:

```r
fit_additive_bank = glm(y ~ job + contact + month + duration + campaign,
                        data = bank_trn, family = binomial)

set.seed(7250)
cv.glm(bank_trn, fit_additive_bank, K = 10)$delta[1]

summary(fit_additive_bank)
```

The misclassification rate is shown as below:

```
> set.seed(7250)
> cv.glm(bank_trn, fit_additive_bank, K = 10)$delta[1]
[1] 0.07542694
```

The misclassification rate of 10-fold cross-validation on train dataset is 0.07542694

3. The coefficients of the model are shown as below:

```
> coef(fit_additive_bank)
     (Intercept)  jobblue-collar  jobentrepreneur     jobhousemaid    jobmanagement       jobretired
     -2.39441877     -1.06679059      -0.48744396      -0.72117976      -0.29024345      -0.41677855
jobself-employed     jobservices       jobstudent    jobtechnician    jobunemployed       jobunknown
     -0.18022171     -0.93620787       0.84013056      -0.45233934      -1.22814953      -0.08693655
contacttelephone  contactunknown         monthaug         monthdec         monthfeb         monthjan
     -0.05155163     -1.70950338      -0.17871184      -1.01820252      -0.13268229      -1.26742737
        monthjul        monthjun         monthmar         monthmay         monthnov         monthoct
     -0.87087379      0.29811583       1.68963232      -0.80712109      -1.65211147       1.39007058
        monthsep        duration         campaign
      1.22971747      0.00507434      -0.11830257
```

Job (value is blue-collar / entrepreneur / housemaid / management / retired / self-employed / services / technician / unemployed / unknown) has a negative coefficient, which means that individual who is blue-collar / entrepreneur / housemaid / management / retired / self-employed / services / technician / unemployed / unknown are more likely to have y = no.

Job (value is student) has a positive coefficient, which means that individual who is student are more likely to have y = yes.

Contact (value is telephone / unknown) has a negative coefficient, which means that individual who has a telephone to contact or whose telephone is unknown are more likely to have y = no.

Month (value is January / February / May / July / August / November / December) has a negative coefficient, which means that when the month is January / February / May / July / August / November / December, it is more likely to have y = no.

Month (value is March / June / September / October) has a positive coefficient, which means that when the month is March / June / September / October, it is more likely to have y = yes.

Duration has a positive coefficient, which means that this variable is more likely to lead to y=yes.

Campaign has a negative coefficient, which means that this variable is more likely to lead to y=no.

4. The confusion matrix of the model is:

```
> bank_mat_additive
         actual
predicted   no  yes
      no  1930  196
      yes   68   67
```

The four confusion matrices evaluated on the test dataset are shown as below:

```
> #####fit_additive_bank: four confusion
> #accuracy
> mean(bank_additive_pred==bank_tst$y)
[1] 0.8832375
> #Prev
> (bank_mat_additive[1,2]+bank_mat_additive[2,2])/nrow(bank_tst)
[1] 0.1163202
> #Sensitivity
> bank_mat_additive[2,2]/(bank_mat_additive[2,2]+bank_mat_additive[1,2])
[1] 0.2547529
> #Specificity
> bank_mat_additive[1,1]/(bank_mat_additive[1,1]+bank_mat_additive[2,1])
[1] 0.965966
```

Thus, for the logistic regression, accuracy is 0.8832375, Prev is 0.1163202, Sensitivity is 0.2547529, and Specificity is 0.965966.