Exercise 7 Advanced Methods for Regression and Classification

Stefan Merdian

2024-12-02

```
df <- read.csv2("bank.csv")
str(df)
```

```
## 'data.frame':    4521 obs. of  17 variables:
##  $ age      : int  30 33 35 30 59 35 36 39 41 43 ...
##  $ job      : chr  "unemployed" "services" "management" "management" ...
##  $ marital  : chr  "married" "married" "single" "married" ...
##  $ education: chr  "primary" "secondary" "tertiary" "tertiary" ...
##  $ default  : chr  "no" "no" "no" "no" ...
##  $ balance  : int  1787 4789 1350 1476 0 747 307 147 221 -88 ...
##  $ housing  : chr  "no" "yes" "yes" "yes" ...
##  $ loan     : chr  "no" "yes" "no" "yes" ...
##  $ contact  : chr  "cellular" "cellular" "cellular" "unknown" ...
##  $ day      : int  19 11 16 3 5 23 14 6 14 17 ...
##  $ month    : chr  "oct" "may" "apr" "jun" ...
##  $ duration : int  79 220 185 199 226 141 341 151 57 313 ...
##  $ campaign : int  1 1 1 4 1 2 1 2 2 1 ...
##  $ pdays    : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
##  $ previous : int  0 4 1 0 0 3 2 0 0 2 ...
##  $ poutcome : chr  "unknown" "failure" "failure" "unknown" ...
##  $ y        : chr  "no" "no" "no" "no" ...
```

```
categorical_cols <- c("job", "marital", "education", "default",
                      "housing", "loan", "contact", "month", "poutcome","y")
df[categorical_cols] <- lapply(df[categorical_cols], as.factor)
```

1)

a)

```
set.seed(1)

df$duration <- NULL
train_indices <- sample(nrow(df), 3000, replace = FALSE)
train_data <- df[train_indices, ]
test_data <- df[-train_indices, ]
```

```r
log.reg <- glm(y ~ ., data = train_data, family = "binomial")
summary(log.reg)
```

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = train_data)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.532e+00  6.762e-01  -2.266 0.023460 *
## age                  1.165e-03  7.799e-03   0.149 0.881227
## jobblue-collar      -3.361e-01  2.658e-01  -1.264 0.206125
## jobentrepreneur     -3.427e-01  4.357e-01  -0.787 0.431494
## jobhousemaid         1.420e-01  4.295e-01   0.331 0.740980
## jobmanagement       -8.876e-02  2.608e-01  -0.340 0.733646
## jobretired           8.166e-01  3.277e-01   2.492 0.012694 *
## jobself-employed     2.279e-01  3.510e-01   0.649 0.516049
## jobservices         -1.271e-01  2.972e-01  -0.428 0.668899
## jobstudent           1.952e-02  4.393e-01   0.044 0.964548
## jobtechnician       -3.508e-01  2.520e-01  -1.392 0.163853
## jobunemployed       -2.173e-01  4.414e-01  -0.492 0.622622
## jobunknown           6.886e-01  5.769e-01   1.194 0.232595
## maritalmarried      -6.758e-01  1.876e-01  -3.602 0.000316 ***
## maritalsingle       -3.767e-01  2.221e-01  -1.696 0.089940 .
## educationsecondary   1.890e-01  2.190e-01   0.863 0.388118
## educationtertiary    3.737e-01  2.552e-01   1.465 0.143018
## educationunknown    -2.916e-01  3.838e-01  -0.760 0.447414
## defaultyes           5.432e-01  4.628e-01   1.174 0.240540
## balance             -1.816e-05  2.518e-05  -0.721 0.470691
## housingyes          -4.363e-02  1.521e-01  -0.287 0.774208
## loanyes             -7.596e-01  2.341e-01  -3.244 0.001177 **
## contacttelephone     8.032e-02  2.351e-01   0.342 0.732656
## contactunknown      -1.014e+00  2.396e-01  -4.232 2.32e-05 ***
## day                  1.347e-02  8.948e-03   1.505 0.132216
## monthaug            -2.617e-01  2.784e-01  -0.940 0.347356
## monthdec             6.847e-01  6.779e-01   1.010 0.312529
## monthfeb             1.522e-01  3.269e-01   0.466 0.641473
## monthjan            -1.174e+00  4.560e-01  -2.575 0.010036 *
## monthjul            -7.291e-01  2.817e-01  -2.588 0.009648 **
## monthjun             5.809e-01  3.290e-01   1.766 0.077443 .
## monthmar             3.920e-01  4.894e-01   0.801 0.423091
## monthmay            -5.635e-01  2.607e-01  -2.161 0.030657 *
## monthnov            -7.694e-01  3.137e-01  -2.452 0.014194 *
## monthoct             1.483e+00  3.723e-01   3.982 6.83e-05 ***
## monthsep             5.177e-01  4.312e-01   1.201 0.229888
## campaign            -8.039e-02  3.179e-02  -2.529 0.011439 *
## pdays                7.256e-04  1.115e-03   0.651 0.515185
## previous             4.686e-02  4.357e-02   1.076 0.282150
## poutcomeother        1.627e-01  3.100e-01   0.525 0.599828
## poutcomesuccess      2.284e+00  3.157e-01   7.235 4.66e-13 ***
## poutcomeunknown      1.781e-01  3.579e-01   0.498 0.618706
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2112.3  on 2999  degrees of freedom
## Residual deviance: 1766.6  on 2958  degrees of freedom
## AIC: 1850.6
##
## Number of Fisher Scoring iterations: 6
```

The Null Deviance (2112.3) represents the deviance of a model with only the intercept, serving as a baseline for comparison. The Residual Deviance (1766.6), significantly lower, indicates that the addition of predictors improves the model's fit. AIC is 1850.6.

A positive coefficient means that as the predictor increases, the likelihood of the outcome also increases, while a negative coefficient indicates a decrease in likelihood. The z-value, calculated as the ratio of the coefficient to its standard error. Higher absolute z-values indicate greater statistical significance, with predictors showing significant contributions to the model having small p-values.

For example, consider the coefficient for Contactunknown (-1.014). This means that if the contact information is unknown, the log-odds of success decrease by 1.014 compared to the baseline category. The corresponding z-value of -4.232 and a p-value less than 0.001 indicate that this effect is highly significant. Similarly, Poutcomesuccess has a positive coefficient of 2.284, suggesting that previous campaign success substantially increases the likelihood of success in the current campaign. Its z-value of 7.235 and very small p-value confirm this predictor's strong significance.

In contrast, the variable Age has a very small coefficient (0.001165) and a low z-value (0.149), indicating it has little impact on the outcome, as its effect is not statistically significant.

Six iterations for Fisher Scoring suggest the model converged without computational issues, indicating stability in parameter estimation.

b)

```r
predicted_probs <- predict(log.reg, newdata = test_data, type = "response")
predicted_labels <- factor(ifelse(predicted_probs >= 0.5, 1, 0), levels = c(0, 1))

# Create the confusion matrix
confusion <- table(Predicted = predicted_labels, Actual = test_data$y)
print(confusion)
```

```
##          Actual
## Predicted   no  yes
##         0 1320  155
##         1   18   28
```

```r
misclassification_rate_0 <- confusion[2, 1] / sum(confusion[, 1])

misclassification_rate_1 <- confusion[1, 2] / sum(confusion[, 2])

cat("Misclassification Rate (Class 0):", misclassification_rate_0, "\n")
```

```
## Misclassification Rate (Class 0): 0.01345291
```

```r
cat("Misclassification Rate (Class 1):", misclassification_rate_1, "\n")
```

```
## Misclassification Rate (Class 1): 0.8469945
```

```r
sensitivity_1 <- 1 - misclassification_rate_1  # Sensitivity (Class 1 Recall)
specificity_0 <- 1 - misclassification_rate_0  # Specificity (Class 0 Recall)

balanced_accuracy <- (sensitivity_1 + specificity_0) / 2
cat("Balanced Accuracy:", balanced_accuracy, "\n")
```

```
## Balanced Accuracy: 0.5697763
```

c)

```r
# Get class proportions
class_counts <- table(train_data$y)

weights <- ifelse(train_data$y == "no",
                  1 / class_counts["no"],
                  1 / class_counts["yes"])
weights <- weights / mean(weights)

summary(weights)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5635  0.5635  0.5635  1.0000  0.5635  4.4379
```

```r
log.reg_weighted <- glm(y ~ ., data = train_data, family = "binomial", weights = weights)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```r
predicted_probs <- predict(log.reg_weighted, newdata = test_data, type = "response")
predicted_labels <- factor(ifelse(predicted_probs >= 0.5, 1, 0), levels = c(0, 1))

confusion <- table(Predicted = predicted_labels, Actual = test_data$y)
print(confusion)
```

```
##          Actual
## Predicted  no yes
##         0 986  82
##         1 352 101
```

```r
misclassification_rate_0 <- confusion[2, 1] / sum(confusion[, 1])
misclassification_rate_1 <- confusion[1, 2] / sum(confusion[, 2])

sensitivity_1 <- 1 - misclassification_rate_1  # Sensitivity (Class 1 Recall)
specificity_0 <- 1 - misclassification_rate_0  # Specificity (Class 0 Recall)

balanced_accuracy <- (sensitivity_1 + specificity_0) / 2
cat("Balanced Accuracy:", balanced_accuracy, "\n")
```

```
## Balanced Accuracy: 0.6444167
```

How do we have to select the weights, and what is the resulting balanced accuracy?:

To handle imbalanced data in logistic regression, weights are used to give more importance to observations from the minority class during model training. So we chose the weights based on the inverse of the class sizes, so the smaller class (yes) gets a higher weight, and the larger class (no) gets a smaller weight. This ensures the model focuses more on the minority class, balancing its influence

Indeed the resulting balanced accuracy increased to : **0.6444167**

d)

```r
full_model <- glm(y ~ ., data = train_data, family = "binomial", weights = weights)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```r
simplified_model <- step(full_model, direction = "both")
```

```
## Start:  AIC=4623.55
## y ~ age + job + marital + education + default + balance + housing +
##     loan + contact + day + month + campaign + pdays + previous +
##     poutcome
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
##             Df Deviance    AIC
## - housing    1   3411.0 4621.6
## - age        1   3411.7 4622.3
## - balance    1   3412.0 4622.6
## - day        1   3412.6 4623.2
## <none>           3411.0 4623.5
## - pdays      1   3413.0 4623.6
## - previous   1   3415.2 4625.8
## - default    1   3415.4 4625.9
## - education  3   3421.7 4628.3
## - campaign   1   3429.3 4639.8
```

```
## - job       11   3452.9 4643.5
## - marital    2   3439.6 4648.1
## - loan       1   3450.7 4661.3
## - contact    2   3453.1 4661.7
## - poutcome   3   3523.4 4730.0
## - month     11   3565.6 4756.2

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

##
## Step:  AIC=4621.72
## y ~ age + job + marital + education + default + balance + loan +
##     contact + day + month + campaign + pdays + previous + poutcome

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

##             Df Deviance    AIC
## - age        1   3411.9 4620.6
## - balance    1   3412.1 4620.7
## - day        1   3412.7 4621.4
## <none>           3411.0 4621.7
## - pdays      1   3413.0 4621.7
## + housing    1   3411.0 4623.7
## - previous   1   3415.3 4624.0
## - default    1   3415.5 4624.2
## - education  3   3421.7 4626.4
## - campaign   1   3429.5 4638.2
## - job       11   3454.2 4642.9
## - marital    2   3439.9 4646.6
## - loan       1   3450.7 4659.4
## - contact    2   3454.4 4661.1
## - poutcome   3   3524.7 4729.4
## - month     11   3572.6 4761.3

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

##
```

```
## Step:  AIC=4620.82
## y ~ job + marital + education + default + balance + loan + contact +
##     day + month + campaign + pdays + previous + poutcome

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

##             Df Deviance    AIC
## - balance    1   3412.8 4619.7
## - day        1   3413.6 4620.5
## - pdays      1   3413.9 4620.8
## <none>           3411.9 4620.8
## + age        1   3411.0 4622.0
## + housing    1   3411.7 4622.7
## - previous   1   3416.0 4623.0
## - default    1   3416.5 4623.4
## - education  3   3421.8 4624.8
## - campaign   1   3430.6 4637.5
## - marital    2   3440.0 4645.0
## - loan       1   3451.8 4658.8
## - contact    2   3455.1 4660.1
## - job       11   3474.5 4661.5
## - poutcome   3   3526.1 4729.1
## - month     11   3574.6 4761.6

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

##
## Step:  AIC=4619.82
## y ~ job + marital + education + default + loan + contact + day +
##     month + campaign + pdays + previous + poutcome

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
##             Df Deviance    AIC
## - day        1   3414.4 4619.5
## - pdays      1   3414.7 4619.8
## <none>           3412.8 4619.8
## + balance    1   3411.9 4620.9
## + age        1   3412.1 4621.1
## + housing    1   3412.6 4621.7
## - previous   1   3416.8 4621.8
## - default    1   3417.7 4622.8
## - education  3   3422.9 4624.0
## - campaign   1   3431.4 4636.5
## - marital    2   3440.9 4644.0
## - loan       1   3452.1 4657.2
## - contact    2   3456.0 4659.1
## - job       11   3474.9 4659.9
## - poutcome   3   3526.8 4727.9
## - month     11   3574.7 4759.8
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
##
## Step:  AIC=4621.65
## y ~ job + marital + education + default + loan + contact + month +
##     campaign + pdays + previous + poutcome
```

```
summary(simplified_model)
```

```
##
## Call:
## glm(formula = y ~ job + marital + education + default + loan +
##     contact + month + campaign + pdays + previous + poutcome,
##     family = "binomial", data = train_data, weights = weights)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         0.6045525  0.3518484   1.718 0.085757 .
## jobblue-collar     -0.3605362  0.1691950  -2.131 0.033098 *
## jobentrepreneur    -0.4572293  0.2705783  -1.690 0.091062 .
## jobhousemaid        0.2398215  0.2872980   0.835 0.403859
## jobmanagement      -0.1459072  0.1764013  -0.827 0.408162
## jobretired          0.9430943  0.2103196   4.484 7.32e-06 ***
## jobself-employed    0.1176597  0.2341912   0.502 0.615380
## jobservices        -0.2483943  0.1942537  -1.279 0.200999
```

```
## jobstudent         -0.0604192  0.3000705  -0.201 0.840425
## jobtechnician      -0.3109582  0.1655145  -1.879 0.060280 .
## jobunemployed      -0.3744129  0.2892049  -1.295 0.195448
## jobunknown          0.4196217  0.4561570   0.920 0.357622
## maritalmarried     -0.5766340  0.1290274  -4.469 7.86e-06 ***
## maritalsingle      -0.1772924  0.1434898  -1.236 0.216617
## educationsecondary  0.0980165  0.1366247   0.717 0.473118
## educationtertiary   0.2821812  0.1606958   1.756 0.079089 .
## educationunknown   -0.4547231  0.2570616  -1.769 0.076906 .
## defaultyes          0.6938237  0.3142865   2.208 0.027271 *
## loanyes            -0.8633982  0.1418661  -6.086 1.16e-09 ***
## contacttelephone    0.0099092  0.1653857   0.060 0.952223
## contactunknown     -0.8760864  0.1377901  -6.358 2.04e-10 ***
## monthaug           -0.3039659  0.1809823  -1.680 0.093048 .
## monthdec            0.7596053  0.5982230   1.270 0.204167
## monthfeb            0.0087635  0.2177176   0.040 0.967892
## monthjan           -1.2956671  0.2993642  -4.328 1.50e-05 ***
## monthjul           -0.6516148  0.1857264  -3.508 0.000451 ***
## monthjun            0.2655794  0.2058748   1.290 0.197049
## monthmar            0.2910334  0.3927285   0.741 0.458660
## monthmay           -0.6082154  0.1746176  -3.483 0.000496 ***
## monthnov           -0.7261550  0.2047941  -3.546 0.000391 ***
## monthoct            1.7392671  0.3351500   5.190 2.11e-07 ***
## monthsep            0.5406562  0.3414463   1.583 0.113324
## campaign           -0.0733619  0.0187820  -3.906 9.39e-05 ***
## pdays               0.0009928  0.0007971   1.246 0.212938
## previous            0.0723716  0.0379192   1.909 0.056317 .
## poutcomeother       0.1888188  0.2181903   0.865 0.386827
## poutcomesuccess     2.4470433  0.2942977   8.315  < 2e-16 ***
## poutcomeunknown     0.3138244  0.2651100   1.184 0.236511
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4158.9  on 2999  degrees of freedom
## Residual deviance: 3414.4  on 2962  degrees of freedom
## AIC: 4621.6
##
## Number of Fisher Scoring iterations: 5
```

```
predicted_probs <- predict(simplified_model, newdata = test_data, type = "response")

predicted_labels <- factor(ifelse(predicted_probs >= 0.5, 1, 0), levels = c(0, 1))

confusion <- table(Predicted = predicted_labels, Actual = test_data$y)
print(confusion)
```

```
##          Actual
## Predicted  no yes
##         0 962  79
##         1 376 104
```

```r
misclassification_rate_0 <- confusion[2, 1] / sum(confusion[, 1])
misclassification_rate_1 <- confusion[1, 2] / sum(confusion[, 2])

sensitivity_1 <- 1 - misclassification_rate_1  # Sensitivity (Class 1 Recall)
specificity_0 <- 1 - misclassification_rate_0  # Specificity (Class 0 Recall)

balanced_accuracy <- (sensitivity_1 + specificity_0) / 2
cat("Balanced Accuracy:", balanced_accuracy, "\n")
```

```
## Balanced Accuracy: 0.6436448
```

Does this also lead to an improvement of the balanced accuracy?:

The performance did not improve; in fact, it became 0.001 worse than before. However, the model is now significantly simpler, achieving nearly the same accuracy with only 11 predictors instead of 16!

2)

```r
library("ISLR")
data(Khan)
str(Khan)
```

```
## List of 4
##  $ xtrain: num [1:63, 1:2308] 0.7733 -0.0782 -0.0845 0.9656 0.0757 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:63] "V1" "V2" "V3" "V4" ...
##   .. ..$ : NULL
##  $ xtest : num [1:20, 1:2308] 0.14 1.164 0.841 0.685 -1.956 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:20] "V1" "V2" "V4" "V6" ...
##   .. ..$ : NULL
##  $ ytrain: num [1:63] 2 2 2 2 2 2 2 2 2 2 ...
##  $ ytest : num [1:20] 3 2 4 2 1 3 4 2 3 1 ...
```

a)

Why would LDA or QDA not work here?:

The dataset contains 2308 gene expression features for a limited number of tissue samples. LDA and QDA rely on the estimation of covariance matrices, which require the number of observations to be much larger than the number of features. When p>n, the covariance matrix becomes singular, making these methods mathematically infeasible.

QDA is even more sensitive to high dimensionality because it estimates a separate covariance matrix for each class, requiring a greater number of observations per class.

Would RDA work?: RDA applies a shrinkage approach to the covariance matrix. By regularizing the covariance matrix, RDA avoids singularity issues even when p>n. But the lambda needs to be picked carefully. When =1, RDA becomes equivalent to LDA, which assumes a single pooled covariance matrix for all classes. When =0 =0, RDA becomes equivalent to QDA, estimating a separate covariance matrix for each class. So we run in the same issues like we would use LDA and RDA by their own. RDA can handle high-dimensional datasets effectively only if is correctly set.

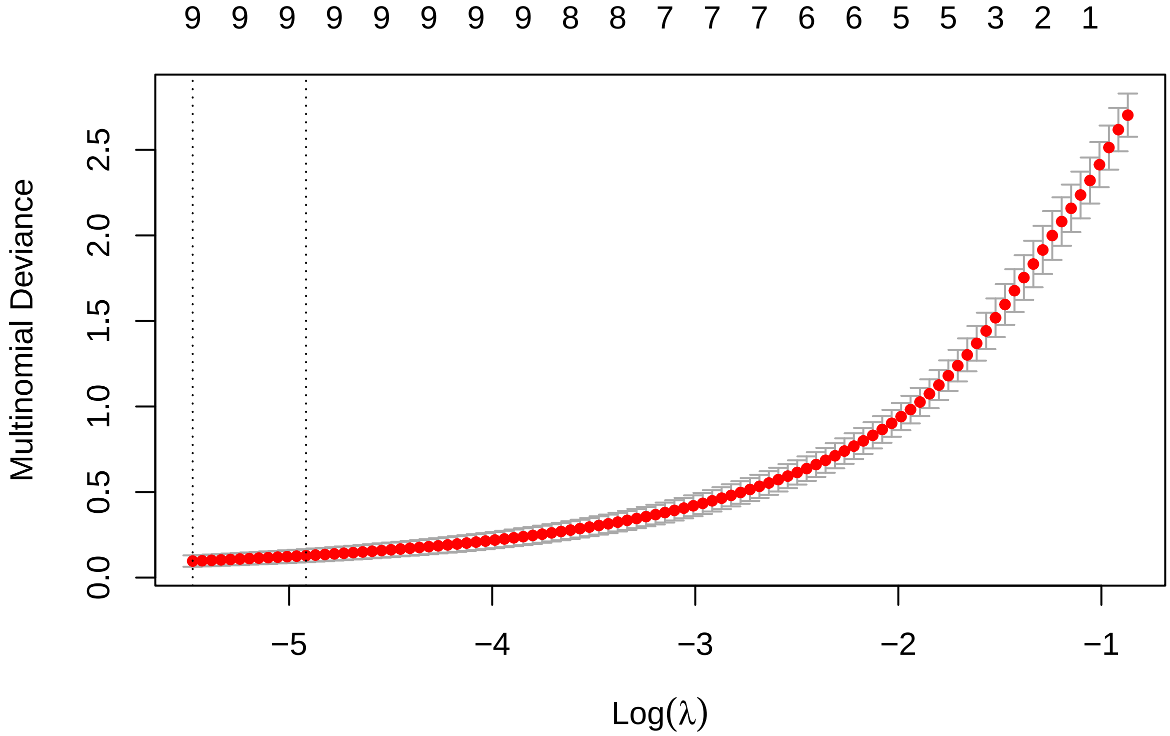
```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.4.2

## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
ytrain <- as.factor(Khan$ytrain)
cv_model <- cv.glmnet(x = Khan$xtrain, y = ytrain, family = "multinomial")
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```r
plot(cv_model)
```



What do you conclude?:

X-axis Represents the logarithm of the regularization parameter . Moving to the left corresponds to smaller values of , meaning less regularization and a more complex model. Y-axis represents the cross-validated multinomial deviance. Lower values are better.

First dashed line: This is the that minimizes the cross-validated multinomial deviance and balances complexity with performance. This provides the best model on the training data

Seconded dashed line: Is the with 1SE from the minimal lambda (first dashed line).It reduces a small amount of performance. This is often preferred in practice as it avoids overfitting and enhances model generalization.

What is the objective function to be minimized?:

For multinomial logistic regression with regularization, glmnet minimizes the following objective function:

Objective Function=Loss Function + Penalty Term

The Loss functio is a Multinomial Negative Log-Likelihood. It calculates the log probability of the predicted class for each observation. The goal is to maximize the likelihood of the true classes. By minimizing the negative log-likelihood, the model is encouraged to assign high probabilities to the correct classes

The penalty term is a value ( ) to prevent overfitting and reduce complexity. By adjusting , the model trades off between these two goals:

Small : The model prioritizes fitting the data well, leading to complex models. Large : The model prioritizes simplicity by heavily penalizing large coefficients, leading to simpler models.

glmnet use elastic net regularization, which combines L1 (lasso) and L2 (ridge) penalties. By default, glmnet uses a mixing parameter =1, which corresponds to lasso regularization.

c)

```
coefficients <- coef(cv_model, s = "lambda.1se")

get_non_zero_coefficients <- function(coef_matrix) {
  non_zero <- which(as.matrix(coef_matrix) != 0, arr.ind = TRUE)

  data.frame(
    variable = rownames(coef_matrix)[non_zero[, 1]],
    coefficient = coef_matrix[non_zero]
  )
}

non_zero_coefficients_list <- lapply(coefficients, get_non_zero_coefficients)
non_zero_coefficients_list
```

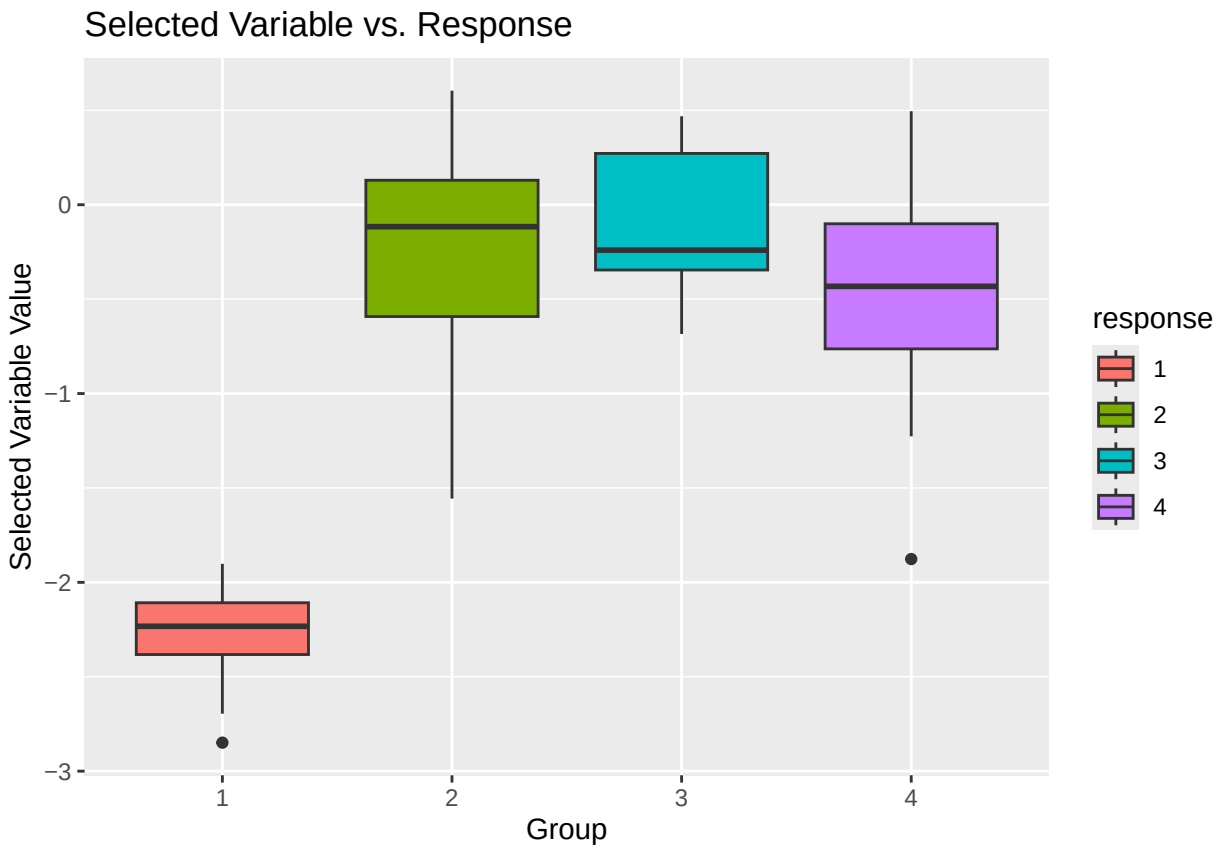
```
## $`1`
##       variable coefficient
## 1  (Intercept) -1.55718883
## 2           V1 -0.17289634
## 3         V123  0.26006234
## 4         V589  0.34082349
## 5         V836  0.31074198
## 6         V846  0.10370715
## 7        V1066 -0.03359345
## 8        V1387  0.13589520
```

```
## 9        V1427 -0.53149218
## 10       V2022 -0.27115095
## 11       V2198 -0.23791675
##
## $`2`
##      variable coefficient
## 1 (Intercept)  -0.8334814
## 2        V246   0.3334059
## 3        V545   0.7038725
## 4       V1319   0.0847777
## 5       V1389   0.7204423
## 6       V1954   0.6834138
## 7       V2050  -0.3967467
##
## $`3`
##      variable coefficient
## 1 (Intercept)  0.92201563
## 2        V255  0.73199424
## 3        V575  0.32110680
## 4        V695  0.21464860
## 5        V742  0.32159946
## 6        V842 -1.21971413
## 7        V879  0.06934387
## 8       V1764  0.04081013
## 9       V1776  0.13432319
##
## $`4`
##       variable coefficient
## 1  (Intercept) 1.468654626
## 2         V174 0.128268335
## 3         V509 0.109465118
## 4         V554 0.003750874
## 5         V910 0.116243327
## 6        V1003 0.550139921
## 7        V1055 0.302283377
## 8        V1105 0.217797522
## 9        V1207 0.286976353
## 10       V1723 0.116727847
## 11       V1955 0.970565002
## 12       V2046 0.324859330
```

d)

```
library(ggplot2)
selected_variable <- Khan$xtrain[, 1427]
response_variable <- Khan$ytrain

data <- data.frame(
variable = selected_variable,
response = factor(response_variable))
ggplot(data, aes(x = response, y = variable, fill = response)) +
geom_boxplot() +
```

```
labs(title = "Selected Variable vs. Response",
x = "Group",
y = "Selected Variable Value")
```



The boxplot shows the distribution of the selected variable's values across the response groups. For the first group (response = 1), the values are notably lower compared to the other groups. In contrast, groups 2, 3, and 4 have higher medians and more overlap, suggesting a distinct difference in the variable's behavior for group 1 compared to the others.

e)

```
y_probs <- predict(cv_model, newx = Khan$xtest, s = "lambda.1se", type = "response")
test_pred <- apply(y_probs, 1, which.max)
predictions <- factor(test_pred, labels = c("Group 1", "Group 2", "Group 3", "Group 4"))
confusion_table <- table(
  Predicted = predictions,
  Actual = factor(Khan$ytest, labels = c("Group 1", "Group 2", "Group 3", "Group 4"))
)

print(confusion_table)
```

```
##           Actual
## Predicted Group 1 Group 2 Group 3 Group 4
##   Group 1       3       0       0       0
```

```
##   Group 2       0       6       0       0
##   Group 3       0       0       6       0
##   Group 4       0       0       0       5
```

All predictions are correct. So the misclassification rate will be 0, as we can see:

```
incorrect_predictions <- sum(confusion_table) - sum(diag(confusion_table))
total_predictions <- sum(confusion_table)
misclassification_rate <- incorrect_predictions / total_predictions
print(paste("Misclassification Rate:", round(misclassification_rate, 4)))
```

```
## [1] "Misclassification Rate: 0"
```