# Exercise 7

Advanced Methods for Regression and Classification

Rita Selimi

03/12/2024

## Task 1

### Load and Explore the Data

```r
# Load the data
d <- read.csv2("bank.csv")

# Exclude the variable 'duration'
d <- d[, !colnames(d) %in% "duration"]

# Convert target variable to a factor
d$y <- as.factor(d$y)

# Check class distribution
table(d$y)
```

```
##
##   no  yes
## 4000  521
```

```r
head(d)
```

```
##   age         job marital education default balance housing loan  contact day
## 1  30  unemployed married   primary      no    1787      no   no cellular  19
## 2  33    services married secondary      no    4789     yes  yes cellular  11
## 3  35  management  single  tertiary      no    1350     yes   no cellular  16
## 4  30  management married  tertiary      no    1476     yes  yes  unknown   3
## 5  59 blue-collar married secondary      no       0     yes   no  unknown   5
## 6  35  management  single  tertiary      no     747      no   no cellular  23
##   month campaign pdays previous poutcome  y
## 1   oct        1    -1        0  unknown no
## 2   may        1   339        4  failure no
## 3   apr        1   330        1  failure no
## 4   jun        4    -1        0  unknown no
## 5   may        1    -1        0  unknown no
## 6   feb        2   176        3  failure no
```

```r
summary(d)
```

```
##       age            job              marital            education
##  Min.   :19.00   Length:4521        Length:4521        Length:4521
##  1st Qu.:33.00   Class :character   Class :character   Class :character
##  Median :39.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :41.17
##  3rd Qu.:49.00
##  Max.   :87.00
##    default             balance        housing              loan
##  Length:4521        Min.   :-3313   Length:4521        Length:4521
##  Class :character   1st Qu.:   69   Class :character   Class :character
##  Mode  :character   Median :  444   Mode  :character   Mode  :character
##                     Mean   : 1423
##                     3rd Qu.: 1480
##                     Max.   :71188
##    contact               day          month              campaign
##  Length:4521        Min.   : 1.00   Length:4521        Min.   : 1.000
##  Class :character   1st Qu.: 9.00   Class :character   1st Qu.: 1.000
##  Mode  :character   Median :16.00   Mode  :character   Median : 2.000
##                     Mean   :15.92                      Mean   : 2.794
##                     3rd Qu.:21.00                      3rd Qu.: 3.000
##                     Max.   :31.00                      Max.   :50.000
##      pdays           previous         poutcome            y
##  Min.   : -1.00   Min.   : 0.0000   Length:4521        no :4000
##  1st Qu.: -1.00   1st Qu.: 0.0000   Class :character   yes: 521
##  Median : -1.00   Median : 0.0000   Mode  :character
##  Mean   : 39.77   Mean   : 0.5426
##  3rd Qu.: -1.00   3rd Qu.: 0.0000
##  Max.   :871.00   Max.   :25.0000
```

The dataset contains information about direct marketing campaigns conducted by a Portuguese bank to predict whether clients subscribe to a term deposit (binary target `y`, with 4000 "no" and 521 "yes"). It includes 16 variables, such as client demographics (e.g., `age`, `job`, `marital`), financial data (`balance`, `housing`, `loan`), and campaign details (`contact`, `campaign`, `pdays`). The dataset is heavily imbalanced, with many more "no" responses, and key variables like `balance`, `previous`, and `pdays` may strongly influence predictions.

## (a) Logistic Regression on Training Set

```r
set.seed(123)

# Randomly select 3000 observations for training
train_indices <- sample(nrow(d), 3000)
train_data <- d[train_indices, ]
test_data <- d[-train_indices, ]

# Fit logistic regression model
logit_model <- glm(y ~ ., data = train_data, family = "binomial")

# Summary of the model
summary(logit_model)
```

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = train_data)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -8.816e-01  6.523e-01  -1.351 0.176562
## age                -5.445e-04  7.694e-03  -0.071 0.943588
## jobblue-collar     -1.524e-01  2.692e-01  -0.566 0.571362
## jobentrepreneur     3.536e-01  3.841e-01   0.920 0.357346
## jobhousemaid        2.691e-01  4.211e-01   0.639 0.522770
## jobmanagement       2.675e-01  2.650e-01   1.009 0.312761
## jobretired          9.708e-01  3.353e-01   2.895 0.003787 **
## jobself-employed    6.173e-01  3.502e-01   1.763 0.077981 .
## jobservices        -9.251e-02  3.058e-01  -0.303 0.762240
## jobstudent          4.103e-01  4.494e-01   0.913 0.361182
## jobtechnician      -2.026e-02  2.536e-01  -0.080 0.936349
## jobunemployed      -2.618e-01  4.791e-01  -0.546 0.584800
## jobunknown         -1.375e-01  7.616e-01  -0.181 0.856685
## maritalmarried     -5.276e-01  1.887e-01  -2.795 0.005185 **
## maritalsingle      -2.033e-01  2.175e-01  -0.935 0.350022
## educationsecondary -5.245e-02  2.146e-01  -0.244 0.806967
## educationtertiary  -4.879e-02  2.497e-01  -0.195 0.845091
## educationunknown   -7.828e-01  4.271e-01  -1.833 0.066833 .
## defaultyes          3.800e-01  4.577e-01   0.830 0.406419
## balance            -2.917e-05  1.979e-05  -1.474 0.140479
## housingyes         -2.842e-01  1.495e-01  -1.901 0.057296 .
## loanyes            -5.633e-01  2.139e-01  -2.633 0.008466 **
## contacttelephone   -2.720e-01  2.552e-01  -1.066 0.286584
## contactunknown     -1.105e+00  2.446e-01  -4.517 6.28e-06 ***
## day                 1.349e-02  8.895e-03   1.516 0.129446
## monthaug           -4.812e-01  2.678e-01  -1.796 0.072420 .
## monthdec            6.294e-01  7.742e-01   0.813 0.416212
## monthfeb           -1.096e-01  3.226e-01  -0.340 0.734060
## monthjan           -1.327e+00  4.313e-01  -3.076 0.002101 **
## monthjul           -8.032e-01  2.683e-01  -2.993 0.002759 **
## monthjun            6.482e-02  3.336e-01   0.194 0.845942
## monthmar            1.129e+00  4.371e-01   2.583 0.009789 **
## monthmay           -5.753e-01  2.506e-01  -2.296 0.021684 *
## monthnov           -8.712e-01  2.930e-01  -2.973 0.002946 **
## monthoct            1.236e+00  3.658e-01   3.379 0.000728 ***
## monthsep            1.416e-01  4.784e-01   0.296 0.767256
## campaign           -5.467e-02  2.897e-02  -1.887 0.059102 .
## pdays              -6.234e-04  1.081e-03  -0.577 0.564210
## previous            8.485e-03  4.143e-02   0.205 0.837705
## poutcomeother       5.916e-01  2.947e-01   2.007 0.044720 *
## poutcomesuccess     2.368e+00  3.244e-01   7.299 2.90e-13 ***
## poutcomeunknown    -2.139e-01  3.422e-01  -0.625 0.532019
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2169.5  on 2999  degrees of freedom
## Residual deviance: 1818.6  on 2958  degrees of freedom
## AIC: 1902.6
##
## Number of Fisher Scoring iterations: 6
```

The logistic regression analysis reveals key factors influencing whether clients subscribe to a term deposit. The most significant predictor is the success of previous campaigns (poutcomesuccess), strongly increasing the likelihood of subscription. Other important variables include specific months, such as October (positive effect) and May (negative effect), and certain client characteristics like being retired. Additionally, using unknown contact methods or being married has a significant impact on the outcome. The reduction in deviance from 2153.3 (null model) to 1761.4 (model with predictors) shows that the regression model significantly improves prediction accuracy by using the provided variables, such as age, job, and contact method. The lower AIC value of 1845.4 indicates a good balance between model fit and simplicity.

## (b) Predictions and Evaluation on Test Set

```r
# Predict probabilities for the test set
test_data$predicted_prob <- predict(logit_model, newdata = test_data, type = "response")

# Convert probabilities to class labels (default threshold = 0.5)
test_data$predicted_class <- ifelse(test_data$predicted_prob > 0.5, "yes", "no")

# Create a confusion matrix
conf_matrix <- table(Predicted = test_data$predicted_class, Actual = test_data$y)

# Calculate misclassification rate for each group
misclass_rate_no <- conf_matrix["yes", "no"]/sum(conf_matrix[, "no"])
misclass_rate_yes <- conf_matrix["no", "yes"]/sum(conf_matrix[, "yes"])

# Calculate balanced accuracy
TPR <- conf_matrix["yes", "yes"]/sum(conf_matrix[, "yes"])  # True Positive Rate
TNR <- conf_matrix["no", "no"]/sum(conf_matrix[, "no"])  # True Negative Rate
balanced_accuracy <- (TPR + TNR)/2

# Print results
cat("Misclassification Rate for 'no':", misclass_rate_no, "\n")
```

```
## Misclassification Rate for 'no': 0.02071006
```

```r
cat("Misclassification Rate for 'yes':", misclass_rate_yes, "\n")
```

```
## Misclassification Rate for 'yes': 0.816568
```

```r
cat("Balanced Accuracy:", balanced_accuracy, "\n")
```

```
## Balanced Accuracy: 0.5813609
```

The model is highly biased toward predicting "no", which aligns with the majority class in this imbalanced dataset. While it correctly identifies "no" with high accuracy, it performs poorly for "yes", the minority class. This suggests that further improvements, such as rebalancing the dataset, are needed to improve the model's ability to identify "yes" clients effectively. The balanced accuracy score of 59.15% is slightly better than random guessing (50%), but it's not strong.

## (c) Weighted Logistic Regression

### Assign Weights Inversely Proportional to Class Frequencies

The idea is to give more weight to the minority class (`yes`) and less weight to the majority class (`no`). This is done using the formula:

$$\text{Weight} = \frac{1}{\text{Frequency of the Class}}$$

For example: - If there are 500 `yes` and 3500 `no` observations: - Weight for `yes` $= \frac{1}{500}$ - Weight for `no` $= \frac{1}{3500}$

```r
# Calculate weights
class_weights <- ifelse(train_data$y == "yes", 1/sum(train_data$y == "yes"), 1/sum(train_data$y ==
    "no"))

# Normalize the weights
class_weights <- class_weights * length(train_data$y)

# Fit weighted logistic regression model
weighted_logit_model <- glm(y ~ ., data = train_data, family = "binomial", weights = class_weights)

# Predict probabilities on the test set
test_data$weighted_prob <- predict(weighted_logit_model, newdata = test_data, type = "response")

# Convert probabilities to class labels (default threshold = 0.5)
test_data$weighted_class <- ifelse(test_data$weighted_prob > 0.5, "yes", "no")

# Create a confusion matrix
conf_matrix_weighted <- table(Predicted = test_data$weighted_class, Actual = test_data$y)

# Calculate misclassification rate for each group
misclass_rate_no_weighted <- conf_matrix_weighted["yes", "no"]/sum(conf_matrix_weighted[,
    "no"])
misclass_rate_yes_weighted <- conf_matrix_weighted["no", "yes"]/sum(conf_matrix_weighted[,
    "yes"])

# Calculate balanced accuracy
TPR_weighted <- conf_matrix_weighted["yes", "yes"]/sum(conf_matrix_weighted[, "yes"])
TNR_weighted <- conf_matrix_weighted["no", "no"]/sum(conf_matrix_weighted[, "no"])
balanced_accuracy_weighted <- (TPR_weighted + TNR_weighted)/2

# Print results
cat("Misclassification Rate for 'no' (weighted):", misclass_rate_no_weighted, "\n")
```

```
## Misclassification Rate for 'no' (weighted): 0.2795858
```

```r
cat("Misclassification Rate for 'yes' (weighted):", misclass_rate_yes_weighted, "\n")
```

```
## Misclassification Rate for 'yes' (weighted): 0.3964497
```

```r
cat("Balanced Accuracy (weighted):", balanced_accuracy_weighted, "\n")
```

```
## Balanced Accuracy (weighted): 0.6619822
```

Adding weights to the logistic regression model reduced the misclassification rate for the minority class ("yes") but increased the error rate for the majority class ("no"). The balanced accuracy showed a slight improvement (59.15% → 66.19%).

Normalizing the weights ensures that the total weight matches the dataset size, keeping the model output consistent.

## (d) Stepwise Variable Selection

```r
# Perform stepwise selection on the weighted logistic regression model
simplified_model <- step(weighted_logit_model, direction = "both")
```

```
## Start:  AIC=6730.78
## y ~ age + job + marital + education + default + balance + housing +
##     loan + contact + day + month + campaign + pdays + previous +
##     poutcome
##
##             Df Deviance    AIC
## - age        1   6848.1 6728.9
## - previous   1   6848.4 6729.2
## - default    1   6849.9 6730.7
## <none>           6848.0 6730.8
## - day        1   6850.1 6730.9
## - pdays      1   6851.1 6732.0
## - balance    1   6855.8 6736.6
## - housing    1   6859.7 6740.5
## - education  3   6867.2 6744.0
## - campaign   1   6864.5 6745.3
## - loan       1   6884.0 6764.9
## - marital    2   6899.7 6778.5
## - job       11   6949.2 6810.1
## - contact    2   6973.4 6852.2
## - poutcome   3   7073.7 6950.5
## - month     11   7129.5 6990.3
##
## Step:  AIC=6728.99
## y ~ job + marital + education + default + balance + housing +
##     loan + contact + day + month + campaign + pdays + previous +
##     poutcome
##
##             Df Deviance    AIC
## - previous   1   6848.5 6727.4
## - default    1   6850.0 6728.9
## <none>           6848.1 6729.0
## - day        1   6850.2 6729.1
## - pdays      1   6851.3 6730.2
## + age        1   6848.0 6730.8
## - balance    1   6856.0 6734.9
## - housing    1   6859.7 6738.6
## - education  3   6867.6 6742.5
## - campaign   1   6864.6 6743.5
## - loan       1   6884.2 6763.1
## - marital    2   6902.8 6779.7
## - job       11   6967.0 6825.9
## - contact    2   6974.5 6851.4
## - poutcome   3   7074.8 6949.7
## - month     11   7129.6 6988.5
##
## Step:  AIC=6727.42
## y ~ job + marital + education + default + balance + housing +
##     loan + contact + day + month + campaign + pdays + poutcome
##
##             Df Deviance    AIC
## - default    1   6850.4 6727.3
## <none>           6848.5 6727.4
## - day        1   6850.6 6727.5
## - pdays      1   6852.0 6728.9
## + previous   1   6848.1 6729.0
## + age        1   6848.4 6729.3
## - balance    1   6856.4 6733.3
## - housing    1   6860.1 6737.0
## - education  3   6868.1 6741.0
## - campaign   1   6864.9 6741.8
## - loan       1   6884.5 6761.4
## - marital    2   6903.4 6778.3
## - job       11   6968.0 6824.9
## - contact    2   6975.4 6850.3
## - poutcome   3   7108.7 6981.6
## - month     11   7130.0 6986.9
##
## Step:  AIC=6727.11
## y ~ job + marital + education + balance + housing + loan + contact +
##     day + month + campaign + pdays + poutcome
##
##             Df Deviance    AIC
## <none>           6850.4 6727.1
## - day        1   6852.4 6727.2
## + default    1   6848.5 6727.2
## - pdays      1   6853.8 6728.5
## + previous   1   6850.0 6728.7
## + age        1   6850.3 6729.0
## - balance    1   6858.9 6733.6
## - housing    1   6862.2 6736.9
## - education  3   6869.3 6740.0
## - campaign   1   6867.2 6741.9
## - loan       1   6885.8 6760.5
## - marital    2   6906.5 6779.2
## - job       11   6969.9 6824.6
## - contact    2   6978.0 6850.7
## - poutcome   3   7109.8 6980.5
## - month     11   7130.9 6985.6
```

```r
# Summarize the simplified model
summary(simplified_model)
```

```
##
## Call:
## glm(formula = y ~ job + marital + education + balance + housing +
##     loan + contact + day + month + campaign + pdays + poutcome,
##     family = "binomial", data = train_data, weights = class_weights)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.309e+00  2.568e-01   5.095 3.49e-07 ***
## jobblue-collar      -7.218e-02  1.211e-01  -0.596 0.551307
## jobentrepreneur      4.760e-01  1.750e-01   2.720 0.006532 **
## jobhousemaid         4.788e-01  2.014e-01   2.377 0.017444 *
## jobmanagement        2.987e-01  1.265e-01   2.361 0.018206 *
## jobretired           1.189e+00  1.558e-01   7.632 2.32e-14 ***
## jobself-employed     6.627e-01  1.673e-01   3.962 7.44e-05 ***
## jobservices         -1.291e-01  1.401e-01  -0.921 0.357125
## jobstudent           4.899e-01  2.235e-01   2.192 0.028367 *
## jobtechnician        5.946e-02  1.185e-01   0.502 0.615765
## jobunemployed       -2.632e-01  2.188e-01  -1.203 0.229066
## jobunknown          -1.398e-02  3.987e-01  -0.035 0.972023
## maritalmarried      -5.259e-01  9.085e-02  -5.789 7.07e-09 ***
## maritalsingle       -1.101e-01  9.989e-02  -1.102 0.270526
## educationsecondary -9.674e-02  9.770e-02  -0.990 0.322093
## educationtertiary  -5.516e-02  1.142e-01  -0.483 0.629110
## educationunknown    -7.887e-01  1.914e-01  -4.121 3.78e-05 ***
## balance             -2.996e-05  1.018e-05  -2.944 0.003243 **
## housingyes          -2.384e-01  6.932e-02  -3.438 0.000585 ***
## loanyes             -5.424e-01  9.257e-02  -5.860 4.64e-09 ***
## contacttelephone    -4.365e-01  1.277e-01  -3.419 0.000629 ***
## contactunknown      -1.084e+00  9.978e-02 -10.862  < 2e-16 ***
## day                  5.969e-03  4.176e-03   1.429 0.152887
## monthaug            -4.639e-01  1.302e-01  -3.562 0.000368 ***
## monthdec             5.702e-01  5.301e-01   1.076 0.282096
## monthfeb            -2.177e-01  1.604e-01  -1.358 0.174619
## monthjan            -1.408e+00  2.031e-01  -6.933 4.12e-12 ***
## monthjul            -7.649e-01  1.293e-01  -5.914 3.34e-09 ***
## monthjun            -1.245e-01  1.562e-01  -0.797 0.425444
## monthmar             1.228e+00  2.765e-01   4.443 8.86e-06 ***
## monthmay            -5.473e-01  1.235e-01  -4.432 9.34e-06 ***
## monthnov            -9.539e-01  1.417e-01  -6.732 1.67e-11 ***
## monthoct             1.394e+00  2.296e-01   6.074 1.25e-09 ***
## monthsep             5.846e-02  2.700e-01   0.217 0.828589
## campaign            -4.975e-02  1.260e-02  -3.950 7.82e-05 ***
## pdays               -1.044e-03  5.637e-04  -1.851 0.064123 .
## poutcomeother        5.830e-01  1.512e-01   3.855 0.000116 ***
## poutcomesuccess      2.384e+00  2.176e-01  10.958  < 2e-16 ***
## poutcomeunknown     -3.784e-01  1.617e-01  -2.340 0.019308 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8317.8  on 2999  degrees of freedom
## Residual deviance: 6850.4  on 2961  degrees of freedom
## AIC: 6727.1
##
## Number of Fisher Scoring iterations: 5
```

```r
# Predict probabilities using the simplified model on the test set
test_data$simplified_prob <- predict(simplified_model, newdata = test_data, type = "response")

# Convert probabilities to class labels (default threshold = 0.5)
test_data$simplified_class <- ifelse(test_data$simplified_prob > 0.5, "yes", "no")

# Ensure confusion matrix includes both 'no' and 'yes'
conf_matrix_simplified <- table(Predicted = factor(test_data$simplified_class, levels = c("no",
    "yes")), Actual = factor(test_data$y, levels = c("no", "yes")))

# Re-run misclassification rate calculations
misclass_rate_no_simplified <- conf_matrix_simplified["yes", "no"]/sum(conf_matrix_simplified[,
    "no"])
misclass_rate_yes_simplified <- conf_matrix_simplified["no", "yes"]/sum(conf_matrix_simplified[,
    "yes"])

# Calculate balanced accuracy
TPR_simplified <- conf_matrix_simplified["yes", "yes"]/sum(conf_matrix_simplified[,
    "yes"])
TNR_simplified <- conf_matrix_simplified["no", "no"]/sum(conf_matrix_simplified[,
    "no"])
balanced_accuracy_simplified <- (TPR_simplified + TNR_simplified)/2

# Print results
cat("Misclassification Rate for 'no' (simplified):", misclass_rate_no_simplified,
    "\n")
```

```
## Misclassification Rate for 'no' (simplified): 0.2773669
```

```r
cat("Misclassification Rate for 'yes' (simplified):", misclass_rate_yes_simplified,
    "\n")
```

```
## Misclassification Rate for 'yes' (simplified): 0.3964497
```

```r
cat("Balanced Accuracy (simplified):", balanced_accuracy_simplified, "\n")
```

```
## Balanced Accuracy (simplified): 0.6630917
```

Yes, **stepwise selection leads to a slight improvement in balanced accuracy**, increasing from **66.20% (weighted model)** to **66.31% (simplified model)**. This improvement, while minor, indicates that removing less impactful predictors helped simplify the model without sacrificing performance.

- Stepwise variable selection slightly improved balanced accuracy (from 66.20% to 66.31%), while keeping the misclassification rates for "yes" and "no" groups nearly the same.
- The simplified model is more efficient, as it uses fewer predictors while maintaining similar performance to the weighted model.

## Task 1

### Load Data and Libraries

```r
# Load required libraries
library(ISLR)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
# Load the dataset
data(Khan)

# Structure of the dataset
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 ...
```

The `Khan` dataset contains gene expression measurements for tissue samples corresponding to four distinct types of small round blue cell tumors. It includes. **Training Data (`xtrain`)**: 63 observations with 2308 gene expression measurements per sample. **Test Data (`xtest`)**: 20 observations with the same 2308 gene expression measurements. **Response Variables**: `ytrain` and `ytest` represent the tumor types for the training and test samples, respectively, encoded as numerical labels (1, 2, 3, 4).

### (a) Why would LDA or QDA not work? Would RDA work?

**Why LDA or QDA Would Not Work:**

1. **High-Dimensional Data**:
    - The dataset has 2308 predictors (genes) and only 63 training observations. Both **LDA (Linear Discriminant Analysis)** and **QDA (Quadratic Discriminant Analysis)** rely on estimating the covariance matrix of the predictors. However, the covariance matrix is $2308 \times 2308$, and its estimation requires more observations than predictors. With only 63 observations, the covariance matrix becomes singular (not invertible), making LDA and QDA inapplicable.
2. **Overfitting Risk**:
    - Even if these methods could handle the dimensionality, the small sample size relative to the large number of predictors would lead to severe overfitting, reducing their ability to generalize to new data.

**Would RDA Work?**

**Regularized Discriminant Analysis (RDA)** introduces regularization to address the overfitting problem by shrinking the covariance matrix. Unlike LDA or QDA, RDA works by combining the identity matrix with the covariance matrix, effectively reducing the complexity of the model. using a tuning parameter to balance between LDA (shared covariance matrix) and QDA (individual covariance matrices for each class).

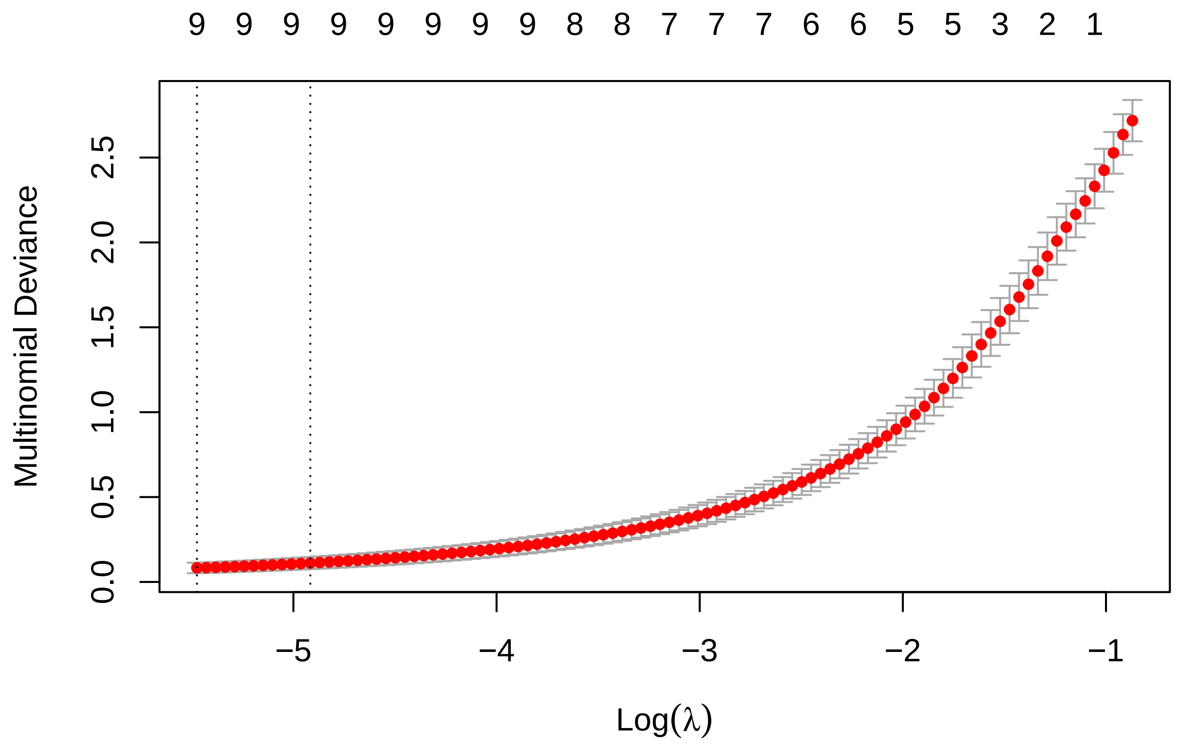
Thus, **RDA could work here** because regularization mitigates the issues of high dimensionality and small sample size. However, it might still face challenges due to the extreme dimensionality of this dataset, and methods like penalized regression (`glmnet`) or dimension reduction may perform better.

## (b) Build a Model Using `cv.glmnet`

```r
# Convert response to a factor
ytrain <- as.factor(Khan$ytrain)

# Fit multinomial model using cross-validation
fit_glmnet <- cv.glmnet(Khan$xtrain, ytrain, family = "multinomial")

# Plot cross-validation results
plot(fit_glmnet)
```



```r
# Display the values of lambda.min and lambda.1se
cat("Lambda.min:", fit_glmnet$lambda.min, "\n")
```

```
## Lambda.min: 0.004190343
```

```r
cat("Lambda.1se:", fit_glmnet$lambda.1se, "\n")
```

```
## Lambda.1se: 0.007322743
```

1. **Mean Multinomial Deviance**:
   - The Y-axis represents the **multinomial deviance**, which measures how well the model fits the data. Lower values indicate better fit.
   - The red dots are the mean deviance for each value of $\lambda$ (the regularization parameter), and the vertical bars show the error around these estimates.
2. **Log($\lambda$)**:
   - The X-axis shows different values of the penalty parameter $\lambda$ (on a log scale). Larger $\lambda$ values correspond to more regularization (simpler models), while smaller $\lambda$ values lead to less regularization (more complex models).
3. **Two Vertical Lines**:
   - The **left dashed line** represents `lambda.min`, the value of $\lambda$ that minimizes the mean deviance (best model fit).
   - The **right dashed line** represents `lambda.1se`, the largest $\lambda$ value within one standard error of `lambda.min`. This gives a simpler model with slightly worse fit but better generalization.

#### What is the Objective Function to Be Minimized?

The objective function being minimized is the **multinomial deviance**, which measures how far off the model's predicted probabilities are from the actual class labels.

Here's what it means:

- **If the model predicts probabilities close to the true class (e.g., high for the correct class, low for others)**, the deviance is small (good).

- **If the model predicts wrong probabilities**, the deviance is large (bad).

The formula is just a way of adding up how "wrong" the model's guesses are for all observations and all classes. The goal is to find the best coefficients and $\lambda$ (penalty) that make the deviance as small as possible, balancing good predictions with avoiding overfitting.

## (c) Which Variables Contribute to the Model?

```r
# Extract coefficients for all groups at lambda.1se
coefficients <- coef(fit_glmnet, s = "lambda.1se")

# Display coefficients for each group
for (group in names(coefficients)) {
    cat("\nCoefficients for group:", group, "\n")
    # print(as.matrix(coefficients[[group]]))
}
```

```
##
## Coefficients for group: 1
##
## Coefficients for group: 2
##
## Coefficients for group: 3
##
## Coefficients for group: 4
```

```r
# Identify relevant coefficients (non-zero) for each group
relevant_coefficients <- lapply(coefficients, function(coef_matrix) {
    coef_matrix <- as.matrix(coef_matrix)
    relevant <- coef_matrix[coef_matrix != 0, , drop = FALSE]
    return(relevant)
})

# Print relevant coefficients
cat("\nRelevant coefficients for each group:\n")
```

```
##
## Relevant coefficients for each group:
```

```r
print(relevant_coefficients)
```

```
## $`1`
##                       1
## (Intercept) -1.55718883
## V1          -0.17289634
## V123         0.26006234
## V589         0.34082349
## V836         0.31074198
## V846         0.10370715
## V1066       -0.03359345
## V1387        0.13589520
## V1427       -0.53149218
## V2022       -0.27115095
## V2198       -0.23791675
##
## $`2`
##                      1
## (Intercept) -0.8334814
## V246         0.3334059
## V545         0.7038725
## V1319        0.0847777
## V1389        0.7204423
## V1954        0.6834138
## V2050       -0.3967467
##
## $`3`
##                       1
## (Intercept)  0.92201563
## V255         0.73199424
## V575         0.32110680
## V695         0.21464860
## V742         0.32159946
## V842        -1.21971413
## V879         0.06934387
## V1764        0.04081013
## V1776        0.13432319
##
## $`4`
##                       1
## (Intercept) 1.468654626
## V174        0.128268335
## V509        0.109465118
## V554        0.003750874
## V910        0.116243327
## V1003       0.550139921
## V1055       0.302283377
## V1105       0.217797522
## V1207       0.286976353
## V1723       0.116727847
## V1955       0.970565002
## V2046       0.324859330
```

**Which Variables Contribute to the Model?** Variables with non-zero coefficients are those that contribute to the prediction of the response variable. These variables are the ones that the model identifies as significant predictors for the respective groups.

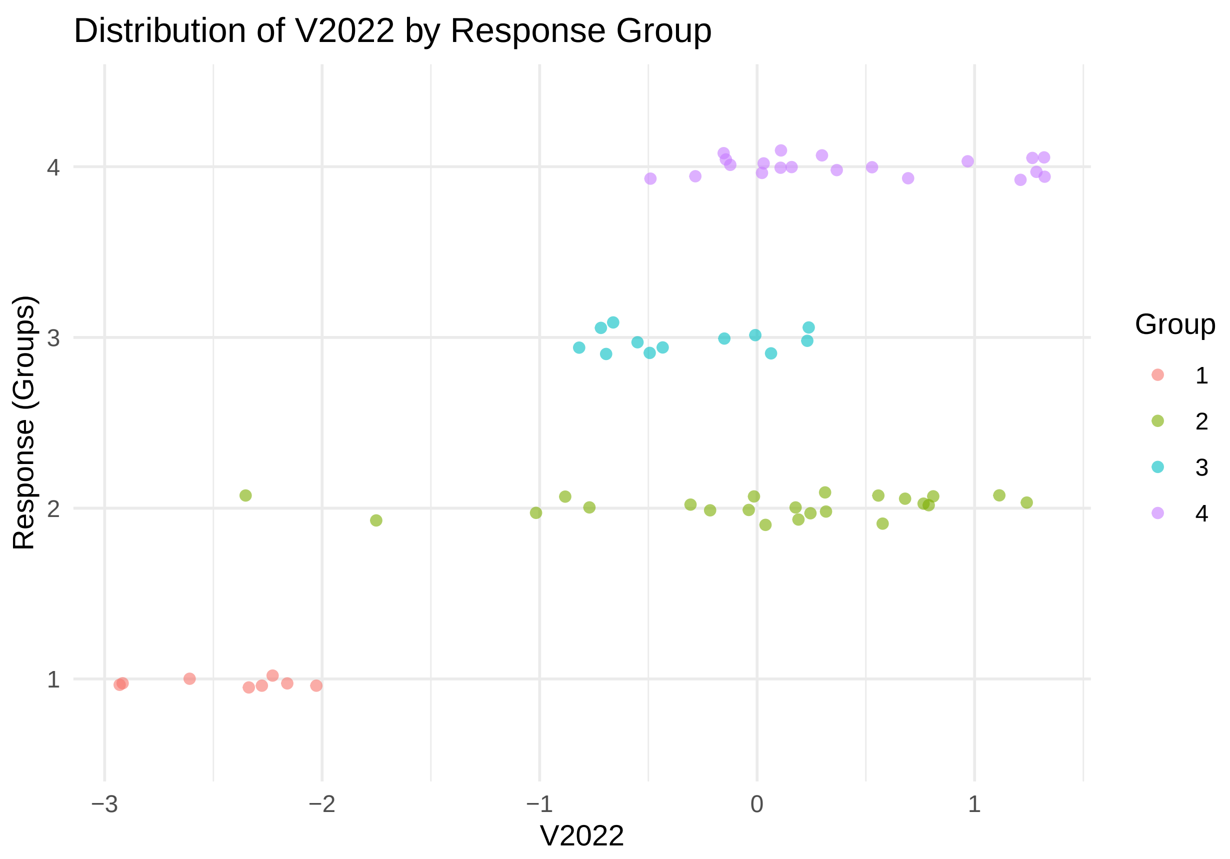
### (d) Plot a Relevant Variable Against the Response

```r
selected_variable <- "V2022"

# Convert the training data to a data frame for ggplot2
training_data <- as.data.frame(Khan$xtrain)
training_data$response <- as.factor(Khan$ytrain)

# Generate the plot
library(ggplot2)

ggplot(training_data, aes_string(x = selected_variable, y = "response", color = "response")) +
    geom_jitter(width = 0.2, height = 0.1, alpha = 0.6) + labs(title = paste("Distribution of",
    selected_variable, "by Response Group"), x = selected_variable, y = "Response (Groups)",
    color = "Group") + theme_minimal()
```



1. **Group 1 (red points)**:
   - The values of `V2022` for Group 1 are distinctively lower compared to other groups, clustering around values between -3 and -2.
   - This separation suggests that `V2022` is highly relevant for distinguishing Group 1 from the other groups.
2. **Group 2 (green points)**:
   - The values for Group 2 are mostly around -1 to 0, showing a different distribution compared to Group 1 and other groups.
3. **Group 3 (blue points)**:
   - The values of `V2022` for Group 3 are slightly above 0, with a tighter clustering compared to Groups 1 and 2.
4. **Group 4 (purple points)**:
   - The values of `V2022` for Group 4 are the highest, with a clear separation from all other groups. This suggests that `V2022` might also contribute to distinguishing Group 4.

**`V2022` is highly relevant for Group 1** as its values for this group are uniquely lower compared to other groups. The variable shows clear separations between groups, which indicates its importance in predicting the group labels. The distinct clustering of values for each group suggests that `V2022` could be a strong predictive feature for this classification problem.

### (e) Predict and Evaluate on Test Data

```r
# Predict probabilities for each class on the test data
predicted_probs <- predict(fit_glmnet, newx = Khan$xtest, s = "lambda.1se", type = "response")

# Select the predicted class for each observation
predicted_classes <- apply(predicted_probs, 1, which.max)

# Create a confusion table
confusion_table <- table(Predicted = predicted_classes, Actual = Khan$ytest)

# Calculate the misclassification error
misclassification_error <- 1 - sum(diag(confusion_table))/sum(confusion_table)

# Print the results
cat("Confusion Table:\n")
```

```
## Confusion Table:
```

```r
print(confusion_table)
```

```
##          Actual
## Predicted 1 2 3 4
##         1 3 0 0 0
##         2 0 6 0 0
##         3 0 0 6 0
##         4 0 0 0 5
```

```r
cat("\nMisclassification Error:", misclassification_error, "\n")
```

```
##
## Misclassification Error: 0
```

The model has perfectly classified all the test samples into their correct groups. This is likely due to the high dimensionality of the dataset and the distinct separation of the classes in the predictor space (as seen in the variable V2022).