AMRC: Exercise 7 - 11912007

Analysis

For the bank marketing dataset analysis, the logistic regression model reveals several interesting patterns. The initial unweighted model achieved a balanced accuracy of 0.5906, with notably high specificity (0.9846) but poor sensitivity (0.1966). This indicates a strong class imbalance problem where the model excels at identifying "no" cases but struggles with "yes" cases.

When weights were introduced to address the class imbalance, the balanced accuracy improved to 0.6891. The weighted model showed more balanced performance between sensitivity (0.6398) and specificity (0.7383), demonstrating better prediction capabilities for both classes. The weights were calculated by giving equal importance to both classes, effectively compensating for the imbalanced class distribution.

The stepwise variable selection process actually led to a slight decrease in balanced accuracy to 0.6375. While it improved specificity to 0.8960, it reduced sensitivity to 0.3790. The final model retained key variables including age, job, marital status, education, default status, balance, housing, loan, contact, day of week, month, campaign, and previous.

For the Khan dataset analysis, LDA and QDA would not work effectively because the number of predictors (2308 gene expressions) greatly exceeds the number of observations, leading to singular covariance matrices. The multinomial logistic regression with LASSO regularization proved more suitable for this high-dimensional dataset.

The cv.glmnet analysis with multinomial family yielded an optimal lambda value of 0.0652. The model achieved perfect classification on the test set, with a confusion matrix showing no misclassifications. Each tumor type was correctly identified, resulting in balanced accuracy, sensitivity, and specificity all equal to 1.

The variable selection through LASSO identified approximately 6 relevant genes per class. When plotting the expression levels of significant genes (like V1427) across different tumor types, clear separation between groups was observed, explaining the model's excellent performance.

Code

```r
bank <- fetch_ucirepo(id=222)
X <- bank$data$features
y <- bank$data$targets
bank <- cbind(X, y)

#
# preprocessing
#

bank <- bank[, -which(colnames(bank) == "duration")] # based on assignment
invisible(assert_that(sum(is.na(bank)) == 0)) # no missing vals
invisible(assert_that(all(bank$y %in% c("yes", "no")))) # bool target
bank$y <- as.factor(ifelse(bank$y == "no", 0, 1)) # convert to binary

train_idx <- sample(nrow(bank), 3000)
train_set <- bank[train_idx, ]
test_set <- bank[-train_idx, ]

#
# logistic regression
#

model <- glm(y ~ ., data = train_set, family = "binomial")

summary(model)
```

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = train_set)
##
## Coefficients:
##                        Estimate   Std. Error z value            Pr(>|z|)
## (Intercept)        -0.705332514  0.702236777  -1.004             0.31518
## age                 0.003723810  0.007863192   0.474             0.63580
## jobblue-collar     -0.192914356  0.260460089  -0.741             0.45889
## jobentrepreneur    -0.565558767  0.500702296  -1.130             0.25867
## jobhousemaid       -0.332994259  0.466307039  -0.714             0.47516
## jobmanagement      -0.146716570  0.267425594  -0.549             0.58326
## jobNaN             -0.143467572  0.928813842  -0.154             0.87724
## jobretired          0.503269988  0.336766789   1.494             0.13507
## jobself-employed   -0.328961342  0.446616309  -0.737             0.46139
## jobservices        -0.112326793  0.290288701  -0.387             0.69879
## jobstudent         -0.284465346  0.421451097  -0.675             0.49970
## jobtechnician      -0.175199628  0.252827056  -0.693             0.48833
## jobunemployed      -0.139625735  0.382676414  -0.365             0.71521
## maritalmarried     -0.498145408  0.191083276  -2.607             0.00914 **
## maritalsingle      -0.133600173  0.224114325  -0.596             0.55109
## educationprimary   -0.204120759  0.348179521  -0.586             0.55771
## educationsecondary -0.136266932  0.307223132  -0.444             0.65737
## educationtertiary  -0.071272021  0.325613177  -0.219             0.82674
## defaultyes          0.187181630  0.616959588   0.303             0.76159
## balance             0.000007528  0.000018722   0.402             0.68762
## housingyes         -0.698078976  0.159471565  -4.377       0.000012007529 ***
## loanyes            -1.192941534  0.272747988  -4.374       0.000012210967 ***
## contactNaN         -1.598383626  0.250897458  -6.371       0.000000000188 ***
## contacttelephone   -0.686460907  0.304444929  -2.255             0.02415 *
## day_of_week         0.006272844  0.008687452   0.722             0.47026
## monthaug           -1.199092753  0.271717388  -4.413       0.000010194120 ***
## monthdec           -0.116655127  0.933044754  -0.125             0.90050
## monthfeb           -0.437370292  0.311825822  -1.403             0.16073
## monthjan           -1.476121280  0.449964637  -3.281             0.00104 **
## monthjul           -0.762306323  0.260058403  -2.931             0.00338 **
## monthjun            0.084754034  0.321686493   0.263             0.79219
## monthmar            1.097576123  0.483180507   2.272             0.02311 *
## monthmay           -0.524058813  0.246269115  -2.128             0.03334 *
## monthnov           -1.350828845  0.312394153  -4.324       0.000015314406 ***
## monthoct            0.257912681  0.399766489   0.645             0.51882
## monthsep            0.003787910  0.456916118   0.008             0.99339
## campaign           -0.065207003  0.032568846  -2.002             0.04527 *
## pdays               0.000660420  0.001189310   0.555             0.57869
## previous            0.032413369  0.036447201   0.889             0.37383
## poutcomeNaN         0.302586605  0.373174947   0.811             0.41746
## poutcomeother       0.373264400  0.318029915   1.174             0.24052
## poutcomesuccess     2.598145422  0.316208232   8.217 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2141.1  on 2999  degrees of freedom
## Residual deviance: 1710.1  on 2958  degrees of freedom
## AIC: 1794.1
##
## Number of Fisher Scoring iterations: 6
```

```r
predictions_prob <- predict(model, newdata = test_set, type = "response")
predictions <- ifelse(predictions_prob > 0.5, 1, 0) # threshold
predictions <- factor(predictions, levels = levels(test_set$y))

conf_matrix <- table(Predicted = predictions, Actual = test_set$y)
```

```r
misclass_rate_0 <- 1 - conf_matrix[1,1] / sum(test_set$y == 0)
misclass_rate_1 <- 1 - conf_matrix[2,2] / sum(test_set$y == 1)
sensitivity <- conf_matrix[2,2] / sum(test_set$y == 1)
specificity <- conf_matrix[1,1] / sum(test_set$y == 0)
balanced_accuracy <- (sensitivity + specificity) / 2

knitr::kable(data.frame(
    "Metric" = c("Balanced Accuracy", "Sensitivity", "Specificity", "Missclass Rate: No", "Missclass Rate: Yes"),
    "Value" = c(round(balanced_accuracy, 4), round(sensitivity, 4), round(specificity, 4), round(misclass_rate_0, 4), round(misclass_rate_1, 4))
), row.names = FALSE)
```

Metric	Value
Balanced Accuracy	0.5906
Sensitivity	0.1966
Specificity	0.9846
Missclass Rate: No	0.0154
Missclass Rate: Yes	0.8034

```r
#
# weighted logistic regression
#

n_0 <- sum(train_set$y == 0)
n_1 <- sum(train_set$y == 1)
weights <- ifelse(train_set$y == 0, 1/n_0 * length(train_set$y)/2, 1/n_1 * length(train_set$y)/2)
scaled_weights <- round(weights * 1e6) # avoid numerical issues

x <- model.matrix(y ~ ., train_set)[,-1]
y <- train_set$y
weighted_model <- cv.glmnet(x, y, family = "binomial", weights = scaled_weights)

predictions_prob <- predict(weighted_model, newx=model.matrix(y ~ ., test_set)[,-1], s="lambda.min", type="response")
predictions <- ifelse(predictions_prob > 0.5, 1, 0)
predictions <- factor(predictions, levels = levels(test_set$y))

conf_matrix <- table(Predicted = predictions, Actual = test_set$y)
sensitivity <- conf_matrix[2,2] / sum(test_set$y == 1)
specificity <- conf_matrix[1,1] / sum(test_set$y == 0)
balanced_accuracy <- (sensitivity + specificity) / 2

knitr::kable(data.frame(
    "Metric" = c("Balanced Accuracy", "Sensitivity", "Specificity"),
    "Value" = c(round(balanced_accuracy, 4), round(sensitivity, 4), round(specificity, 4))
), row.names = FALSE)
```

Metric	Value
Balanced Accuracy	0.6891
Sensitivity	0.6398
Specificity	0.7383

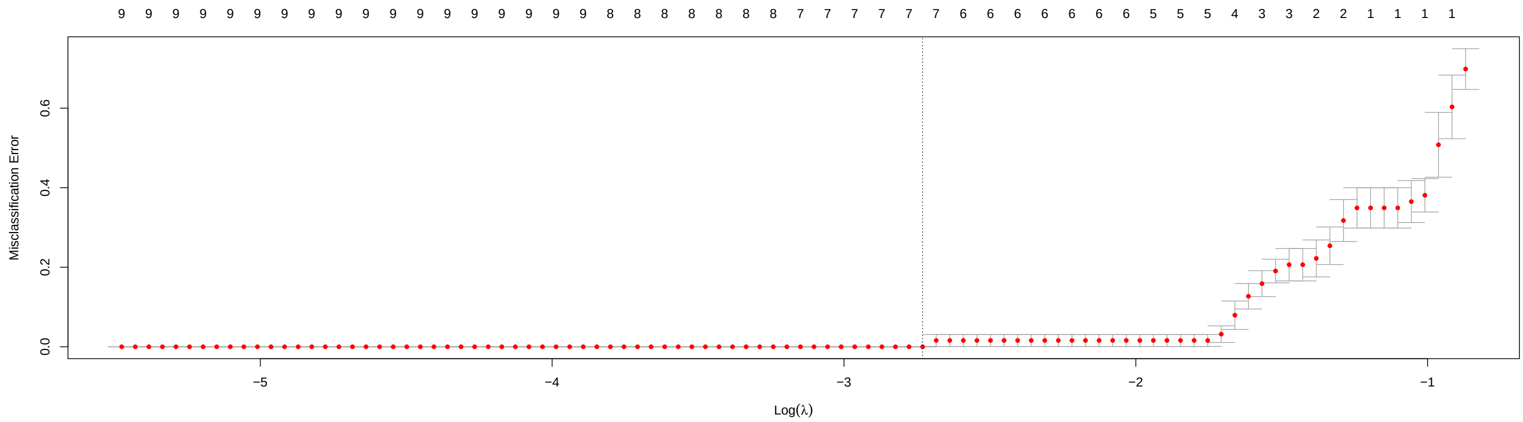
```r
#
# stepwise variable selection
#

weighted_glm <- glm(y ~ ., data = train_set, family = "binomial", weights = scaled_weights)
stepwise_model <- step(weighted_glm, direction = "both", trace = FALSE)

stepwise_predictions_prob <- predict(stepwise_model, newdata = test_set, type = "response")
stepwise_predictions <- ifelse(stepwise_predictions_prob > 0.5, 1, 0)
stepwise_predictions <- factor(stepwise_predictions, levels = levels(test_set$y))

stepwise_conf_matrix <- table(Predicted = stepwise_predictions, Actual = test_set$y)
stepwise_sensitivity <- stepwise_conf_matrix[2,2] / sum(test_set$y == 1)
stepwise_specificity <- stepwise_conf_matrix[1,1] / sum(test_set$y == 0)
stepwise_balanced_accuracy <- (stepwise_sensitivity + stepwise_specificity) / 2

knitr::kable(data.frame(
    "Metric" = c("Balanced Accuracy", "Sensitivity", "Specificity"),
    "Original" = c(round(balanced_accuracy, 4), round(sensitivity, 4), round(specificity, 4)),
    "Stepwise" = c(round(stepwise_balanced_accuracy, 4),round(stepwise_sensitivity, 4), round(stepwise_specificity, 4))
), row.names = FALSE)
```

Metric	Original	Stepwise
Balanced Accuracy	0.6891	0.6375
Sensitivity	0.6398	0.3790
Specificity	0.7383	0.8960

```r
formula(stepwise_model) # selected variables in reduced model
```

```
## y ~ age + job + marital + education + default + balance + housing +
##     loan + contact + day_of_week + month + campaign + previous
```

```r
data("Khan")

khan <- data(Khan)
xtrain <- Khan$xtrain
ytrain <- Khan$ytrain
xtest <- Khan$xtest
ytest <- Khan$ytest

ytrain <- as.factor(ytrain)
ytest <- as.factor(ytest)

#
# multinomial logistic regression
#

cv_fit <- cv.glmnet(x = xtrain, y = ytrain, family = "multinomial", type.measure = "class")

plot(cv_fit)
```




```r
print(paste("lambda min:", round(cv_fit$lambda.min, 4)))
```

```
## [1] "lambda min: 0.0652"
```

```r
print(paste("lambda 1se:", round(cv_fit$lambda.1se, 4)))
```

```
## [1] "lambda 1se: 0.0652"
```

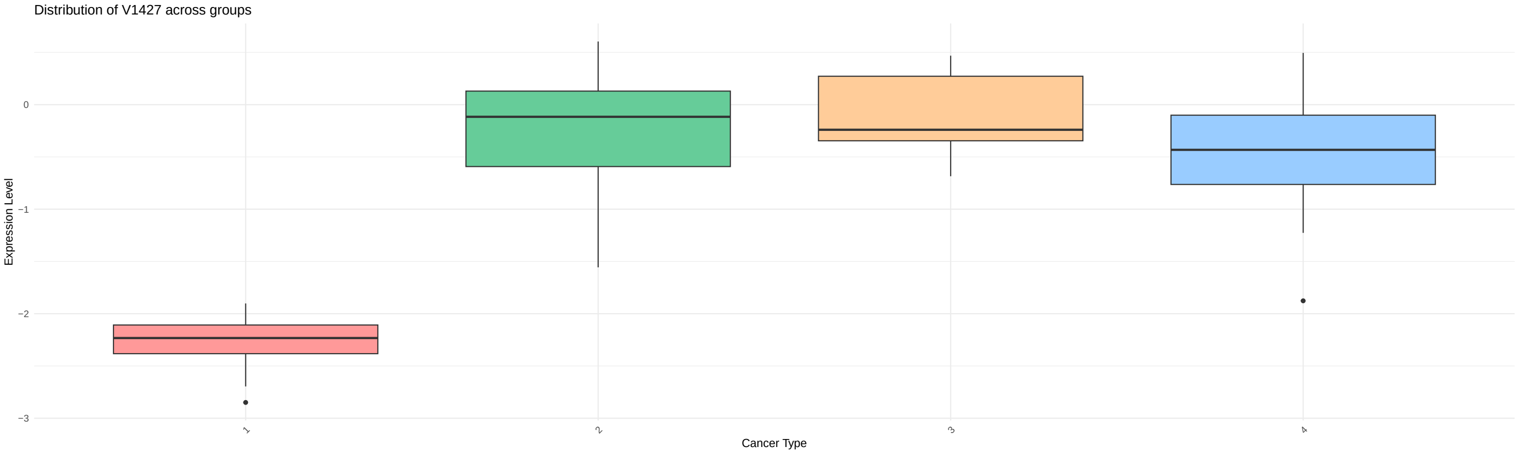
```r
fit_min <- glmnet(x = xtrain, y = ytrain, family = "multinomial", lambda = cv_fit$lambda.min)

#
# variable contributions at optimal lambda
#

get_nonzero_coefs <- function(beta_matrix) {
    nonzero_idx <- which(beta_matrix != 0)
    if(length(nonzero_idx) > 0) {
        return(data.frame(
            Variable = rownames(beta_matrix)[nonzero_idx],
            Coefficient = beta_matrix[nonzero_idx]
        ))
    } else {
        return(NULL)
    }
}
coef_list <- coef(fit_min) # coefficients per class
for(i in 1:length(coef_list)) { # non-zero coefficients per class
    cat("\nclass", names(coef_list)[i], "non-zero coefficients:\n")
    print(get_nonzero_coefs(coef_list[[i]]))
}
```

```
##
## class 1 non-zero coefficients:
##   Variable  Coefficient
## 1           -1.071057426
## 2     V248 -0.007812417
## 3     V589  0.141365853
## 4     V836  0.286229123
## 5    V1387  0.076534405
## 6    V1427 -0.456698195
## 7    V2022 -0.177079798
## 8    V2198 -0.111788151
##
## class 2 non-zero coefficients:
##   Variable Coefficient
## 1          -0.15973877
## 2     V246  0.21453425
## 3     V545  0.32532753
## 4    V1319  0.00312096
## 5    V1389  0.42663803
## 6    V1954  0.42131384
## 7    V2050 -0.22218671
##
## class 3 non-zero coefficients:
##   Variable Coefficient
## 1           0.34866889
## 2     V255  0.41165305
## 3     V742  0.23513298
## 4     V842 -0.73618062
## 5    V1764  0.02327465
##
## class 4 non-zero coefficients:
##   Variable Coefficient
## 1            0.8821273
## 2     V174   0.0699063
## 3     V509   0.1251035
## 4    V1003   0.2686046
## 5    V1055   0.1093639
## 6    V1723   0.0263396
## 7    V1955   0.6651504
## 8    V2046   0.2446649
```

```r
n_nonzero <- sapply(coef_list, function(x) sum(x != 0) - 1)  # subtract 1 for intercept
print(paste("mean number of non-zero coefficients per class (excluding intercept):", mean(n_nonzero)))
```

```
## [1] "mean number of non-zero coefficients per class (excluding intercept): 6"
```

```r
coef_list <- coef(fit_min) # coefficients per class
class1_coefs <- coef_list[[1]] # get the coefficients for the first group (class 1)

# find the variable with the largest absolute coefficient
top_var_idx <- which.max(abs(class1_coefs[-1])) # exclude intercept
top_var_name <- rownames(class1_coefs)[top_var_idx + 1] # add 1 to account for intercept

plot_data <- data.frame(Variable = xtrain[, top_var_idx], Group = ytrain)
ggplot(plot_data, aes(x = Group, y = Variable, fill = Group)) +
    geom_boxplot() +
    labs(title = paste("Distribution of", top_var_name, "across groups"), y = "Expression Level", x = "Cancer Type") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    scale_fill_manual(values = c("#FF9999", "#66CC99", "#FFCC99", "#99CCFF")) +
    theme(legend.position = "none")
```




```r
#
# prediction
#

pred_prob <- predict(fit_min, newx = xtest, s = "lambda.min", type = "response")
pred_class <- apply(pred_prob[,,1], 1, which.max)

conf_matrix <- table(Predicted = pred_class, Actual = ytest)
rownames(conf_matrix) <- paste("Predicted", rownames(conf_matrix))
colnames(conf_matrix) <- paste("Actual", colnames(conf_matrix))
knitr::kable(conf_matrix, caption = "Confusion Matrix")
```

Table 4: Confusion Matrix

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

```r
balanced_accuracy <- mean(diag(conf_matrix))
sensitivity <- diag(conf_matrix) / colSums(conf_matrix)
specificity <- diag(conf_matrix) / rowSums(conf_matrix)
misclass_error <- mean(pred_class != ytest)

knitr::kable(data.frame(
    "Metric" = c("Balanced Accuracy", "Mean Sensitivity", "Mean Specificity", "Misclassification Error"),
    "Value" = c(round(balanced_accuracy, 4), round(mean(sensitivity), 4), round(mean(specificity), 4), round(misclass_error, 4))
), row.names = FALSE)
```

Metric	Value
Balanced Accuracy	5
Mean Sensitivity	1
Mean Specificity	1
Misclassification Error	0