AMRC: Exercise 6 - 11912007

We want to apply discriminant analysis methods to the Loan dataset to predict the Status variable. We use Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Regularized Discriminant Analysis (RDA), evaluating their performance using misclassification rate and balanced accuracy metrics.

An additional preprocessing step was necessary in which we used the LASSO method to remove multicollinear features from the training data.

The initial LDA implementation on the preprocessed data achieved a misclassification rate of 0.152 and balanced accuracy of 0.523 on the training set, with similar performance on the test set (misclassification rate: 0.147, balanced accuracy: 0.501). This indicates potential class imbalance issues in the dataset.

To address the class imbalance and maximize balanced accuracy, two sampling strategies were implemented. The undersampling approach with LDA showed improved balanced accuracy, reaching 0.652 on the training set and 0.632 on the test set, though with higher misclassification rates (0.348 and 0.393 respectively). The oversampling strategy produced slightly lower but comparable results, with balanced accuracy of 0.641 for training and 0.619 for test data.

QDA implementation showed different patterns. The base QDA model achieved a misclassification rate of 0.173 and balanced accuracy of 0.577 on training data, with slightly degraded performance on test data (0.183 and 0.524 respectively). When combined with undersampling, QDA improved with balanced accuracy of 0.641 for training and 0.608 for test data. The oversampling approach with QDA yielded similar results (training: 0.644, test: 0.594).

RDA, which combines aspects of both LDA and QDA, demonstrated intermediate performance. The base RDA model achieved balanced accuracy of 0.554 on training and 0.517 on test data. With undersampling, RDA's balanced accuracy improved to 0.641 for training and 0.614 for test data. The oversampling approach yielded slightly lower but stable results (training: 0.622, test: 0.606).

The gamma and lambda parameters in RDA control the model's complexity. Gamma regulates the transition between LDA ($\gamma = 1$) and QDA ($\gamma = 0$), while lambda controls the shrinkage of the covariance matrix estimation. The optimal values for these parameters were determined through cross-validation to balance model flexibility and stability.

Overall, the undersampling strategy proved more successful than oversampling across all methods, likely because it reduces noise and prevents overfitting that can occur with duplicate observations in oversampling. The balanced accuracy improvements came at the cost of higher misclassification rates, representing a trade-off between overall accuracy and balanced class prediction.

```r
data(Loan)

#
# preprocessing
#

# encoding categorical variables
Loan$Status <- as.numeric(Loan$Status == "FP")
Loan$EmpLen_A <- as.numeric(Loan$EmpLen == "A")
Loan$EmpLen_B <- as.numeric(Loan$EmpLen == "B")
Loan$EmpLen_C <- as.numeric(Loan$EmpLen == "C")
Loan$EmpLen_D <- as.numeric(Loan$EmpLen == "D")
Loan$EmpLen_U <- as.numeric(Loan$EmpLen == "U")
Loan$EmpLen <- NULL
Loan$Home_MORTGAGE <- as.numeric(Loan$Home == "MORTGAGE")
Loan$Home_OWN <- as.numeric(Loan$Home == "OWN")
Loan$Home_RENT <- as.numeric(Loan$Home == "RENT")
Loan$Home <- NULL
invisible(assert_that(all(sapply(Loan, is.numeric))))

# no missing vals
invisible(assert_that(all(sapply(Loan, function(x) sum(is.na(x)) == 0))))

# drop single valued cols
Loan$Term <- NULL
invisible(assert_that(all(sapply(Loan, function(x) length(unique(x)) > 1))))

# scale columns that aren't encoded
Loan$Amount <- scale(Loan$Amount)
Loan$IntRate <- scale(Loan$IntRate)
Loan$Income <- scale(Loan$Income)
Loan$Score <- scale(Loan$Score)

# train/test split
train_idx <- sample(nrow(Loan), size = round(2/3 * nrow(Loan)))
train_data <- Loan[train_idx, ]
test_data <- Loan[-train_idx, ]

# drop correlated features with lasso
X_train <- as.matrix(train_data[, !colnames(train_data) %in% c("Status")])
y_train <- train_data$Status
cv_fit <- cv.glmnet(X_train, y_train, alpha=1, family="binomial")
coef_matrix <- coef(cv_fit, s="lambda.min")
selected_features <- rownames(coef_matrix)[which(coef_matrix != 0)][-1] # drop intercept
train_data <- train_data[, c("Status", selected_features)]
test_data <- test_data[, c("Status", selected_features)]

#
# LDA
#

lda_model <- lda(Status ~ ., data = train_data)

# train eval
train_pred <- predict(lda_model, train_data)
train_conf_matrix <- confusionMatrix(factor(train_pred$class), factor(train_data$Status))
train_misclass <- (train_conf_matrix$table[1,2] + train_conf_matrix$table[2,1]) / sum(train_conf_matrix$table)
train_balanced_acc <- (train_conf_matrix$byClass["Sensitivity"] + train_conf_matrix$byClass["Specificity"]) / 2

# test eval
test_pred <- predict(lda_model, test_data)
test_conf_matrix <- confusionMatrix(factor(test_pred$class), factor(test_data$Status))
test_misclass <- (test_conf_matrix$table[1,2] + test_conf_matrix$table[2,1]) / sum(test_conf_matrix$table)
test_balanced_acc <- (test_conf_matrix$byClass["Sensitivity"] + test_conf_matrix$byClass["Specificity"]) / 2

knitr::kable(data.frame(
    "Set" = c("Train", "Test"),
    "Misclassification Rate" = c(train_misclass, test_misclass),
    "Balanced Accuracy" = c(train_balanced_acc, test_balanced_acc)
), row.names = FALSE)
```

Set	Misclassification.Rate	Balanced.Accuracy
Train	0.1516667	0.5232369
Test	0.1466667	0.5013263

```r
#
# LDA + undersampling
#

status_0 <- train_data[train_data$Status == 0, ]
status_1 <- train_data[train_data$Status == 1, ]
min_samples <- min(nrow(status_0), nrow(status_1))

balanced_status_0 <- status_0[sample(nrow(status_0), min_samples), ]
balanced_status_1 <- status_1[sample(nrow(status_1), min_samples), ]
balanced_train_data <- rbind(balanced_status_0, balanced_status_1)

lda_model <- lda(Status ~ ., data = balanced_train_data)
```

```r
# train eval
train_pred <- predict(lda_model, balanced_train_data)
train_conf_matrix <- confusionMatrix(factor(train_pred$class), factor(balanced_train_data$Status))
train_misclass <- (train_conf_matrix$table[1,2] + train_conf_matrix$table[2,1]) / sum(train_conf_matrix$table)
train_balanced_acc <- (train_conf_matrix$byClass["Sensitivity"] + train_conf_matrix$byClass["Specificity"]) / 2

# test eval
test_pred <- predict(lda_model, test_data)
test_conf_matrix <- confusionMatrix(factor(test_pred$class), factor(test_data$Status))
test_misclass <- (test_conf_matrix$table[1,2] + test_conf_matrix$table[2,1]) / sum(test_conf_matrix$table)
test_balanced_acc <- (test_conf_matrix$byClass["Sensitivity"] + test_conf_matrix$byClass["Specificity"]) / 2

knitr::kable(data.frame(
    "Set" = c("Train", "Test"),
    "Misclassification Rate" = c(train_misclass, test_misclass),
    "Balanced Accuracy" = c(train_balanced_acc, test_balanced_acc)
), row.names = FALSE)
```

Set	Misclassification.Rate	Balanced.Accuracy
Train	0.3478261	0.6521739
Test	0.3933333	0.6321839

```r
#
# LDA + oversampling
#

status_0 <- train_data[train_data$Status == 0, ]
status_1 <- train_data[train_data$Status == 1, ]
max_samples <- max(nrow(status_0), nrow(status_1))
if (nrow(status_0) < nrow(status_1)) { # sample with replacement from the smaller group
    balanced_status_0 <- status_0[sample(nrow(status_0), max_samples, replace = TRUE), ]
    balanced_status_1 <- status_1
} else {
    balanced_status_0 <- status_0
    balanced_status_1 <- status_1[sample(nrow(status_1), max_samples, replace = TRUE), ]
}
balanced_train_data <- rbind(balanced_status_0, balanced_status_1)

lda_model <- lda(Status ~ ., data = balanced_train_data)

# train eval
train_pred <- predict(lda_model, balanced_train_data)
train_conf_matrix <- confusionMatrix(factor(train_pred$class), factor(balanced_train_data$Status))
train_misclass <- (train_conf_matrix$table[1,2] + train_conf_matrix$table[2,1]) / sum(train_conf_matrix$table)
train_balanced_acc <- (train_conf_matrix$byClass["Sensitivity"] + train_conf_matrix$byClass["Specificity"]) / 2

# test eval
test_pred <- predict(lda_model, test_data)
test_conf_matrix <- confusionMatrix(factor(test_pred$class), factor(test_data$Status))
test_misclass <- (test_conf_matrix$table[1,2] + test_conf_matrix$table[2,1]) / sum(test_conf_matrix$table)
test_balanced_acc <- (test_conf_matrix$byClass["Sensitivity"] + test_conf_matrix$byClass["Specificity"]) / 2

knitr::kable(data.frame(
    "Set" = c("Train", "Test"),
    "Misclassification Rate" = c(train_misclass, test_misclass),
    "Balanced Accuracy" = c(train_balanced_acc, test_balanced_acc)
), row.names = FALSE)
```

Set	Misclassification.Rate	Balanced.Accuracy
Train	0.359252	0.6407480
Test	0.360000	0.6186266

```r
#
# QDA
#

qda_model <- qda(Status ~ ., data = train_data)

# train eval
train_pred <- predict(qda_model, train_data)
train_conf_matrix <- confusionMatrix(factor(train_pred$class), factor(train_data$Status))
train_misclass <- (train_conf_matrix$table[1,2] + train_conf_matrix$table[2,1]) / sum(train_conf_matrix$table)
train_balanced_acc <- (train_conf_matrix$byClass["Sensitivity"] + train_conf_matrix$byClass["Specificity"]) / 2

# test eval
test_pred <- predict(qda_model, test_data)
test_conf_matrix <- confusionMatrix(factor(test_pred$class), factor(test_data$Status))
test_misclass <- (test_conf_matrix$table[1,2] + test_conf_matrix$table[2,1]) / sum(test_conf_matrix$table)
test_balanced_acc <- (test_conf_matrix$byClass["Sensitivity"] + test_conf_matrix$byClass["Specificity"]) / 2

knitr::kable(data.frame(
    "Set" = c("Train", "Test"),
    "Misclassification Rate" = c(train_misclass, test_misclass),
    "Balanced Accuracy" = c(train_balanced_acc, test_balanced_acc)
), row.names = FALSE)
```

Set	Misclassification.Rate	Balanced.Accuracy
Train	0.1733333	0.5771996
Test	0.1833333	0.5238727

```r
#
# QDA + undersampling
#

status_0 <- train_data[train_data$Status == 0, ]
status_1 <- train_data[train_data$Status == 1, ]
min_samples <- min(nrow(status_0), nrow(status_1))

balanced_status_0 <- status_0[sample(nrow(status_0), min_samples), ]
balanced_status_1 <- status_1[sample(nrow(status_1), min_samples), ]
balanced_train_data <- rbind(balanced_status_0, balanced_status_1)

qda_model <- qda(Status ~ ., data = balanced_train_data)

# train eval
train_pred <- predict(qda_model, balanced_train_data)
train_conf_matrix <- confusionMatrix(factor(train_pred$class), factor(balanced_train_data$Status))
train_misclass <- (train_conf_matrix$table[1,2] + train_conf_matrix$table[2,1]) / sum(train_conf_matrix$table)
train_balanced_acc <- (train_conf_matrix$byClass["Sensitivity"] + train_conf_matrix$byClass["Specificity"]) / 2

# test eval
test_pred <- predict(qda_model, test_data)
test_conf_matrix <- confusionMatrix(factor(test_pred$class), factor(test_data$Status))
test_misclass <- (test_conf_matrix$table[1,2] + test_conf_matrix$table[2,1]) / sum(test_conf_matrix$table)
```

```r
test_balanced_acc <- (test_conf_matrix$byClass["Sensitivity"] + test_conf_matrix$byClass["Specificity"]) / 2

knitr::kable(data.frame(
    "Set" = c("Train", "Test"),
    "Misclassification Rate" = c(train_misclass, test_misclass),
    "Balanced Accuracy" = c(train_balanced_acc, test_balanced_acc)
), row.names = FALSE)
```

Set	Misclassification.Rate	Balanced.Accuracy
Train	0.3586957	0.6413043
Test	0.3033333	0.6075744

```r
#
# QDA + oversampling
#

status_0 <- train_data[train_data$Status == 0, ]
status_1 <- train_data[train_data$Status == 1, ]
max_samples <- max(nrow(status_0), nrow(status_1))
if (nrow(status_0) < nrow(status_1)) {
    balanced_status_0 <- status_0[sample(nrow(status_0), max_samples, replace = TRUE), ]
    balanced_status_1 <- status_1
} else {
    balanced_status_0 <- status_0
    balanced_status_1 <- status_1[sample(nrow(status_1), max_samples, replace = TRUE), ]
}
balanced_train_data <- rbind(balanced_status_0, balanced_status_1)

qda_model <- qda(Status ~ ., data = balanced_train_data)

# train eval
train_pred <- predict(qda_model, balanced_train_data)
train_conf_matrix <- confusionMatrix(factor(train_pred$class), factor(balanced_train_data$Status))
train_misclass <- (train_conf_matrix$table[1,2] + train_conf_matrix$table[2,1]) / sum(train_conf_matrix$table)
train_balanced_acc <- (train_conf_matrix$byClass["Sensitivity"] + train_conf_matrix$byClass["Specificity"]) / 2

# test eval
test_pred <- predict(qda_model, test_data)
test_conf_matrix <- confusionMatrix(factor(test_pred$class), factor(test_data$Status))
test_misclass <- (test_conf_matrix$table[1,2] + test_conf_matrix$table[2,1]) / sum(test_conf_matrix$table)
test_balanced_acc <- (test_conf_matrix$byClass["Sensitivity"] + test_conf_matrix$byClass["Specificity"]) / 2

knitr::kable(data.frame(
    "Set" = c("Train", "Test"),
    "Misclassification Rate" = c(train_misclass, test_misclass),
    "Balanced Accuracy" = c(train_balanced_acc, test_balanced_acc)
), row.names = FALSE)
```

Set	Misclassification.Rate	Balanced.Accuracy
Train	0.3562992	0.6437008
Test	0.3833333	0.5943118

```r
#
# RDA
#

rda_model <- rda(Status ~ ., data = train_data)

# train eval
train_pred <- predict(rda_model, train_data)
train_conf_matrix <- confusionMatrix(factor(train_pred$class), factor(train_data$Status))
train_misclass <- (train_conf_matrix$table[1,2] + train_conf_matrix$table[2,1]) / sum(train_conf_matrix$table)
train_balanced_acc <- (train_conf_matrix$byClass["Sensitivity"] + train_conf_matrix$byClass["Specificity"]) / 2

# test eval
test_pred <- predict(rda_model, test_data)
test_conf_matrix <- confusionMatrix(factor(test_pred$class), factor(test_data$Status))
test_misclass <- (test_conf_matrix$table[1,2] + test_conf_matrix$table[2,1]) / sum(test_conf_matrix$table)
test_balanced_acc <- (test_conf_matrix$byClass["Sensitivity"] + test_conf_matrix$byClass["Specificity"]) / 2

knitr::kable(data.frame(
    "Set" = c("Train", "Test"),
    "Misclassification Rate" = c(train_misclass, test_misclass),
    "Balanced Accuracy" = c(train_balanced_acc, test_balanced_acc)
), row.names = FALSE)
```

Set	Misclassification.Rate	Balanced.Accuracy
Train	0.1516667	0.5543906
Test	0.1566667	0.5173887

```r
#
# RDA + undersampling
#

status_0 <- train_data[train_data$Status == 0, ]
status_1 <- train_data[train_data$Status == 1, ]
min_samples <- min(nrow(status_0), nrow(status_1))

balanced_status_0 <- status_0[sample(nrow(status_0), min_samples), ]
balanced_status_1 <- status_1[sample(nrow(status_1), min_samples), ]
balanced_train_data <- rbind(balanced_status_0, balanced_status_1)

rda_model <- rda(Status ~ ., data = balanced_train_data)

# train eval
train_pred <- predict(rda_model, balanced_train_data)
train_conf_matrix <- confusionMatrix(factor(train_pred$class), factor(balanced_train_data$Status))
train_misclass <- (train_conf_matrix$table[1,2] + train_conf_matrix$table[2,1]) / sum(train_conf_matrix$table)
train_balanced_acc <- (train_conf_matrix$byClass["Sensitivity"] + train_conf_matrix$byClass["Specificity"]) / 2

# test eval
test_pred <- predict(rda_model, test_data)
test_conf_matrix <- confusionMatrix(factor(test_pred$class), factor(test_data$Status))
test_misclass <- (test_conf_matrix$table[1,2] + test_conf_matrix$table[2,1]) / sum(test_conf_matrix$table)
test_balanced_acc <- (test_conf_matrix$byClass["Sensitivity"] + test_conf_matrix$byClass["Specificity"]) / 2

knitr::kable(data.frame(
    "Set" = c("Train", "Test"),
    "Misclassification Rate" = c(train_misclass, test_misclass),
    "Balanced Accuracy" = c(train_balanced_acc, test_balanced_acc)
), row.names = FALSE)
```

Set	Misclassification.Rate	Balanced.Accuracy
Train	0.3586957	0.6413043
Test	0.3866667	0.6142057

```r
#
# RDA + oversampling
#

status_0 <- train_data[train_data$Status == 0, ]
status_1 <- train_data[train_data$Status == 1, ]
max_samples <- max(nrow(status_0), nrow(status_1))
if (nrow(status_0) < nrow(status_1)) {
    balanced_status_0 <- status_0[sample(nrow(status_0), max_samples, replace = TRUE), ]
    balanced_status_1 <- status_1
} else {
    balanced_status_0 <- status_0
    balanced_status_1 <- status_1[sample(nrow(status_1), max_samples, replace = TRUE), ]
}
balanced_train_data <- rbind(balanced_status_0, balanced_status_1)

rda_model <- rda(Status ~ ., data = balanced_train_data)

# train eval
train_pred <- predict(rda_model, balanced_train_data)
train_conf_matrix <- confusionMatrix(factor(train_pred$class), factor(balanced_train_data$Status))
train_misclass <- (train_conf_matrix$table[1,2] + train_conf_matrix$table[2,1]) / sum(train_conf_matrix$table)
train_balanced_acc <- (train_conf_matrix$byClass["Sensitivity"] + train_conf_matrix$byClass["Specificity"]) / 2

# test eval
test_pred <- predict(rda_model, test_data)
test_conf_matrix <- confusionMatrix(factor(test_pred$class), factor(test_data$Status))
test_misclass <- (test_conf_matrix$table[1,2] + test_conf_matrix$table[2,1]) / sum(test_conf_matrix$table)
test_balanced_acc <- (test_conf_matrix$byClass["Sensitivity"] + test_conf_matrix$byClass["Specificity"]) / 2

knitr::kable(data.frame(
    "Set" = c("Train", "Test"),
    "Misclassification Rate" = c(train_misclass, test_misclass),
    "Balanced Accuracy" = c(train_balanced_acc, test_balanced_acc)
), row.names = FALSE)
```

Set	Misclassification.Rate	Balanced.Accuracy
Train	0.3779528	0.6220472
Test	0.3066667	0.6056587