Rhw4

Priya Mantraratnam

2025-10-11

```
knitr::opts_chunk$set(echo = TRUE)
```

```
options(repos = c(CRAN = "https://cran.rstudio.com/"))

install.packages("ggfortify")
```

```
## Installing package into 'C:/Users/harip/AppData/Local/R/win-library/4.5'
## (as 'lib' is unspecified)

## package 'ggfortify' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\harip\AppData\Local\Temp\RtmpqK79jR\downloaded_packages
```

```
install.packages("mvnormtest")
```

```
## Installing package into 'C:/Users/harip/AppData/Local/R/win-library/4.5'
## (as 'lib' is unspecified)

## package 'mvnormtest' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\harip\AppData\Local\Temp\RtmpqK79jR\downloaded_packages
```

```
install.packages("datarium")
```

```
## Installing package into 'C:/Users/harip/AppData/Local/R/win-library/4.5'
## (as 'lib' is unspecified)

## package 'datarium' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\harip\AppData\Local\Temp\RtmpqK79jR\downloaded_packages
```

```
install.packages("ggplot2")
```

```
## Installing package into 'C:/Users/harip/AppData/Local/R/win-library/4.5'
## (as 'lib' is unspecified)
```

```
## package 'ggplot2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\harip\AppData\Local\Temp\RtmpqK79jR\downloaded_packages
```

```
install.packages("caret")
```

```
## Installing package into 'C:/Users/harip/AppData/Local/R/win-library/4.5'
## (as 'lib' is unspecified)

## package 'caret' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\harip\AppData\Local\Temp\RtmpqK79jR\downloaded_packages
```

```
install.packages("mvtnorm")
```

```
## Installing package into 'C:/Users/harip/AppData/Local/R/win-library/4.5'
## (as 'lib' is unspecified)

## package 'mvtnorm' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\harip\AppData\Local\Temp\RtmpqK79jR\downloaded_packages
```

```
install.packages("pROC")
```

```
## Installing package into 'C:/Users/harip/AppData/Local/R/win-library/4.5'
## (as 'lib' is unspecified)

## package 'pROC' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\harip\AppData\Local\Temp\RtmpqK79jR\downloaded_packages
```

```
install.packages("tinytex")
```

```
## Installing package into 'C:/Users/harip/AppData/Local/R/win-library/4.5'
## (as 'lib' is unspecified)

## package 'tinytex' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\harip\AppData\Local\Temp\RtmpqK79jR\downloaded_packages
```

```
library(MASS)
library(datarium)
library(ggplot2)
library(broom)
library(ggfortify)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.1     v stringr   1.5.2
## v lubridate 1.9.4     v tibble    3.3.0
## v purrr     1.1.0     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(mvnormtest)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(dplyr)
library(tinytex)

#define helper for decision boundary visualization
decisionplot <- function(model, data, class = NULL, predict_type = "class",
                         resolution = 200, ...) {
  require(data.table)
  require(ggplot2)

  if (!is.data.table(data)) data <- as.data.table(data)

  if (!is.null(class)) {
    cl <- data[[class]]
```

```
  } else {
    stop("You must provide the class column name using `class =`")
  }

  data_xy <- data[, 1:2, with = FALSE]
  k <- length(unique(cl))

  # Build grid
  r <- sapply(data_xy, range, na.rm = TRUE)
  grid_x1 <- seq(r[1, 1], r[2, 1], length.out = resolution)
  grid_x2 <- seq(r[1, 2], r[2, 2], length.out = resolution)
  grid <- as.data.table(expand.grid(x1 = grid_x1, x2 = grid_x2))

  # Predict over grid
  p <- predict(model, newdata = grid, type = predict_type)
  if (is.list(p)) p <- p$class
  grid[, yhat := as.factor(p)]

  # Return ggplot object
  plt <- ggplot() +
    geom_point(data = grid, aes(x1, x2, color = yhat), alpha = 0.05, shape = 15) +
    geom_point(data = data, aes(x1, x2, color = get(class)), shape = 1) +
    labs(title = "Decision Boundary", color = "Class") +
    theme_minimal()

  print(plt)
  invisible(plt)
}

data <- read.csv("C:/Users/harip/Downloads/adult/adult.data")
head(data)
```

```
##   X39         State.gov X77516  Bachelors X13          Never.married
## 1  50  Self-emp-not-inc  83311  Bachelors  13     Married-civ-spouse
## 2  38           Private 215646    HS-grad   9               Divorced
## 3  53           Private 234721       11th   7     Married-civ-spouse
## 4  28           Private 338409  Bachelors  13     Married-civ-spouse
## 5  37           Private 284582    Masters  14     Married-civ-spouse
## 6  49           Private 160187        9th   5  Married-spouse-absent
##          Adm.clerical  Not.in.family  White    Male X2174 X0 X40  United.States
## 1    Exec-managerial         Husband  White    Male     0  0  13  United-States
## 2  Handlers-cleaners  Not-in-family  White    Male     0  0  40  United-States
## 3  Handlers-cleaners         Husband  Black    Male     0  0  40  United-States
## 4     Prof-specialty            Wife  Black  Female     0  0  40           Cuba
## 5    Exec-managerial            Wife  White  Female     0  0  40  United-States
## 6      Other-service  Not-in-family  Black  Female     0  0  16        Jamaica
##   X..50K
## 1  <=50K
## 2  <=50K
## 3  <=50K
## 4  <=50K
## 5  <=50K
## 6  <=50K
```

```r
data <- select(data, X39, X13, X40, X..50K)
names(data) <- c("age", "education-num", "hours-per-week", "y")
head(data)
```

```
##   age education-num hours-per-week      y
## 1  50            13             13  <=50K
## 2  38             9             40  <=50K
## 3  53             7             40  <=50K
## 4  28            13             40  <=50K
## 5  37            14             40  <=50K
## 6  49             5             16  <=50K
```

```r
data$y <- ifelse(data$y == " >50K", 2, 1)
head(data)
```

```
##   age education-num hours-per-week y
## 1  50            13             13 1
## 2  38             9             40 1
## 3  53             7             40 1
## 4  28            13             40 1
## 5  37            14             40 1
## 6  49             5             16 1
```

```r
X1 <- filter(data, y == 1)
head(X1)
```

```
##   age education-num hours-per-week y
## 1  50            13             13 1
## 2  38             9             40 1
## 3  53             7             40 1
## 4  28            13             40 1
## 5  37            14             40 1
## 6  49             5             16 1
```

```r
X2 <- filter(data, y == 2)
head(X2)
```

```
##   age education-num hours-per-week y
## 1  52             9             45 2
## 2  31            14             50 2
## 3  42            13             40 2
## 4  37            10             80 2
## 5  30            13             40 2
## 6  40            11             40 2
```

```r
X <- rbind(X1, X2)
head(X)
```

```
##   age education-num hours-per-week y
## 1  50            13             13 1
```

```
## 2  38             9            40 1
## 3  53             7            40 1
## 4  28            13            40 1
## 5  37            14            40 1
## 6  49             5            16 1
```

```
y <- factor(c(rep(0, nrow(X1)), rep(1, nrow(X2))))
head(y)
```

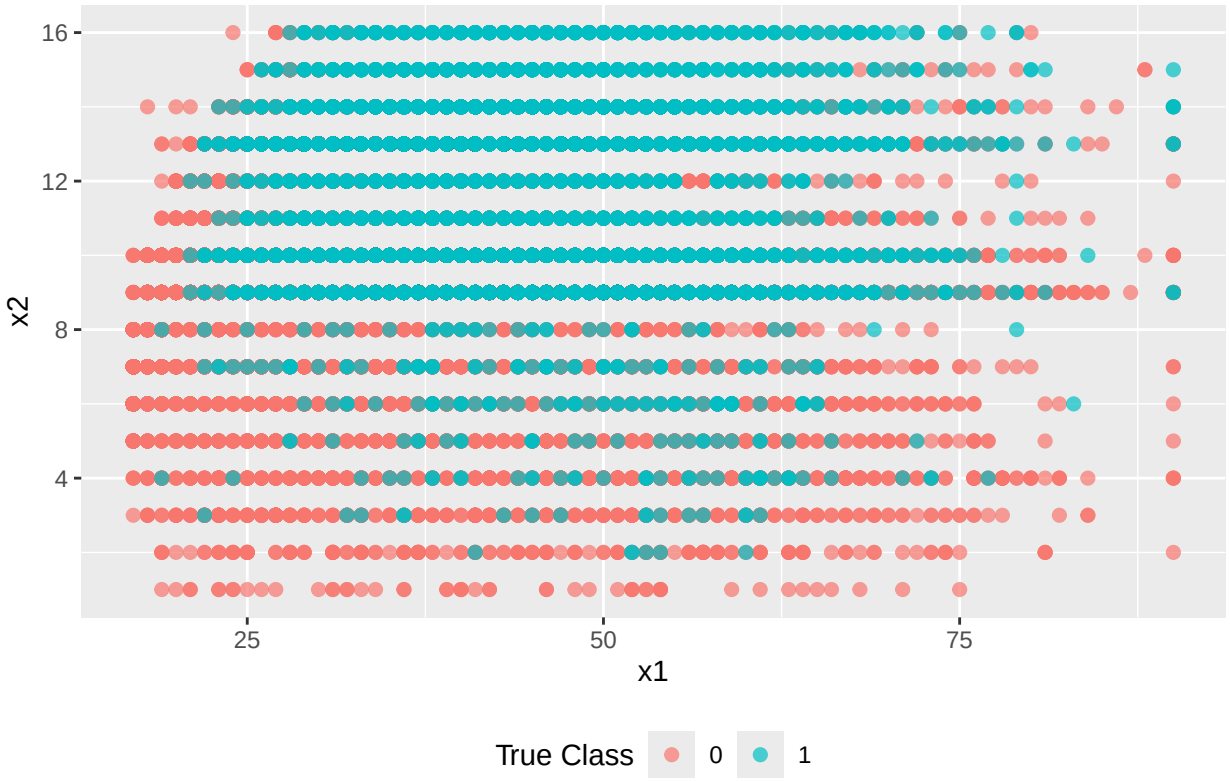
```
## [1] 0 0 0 0 0 0
## Levels: 0 1
```

```
data <- data.table(x1 = X[, 1], x2 = X[, 2], y = y)
head(data)
```

```
##       x1    x2      y
##    <int> <int> <fctr>
## 1:    50    13      0
## 2:    38     9      0
## 3:    53     7      0
## 4:    28    13      0
## 5:    37    14      0
## 6:    49     5      0
```

```
ggplot(data, aes(x = x1, y = x2, color = y)) +
  geom_point(size = 2, alpha = 0.7) +
  scale_shape_manual(values = c(16, 17, 18, 19, 20, 21)) +
  ggtitle("") +
  labs(color = "True Class", shape = "Predicted Class") +
  theme(legend.position = "bottom")
```

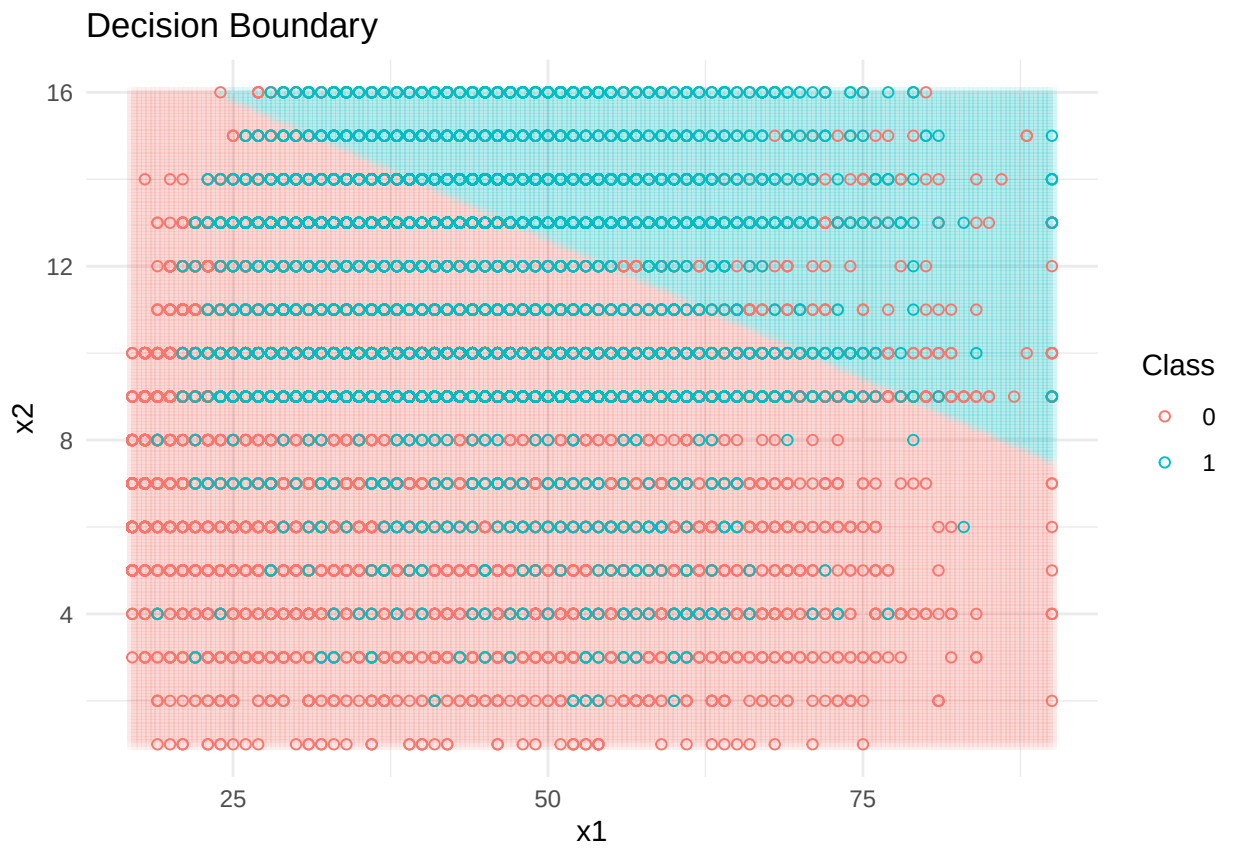
```
## Ignoring unknown labels:
## * shape : "Predicted Class"
```




```
#LDA model and decision boundary

lda_model <- lda(y ~ x1 + x2, data = data)
lda_pred <- predict(lda_model, data)
table("LDA" = lda_pred$class, "True" = data$y)
```

```
##     True
## LDA     0     1
##   0 23339  5761
##   1  1380  2080
```

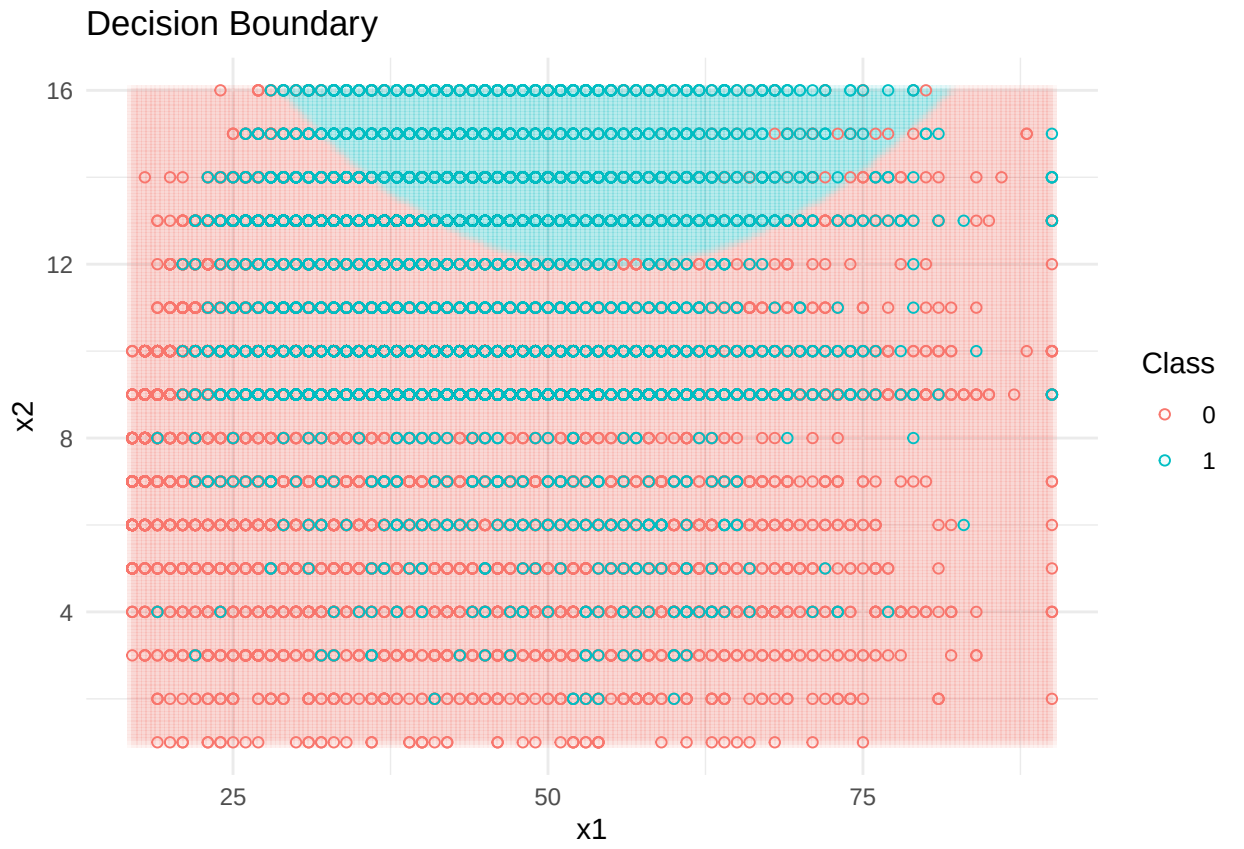
```
decisionplot(lda_model, data, class="y")
```




```
#QDA and comparison

qda_model <- qda(y ~ x1 + x2, data = data)
qda_pred <- predict(qda_model, data)
table("QDA" = qda_pred$class, "True" = data$y)
```

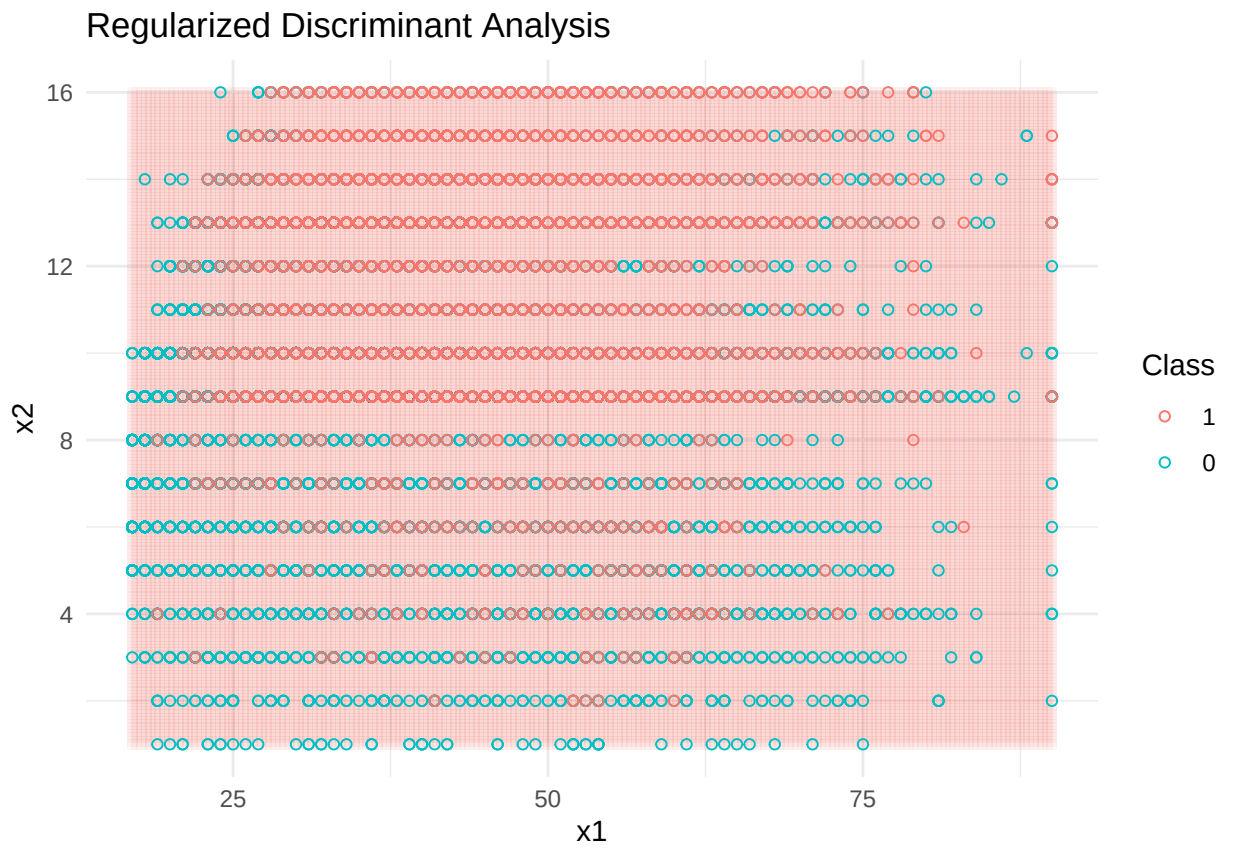
```
##    True
## QDA     0     1
##   0 23148  5225
##   1  1571  2616
```

```
decisionplot(qda_model, data, class="y")
```




```r
#RDA
library(mvtnorm)

K <- 2

mu_list <- lapply(1:K, function(k) {
  colMeans(data[y == k, .(x1, x2)])
})

n_k <- as.numeric(table(y))
prior <- prop.table(n_k)

cov_list <- lapply(1:K, function(k) {
  X_k <- data[y == k, .(x1, x2)]
  cov(X_k)
})

Sigma_pooled <- Reduce("+", lapply(1:2, function(k) (n_k[k]-1)*cov_list[[k]])) / (sum(n_k) - K)

sigma2_hat <- mean(sapply(cov_list, function(S) mean(diag(S))))

#compute sigma_k^tilde and final RDA covariances
lambda <- 0.5
gamma <- 0.5

tilde_covs <- lapply(1:K, function(k) {
```

```
  lambda * cov_list[[k]] + (1 - lambda) * sigma2_hat * diag(2)
})

rda_covs <- lapply(1:K, function(k) {
  gamma * tilde_covs[[k]] + (1 - gamma) * Sigma_pooled
})

#RDA prediction function
rda_predict <- function(Xnew, mu_list, cov_list, prior) {
  log_post <- sapply(1:length(mu_list), function(k) {
    dmvnorm(Xnew, mean = mu_list[[k]], sigma = cov_list[[k]], log = TRUE) + log(prior[k])
  })
  apply(log_post, 1, which.max)
}

#plot
grid_x1 <- seq(min(data$x1), max(data$x1), length.out = 200)
grid_x2 <- seq(min(data$x2), max(data$x2), length.out = 200)
grid <- as.data.table(expand.grid(x1 = grid_x1, x2 = grid_x2))

yhat <- rda_predict(as.matrix(grid), mu_list, rda_covs, prior)
grid[, yhat := as.factor(yhat)]

ggplot() +
  geom_point(data = grid, aes(x1, x2, color = yhat), alpha = 0.05, shape = 15) +
  geom_point(data = data, aes(x1, x2, color = y), shape = 1) +
  labs(title = "Regularized Discriminant Analysis", color = "Class") +
  theme_minimal()
```




```
#computation for QDA via eigen-decomposition
#compute eigen-decomposition for sigma_k
cov1 <- cov(X1)
eig1 <- eigen(cov1)

cat("Eigenvalues (class 1):", eig1$values, "\n")
```

```
## Eigenvalues (class 1): 197.9632 150.4318 5.87388 0
```

```
cat("Orthonormal matrix U_k (class 1):\n")
```

```
## Orthonormal matrix U_k (class 1):
```

```
print(eig1$vectors)
```

```
##              [,1]        [,2]         [,3] [,4]
## [1,]  0.985394745 -0.16999429  0.009956804    0
## [2,] -0.006839329  0.01891447  0.999797713    0
## [3,]  0.170148230  0.98526351 -0.017475568    0
## [4,]  0.000000000  0.00000000  0.000000000    1
```

```
#reduced-rank LDA
#mean vectors for each class
```

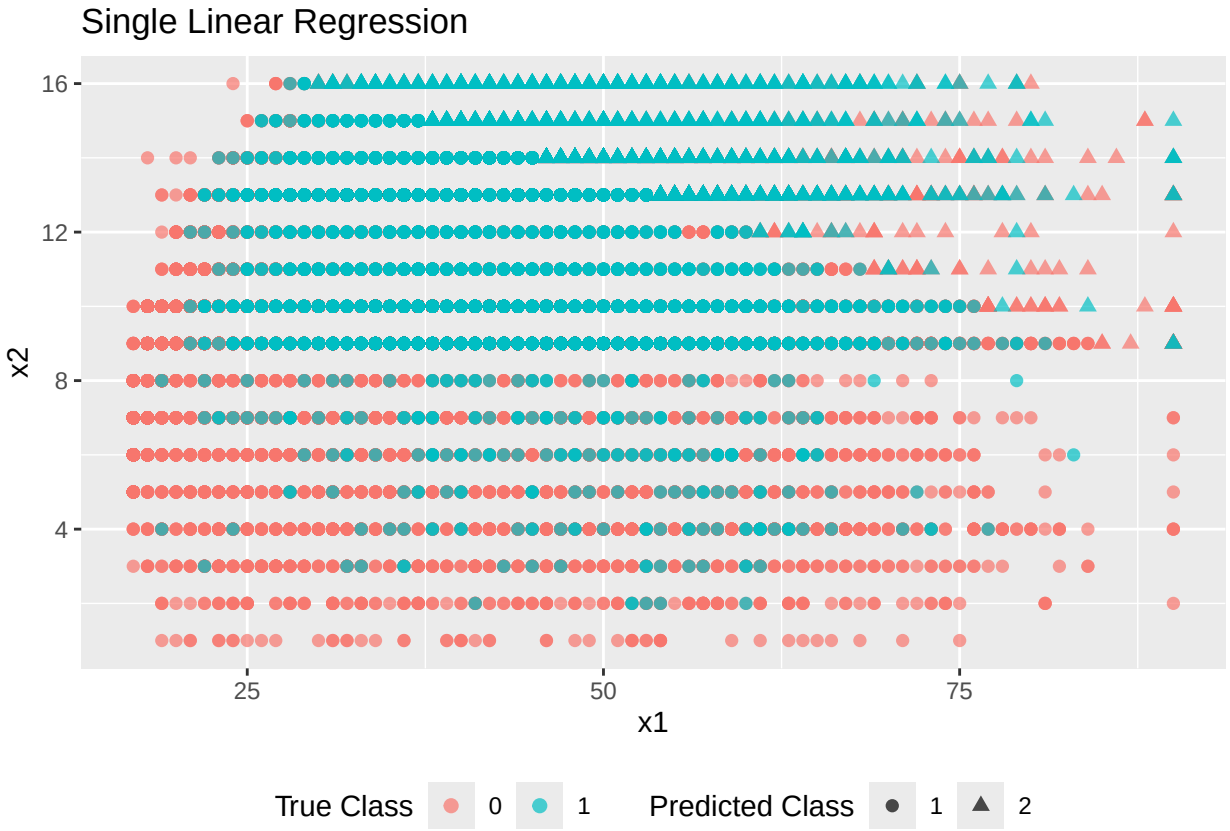
```r
x3 <- X[, 3]
x4 <- X[, 4]

means <- by(data[, .(x1, x2, x3, x4)], data$y, colMeans)
means_matrix <- do.call(rbind, means)

#compute between-class and within-class scatter matrices
grand_mean <- colMeans(data[, .(x1, x2, x3, x4)])
B <- t(means_matrix - grand_mean) %*% (means_matrix - grand_mean)
W <- var(X)

#eigen-decomposition of W^{-1}B
eig <- eigen(solve(W) %*% B)
eig$vectors  # Discriminant directions
```

```
##              [,1]        [,2]        [,3]         [,4]
## [1,] -0.006674496  0.09035144 -0.14982961 -0.717965040
## [2,] -0.031573979  0.77919431  0.77989452 -0.008320287
## [3,] -0.006475414  0.09689673 -0.07092993  0.694650358
## [4,]  0.999458155 -0.61262049 -0.60355992  0.043793310
```

```r
#single linear regression
linear_model <- lm(as.numeric(y) ~ x1 + x2, data = data)
linear_pred <- as.numeric(predict(linear_model, data))

linear_pred_class <- ifelse(linear_pred < 1.5, 1, ifelse(linear_pred < 2.5, 2, 3))
table("Linear Regression" = linear_pred_class, "True" = data$y)
```

```
##                  True
## Linear Regression     0     1
##                 1 23873  6398
##                 2   846  1443
```

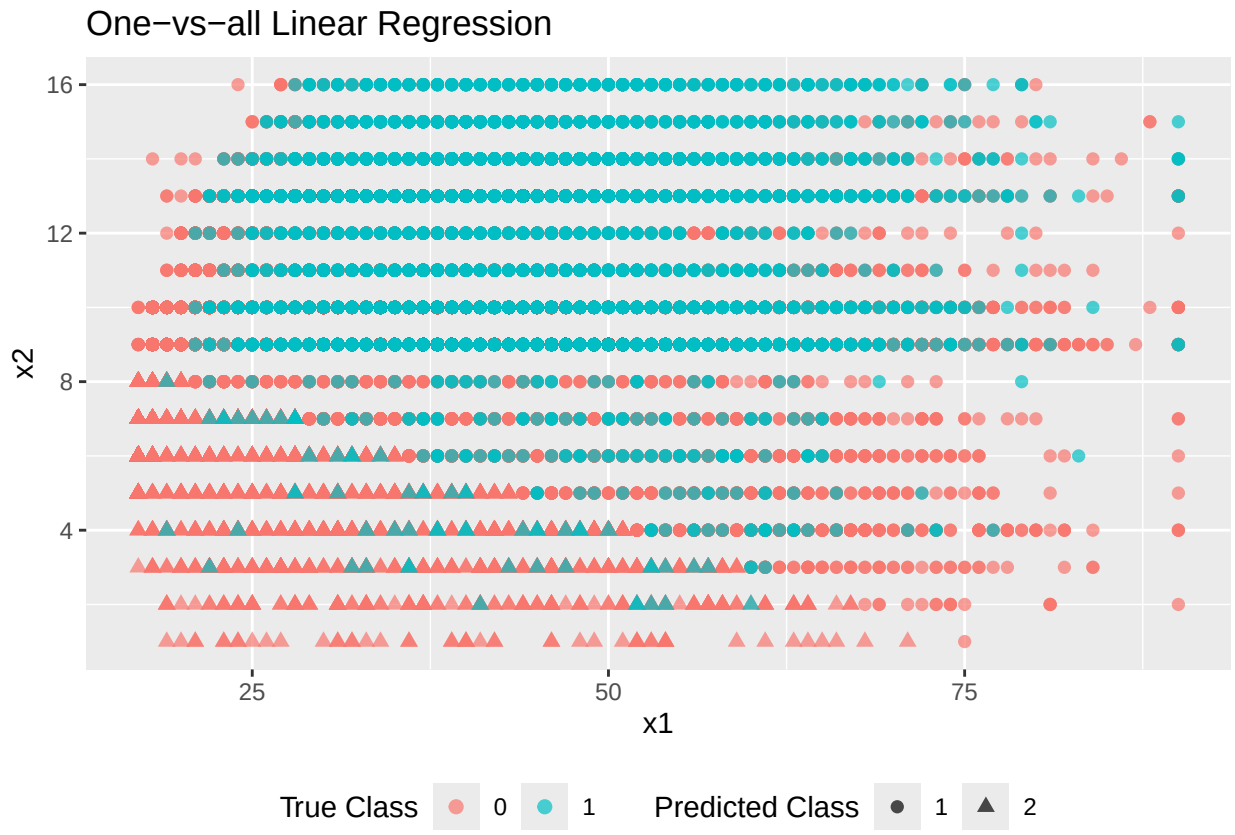
```r
result1 <- cbind(data,Linear_Regression=as.factor(linear_pred_class))
ggplot(result1, aes(x = x1, y = x2, color = y)) +
  geom_point(aes(shape = Linear_Regression), size = 2, alpha = 0.7) +
  scale_shape_manual(values = c(16, 17, 18, 19, 20, 21)) +
  ggtitle("Single Linear Regression") +
  labs(color = "True Class", shape = "Predicted Class") +
  theme(legend.position = "bottom")
```




```
#one-vs-all linear regression

#linear regression for class 1
linear_model_1 <- lm(I(y == 1) ~ x1 + x2, data = data)
linear_pred_1 <- predict(linear_model_1, data)

#for class 2
linear_model_2 <- lm(I(y == 2) ~ x1 + x2, data = data)
linear_pred_2 <- predict(linear_model_2, data)

#combine
linear_pred <- data.frame(
  class1 = linear_pred_1,
  class2 = linear_pred_2
)

#highest predicted value
linear_pred_class <- apply(linear_pred, 1, function(x) names(x)[which.max(x)])
linear_pred_class <- gsub("class","",linear_pred_class)

table("Linear Regression" = linear_pred_class, "True" = data$y)
```

```
##                   True
## Linear Regression     0     1
##                 1 22430  7780
##                 2  2289    61
```

```
result2 <- cbind(data,Linear_Regression=linear_pred_class)
ggplot(result2, aes(x = x1, y = x2, color = y)) +
  geom_point(aes(shape = Linear_Regression), size = 2, alpha = 0.7) +
  scale_shape_manual(values = c(16, 17, 18, 19, 20, 21)) +
  ggtitle("One-vs-all Linear Regression") +
  labs(color = "True Class", shape = "Predicted Class") +
  theme(legend.position = "bottom")
```




```
#confusion matrices
dc_data <- as.matrix(data[,1:2])%*%as.matrix(lda_model$scaling)
dc_data <- as.data.frame(dc_data)
dc_data$True <- data$y
dc_data$LDA <- lda_pred$class
dc_data$Linear_Regression <- factor(linear_pred_class)
head(dc_data)
```

```
##        LD1 True LDA Linear_Regression
## 1 6.701477    0   1                 1
## 2 4.789193    0   0                 1
## 3 4.761927    0   0                 1
## 4 5.728368    0   0                 1
## 5 6.471832    0   0                 1
## 6 3.894250    0   0                 1
```

```
#error: object 'LD2' not found
#ggplot(dc_data, aes(x = LD1, y = LD2, color = True)) +
#   geom_point() +
#  geom_point(aes( shape = LDA), size = 4, alpha = 0.3) +
#  geom_point(aes(shape = Linear_Regression), size = 2, alpha = 0.7) +
#  scale_shape_manual(values = c(16, 17, 18, 19, 20, 21)) +
#  ggtitle("Comparison of LDA and Linear Regression") +
#  labs(color = "True Class", shape = "Predicted Class") +
#  theme(legend.position = "bottom")

#logistic regression for multinomial
library(tidyverse)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(nnet)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
str(data)
```

```
## Classes 'data.table' and 'data.frame':   32560 obs. of  3 variables:
##  $ x1: int  50 38 53 28 37 49 23 32 34 25 ...
##  $ x2: int  13 9 7 13 14 5 13 12 4 9 ...
##  $ y : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
set.seed(100)
train.sample1 <- sample(seq_len(nrow(data)), size=0.8*nrow(data))

length(train.sample1)
```

```
## [1] 26048
```

```
set.seed(100)
train.sample2 <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE, prob=c(0.8,0.2))
sum(train.sample2)
```

```
## [1] 26143
```

```
train.data <- data[train.sample1, ]
dim(train.data)
```

```
## [1] 26048     3
```

```
test.data <- data[-train.sample1, ]
dim(test.data)
```

```
## [1] 6512    3
```

```
#one vs. all
y.uni <- as.character(unique(data$y))
y.uni
```

```
## [1] "0" "1"
```

```
#create an empty list to hold each binary classifier
classifiers_ova <- list()
train.data2 <- train.data
head(train.data2)
```

```
##       x1    x2      y
##    <int> <int> <fctr>
## 1:    62     9      0
## 2:    21    10      0
## 3:    57    13      0
## 4:    25    13      0
## 5:    43    13      1
## 6:    33    13      1
```

```
#loop to train a classifier for each class
#for (k in y.uni) {
#  y_binary <- ifelse(train.data2$y == k, 1, 0)
#  assign("train.data2", within(train.data2, assign(k, y_binary)))
#  formula.glm <- paste(k,"~ x1 + x2")
#  fit <- glm(formula.glm, data = train.data2, family = binomial)
#  classifiers_ova[[k]] <- fit
#}

for (k in y.uni) {
  y_binary <- ifelse(train.data2$y == k, 1, 0)
  formula_str <- paste("y_binary ~ x1 + x2")
  temp_data_for_glm <- data.frame(
    y_binary = y_binary,
    x1 = train.data2$x1,
    x2 = train.data2$x2
  )
  fit <- glm(as.formula(formula_str), data = temp_data_for_glm, family = binomial)
  classifiers_ova[[k]] <- fit
```

```
  rm(temp_data_for_glm)
}

#prediction function for OvA
predict_OvA <- function(newdata, classifiers) {
  scores <- sapply(classifiers, function(fit) predict(fit, newdata = data.frame(newdata), type = "respo
  return(y.uni[apply(scores, 1, which.max)])
}

#test the prediction function
OvA_pred <- predict_OvA(test.data, classifiers_ova)

mean(OvA_pred == test.data$y)
```

```
## [1] 0.7862408
```

```
#confusion matrix
conf.ova <- confusionMatrix(as.factor(OvA_pred), test.data$y)
print(conf.ova)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4685 1075
##          1  317  435
##
##                Accuracy : 0.7862
##                  95% CI : (0.7761, 0.7961)
##     No Information Rate : 0.7681
##     P-Value [Acc > NIR] : 0.0002508
##
##                   Kappa : 0.2724
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9366
##             Specificity : 0.2881
##          Pos Pred Value : 0.8134
##          Neg Pred Value : 0.5785
##              Prevalence : 0.7681
##          Detection Rate : 0.7194
##    Detection Prevalence : 0.8845
##       Balanced Accuracy : 0.6124
##
##        'Positive' Class : 0
##
```

```
#multinomial/softmaxx
logit_model <- nnet::multinom(y ~., data = train.data)
```

```
## # weights:  4 (3 variable)
```

```
## initial  value 18055.097759
## final  value 12156.894240
## converged
```

```
summary(logit_model)
```

```
## Call:
## nnet::multinom(formula = y ~ ., data = train.data)
##
## Coefficients:
##                   Values   Std. Err.
## (Intercept) -6.74882021 0.099163241
## x1           0.04306663 0.001218634
## x2           0.36489561 0.007129258
##
## Residual Deviance: 24313.79
## AIC: 24319.79
```

```
logit_pred <- logit_model %>% predict(test.data)
head(logit_pred)
```

```
## [1] 0 0 0 0 0 0
## Levels: 0 1
```

```
#levels
mean(logit_pred == test.data$y)
```

```
## [1] 0.7862408
```

```
conf.logit <- confusionMatrix(as.factor(logit_pred), test.data$y)
print(conf.logit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4684 1074
##          1  318  436
##
##                Accuracy : 0.7862
##                  95% CI : (0.7761, 0.7961)
##     No Information Rate : 0.7681
##     P-Value [Acc > NIR] : 0.0002508
##
##                   Kappa : 0.2729
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9364
##             Specificity : 0.2887
##          Pos Pred Value : 0.8135
```

```
##          Neg Pred Value : 0.5782
##              Prevalence : 0.7681
##          Detection Rate : 0.7193
##    Detection Prevalence : 0.8842
##       Balanced Accuracy : 0.6126
##
##        'Positive' Class : 0
##
```

```
#LDA
lda_model <- lda(y ~ ., data = train.data)

lda_pred <- lda_model %>%
  predict(test.data) %>%
  `[[`("class")

lda_pred <- lda_model %>%
  predict(test.data) %>%
  pluck("class")

mean(lda_pred==test.data$y)
```

```
## [1] 0.7847052
```

```
conf.lda <- confusionMatrix(as.factor(lda_pred), test.data$y)
print(conf.lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4674 1074
##          1  328  436
##
##                Accuracy : 0.7847
##                  95% CI : (0.7745, 0.7946)
##     No Information Rate : 0.7681
##     P-Value [Acc > NIR] : 0.000734
##
##                   Kappa : 0.2697
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9344
##             Specificity : 0.2887
##          Pos Pred Value : 0.8132
##          Neg Pred Value : 0.5707
##              Prevalence : 0.7681
##          Detection Rate : 0.7178
##    Detection Prevalence : 0.8827
##       Balanced Accuracy : 0.6116
##
##        'Positive' Class : 0
##
```

```r
#linear
linear_model_1 <- lm(I(y == 1) ~ ., data = train.data)
linear_model_2 <- lm(I(y == 2) ~ ., data = train.data)

linear_pred_1 <- predict(linear_model_1, test.data)
linear_pred_2 <- predict(linear_model_2, test.data)

linear_pred <- data.frame(
  1 == linear_pred_1,
  2 == linear_pred_2
)

linear_pred_class <- apply(linear_pred, 1, function(x) names(x)[which.max(x)])
mean(linear_pred_class==test.data$y)
```

```
## [1] 0
```

```r
#plot
dc_data <- as.matrix(test.data[,1:2])%*%as.matrix(lda_model$scaling)
dc_data <- as.data.frame(dc_data)
dc_data$True <- test.data$y
dc_data$Logit <- logit_pred
dc_data$LDA <- lda_pred
dc_data$Linear <- factor(linear_pred_class)
head(dc_data)
```

```
##        LD1 True Logit LDA              Linear
## 1 6.483500    0     0   0 X1....linear_pred_1
## 2 4.096457    0     0   0 X1....linear_pred_1
## 3 5.492290    0     0   0 X1....linear_pred_1
## 4 3.958201    0     0   0 X1....linear_pred_1
## 5 5.617021    0     0   0 X1....linear_pred_1
## 6 5.185472    0     0   0 X1....linear_pred_1
```

```r
#error: object LD2 not found
#ggplot(dc_data, aes(x = LD1, y = LD2, color = True)) +
#  geom_point(aes(shape = Logit),  size = 4, alpha = 0.3) +
#  geom_point(aes(shape = LDA),  size = 2, alpha = 0.9) +
#  # geom_point(aes(shape = Linear), size = 2, alpha = 0.7) +
#  scale_shape_manual(values = c(16, 17, 18, 19, 20, 21)) +
#  ggtitle("PCA Visualization of Actual and Predicted Classes") +
#  labs(shape = "Predicted Class",
#       color = "Actual Class") +
#  theme(legend.position = "bottom")
```

```r
library(MASS)
library(datarium)
library(ggplot2)
library(broom)
library(ggfortify)
library(tidyverse)
library(mvnormtest)
```

```r
library(data.table)
library(gridExtra)
library(dplyr)
library(tinytex)

#logistic regression for multinomial
library(tidyverse)
library(caret)
library(nnet)
library(pROC)

data <- read.csv("C:/Users/harip/Downloads/predict+students+dropout+and+academic+success/data.csv", sep=
data <- select(data, Previous.qualification..grade., Admission.grade, Target)

str(data)
```

```
## 'data.frame':    4424 obs. of  3 variables:
##  $ Previous.qualification..grade.: num  122 160 122 122 100 ...
##  $ Admission.grade               : num  127 142 125 120 142 ...
##  $ Target                        : chr  "Dropout" "Graduate" "Dropout" "Graduate" ...
```

```r
set.seed(100)
train.sample1 <- sample(seq_len(nrow(data)), size=0.8*nrow(data))

length(train.sample1)
```

```
## [1] 3539
```

```r
set.seed(100)
train.sample2 <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE, prob=c(0.8,0.2))
sum(train.sample2)
```

```
## [1] 3527
```

```r
train.data <- data[train.sample1, ]
dim(train.data)
```

```
## [1] 3539    3
```

```r
test.data <- data[-train.sample1, ]
dim(test.data)
```

```
## [1] 885   3
```

```r
#one vs. all
Target.uni <- as.character(unique(data$Target))
Target.uni
```

```
## [1] "Dropout"  "Graduate" "Enrolled"
```

```r
#create an empty list to hold each binary classifier
classifiers_ova <- list()
train.data2 <- train.data
head(train.data2)
```

```
##      Previous.qualification..grade. Admission.grade   Target
## 3786                          150.0           150.0 Enrolled
## 503                           125.0           132.0 Enrolled
## 3430                          140.0           140.0  Dropout
## 3696                          133.1           149.4  Dropout
## 4090                          133.1           100.0  Dropout
## 3052                          139.0           112.3 Enrolled
```

```r
#loop to train a classifier for each class
for (k in Target.uni) {
  Target_binary <- ifelse(train.data2$Target == k, 1, 0)
  assign("train.data2", within(train.data2, assign(k, Target_binary)))
  formula.glm <- paste(k,"~ Previous.qualification..grade. + Admission.grade")
  fit <- glm(formula.glm, data = train.data2, family = binomial)
  classifiers_ova[[k]] <- fit
}

#prediction function for OvA
predict_OvA <- function(newdata, classifiers) {
  scores <- sapply(classifiers, function(fit) predict(fit, newdata = data.frame(newdata), type = "respo
  return(Target.uni[apply(scores, 1, which.max)])
}

#test the prediction function
OvA_pred <- predict_OvA(test.data, classifiers_ova)

mean(OvA_pred == test.data$Target)
```

```
## [1] 0.4960452
```

```r
all_possible_classes <- c("Dropout", "Enrolled", "Graduate")
OvA_pred <- factor(OvA_pred, levels = all_possible_classes)
test.data$Target <- factor(test.data$Target, levels = all_possible_classes)

#confusion matrix
conf.ova <- confusionMatrix(as.factor(OvA_pred), as.factor(test.data$Target))
print(conf.ova)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Dropout Enrolled Graduate
##   Dropout       16        6       15
##   Enrolled       0        0        0
##   Graduate     286      139      423
##
## Overall Statistics
```

```
##
##                Accuracy : 0.496
##                  95% CI : (0.4626, 0.5295)
##     No Information Rate : 0.4949
##     P-Value [Acc > NIR] : 0.4865
##
##                   Kappa : 0.0148
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: Dropout Class: Enrolled Class: Graduate
## Sensitivity                 0.05298          0.0000         0.96575
## Specificity                 0.96398          1.0000         0.04922
## Pos Pred Value              0.43243             NaN         0.49882
## Neg Pred Value              0.66274          0.8362         0.59459
## Prevalence                  0.34124          0.1638         0.49492
## Detection Rate              0.01808          0.0000         0.47797
## Detection Prevalence        0.04181          0.0000         0.95819
## Balanced Accuracy           0.50848          0.5000         0.50749
```

```
#multinomial/softmaxx
logit_model <- nnet::multinom(Target ~., data = train.data)
```

```
## # weights:  12 (6 variable)
## initial  value 3887.988890
## iter  10 value 3579.115052
## final  value 3579.114642
## converged
```

```
summary(logit_model)
```

```
## Call:
## nnet::multinom(formula = Target ~ ., data = train.data)
##
## Coefficients:
##          (Intercept) Previous.qualification..grade. Admission.grade
## Enrolled  -0.8297618                   -0.001024065     0.003348595
## Graduate  -2.6840321                    0.010153848     0.014173979
##
## Std. Errors:
##          (Intercept) Previous.qualification..grade. Admission.grade
## Enrolled   0.5494669                    0.004584842     0.004212838
## Graduate   0.4273548                    0.003583254     0.003264914
##
## Residual Deviance: 7158.229
## AIC: 7170.229
```

```
logit_pred <- logit_model %>% predict(test.data)
head(logit_pred)
```

```
## [1] Graduate Graduate Graduate Graduate Graduate Dropout
## Levels: Dropout Enrolled Graduate
```

```
#levels
mean(logit_pred == test.data$Target)
```

```
## [1] 0.4949153
```

```
conf.logit <- confusionMatrix(as.factor(logit_pred), test.data$Target)
print(conf.logit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Dropout Enrolled Graduate
##   Dropout       15        6       15
##   Enrolled       0        0        0
##   Graduate     287      139      423
##
## Overall Statistics
##
##                Accuracy : 0.4949
##                  95% CI : (0.4615, 0.5284)
##     No Information Rate : 0.4949
##     P-Value [Acc > NIR] : 0.5134
##
##                   Kappa : 0.0122
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: Dropout Class: Enrolled Class: Graduate
## Sensitivity                 0.04967          0.0000         0.96575
## Specificity                 0.96398          1.0000         0.04698
## Pos Pred Value              0.41667             NaN         0.49823
## Neg Pred Value              0.66196          0.8362         0.58333
## Prevalence                  0.34124          0.1638         0.49492
## Detection Rate              0.01695          0.0000         0.47797
## Detection Prevalence        0.04068          0.0000         0.95932
## Balanced Accuracy           0.50682          0.5000         0.50637
```

```
#LDA
lda_model <- lda(Target ~ ., data = train.data)

lda_pred <- lda_model %>%
  predict(test.data) %>%
  `[[`("class")

lda_pred <- lda_model %>%
  predict(test.data) %>%
  pluck("class")
```

```r
mean(lda_pred==test.data$Target)
```

```
## [1] 0.4949153
```

```r
conf.lda <- confusionMatrix(as.factor(lda_pred), test.data$Target)
print(conf.lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Dropout Enrolled Graduate
##   Dropout       15        6       15
##   Enrolled       0        0        0
##   Graduate     287      139      423
##
## Overall Statistics
##
##                Accuracy : 0.4949
##                  95% CI : (0.4615, 0.5284)
##     No Information Rate : 0.4949
##     P-Value [Acc > NIR] : 0.5134
##
##                   Kappa : 0.0122
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: Dropout Class: Enrolled Class: Graduate
## Sensitivity                 0.04967          0.0000         0.96575
## Specificity                 0.96398          1.0000         0.04698
## Pos Pred Value              0.41667             NaN         0.49823
## Neg Pred Value              0.66196          0.8362         0.58333
## Prevalence                  0.34124          0.1638         0.49492
## Detection Rate              0.01695          0.0000         0.47797
## Detection Prevalence        0.04068          0.0000         0.95932
## Balanced Accuracy           0.50682          0.5000         0.50637
```

```r
#linear
linear_model_Dropout <- lm(I(Target == "Dropout") ~ ., data = train.data)
linear_model_Enrolled <- lm(I(Target == "Enrolled") ~ ., data = train.data)
linear_model_Graduate <- lm(I(Target == "Graduate") ~ ., data = train.data)

linear_pred_Dropout <- predict(linear_model_Dropout, test.data)
linear_pred_Enrolled <- predict(linear_model_Enrolled, test.data)
linear_pred_Graduate <- predict(linear_model_Graduate, test.data)

linear_pred <- data.frame(
  Dropout = linear_pred_Dropout,
  Enrolled = linear_pred_Enrolled,
  Graduate = linear_pred_Graduate
)
```

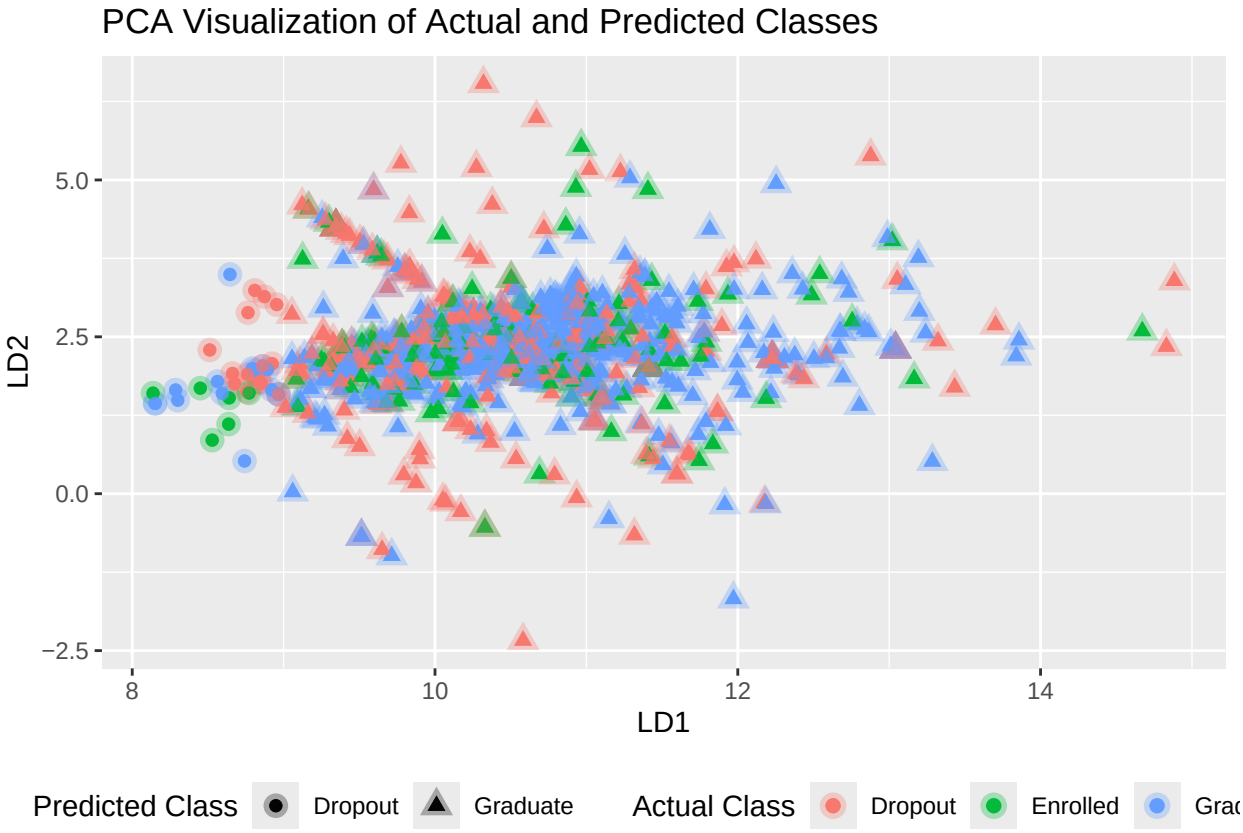
```r
linear_pred_class <- apply(linear_pred, 1, function(x) names(x)[which.max(x)])
mean(linear_pred_class==test.data$Target)
```

```
## [1] 0.4949153
```

```r
#plot
dc_data <- as.matrix(test.data[,1:2])%*%as.matrix(lda_model$scaling)
dc_data <- as.data.frame(dc_data)
dc_data$True <- test.data$Target
dc_data$Logit <- logit_pred
dc_data$LDA <- lda_pred
dc_data$Linear <- factor(linear_pred_class)
head(dc_data)
```

```
##          LD1      LD2     True    Logit      LDA   Linear
## 8   9.431833 2.130359  Dropout Graduate Graduate Graduate
## 15 11.604585 2.986213 Graduate Graduate Graduate Graduate
## 22 10.115729 2.196106 Enrolled Graduate Graduate Graduate
## 24  9.730084 2.513462 Graduate Graduate Graduate Graduate
## 32 10.416122 1.446739 Graduate Graduate Graduate Graduate
## 43  8.286662 1.652711 Graduate  Dropout  Dropout  Dropout
```

```r
ggplot(dc_data, aes(x = LD1, y = LD2, color = True)) +
  geom_point(aes(shape = Logit),  size = 4, alpha = 0.3) +
  geom_point(aes(shape = LDA),  size = 2, alpha = 0.9) +
  geom_point(aes(shape = Linear), size = 2, alpha = 0.7) +
  scale_shape_manual(values = c(16, 17, 18, 19, 20, 21)) +
  ggtitle("PCA Visualization of Actual and Predicted Classes") +
  labs(shape = "Predicted Class",
       color = "Actual Class") +
  theme(legend.position = "bottom")
```


PCA Visualization of Actual and Predicted Classes
LD2
5.0
2.5
0.0
−2.5
8
10
12
14
LD1
Predicted Class
Dropout
Graduate
Actual Class
Dropout
Enrolled
Grad