# Classification II

Yifei Sun, Runze Cui

## Contents

**Diabetes data** **2**
- LDA . . . 3
- QDA . . . 6
- Naive Bayes (NB) . . . 7
- Model comparison . . . 9

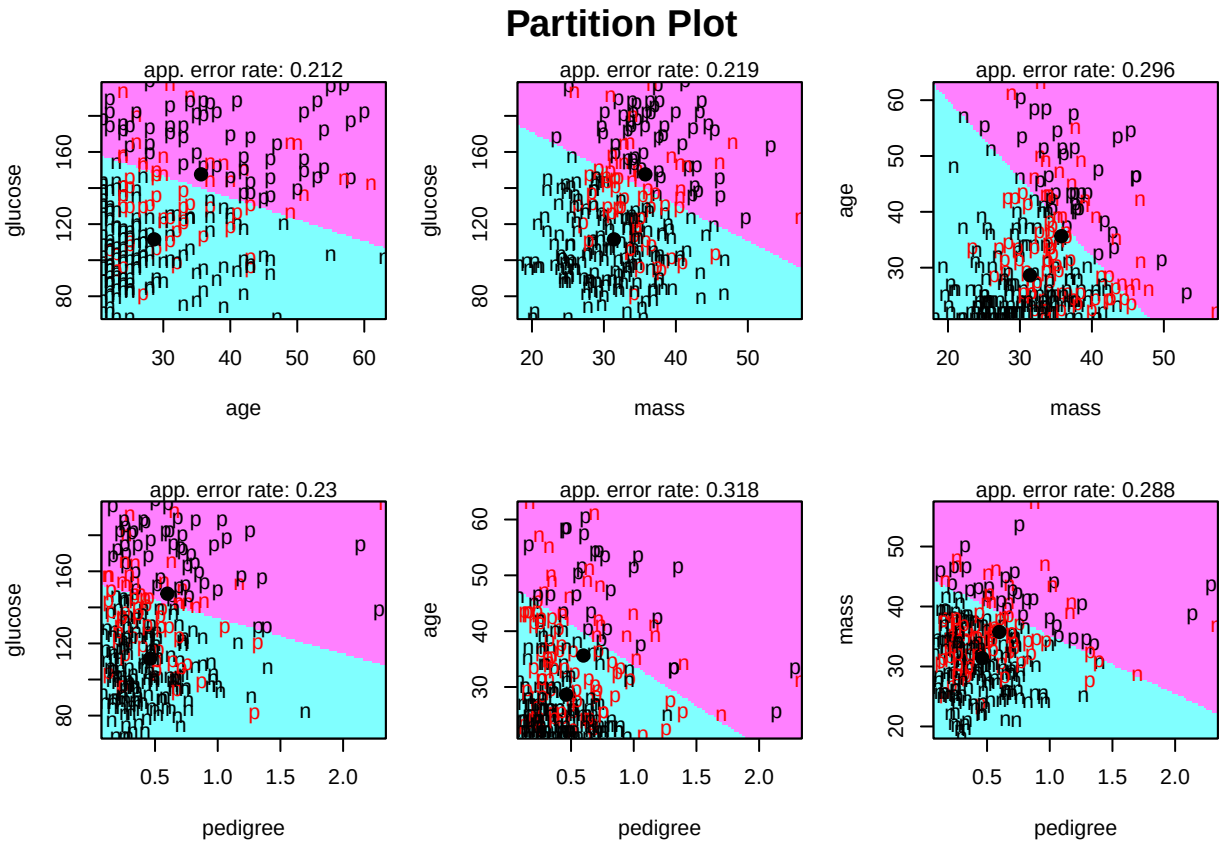
**Iris data (K = 3)** **11**

```
library(caret)
library(tidymodels)
library(discrim)
library(MASS)
library(mlbench)
library(pROC)
library(klaR)
library(plotmo)
```

## Diabetes data

We use the Pima Indians Diabetes Database for illustration. The data contain 768 observations and 9 variables. The outcome is a binary variable `diabetes`. We start from some simple visualization of the data.

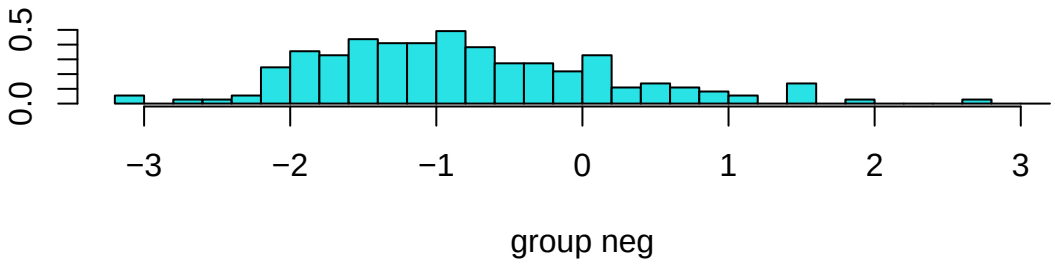
```
data(PimaIndiansDiabetes2)
dat <- na.omit(PimaIndiansDiabetes2)

set.seed(1)
data_split <- initial_split(dat, prop = 0.7)

# Extract the training and test data
training_data <- training(data_split)
testing_data <- testing(data_split)

# Exploratory analysis: LDA based on every combination of two variables
partimat(diabetes ~ glucose + age + mass + pedigree,
         data = training_data, method = "lda")
```



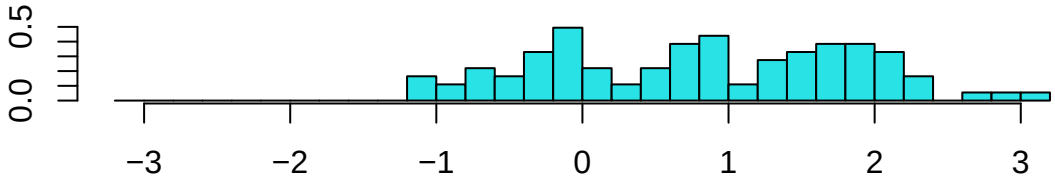

## LDA

We use the function `lda` in library `MASS` to conduct LDA.

```
lda.fit <- lda(diabetes~., data = training_data)
plot(lda.fit)
```




group neg




group pos

```
lda.fit$scaling
```

```
##                   LD1
## pregnant  0.0168948055
## glucose   0.0300129787
## pressure  0.0078546614
## triceps   0.0078028827
## insulin  -0.0008858014
## mass      0.0478547194
## pedigree  0.6026890812
## age       0.0261954726
```

```
head(predict(lda.fit)$x)
```

```
##            LD1
## 634 -0.6665273
## 332 -1.6022713
## 274 -1.5973480
## 575  0.4052179
## 528 -1.0121002
## 374 -0.7711331
```

```
mean(predict(lda.fit)$x)
```

```
## [1] -5.566864e-17
```

```
dat_t <- training_data
x_n_tr <- dat_t[dat_t$diabetes == "neg", 1:8]
x_p_tr <- dat_t[dat_t$diabetes == "pos", 1:8]
cov.neg <- cov(x_n_tr)
cov.pos <- cov(x_p_tr)
n.neg <- nrow(x_n_tr)
n.pos <- nrow(x_p_tr)
n <- n.neg + n.pos
```

```r
K <- 2
W <- 1/(n - K) * (cov.neg * (n.neg - 1) + cov.pos * (n.pos - 1))
t(lda.fit$scaling) %*% W %*% lda.fit$scaling
```

```
##     LD1
## LD1   1
```

```r
lda.pred <- predict(lda.fit, newdata = testing_data)
head(lda.pred$posterior)
```

```
##           neg        pos
## 7  0.98353968 0.01646032
## 9  0.08112538 0.91887462
## 14 0.16358294 0.83641706
## 15 0.23122065 0.76877935
## 17 0.54224666 0.45775334
## 19 0.87241326 0.12758674
```

Using caret:

```r
ctrl <- trainControl(method = "repeatedcv", repeats = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

set.seed(11)
model.lda <- train(x = training_data[, 1:8],
                   y = training_data$diabetes,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)

lda.pred2 <- predict(model.lda, newdata = testing_data, type = "prob")
head(lda.pred2)
```

```
##           neg        pos
## 7  0.98353968 0.01646032
## 9  0.08112538 0.91887462
## 14 0.16358294 0.83641706
## 15 0.23122065 0.76877935
## 17 0.54224666 0.45775334
## 19 0.87241326 0.12758674
```

Using tidymodels:

```r
set.seed(11)
cv_folds <- vfold_cv(training_data, v = 10, repeats = 5)

# Model specification for LDA
lda_spec <- discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")

# Set up the workflow
lda_workflow <- workflow() %>%
  add_model(lda_spec) %>%
  add_formula(diabetes ~ .)
```

```r
# Fit the model
lda_fit <- lda_workflow %>%
  fit(data = training_data)

# Prediction using test data
lda_pred <- predict(lda_fit, new_data = testing_data, type = "prob")
head(lda_pred)
```

```
## # A tibble: 6 x 2
##   .pred_neg .pred_pos
##       <dbl>     <dbl>
## 1    0.984     0.0165
## 2    0.0811    0.919
## 3    0.164     0.836
## 4    0.231     0.769
## 5    0.542     0.458
## 6    0.872     0.128
```

## QDA

```r
qda.fit <- qda(diabetes~., data = training_data)

qda.pred <- predict(qda.fit, newdata = testing_data)
head(qda.pred$posterior)
```

```
##             neg        pos
## 7  9.901635e-01 0.00983649
## 9  3.490900e-05 0.99996509
## 14 2.729319e-12 1.00000000
## 15 1.014859e-01 0.89851414
## 17 7.432413e-01 0.25675868
## 19 7.971531e-01 0.20284690
```

Using caret:

```r
set.seed(11)
model.qda <- train(x = training_data[, 1:8],
                   y = training_data$diabetes,
                   method = "qda",
                   metric = "ROC",
                   trControl = ctrl)

qda.pred2 <- predict(model.qda, newdata = testing_data, type = "prob")
head(qda.pred2)
```

```
##             neg        pos
## 7  9.901635e-01 0.00983649
## 9  3.490900e-05 0.99996509
## 14 2.729319e-12 1.00000000
## 15 1.014859e-01 0.89851414
## 17 7.432413e-01 0.25675868
## 19 7.971531e-01 0.20284690
```

Using tidymodels:

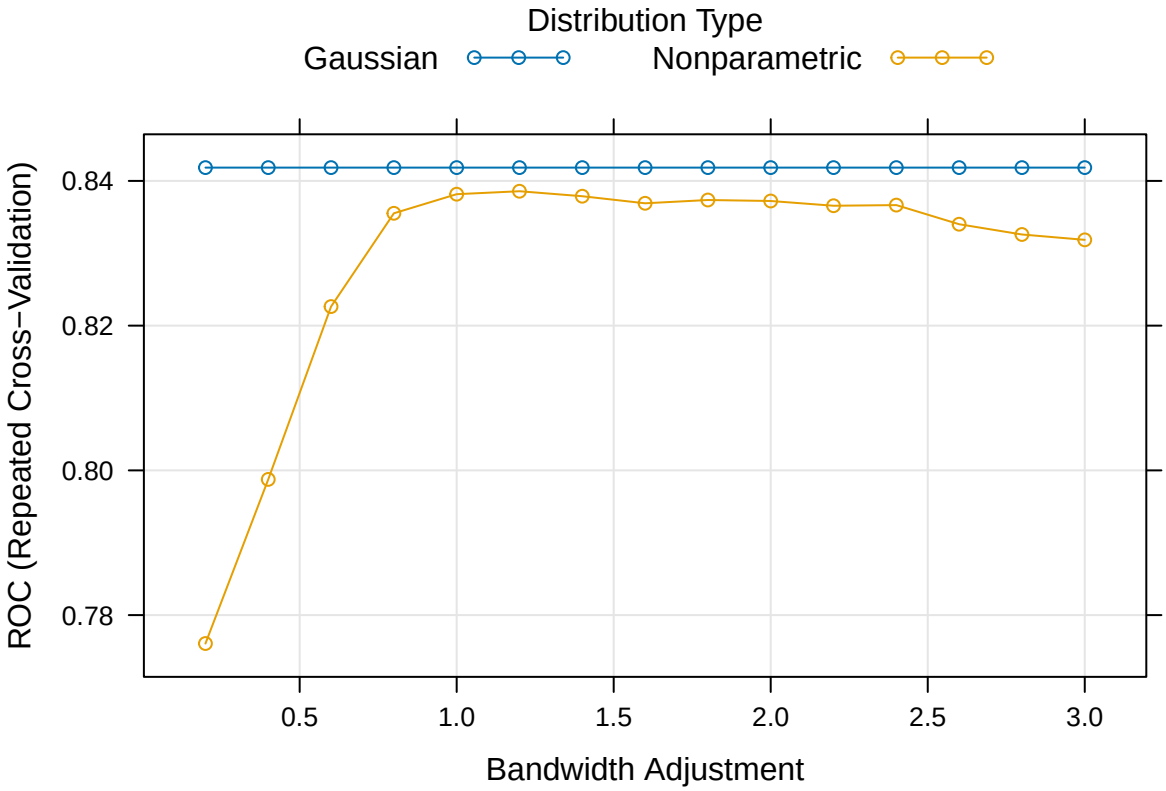
```
# Model specification for QDA
qda_spec <- discrim_quad() %>%
  set_engine("MASS") %>%
  set_mode("classification")

# Set up the workflow
qda_workflow <- workflow() %>%
  add_model(qda_spec) %>%
  add_formula(diabetes ~ .)

# Fit the model
qda_fit <- qda_workflow %>%
  fit(data = training_data)

# Prediction using test data
qda_pred <- predict(qda_fit, new_data = testing_data, type = "prob")
head(qda_pred)
```

```
## # A tibble: 6 x 2
##   .pred_neg .pred_pos
##       <dbl>     <dbl>
## 1  9.90e- 1   0.00984
## 2  3.49e- 5   1.00
## 3  2.73e-12   1.00
## 4  1.01e- 1   0.899
## 5  7.43e- 1   0.257
## 6  7.97e- 1   0.203
```

## Naive Bayes (NB)

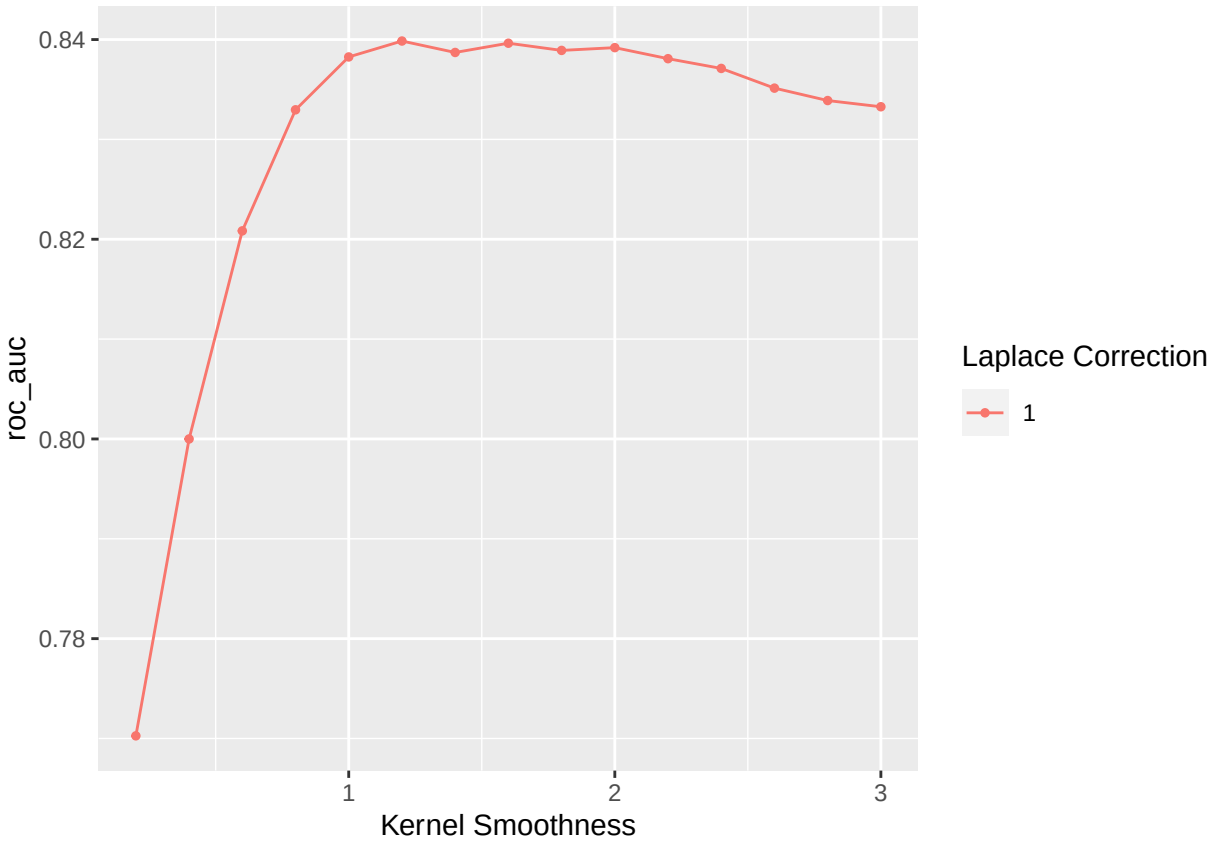
There is one practical issue with the NB classifier when nonparametric estimators are used. When a new data point includes a feature value that never occurs for some response class, the posterior probability can become zero. To avoid this, we increase the count of the value with a zero occurrence to a small value, so that the overall probability doesn't become zero. In practice, a value of one or two is a common choice. This correction is called "Laplace Correction," and is implemented via the parameter `fL`. The parameter `adjust` adjusts the bandwidths of the kernel density estimates, and a larger value means a more flexible estimate.

```
nbGrid <- expand.grid(usekernel = c(FALSE, TRUE),
                      fL = 1,
                      adjust = seq(.2, 3, by = .2))

set.seed(11)
model.nb <- train(x = training_data[, 1:8],
                  y = training_data$diabetes,
                  method = "nb",
                  tuneGrid = nbGrid,
                  metric = "ROC",
                  trControl = ctrl)

plot(model.nb)
```




Using tidymodels:

```
# Model specification for Naive Bayes
nb_spec <- naive_Bayes(smoothness = tune(), Laplace = tune()) %>%
  set_engine("klaR") %>%
  set_mode("classification")

# nb_spec %>% extract_parameter_dials("Laplace")
# nb_spec %>% extract_parameter_dials("smoothness")

# Tuning grid
nb_grid_set <- parameters(Laplace(range = c(1, 1)), smoothness(range = c(0.2, 3)))
nb_grid <- grid_regular(nb_grid_set, levels = c(1, 15))

# Set up the workflow
nb_workflow <- workflow() %>%
  add_model(nb_spec) %>%
  add_formula(diabetes ~ .)

nb_tune <- nb_workflow %>%
  tune_grid(resamples = cv_folds,
            grid = nb_grid)

autoplot(nb_tune, metric = "roc_auc")
```




```
nb_best <- select_best(nb_tune, metric = "roc_auc")

# Update the model spec
final_nb_spec <- nb_spec %>%
  update(Laplace = nb_best$Laplace,
         smoothness = nb_best$smoothness)

nb_fit <- fit(final_nb_spec, formula = diabetes ~ ., data = training_data)
```

## Model comparison

To compare the CV performance across LDA, QDA and NB models in caret:

```
res <- resamples(list(LDA = model.lda, QDA = model.qda, NB = model.nb))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: LDA, QDA, NB
## Number of resamples: 50
##
## ROC
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA 0.4876543 0.8014945 0.8650097 0.8466485 0.9071637 0.9753086    0
## QDA 0.5617284 0.7808642 0.8499025 0.8354756 0.8875000 0.9629630    0
## NB  0.4506173 0.8138158 0.8421053 0.8418395 0.9058642 0.9753086    0
```

```
##
## Sens
##          Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## LDA 0.6666667 0.8333333 0.8947368 0.8829240 0.9444444    1    0
## QDA 0.5555556 0.7777778 0.8421053 0.8458480 0.8932749    1    0
## NB  0.5555556 0.7777778 0.8333333 0.8204094 0.8888889    1    0
##
## Spec
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA 0.2222222 0.5555556 0.5555556 0.6091111 0.7777778 0.8888889    0
## QDA 0.2222222 0.5138889 0.6666667 0.6257778 0.7777778 0.9000000    0
## NB  0.2222222 0.5555556 0.6666667 0.6328889 0.7777778 0.9000000    0
```

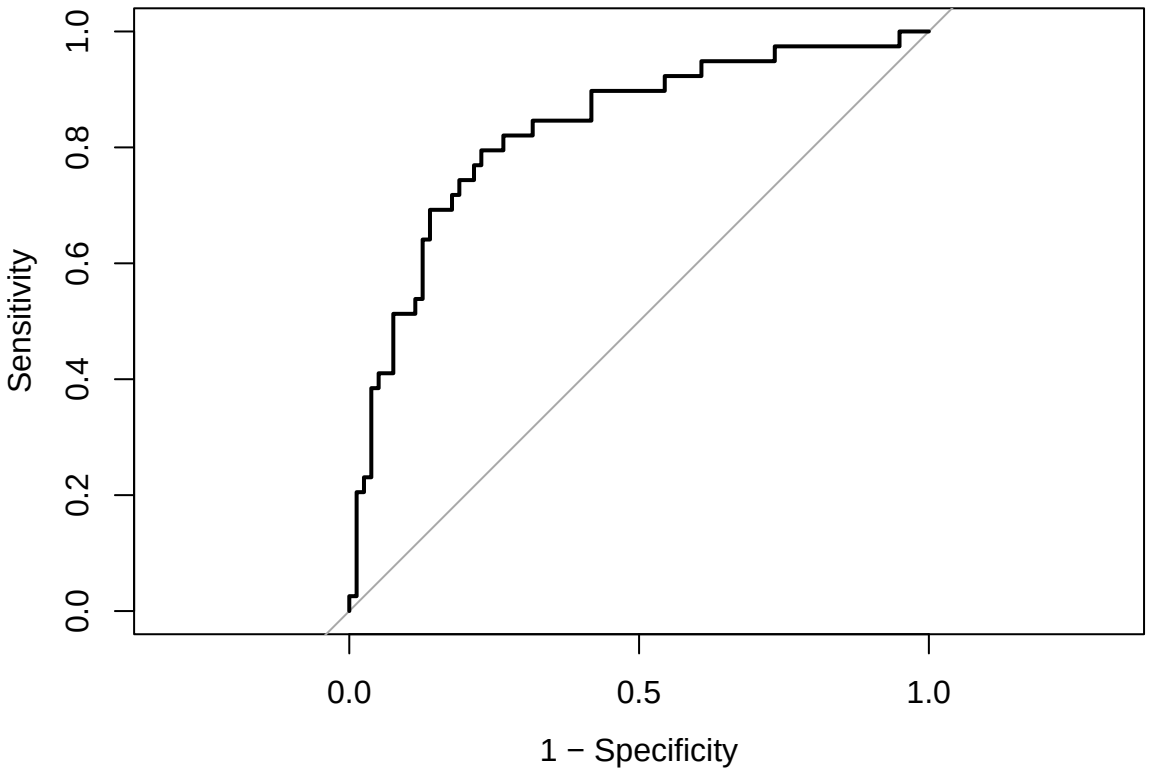
To compare the CV performance across LDA, QDA and NB models in tidymodels:

```
model_compare <- workflow_set(preproc = list(diabetes ~ .),
                              models = list(lda = lda_spec,
                                            qda = qda_spec,
                                            nb = final_nb_spec)) %>%
  workflow_map(resamples = cv_folds) %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  dplyr::select(wflow_id, mean) %>%
  print()
```

```
## # A tibble: 3 x 2
##   wflow_id     mean
##   <chr>       <dbl>
## 1 formula_lda 0.849
## 2 formula_qda 0.825
## 3 formula_nb  0.840
```

Test performance

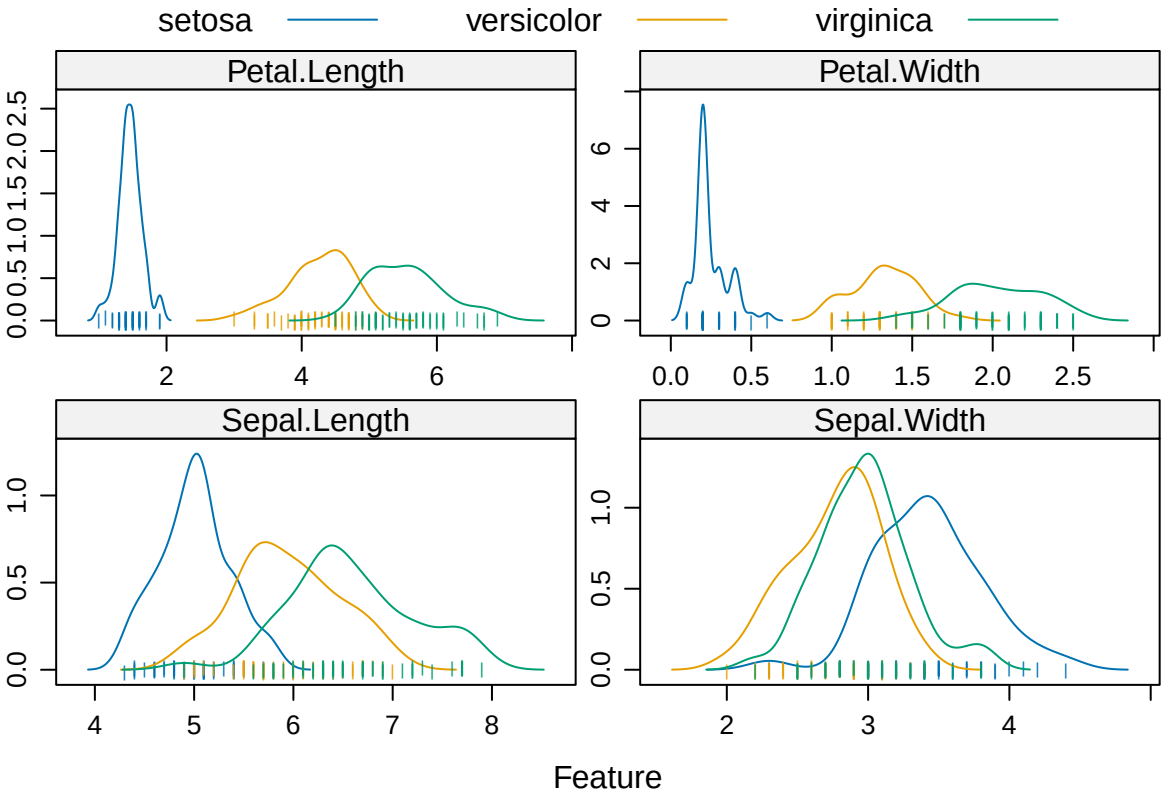
```
roc.lda <- roc(testing_data$diabetes, lda.pred2[,2])
# roc.lda <- roc(testing_data$diabetes, lda_pred$.pred_pos)

plot(roc.lda, legacy.axes = TRUE)
```



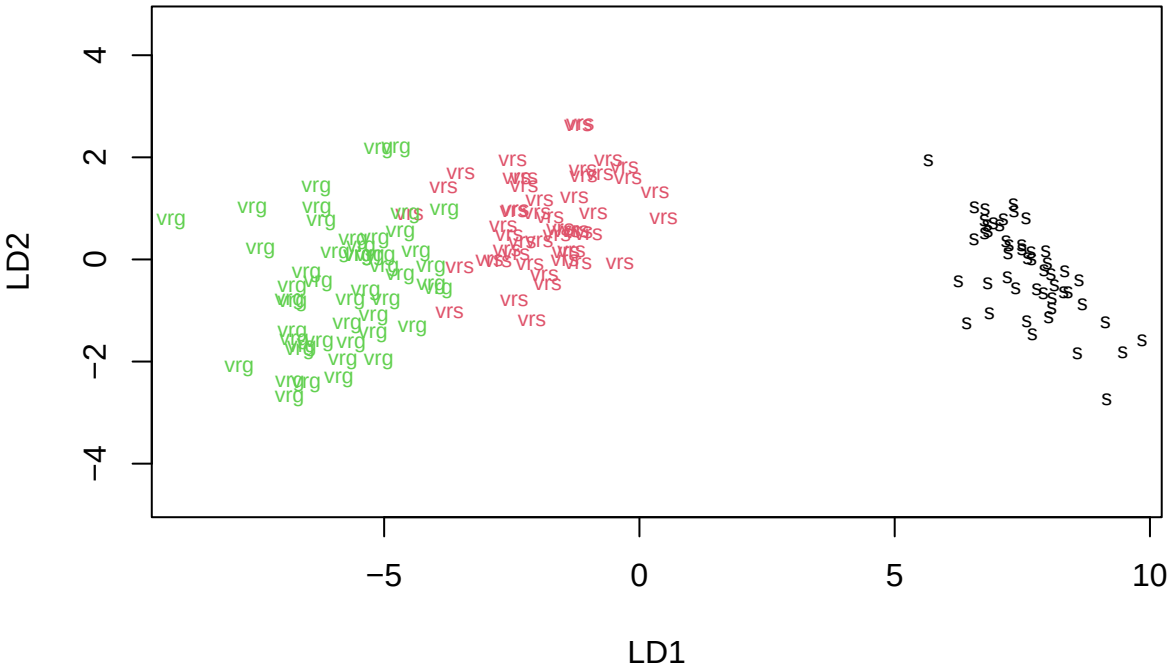

## Iris data (K = 3)

The famous iris data!

```
data(iris)
dat2 <- iris

featurePlot(x = dat2[, 1:4],
            y = dat2$Species,
            scales = list(x=list(relation="free"),
                          y=list(relation="free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 3))
```




```
lda.fit2 <- lda(Species~., data = dat2)
plot(lda.fit2, col = as.numeric(dat2$Species), abbrev = TRUE)
```




```
ctrl2 <- trainControl(method = "cv")

set.seed(1)
model.lda2 <- train(x = dat2[,1:4],
```

```
                    y = dat2$Species,
                    method = "lda",
                    trControl = ctrl2)

set.seed(1)
model.qda2 <- train(x = dat2[,1:4],
                    y = dat2$Species,
                    method = "qda",
                    trControl = ctrl2)



res2 <- resamples(list(LDA = model.lda2,
                       QDA = model.qda2))
summary(res2)
```

```
##
## Call:
## summary.resamples(object = res2)
##
## Models: LDA, QDA
## Number of resamples: 10
##
## Accuracy
##          Min.   1st Qu. Median      Mean 3rd Qu. Max. NA's
## LDA 0.9333333 0.9500000      1 0.9800000       1    1    0
## QDA 0.9333333 0.9333333      1 0.9733333       1    1    0
##
## Kappa
##     Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## LDA  0.9   0.925      1 0.97       1    1    0
## QDA  0.9   0.900      1 0.96       1    1    0
```