

LDA but with TidyModels

Luca Baggi

Load the Libraries

```
library(tidymodels) # for data preprocessing
library(discrim) # for the LDA
library(MASS) # for the LDA
library(ggplot2) # for plotting the distributions
library(gridExtra)
```

Load the data

```
dir = '/Users/luca/OneDrive - Università degli Studi di Milano/uni/data-mining/lect/lect9/DFA_df.txt'
data <- read.delim(dir, sep = ',') # load the data
```

Feature selection & factorisation of the response variable:

```
df <- data %>%
  dplyr::select(-code) %>%
  mutate(y = as.factor(y))
```

I cannot standardise the data before partitioning: that's data leakage!

Visualisations

We need to check whether the data is normally distributed.

```
hist1 <- ggplot(df, aes(x1)) +
  geom_histogram() +
  geom_vline(aes(xintercept = mean(x1))) +
  geom_density()

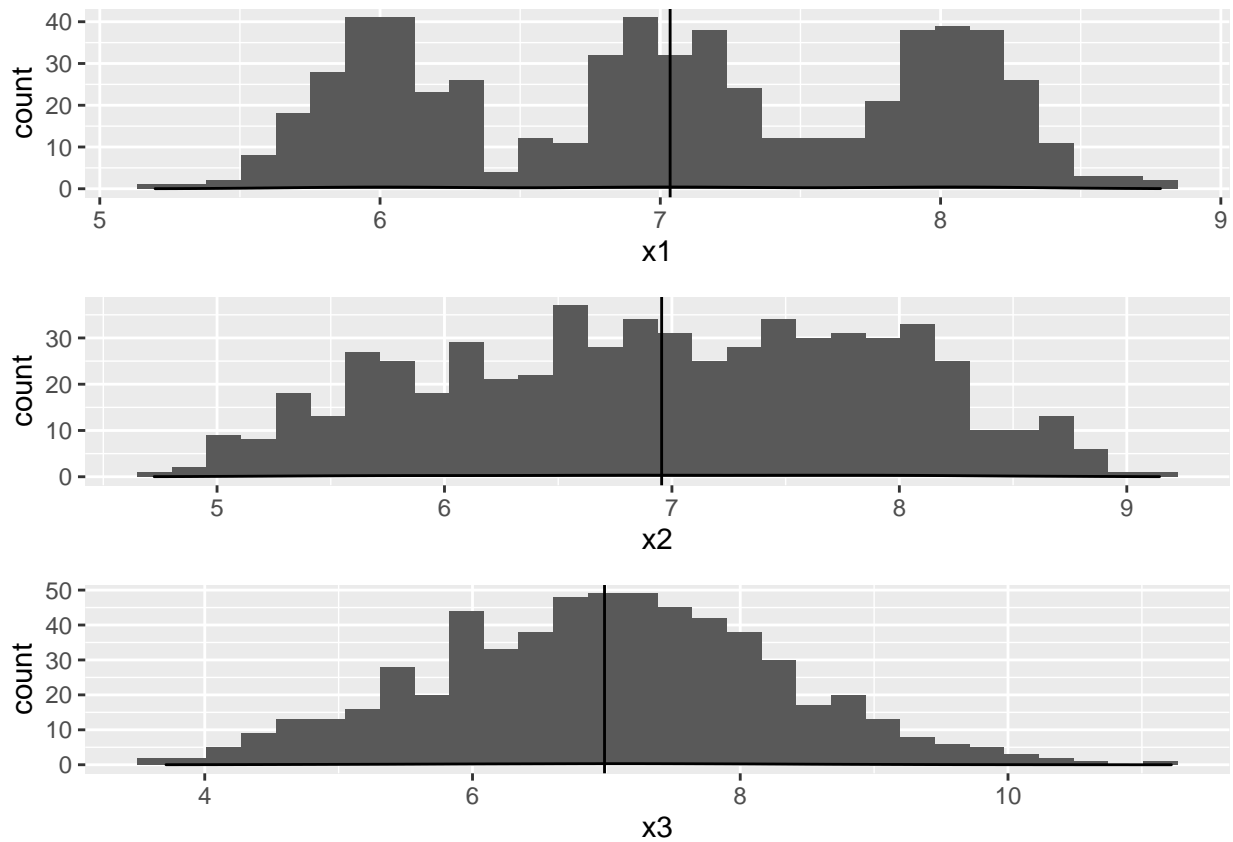
hist2 <- ggplot(df, aes(x2)) +
  geom_histogram() +
  geom_vline(aes(xintercept = mean(x2))) +
  geom_density()

hist3 <- ggplot(df, aes(x3)) +
  geom_histogram() +
  geom_vline(aes(xintercept = mean(x3))) +
  geom_density()
```

Let's plot them altogether

```
grid.arrange(hist1, hist2, hist3)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Partition the data

Create the split with `rsample`

```
df_split <- initial_split(df, strata = y)
```

View the training and test sets:

```
df_split %>%  
  training() %>%  
  glimpse()  
  
df_split %>%  
  testing() %>%  
  glimpse()
```

Data pre-processing

Specify the recipe

```
df_recipe <- training(df_split) %>%  
  recipe(y ~ .) %>% # write the formula  
  step_center(all_predictors()) %>% # center all predictors  
  step_scale(all_predictors()) %>%  
  prep()
```

Obtain train and test data

By juicing and baking the recipe:

```
train_set <- juice(df_recipe)  
  
test_set <- df_recipe %>%  
  bake(testing(df_split))
```

We can also display train and test set:

```
train_set  
test_set
```

Train and test set plotting

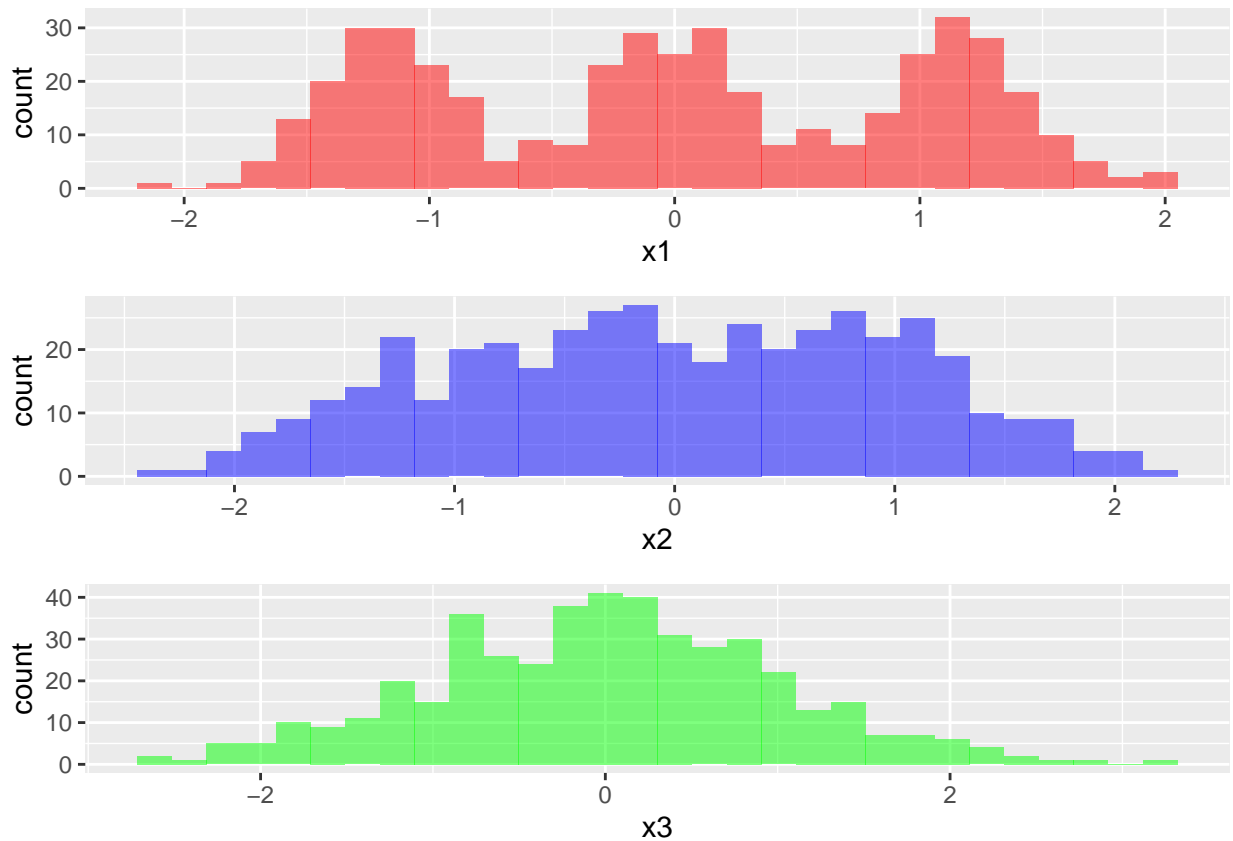
Much better, we can plot the train and the test set!

```
norm_1 <- train_set %>%  
  ggplot() +  
  geom_histogram(aes(x1), alpha = 1/2, fill = 'red')  
  
norm_2 <- train_set %>%  
  ggplot() +  
  geom_histogram(aes(x2), alpha = 1/2, fill = 'blue')  
  
norm_3 <- train_set %>%  
  ggplot() +  
  geom_histogram(aes(x3), alpha = 1/2, fill = 'green')
```

And display them in a single plot:

```
grid.arrange(norm_1, norm_2, norm_3)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



X does not seem normally distributed. One should proceed with further statistical tests, such as the KS.

LDA Model Fitting

Let's train the model:

```
lda_classifier <- discrim_linear(mode = 'classification') %>%
  set_engine('MASS') %>%
  fit(y ~ ., data = train_set)
```

Prediction

Then let's predict!

```
lda_classifier %>%
  predict(test_set) %>%
  bind_cols(test_set) %>%
  glimpse()
```

```
## Rows: 149
## Columns: 5
## $ .pred_class <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ x1          <dbl> -1.1099966, -0.7992768, -0.8165241, -1.0632063, -1.0296...
## $ x2          <dbl> -1.50211609, -1.27833756, -0.88042758, -1.27549405, -0....
## $ x3          <dbl> 0.46887800, -2.41127886, -1.71937806, 0.30266560, -0.89...
## $ y          <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
```

Tidymodels will create the column `.pred_class`: that is our prediction.

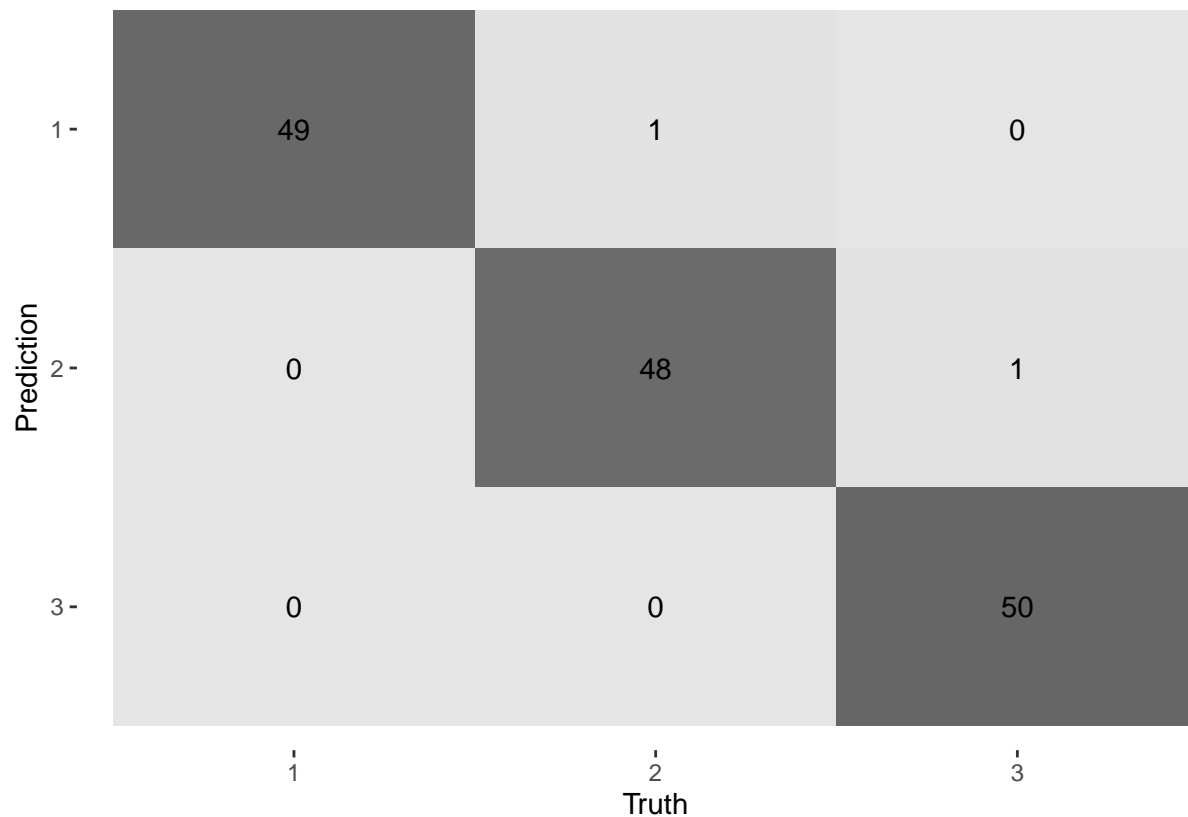
Model Validation

```
lda_classifier %>%  
  predict(test_set) %>%  
  bind_cols(test_set) %>%  
  metrics(truth = y, estimate = .pred_class)
```

```
## # A tibble: 2 x 3  
##   .metric .estimator .estimate  
##   <chr>   <chr>      <dbl>  
## 1 accuracy multiclass    0.987  
## 2 kap     multiclass    0.980
```

Confusion Matrix

```
lda_classifier %>%  
  predict(test_set) %>%  
  bind_cols(test_set) %>%  
  conf_mat(y, .pred_class) %>%  
  autoplot(type = 'heatmap')
```

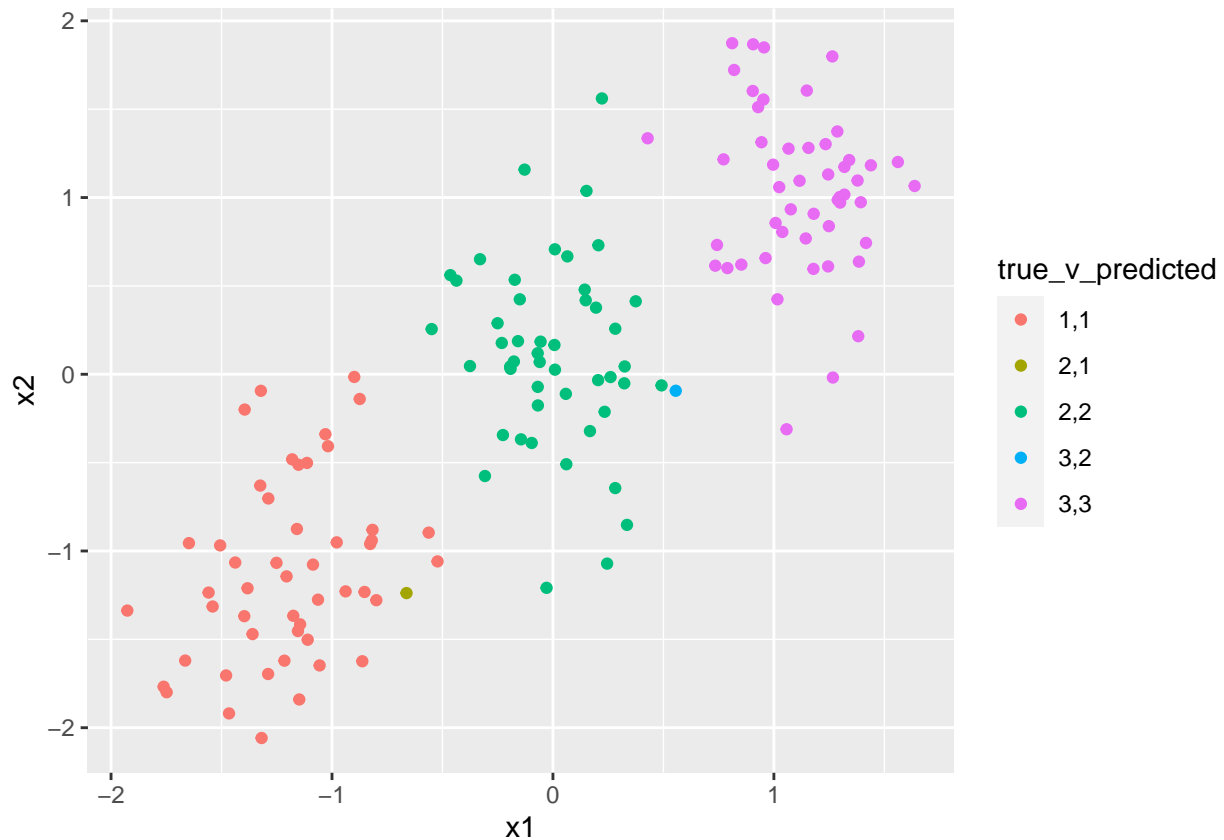


Let's visualise it once more

Then, let's add a layer of complications to visualise how we plotted it:

```
test_set <- lda_classifier %>%
  predict(test_set) %>%
  bind_cols(test_set)

test_set %>%
  mutate(true_v_predicted = paste(test_set$y, test_set$.pred_class, sep = ',')) %>%
  ggplot() +
  geom_point(aes(x1, x2, col = true_v_predicted))
```



Per-classifier metrics

We can obtain the metrics for each class by simply specifying a different argument in `predict`:

```
lda_probs <- lda_classifier %>%
  predict(test_set, type = 'prob') %>%
  bind_cols(test_set)
```

And we can immediately plot some curves:

```
lda_probs %>%
  roc_curve(y, .pred_1:.pred_3) %>%
  autoplot()
```

