Machine Learning example using R

This lab is a quick overview on graphics and data handling and machine learning in R.

Introduction to R Markdown

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

See https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf for more details.

Packages

We will use the twoClass dataset from *Applied Predictive Modeling*, the book of M. Kuhn and K. Johnson to illustrate the most classical supervised classification algorithms. We will use some *advanced* **R** packages: the **ggplot2** package for the figures and the **caret** package for the learning part. **caret** that provides an unified interface to many other packages.

Install and load the following packages.

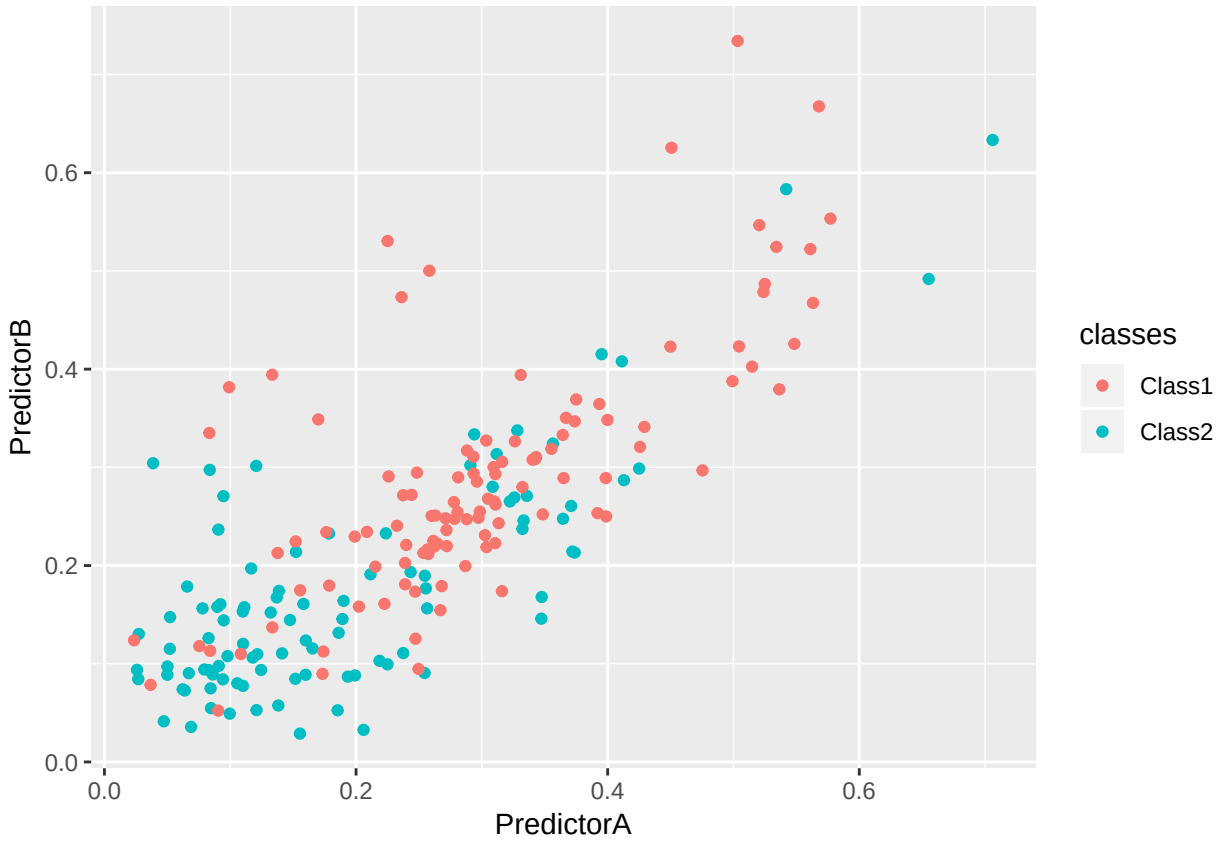
```
library("plyr")
library("dplyr")
library("ggplot2")
library("grid")
library("gridExtra")
library("caret")
```

The twoClass dataset

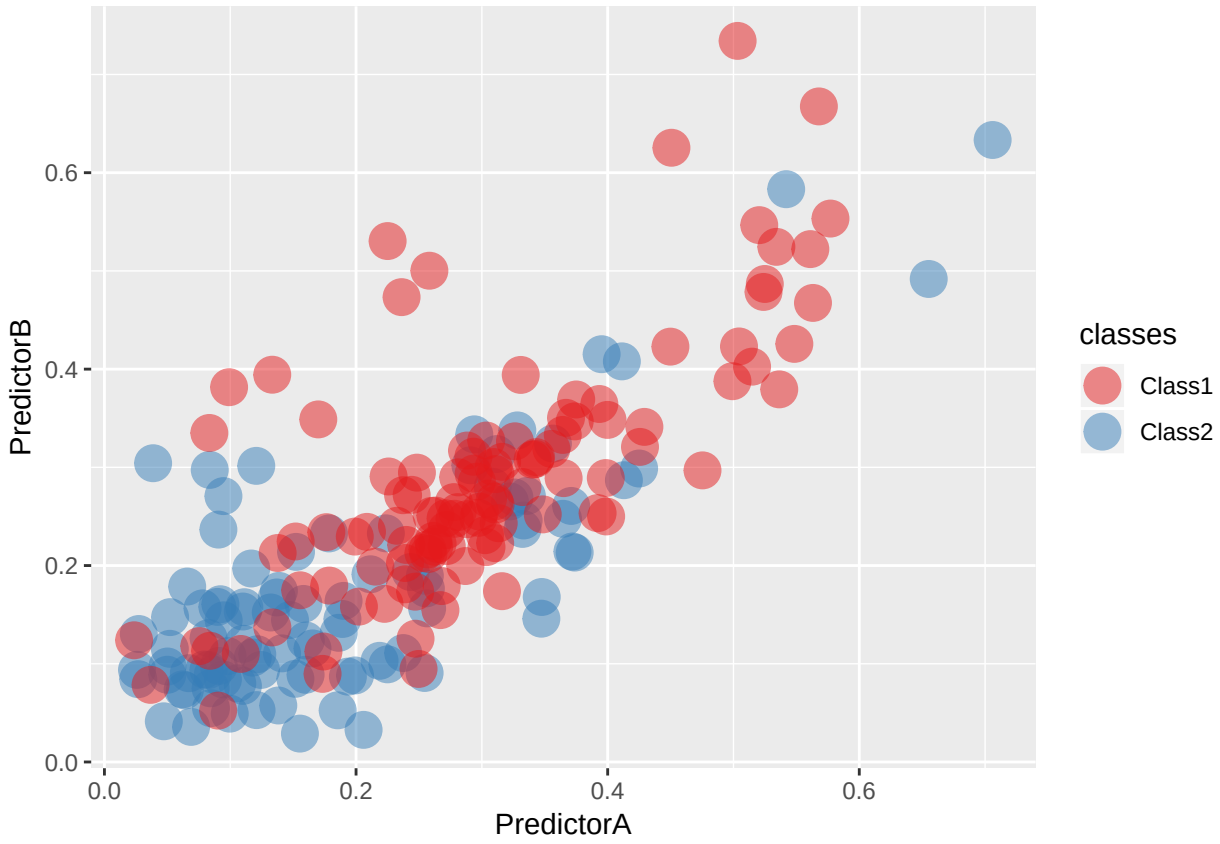
Data and graphical representation

- We read first the dataset and use **ggplot2** to display it.

```
library(AppliedPredictiveModeling)
library(RColorBrewer)
data(twoClassData)
twoClass=cbind(as.data.frame(predictors),classes)

ggplot(data = twoClass,aes(x = PredictorA, y = PredictorB)) +
  geom_point(aes(color = classes))
```



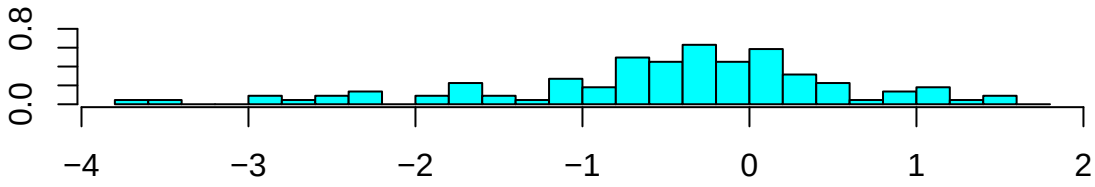

```
twoClassColor <- brewer.pal(3,'Set1')[1:2]
names(twoClassColor) <- c('Class1','Class2')

ggplot(data = twoClass,aes(x = PredictorA, y = PredictorB)) +
  geom_point(aes(color = classes), size = 6, alpha = .5) +
  scale_colour_manual(name = 'classes', values = twoClassColor)
```



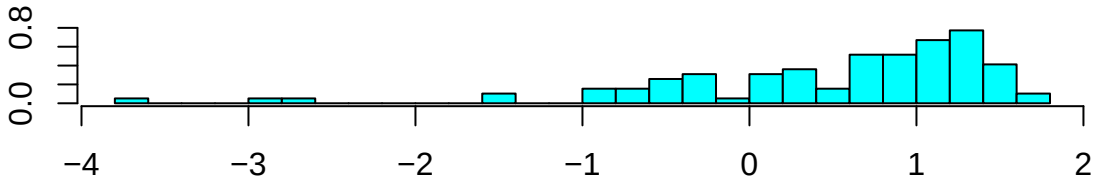
First classification : Linear discriminant analysis

via the MASS package

```
library(MASS)
?lda
lda_fit = lda(classes ~ ., data=twoClass)
plot(lda_fit)
```



group Class1



group Class2

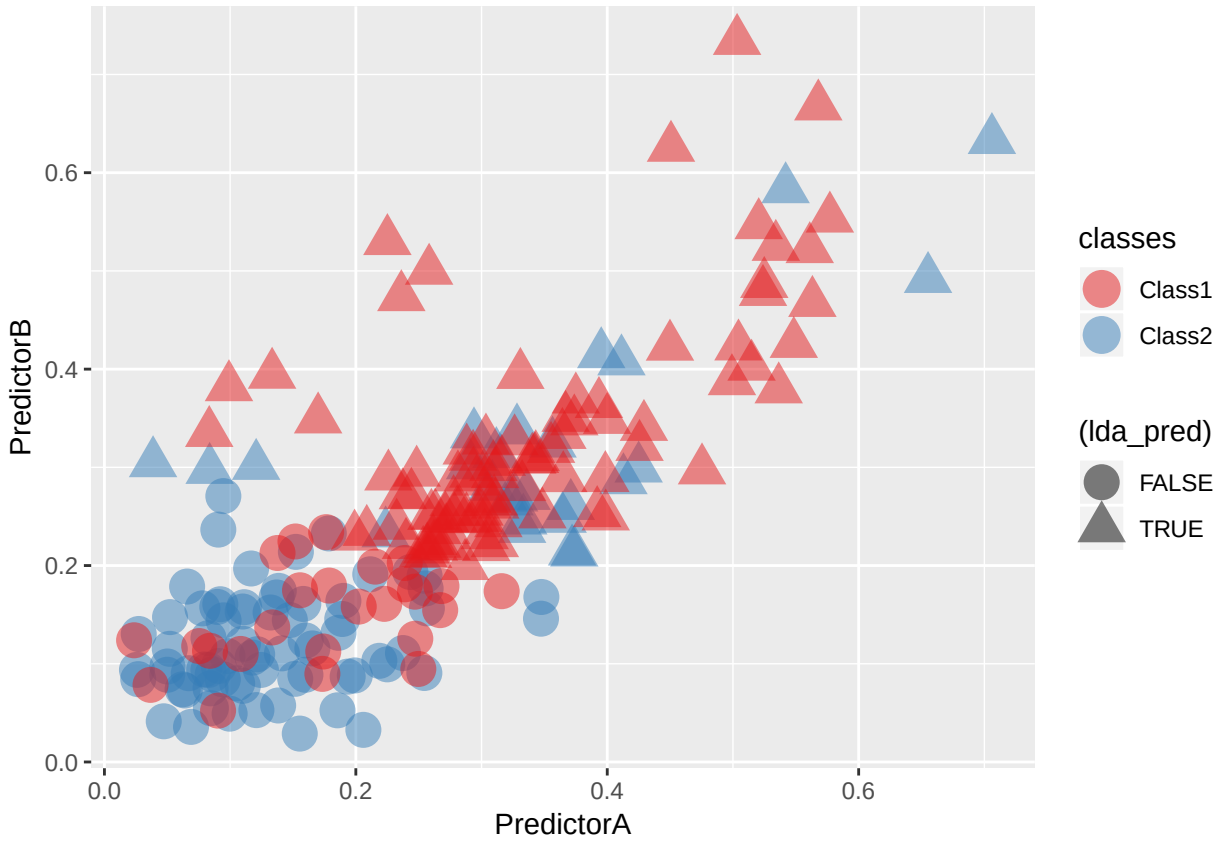
Graphical representation

- Create a new dataset with the predictions from the LDA (keep the posterior probabilities of being in "Class 1") https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

```
ML_fits = mutate(twoClass , lda_prob = predict(lda_fit, data=twoClass)$posterior[,1],lda_pred = predict
```

- Represent on the same graph the true classes and the LDA predictions. https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

```
ggplot(data = ML_fits,aes(x = PredictorA, y = PredictorB)) +
  geom_point(aes(color = classes,shape=(lda_pred)), size = 6, alpha = .5) +
  scale_colour_manual(name = 'classes', values = twoClassColor)
```



Errors measures

- Set the "Class 1" level as 1 (and the "Class 2" level as 0). Compute via the **MLmetrics package**, the confusion matrix, specificity, sensitivity and AUC for the LDA model.

```
library(MLmetrics)
MLmetrics::ConfusionMatrix(ML_fits$lda_pred,ML_fits$classes=="Class1")
```

```
##        y_pred
## y_true  FALSE TRUE
##   FALSE    71   26
##   TRUE     25   86
```

```
MLmetrics::AUC(ML_fits$lda_prob,ML_fits$classes=="Class1")
```

```
## [1] 0.7850841
```

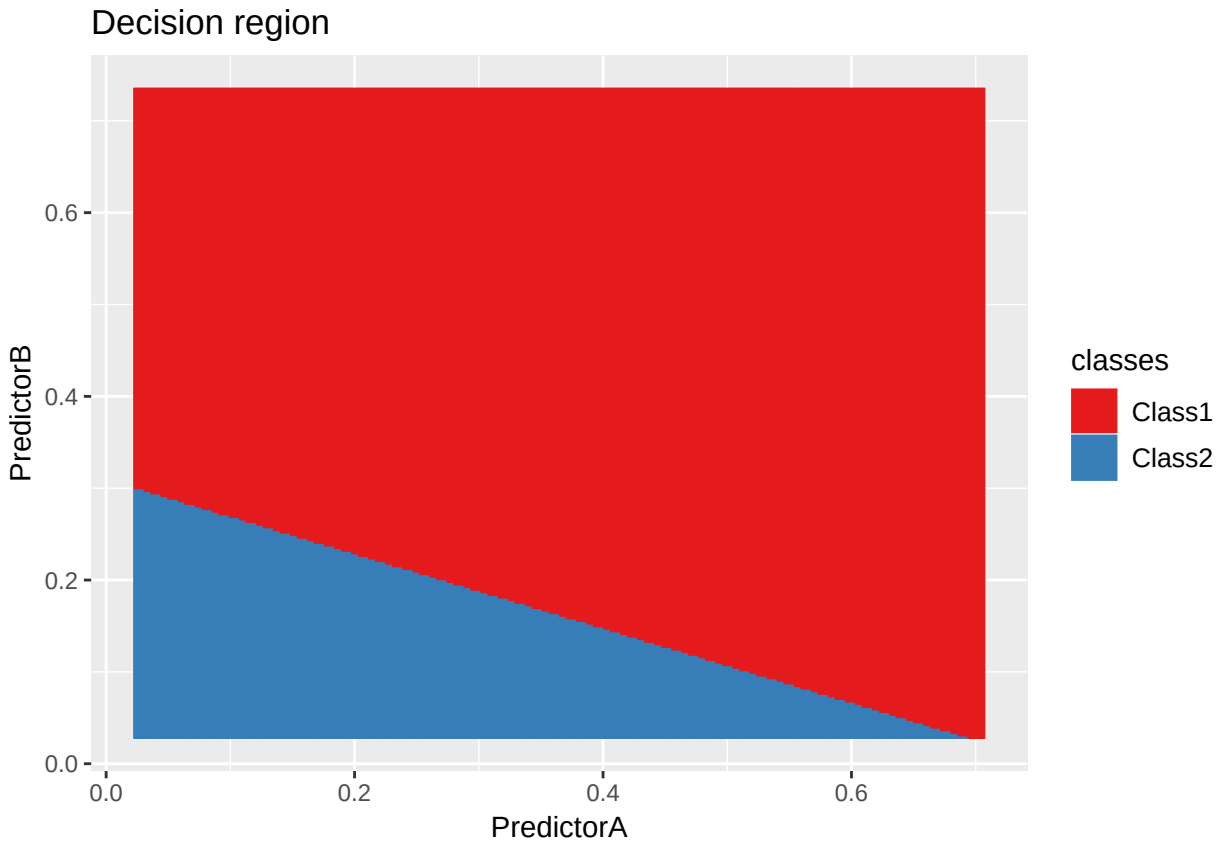
```
MLmetrics::Accuracy(ML_fits$lda_pred,ML_fits$classes=="Class1")
```

```
## [1] 0.7548077
```

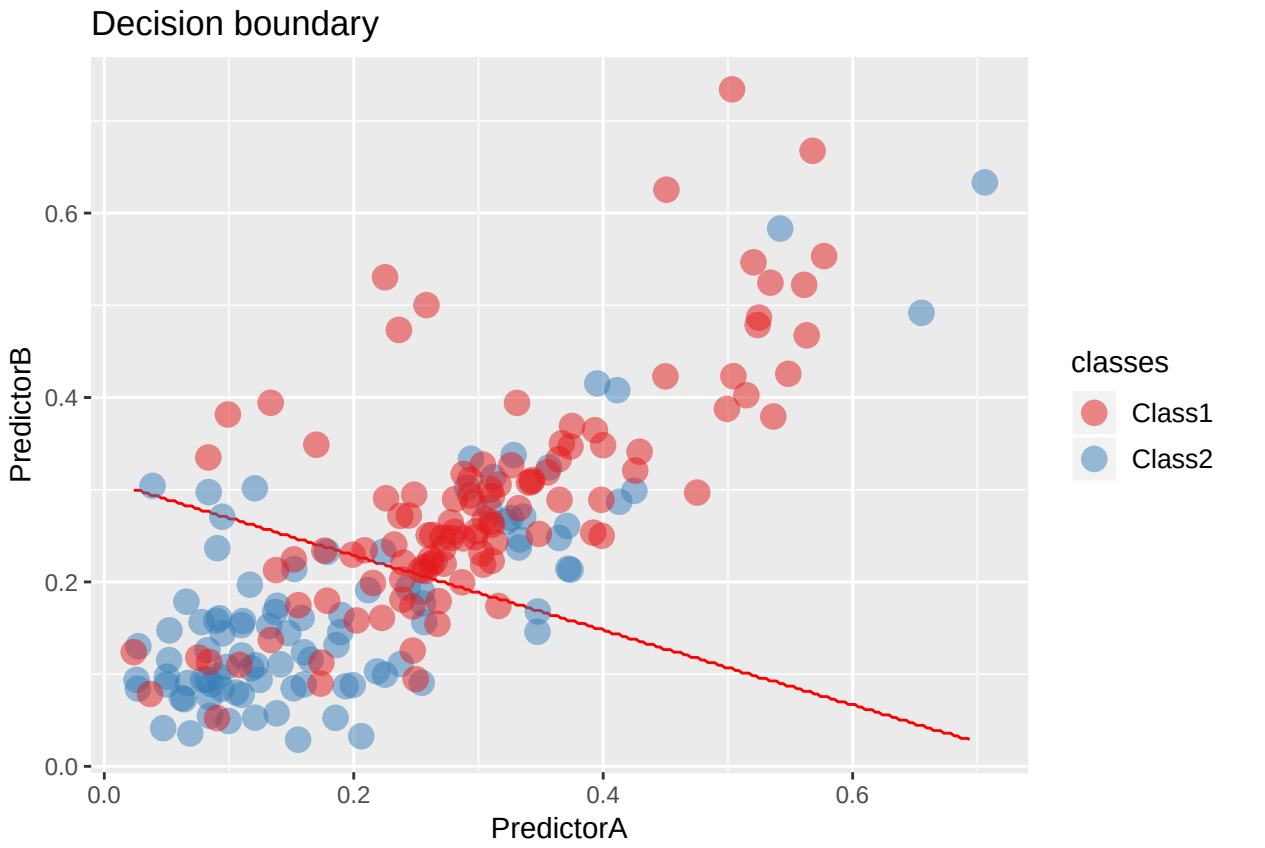
A better graphical representation

- Create two graphics, one with the decision surfaces for each classes and one with the decision boundary.
- Towards that end, create a grid of 250^2 points on the predictors plan, compute the predictions for these points and use the *geom_tile* and *geom_contour* functions of package **ggplot2**. Use the **gridExtra** to display them on the same plot.

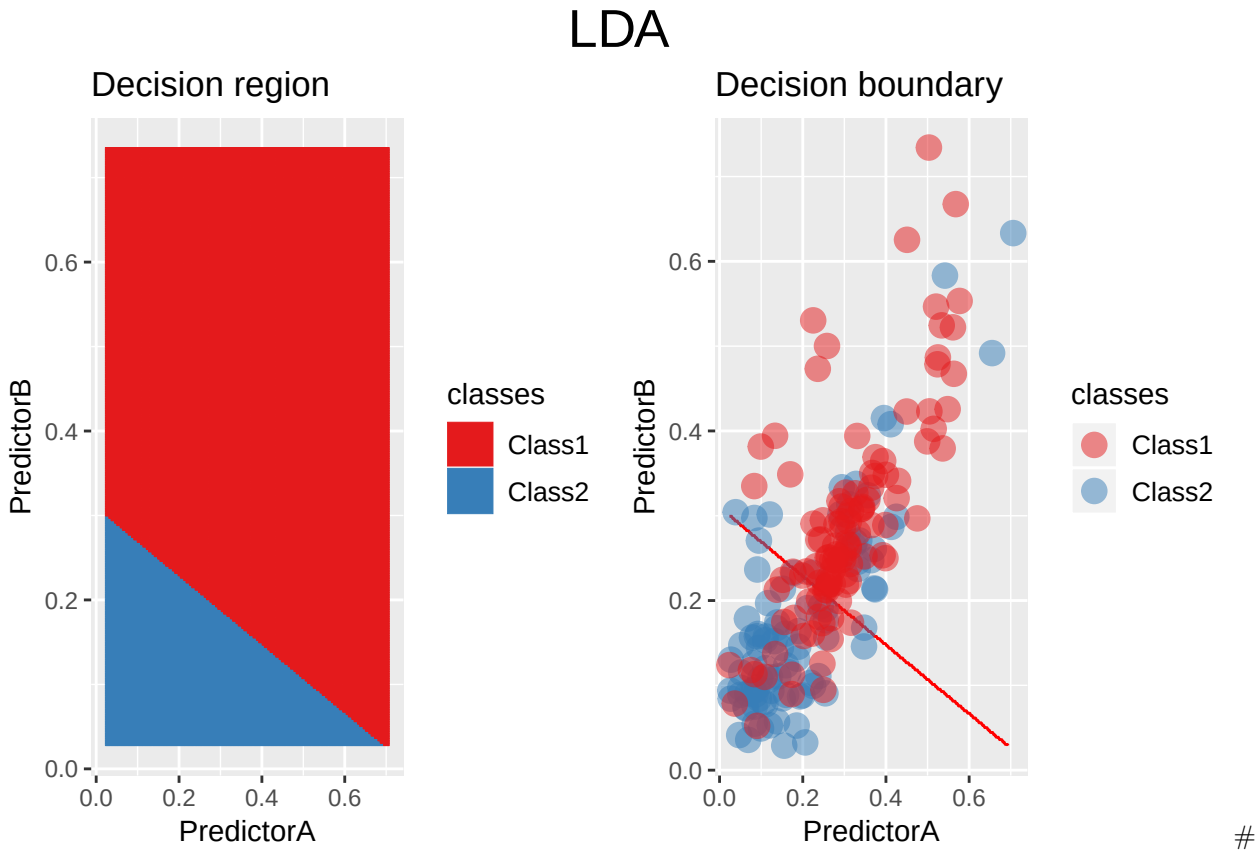
```
nbp <- 250;
PredA <- seq(min(twoClass$PredictorA), max(twoClass$PredictorA), length = nbp)
PredB <- seq(min(twoClass$PredictorB), max(twoClass$PredictorB), length = nbp)
Grid <- expand.grid(PredictorA = PredA, PredictorB = PredB)
regions = ggplot(data = ML_fits, aes(x = PredictorA, y = PredictorB,
                                     color = classes)) +
          geom_tile(data = cbind(Grid, classes = predict(lda_fit,Grid)$class), aes(fill = classes)) +
          scale_fill_manual(name = 'classes', values = twoClassColor) +
          ggtitle("Decision region") + theme(legend.text = element_text(size = 10)) +
          scale_colour_manual(name = 'classes', values = twoClassColor)
show(regions)
```




```
boundary = ggplot(data = twoClass, aes(x = PredictorA, y = PredictorB,
                                       color = classes)) +
          geom_contour(data = cbind(Grid, classes = predict(lda_fit,Grid)$class),
                       aes(z = as.numeric(classes)), color = "red", breaks = c(1.5)) +
          geom_point(size = 4, alpha = .5) +
          ggtitle("Decision boundary") +
          theme(legend.text = element_text(size = 10)) +
          scale_colour_manual(name = 'classes', values = twoClassColor)
show(boundary)
```




```
grid.arrange(regions, boundary,ncol=2 ,top = textGrob("LDA", gp = gpar(fontsize = 20)))
```

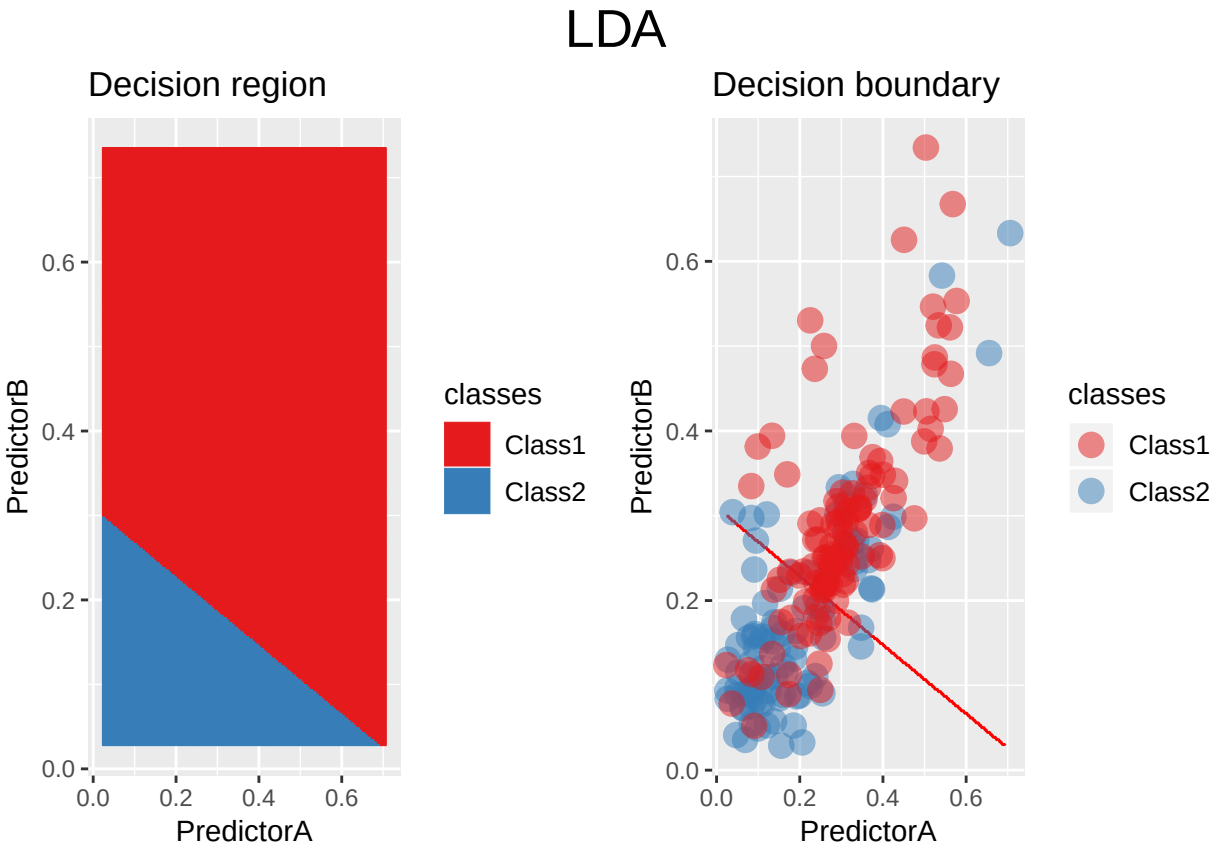


#

Functions - Create a function that takes the Grid, fit and a title as argument and return the plots created above.

```
nbp <- 250;
PredA <- seq(min(twoClass$PredictorA), max(twoClass$PredictorA), length = nbp)
PredB <- seq(min(twoClass$PredictorB), max(twoClass$PredictorB), length = nbp)
Grid <- expand.grid(PredictorA = PredA, PredictorB = PredB)

create_plots = function(Grid,fit,title){
regions = ggplot(data = ML_fits, aes(x = PredictorA, y = PredictorB,
                                     color = classes)) +
          geom_tile(data = cbind(Grid, classes = predict(fit,Grid)$class), aes(fill = classes)) +
          scale_fill_manual(name = 'classes', values = twoClassColor) +
          ggtitle("Decision region") + theme(legend.text = element_text(size = 10)) +
          scale_colour_manual(name = 'classes', values = twoClassColor)
boundary = ggplot(data = twoClass, aes(x = PredictorA, y = PredictorB,
                                   color = classes)) +
          geom_contour(data = cbind(Grid, classes = predict(fit,Grid)$class),
                       aes(z = as.numeric(classes)), color = "red", breaks = c(1.5)) +
          geom_point(size = 4, alpha = .5) +
          ggtitle("Decision boundary") +
          theme(legend.text = element_text(size = 10)) +
          scale_colour_manual(name = 'classes', values = twoClassColor)
grid.arrange(regions, boundary,ncol=2 ,top = textGrob(title, gp = gpar(fontsize = 20)))
}
```

```
create_plots(Grid,lda_fit,"LDA")
```



The caret package

As explained in the introduction, we will use **caret** for the learning part. This package provides a unified interface to a huge number of classifier available in **R**. It is a very powerful tool when exploring the different models. In particular, it proposes to compute a *resampling* accuracy estimate and gives the user the choice of the specific methodology. We will use a repeated V-fold strategy with 10 folds and 4 repetitions. We will reuse the same *seed* for each model in order to be sure that the same folds are used.

```
library("caret")
V <- 10
T <- 4
TrControl <- trainControl(method = "repeatedcv",
                          number = V,
                          repeats = T, classProbs = TRUE,
        summaryFunction = twoClassSummary)

Seed <- 345
```

- Train the LDA via the **caret** package. And change the options so that the AUC, sensititvy and specificity are computed. See https://topepo.github.io/caret/model-training-and-tuning.html#metrics

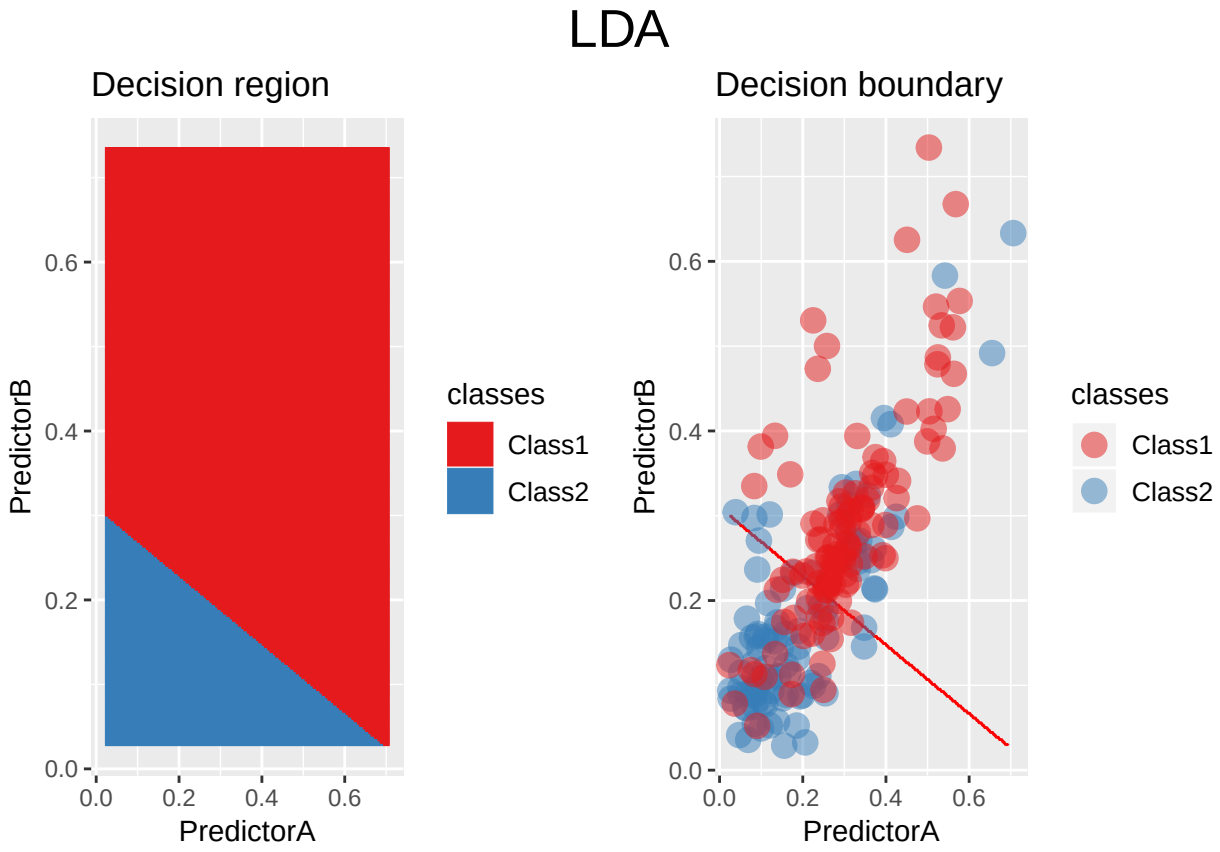
```
library(caret)
lda_fit_caret = train(classes ~., data = twoClass, method = "lda", trControl = TrControl, metric = "ROC
print(lda_fit_caret)
```

```
## Linear Discriminant Analysis
##
```

```
## 208 samples
##   2 predictor
##   2 classes: 'Class1', 'Class2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 4 times)
## Summary of sample sizes: 187, 187, 186, 188, 188, 187, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7783333  0.7640152  0.7097222
```

- Change your **create_plots** function so that it handles the object returned by the **caret** package.

```
create_plots = function(Grid,fit,title){
regions = ggplot(data = ML_fits, aes(x = PredictorA, y = PredictorB,
                                     color = classes)) +
          geom_tile(data = cbind(Grid, classes = predict(fit,Grid)), aes(fill = classes)) +
          scale_fill_manual(name = 'classes', values = twoClassColor) +
          ggtitle("Decision region") + theme(legend.text = element_text(size = 10)) +
          scale_colour_manual(name = 'classes', values = twoClassColor)
boundary = ggplot(data = twoClass, aes(x = PredictorA, y = PredictorB,
                                   color = classes)) +
          geom_contour(data = cbind(Grid, classes = predict(fit,Grid)),
                       aes(z = as.numeric(classes)), color = "red", breaks = c(1.5)) +
          geom_point(size = 4, alpha = .5) +
          ggtitle("Decision boundary") +
          theme(legend.text = element_text(size = 10)) +
          scale_colour_manual(name = 'classes', values = twoClassColor)

grid.arrange(regions, boundary,ncol=2 ,top = textGrob(title, gp = gpar(fontsize = 20)))
}

create_plots(Grid,lda_fit_caret,"LDA")
```



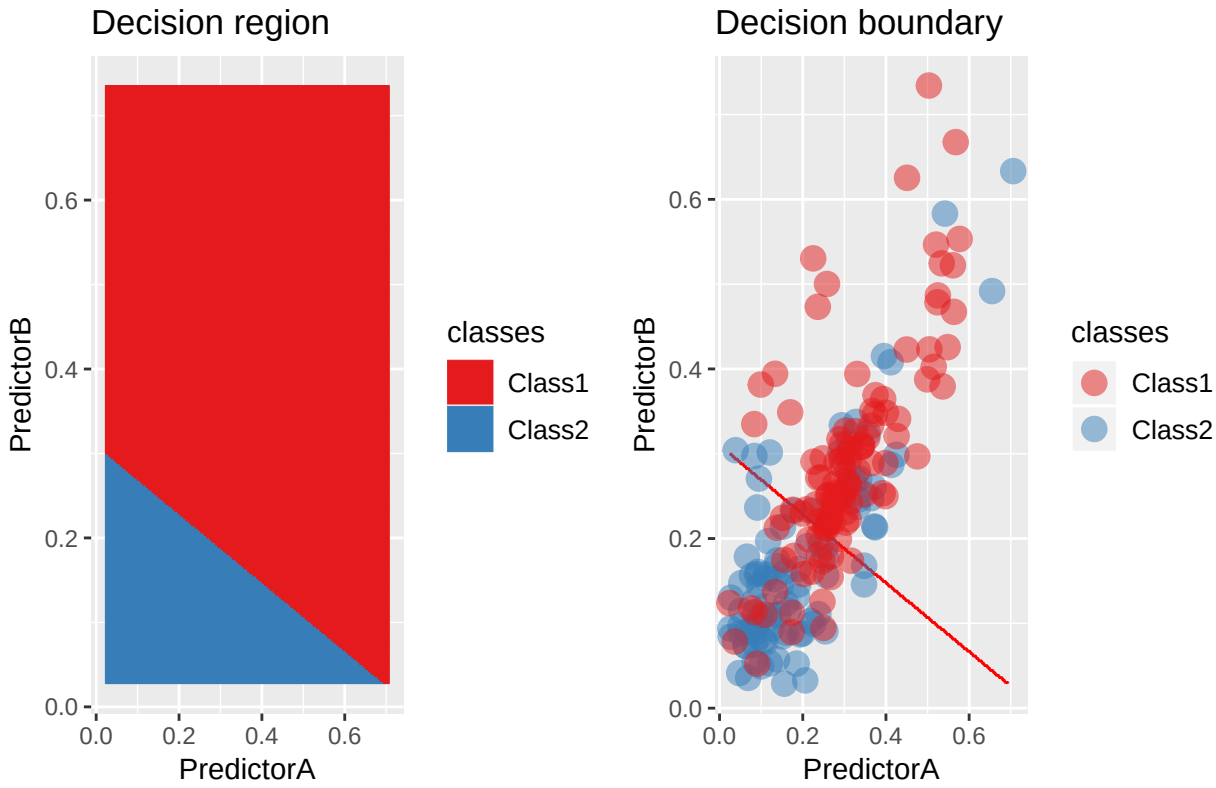
Comparison of classification methods

- Use these functions to compare the performances of the classification methods that you already know (Naive Bayes, logistic, SVM, kNN, etc). See also https://web.stanford.edu/~hastie/Papers/ESLII.pdf for an overview of ML algorithms.

```
Errs = data.frame()
```

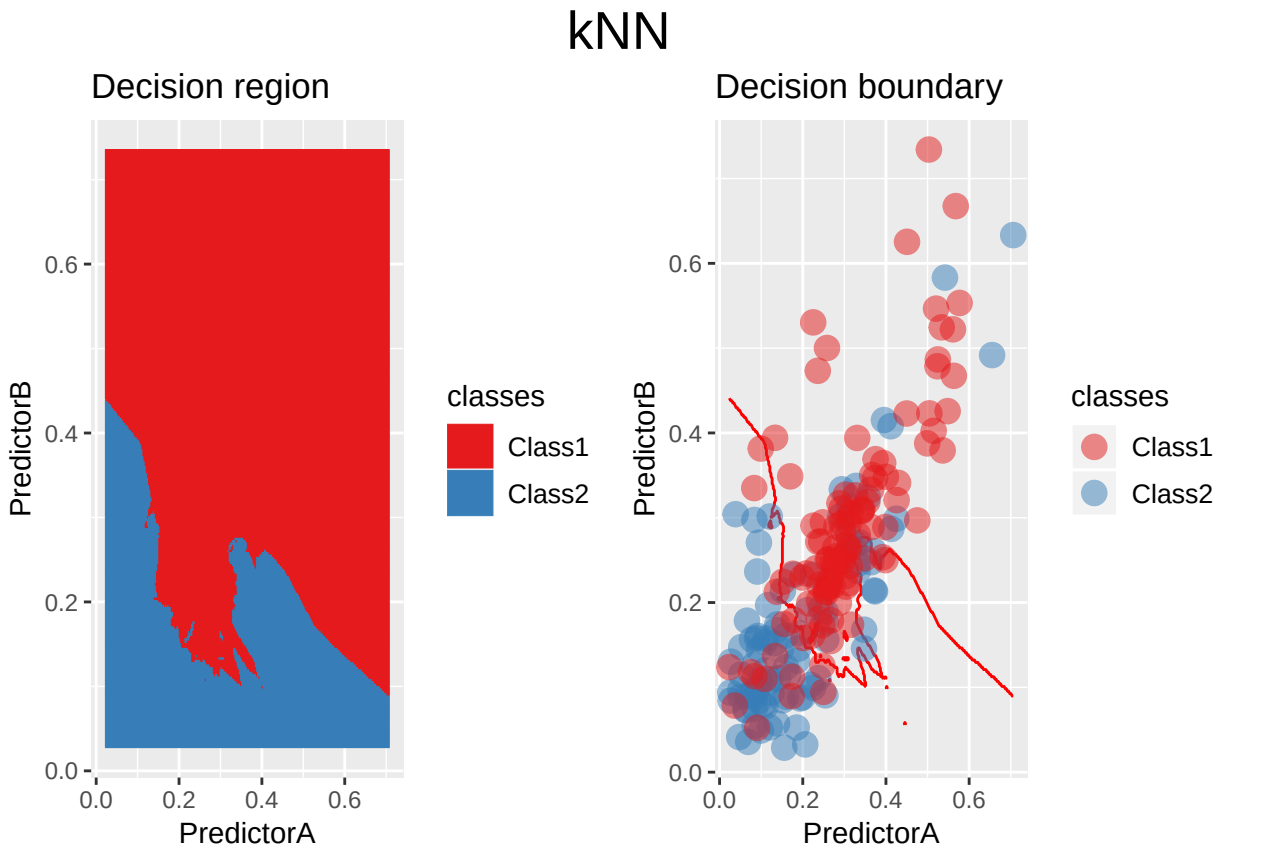
```
Method = "lda"
Name = "LDA"

Formula = "classes~."
Model = train(as.formula(Formula), data = twoClass, method = Method, trControl = TrControl, metric = "R
create_plots(Grid, Model ,Name)
```

LDA




```
Errs <- rbind(Errs,cbind(Model$resample,rep(Name,40)))
```

```
Method = "knn"
Name = "kNN"

Formula = "classes~."
Model = train(as.formula(Formula), data = twoClass, method = Method, trControl = TrControl, metric = "R
create_plots(Grid, Model ,Name)
```




```
Errs <- rbind(Errs,cbind(Model$resample,rep(Name,40)))
```