Introduction to Machine Learning

Working Group “Computational Statistics” – Bernd Bischl et al.

Code demo for Random Forests

Variable Importance from mlr

```r
library(mlr)
library(mlbench)
library(dplyr)

data("Servo")

# transform ordered factors to numeric
servo <- Servo %>%
  mutate_at(c("Pgain", "Vgain"), as.character) %>%
  mutate_at(c("Pgain", "Vgain"), as.numeric)
rm(Servo)
str(servo)
```

```
## 'data.frame':    167 obs. of  5 variables:
##  $ Motor: Factor w/ 5 levels "A","B","C","D",..: 5 2 4 2 4 5 3 ..
##  $ Screw: Factor w/ 5 levels "A","B","C","D",..: 5 4 4 1 2 3 1 ..
##  $ Pgain: num  5 6 4 3 6 4 3 3 ...
##  $ Vgain: num  4 5 3 2 5 3 2 2 ...
##  $ Class: num  4 11 6 48 6 20 46 49 ...
```

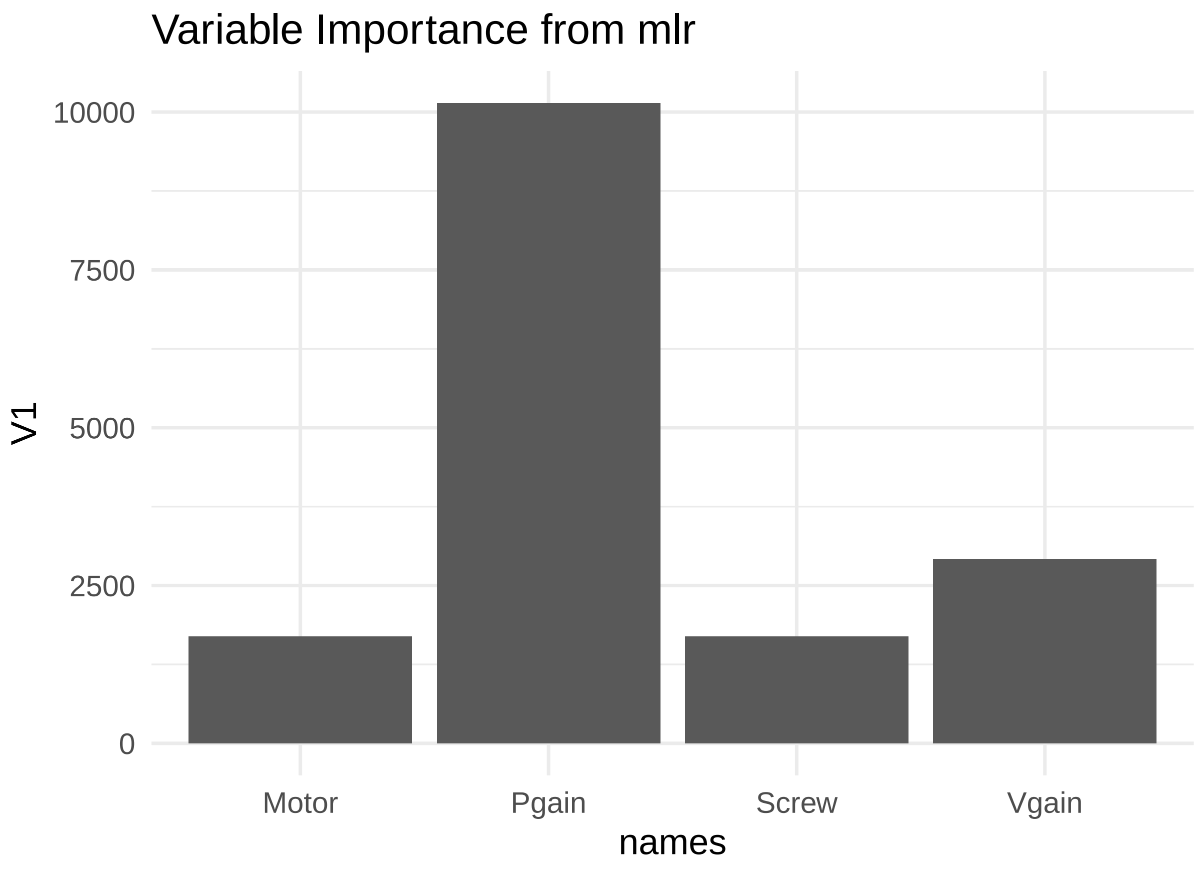
```r
head(servo)
```

```
##   Motor Screw Pgain Vgain Class
## 1     E     E     5     4     4
## 2     B     D     6     5    11
## 3     D     D     4     3     6
## 4     B     A     3     2    48
## 5     D     B     6     5     6
## 6     E     C     4     3    20
```

```r
# split in train and test with two different seeds to show data dependency
train_size <- 3 / 4

set.seed(1333)
train_indices <- sample(
  x = seq(1, nrow(servo), by = 1),
  size = ceiling(train_size * nrow(servo)), replace = FALSE
)
train_1 <- servo[ train_indices, ]
test_1 <- servo[ -train_indices, ]
```

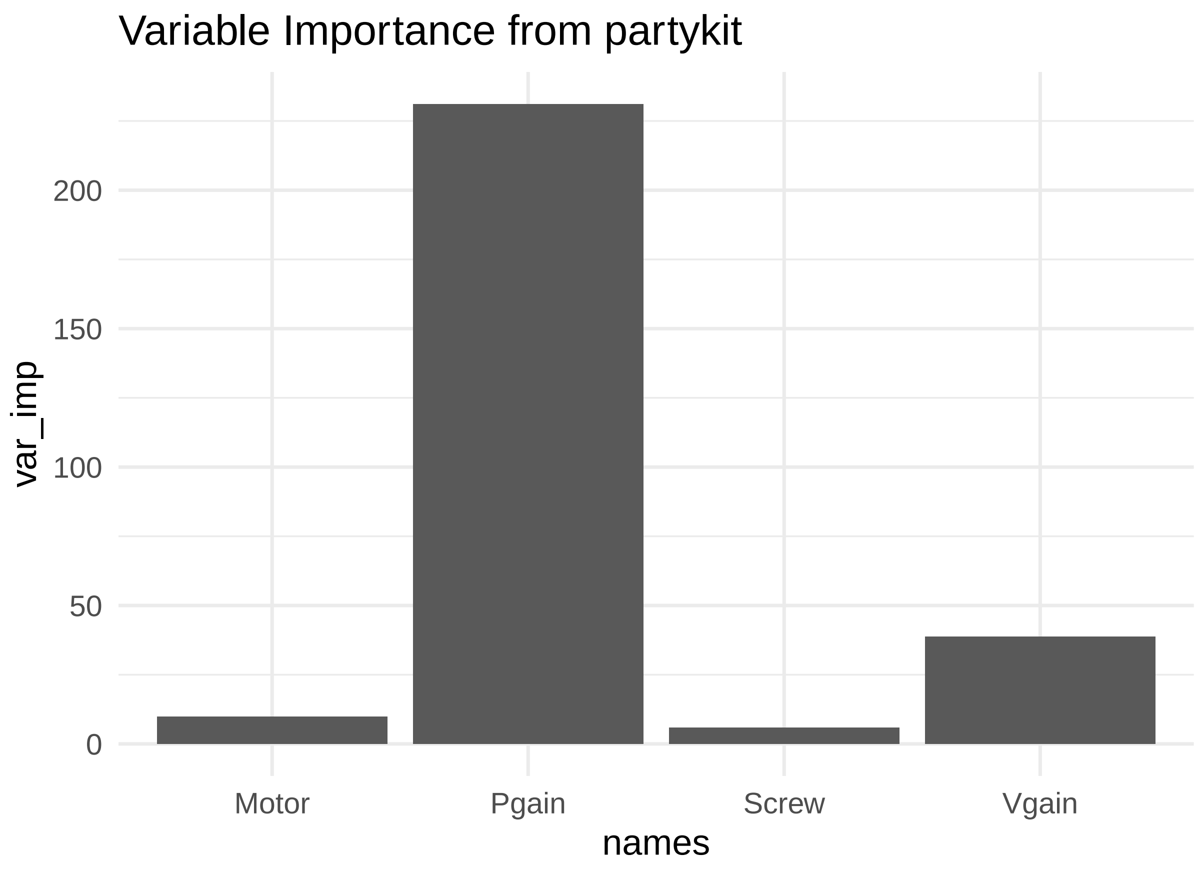
```r
library(ggplot2)

task <- makeRegrTask(data = train_1, target = "Class")
lrn1 <- makeLearner("regr.randomForest")
mod <- mlr::train(learner = lrn1, task = task)
```

```r
var_imp <- getFeatureImportance(mod)
var <- as.data.frame(t(var_imp$res))
var$names <- names(var_imp$res)
p <- ggplot(data = var, aes(x = names, y = V1)) + geom_bar(stat = "identity") +
  ggtitle(label = "Variable Importance from mlr")
p
```



Variable importance from partykit

```r
library(partykit)

modForest <- cforest(Class ~ Motor + Screw + Pgain + Vgain, data = train_1)
var <- as.data.frame(varimp(modForest))
var$names <- rownames(var)
colnames(var) <- c("var_imp", "names")
p <- ggplot(data = var, aes(x = names, y = var_imp)) +
  geom_bar(stat = "identity") +
  ggtitle(label = "Variable Importance from partykit")
p
```



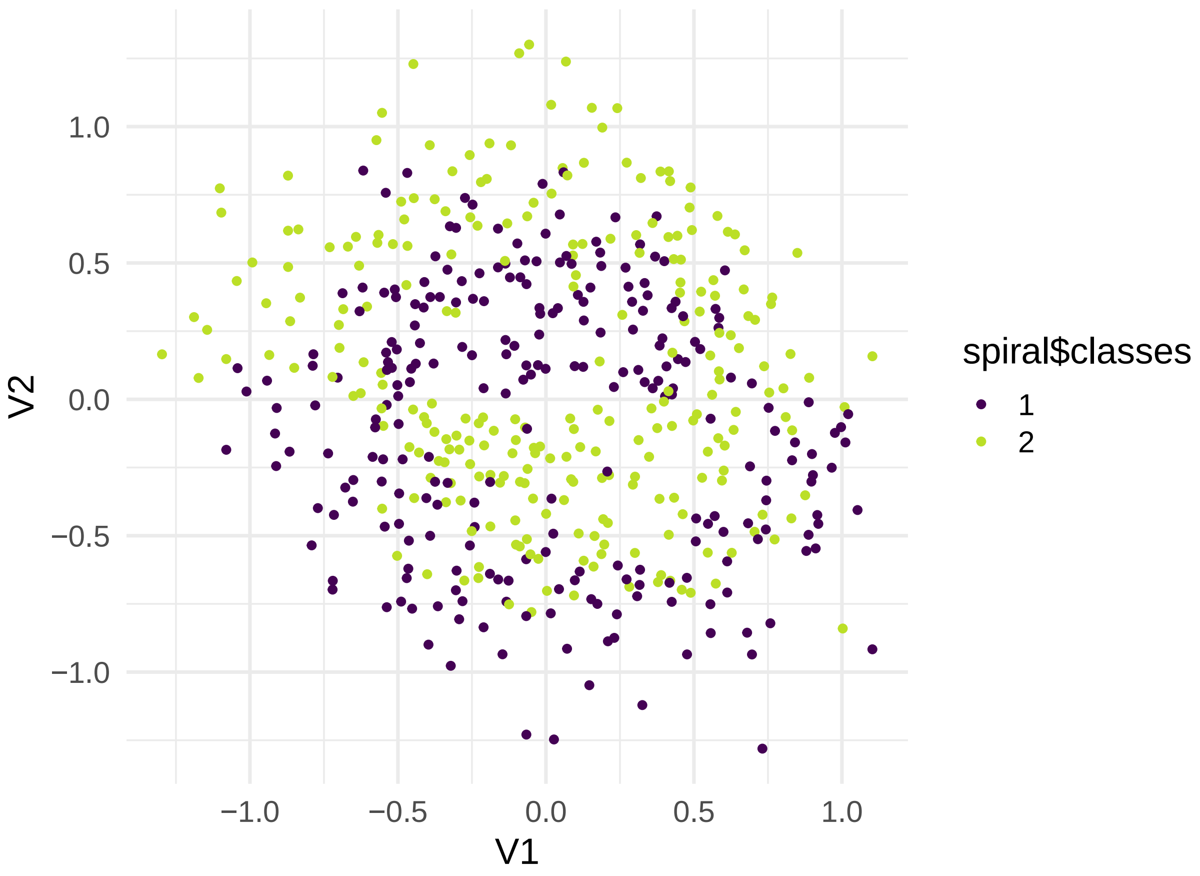
Proximity measures in Random Forests

One neat feature of random forests is, that they calculate a proximity measure (how close is one data point to another) on the fly: frequencies of how often two (OOB-)observations end up in the same terminal node are calculated during the fit – if two observations are "close" in terms of the features that matter for the random forest, they will have a high proximity. We can try to interpret them and do fancy stuff such as **multi dimensional scaling**. Check this blog post for a wonderful example of MDS for European capital cities.

We apply this to the spiral data set and try to see, if we could also classify the points based on their proximity.

First, look a the data:

```r
# get data
spiral <- mlbench::mlbench.spirals(500, cycles = 1, sd = 0.2)
p <- ggplot(data = as.data.frame(spiral$x), aes(
  x = V1, y = V2,
  colour = spiral$classes
)) +
  geom_point()
p
```



Now we fit a random forest and extract the proximity distance matrix:

```r
library(randomForest)
# fit forest and extract proximity measure
set.seed(1337)
spiral_rf <- randomForest(
  x = spiral$x, y = spiral$classes,
  ntree = 1000,
  proximity = TRUE, oob.prox = TRUE,
)
spiral_proximity <- spiral_rf$proximity
spiral_proximity[1:5, 1:5]
```

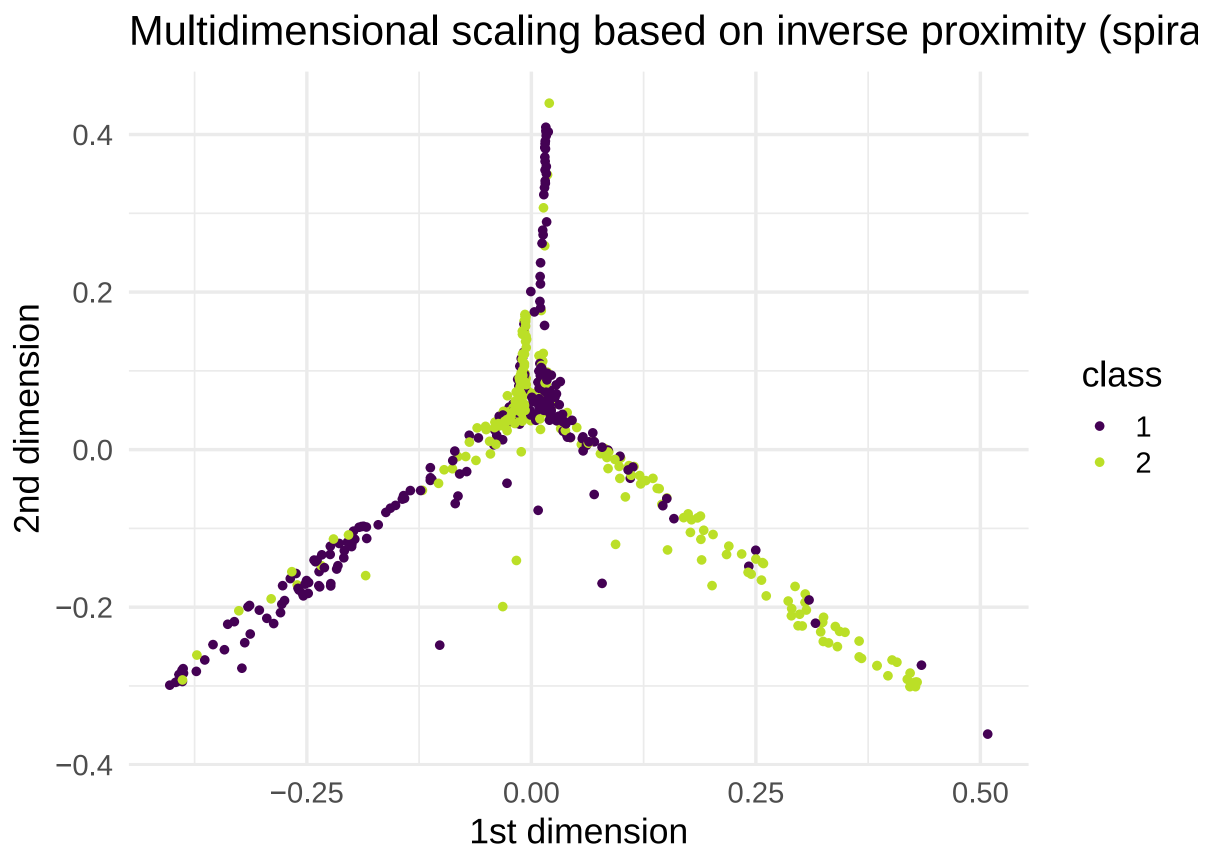
```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

Run MDS on the distance matrix derived from the (inverse) proximities and plot the data based on the two dimensions, this won't work *that* well here because so many data points have mutual proximities of 0, i.e., an *"infinite"* distance – they never ended up in the same leaf of any tree, ever. . .

```r
# convert proximities into distances
proximity_to_dist <- function(proximity) {
  1 - proximity
}

spiral_dist <- proximity_to_dist(spiral_proximity)
spiral_dist[1:5, 1:5]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    1    1    1
## [2,]    1    0    1    1    1
## [3,]    1    1    0    1    1
## [4,]    1    1    1    0    1
## [5,]    1    1    1    1    0
```

```r
spiral_mds <- as.data.frame(cmdscale(spiral_dist))
spiral_mds$class <- spiral$classes

# plot the result, sweet
p <- ggplot(data = spiral_mds, aes(x = V1, y = V2, colour = class)) +
  geom_point() +
  labs(x = "1st dimension", y = "2nd dimension",
       title = "Multidimensional scaling based on inverse proximity (spirals data)")
p
```

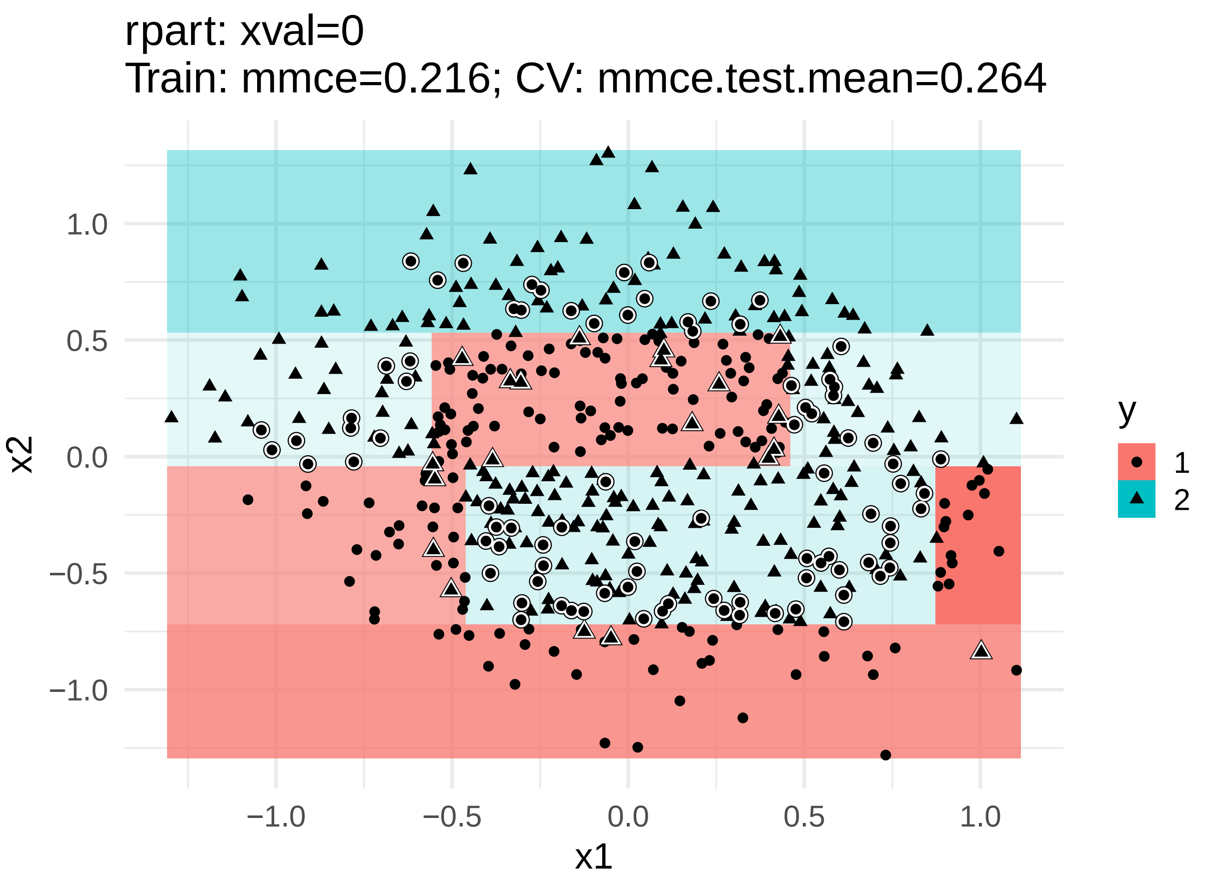


Some structure seems visible, but this is really most useful for high-dimensional problems where we have much more than just 2 features…

Decision Regions CART vs. Random Forest

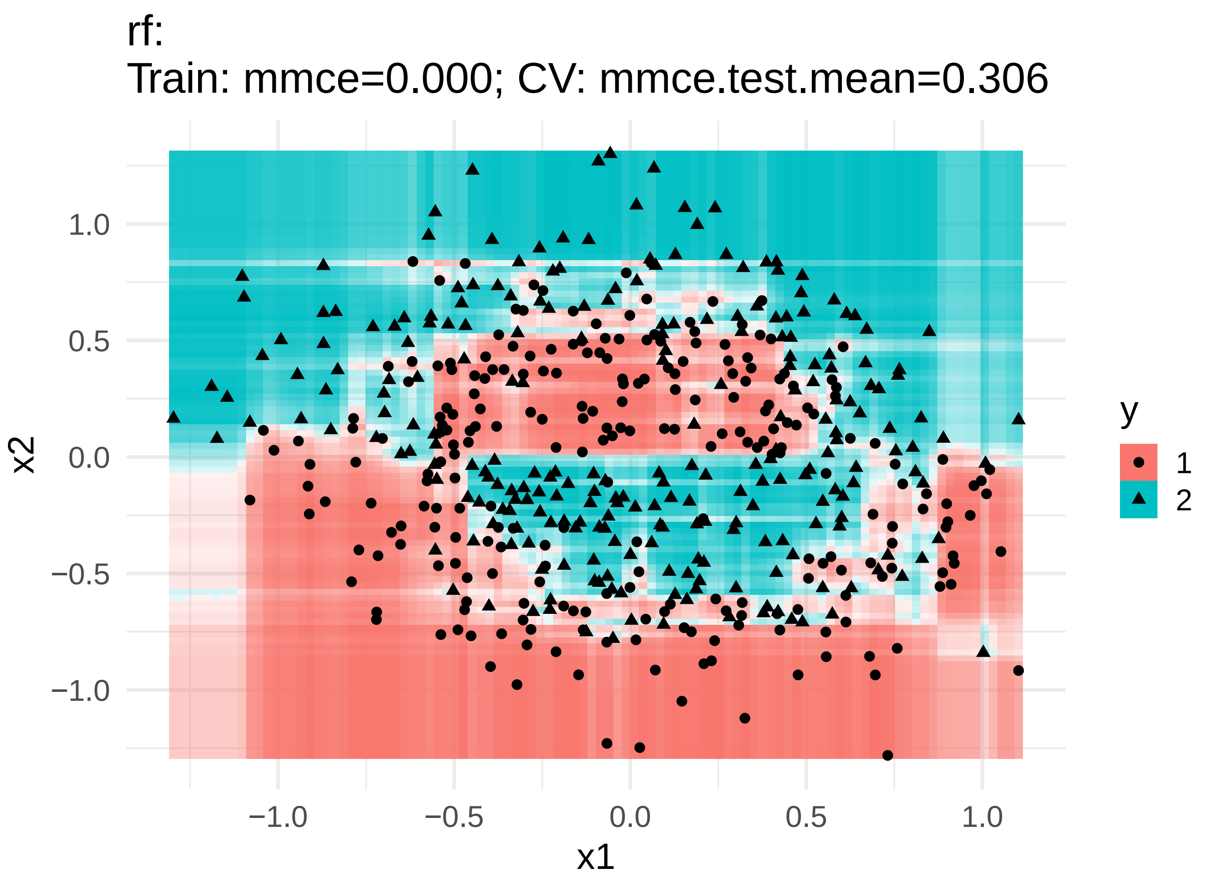
Decision regions for the CART:

```r
library(mlr)
spiral_data <- data.frame(spiral$x, y = factor(spiral$classes))
colnames(spiral_data) <- c("x1", "x2", "y")
# set features that should be inspected within the regions
features <- c("x1", "x2")
spiral_task <- mlr::makeClassifTask(data = spiral_data, target = "y")

mlr::plotLearnerPrediction(
  learner = "classif.rpart", task = spiral_task,
  features = features
)
```



Decision regions for the Random Forest:

```r
mlr::plotLearnerPrediction(
  learner = "classif.randomForest", task = spiral_task,
  features = features
)
```



MNIST example

In this exercise we want to train a random forest classifier, s.t. it can be used to identify the handwritten digits 8 and 9. For that, we'll use the iconic MNIST data set that contains 28×28 greyscale pixel images of handwritten digits.

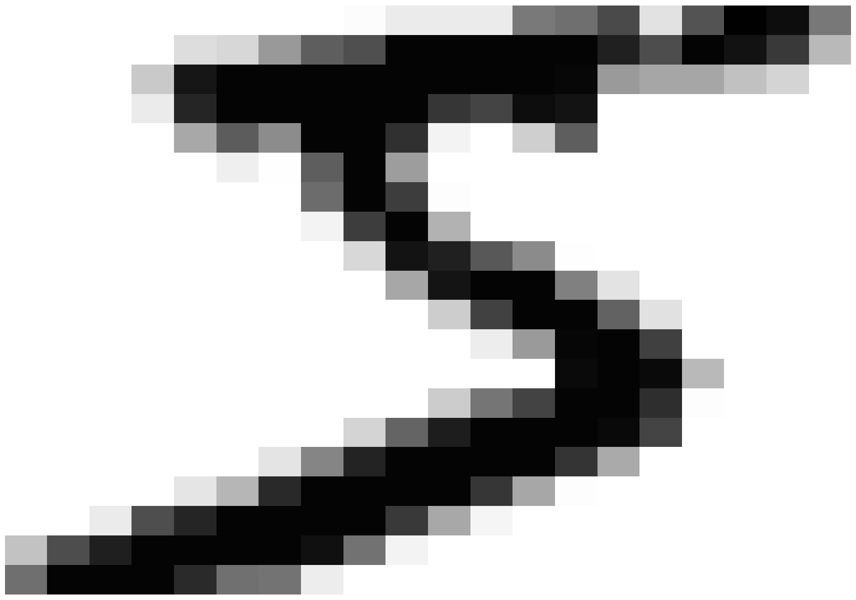
```r
library(readr)
library(dplyr)

mnist_raw <- read_csv("https://www.python-course.eu/data/mnist/mnist_train.csv",
  col_names = FALSE
)
```

Every row of the MNIST data set contains the label (0-9) and the grayscale of the $28*28 = 784$ pixels of the corresponding image. For example:

```r
rotate <- function(x) t(apply(x, 2, rev))

# function which plots a 28*28 raster image
plot_mnist_raster <- function(pixels, title = "") {
  pixels <- rotate(t(matrix(as.numeric(pixels), nrow = 28, byrow = F)))

  #transform matrix m[i.j] into a data.frame with rows (x = i, y = j, value = m[i,j])
  df <- data.frame(x = as.vector(row(pixels)),
                   y = as.vector(col(pixels)),
                   value  = as.vector(pixels))

  ggplot(df, aes(x,y)) + geom_raster(aes(fill=value)) +
    scale_fill_gradient(low = "white", high = "black",
                        limits = c(0,256), guide = "none") +
    ggtitle(title) + theme_void()
}

as.numeric(mnist_raw[1, 1])
```

```
## [1] 5
```

```
plot_mnist_raster(mnist_raw[1, 2:ncol(mnist_raw)])
```



We are only interested in the digits 8 and 9:

```
mnist_8 <- mnist_raw[ mnist_raw[, 1] == 8, ]
mnist_9 <- mnist_raw[ mnist_raw[, 1] == 9, ]

train_size <- 150
set.seed(123)
train_8_indices <- sample(nrow(mnist_8), train_size / 2, replace = FALSE)
train_9_indices <- sample(nrow(mnist_9), train_size / 2, replace = FALSE)

mnist_89_train <- as.data.frame(rbind(
  mnist_8 [train_8_indices, ],
  mnist_9 [train_9_indices, ]
))

mnist_89_train$X1 <- as.factor(mnist_89_train$X1)

mnist_89_test <- as.data.frame(rbind(
  mnist_8 [-train_8_indices, ],
  mnist_9 [-train_9_indices, ]
))

mnist_89_test$X1 <- as.factor(mnist_89_test$X1)
```

Train a random forest model:

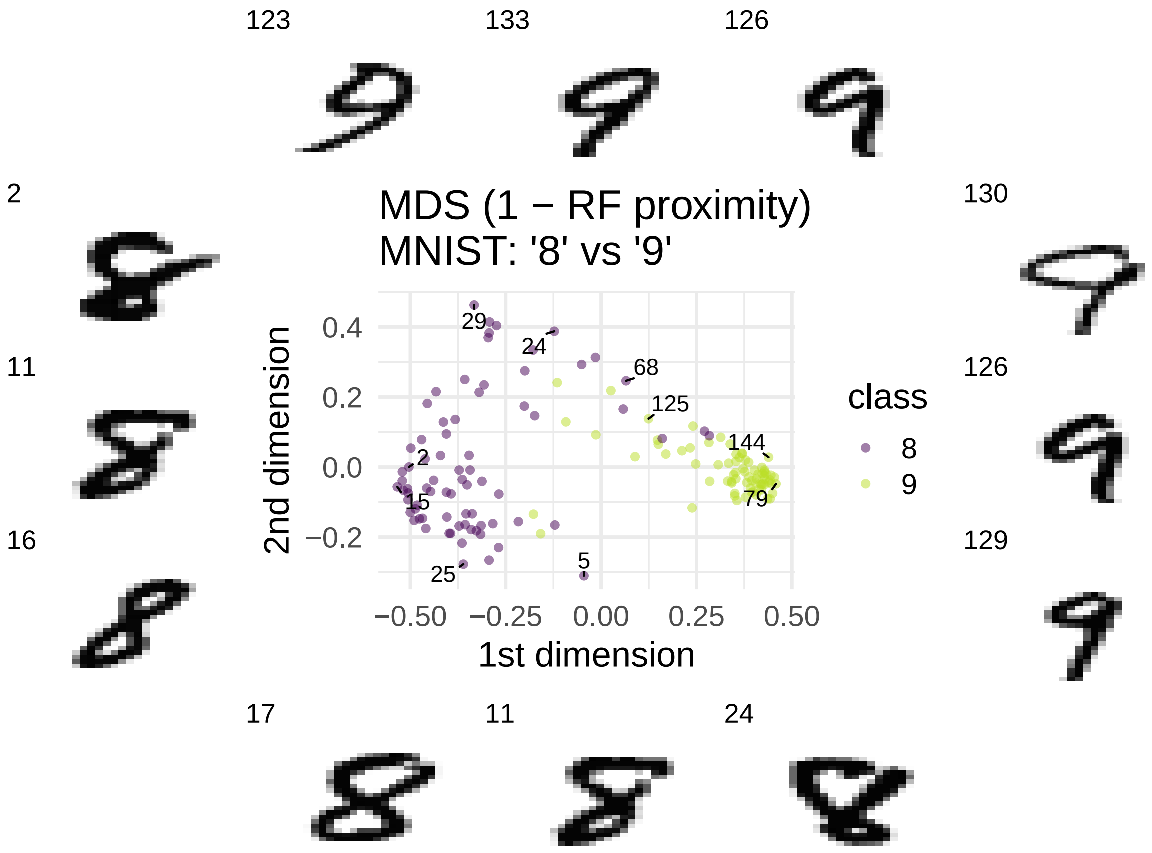
```
set.seed(123)
mnist_89_task <- makeClassifTask(data = mnist_89_train, target = "X1")
mnist_89_learner <- makeLearner("classif.randomForest", proximity = TRUE, oob.prox = TRUE)
mnist_89_rf <- mlr::train(learner = mnist_89_learner, task = mnist_89_task)

# evaluate performance on test data
mnist_89_pred <- predict(mnist_89_rf, newdata = mnist_89_test)
performance(mnist_89_pred)
```

```
##  mmce
## 0.045
```

With proximities we are able to explore how similar the images of our training set are:

```
library(ggrepel)
library(gridExtra)

# turn proximity matrix into dissimilarity matrix
# (proximity of 0 gets turned into a very small value)
mnist_89_dist <- proximity_to_dist(mnist_89_rf$learner.model$proximity)

# apply mds on the proximity matrix
mnist_89_mds <- as.data.frame(cmdscale(mnist_89_dist))
mnist_89_mds$class <- mnist_89_train$X1
mnist_89_mds$id <- 1:nrow(mnist_89_mds)

# pick points on the outside of the point cloud for viz
library(dplyr)
mnist_89_mds <- mnist_89_mds %>%
  mutate(
    dist = sqrt(V1^2 + V2^2),
    sector = cut(atan2(V1, V2), seq(-pi, pi, l = 13)), ordered = TRUE) %>%
  group_by(sector) %>% arrange(dist) %>%
  mutate(example = (dist == max(dist)))

mnist_89_examples <- filter(mnist_89_mds, example)
```

```r
# plot the result
p <- ggplot(data = mnist_89_mds, aes(x = V1, y = V2, colour = class)) +
  geom_point(alpha = 0.5) +
  geom_text_repel(
    data = mnist_89_examples,
    aes(V1, V2, label = id), color = "black",
    min.segment.length = unit(0, "lines")
  ) +

  labs(
    x = "1st dimension", y = "2nd dimension", title =
      "MDS (1 - RF proximity)\nMNIST: '8' vs '9'"
  )

p_examples <- lapply(mnist_89_examples$id, function(i) {
  plot_mnist_raster(mnist_89_train[as.numeric(i), -1])
})

hlay <- rbind(c(NA,8,9,10,NA),
              c(1,4,4,4,5),
              c(2,4,4,4,6),
              c(3,4,4,4,7),
              c(NA,11,12,13,NA))

grid.arrange(plot_mnist_raster(mnist_89_train[2, 2:ncol(mnist_raw)],"2"),
             plot_mnist_raster(mnist_89_train[11, 2:ncol(mnist_raw)],"11"),
             plot_mnist_raster(mnist_89_train[16, 2:ncol(mnist_raw)],"16"),
             p,
             plot_mnist_raster(mnist_89_train[130, 2:ncol(mnist_raw)],"130"),
             plot_mnist_raster(mnist_89_train[126, 2:ncol(mnist_raw)],"126"),
             plot_mnist_raster(mnist_89_train[129, 2:ncol(mnist_raw)],"129"),
             plot_mnist_raster(mnist_89_train[123, 2:ncol(mnist_raw)],"123"),
             plot_mnist_raster(mnist_89_train[133, 2:ncol(mnist_raw)],"133"),
             plot_mnist_raster(mnist_89_train[126, 2:ncol(mnist_raw)],"126"),
             plot_mnist_raster(mnist_89_train[17, 2:ncol(mnist_raw)],"17"),
             plot_mnist_raster(mnist_89_train[11, 2:ncol(mnist_raw)],"11"),
             plot_mnist_raster(mnist_89_train[24, 2:ncol(mnist_raw)],"24"),
             layout_matrix=hlay)
```

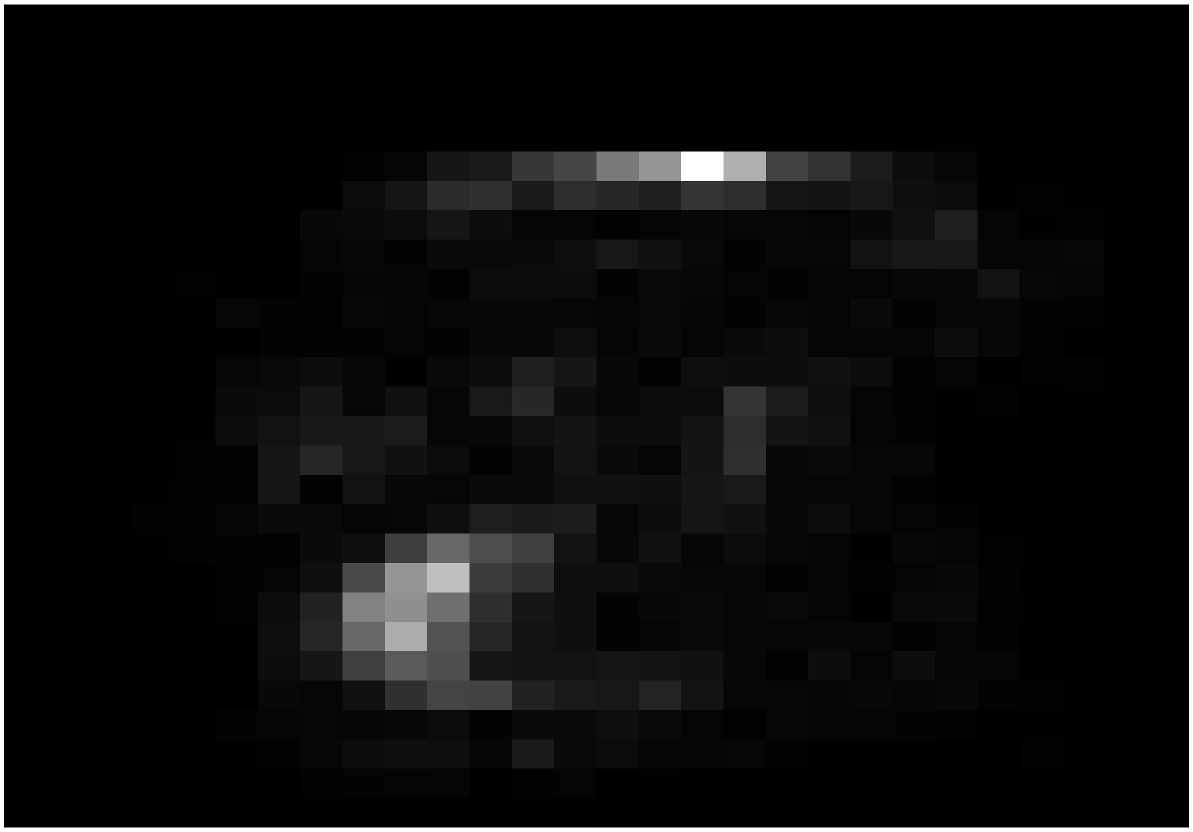


By looking at the variations of the images along the axes we see that while the first axis seems to represent mainly the shape of the lower half of the image, the second coordinate seems to vary with the size of the upper circle. This suggests that we were able to find a meaningful two dimensional representation of our data set, which gives us a good overview of the distribution of the occurring shapes and enables us to visualize the similarity of the digits.

(A further discussion of the problems of visualizing the MNIST data set and possible solutions can be found here)

With variable importance we can now investigate which pixels are the "most important" when we want to tell the digits 8 and 9 apart:

```r
mnist_89_var_imp <- getFeatureImportance(mnist_89_rf)$res
mnist_89_var_imp <- as.data.frame(t(mnist_89_var_imp))

# rescale importance to [0, 256] so they can be used as grayscale-image pixels:
mnist_89_var_imp_rescaled <- 256 - mnist_89_var_imp$V1  / max(mnist_89_var_imp$V1) * 256
plot_mnist_raster(mnist_89_var_imp_rescaled)
```



(white means high importance, black means low importance)

Can you think of why these regions are important?