You can open the lab_5.txt file found on Opencourses with the full code in one file

Step 0: Check if a library is installed

```
# Specify the library name
library_name <- "rpart"

# Check if the package is installed, and install it if not
if (!require(package = library_name, character.only = TRUE)) {
  install.packages(library_name)
  library(library_name)
}
```

Και ένας εναλλακτικός τρόπος που κάνει την ίδια δουλειά:

```
if (!requireNamespace("rpart", quietly = TRUE)) {
  install.packages("rpart")
}
library(rpart)
```

Step 1: Prepare the Environment

First, ensure you have the necessary R packages installed. If not, install them:

```
install.packages(c("caret", "rpart", "class", "nnet"))
```

Step 2: Load the Data

We'll use the `iris` dataset but filter it for a binary classification task:

```
# Load and prepare the data
data(iris)
iris_binary <- subset(iris, Species != "virginica")

# Convert Species to factor with two levels
```

```r
iris_binary$Species <- factor(iris_binary$Species)

# Split the data into training and testing sets
library(caret)        # For data splitting and evaluation
set.seed(123)
train_index <- createDataPartition(iris_binary$Species, p = 0.7, list = FALSE)
train_data <- iris_binary[train_index,]
test_data <- iris_binary[-train_index,]
```

Step 3: Train a Decision Tree

```r
library(rpart)
tree_model <- rpart(Species ~ ., data = train_data, method = "class")

# Predict on test data
tree_predictions <- predict(tree_model, test_data, type = "class")

# Evaluate performance
tree_conf_matrix <- confusionMatrix(tree_predictions, test_data$Species)
print(tree_conf_matrix)
```

Step 4: Train a k-Nearest Neighbors (kNN) Model

```r
library(class)
knn_predictions <- knn(train = train_data[, -5], test = test_data[, -5], cl =
train_data$Species, k = 3)

# Evaluate performance
knn_conf_matrix <- confusionMatrix(knn_predictions, test_data$Species)
print(knn_conf_matrix)
```

Step 5: Train a Simple Neural Network (Perceptron)

We shall train a simple neural network using the `nnet` package. There are, of course, plenty of other libraries we could use, but for the time being ww'll stick with nnet:

```r
library(nnet)
nn_model <- nnet(Species ~ ., data = train_data, size = 1, linout = FALSE, trace
= FALSE)
```

```r
# Predict
nn_probabilities <- predict(nn_model, test_data, type = "class")

# print the model's coefficients
coef(nn_model)

# Evaluate performance
nn_conf_matrix <- confusionMatrix(as.factor(nn_probabilities),test_data$Species)
print(nn_conf_matrix)
```

Step 6: Compare Models

Άσκηση: Δοκιμάστε διαφορετικές παραμετροποιήσεις στους παραπάνω αλγορίθμους και δείτε την επίδρασή τους στην αξιολόγηση.

Step 5***: Custom Perceptron code

```r
# Perceptron training (binary labels must be -1 and +1)
perceptron_train <- function(X, y, lr = 1, epochs = 1000, verbose = FALSE) {
  # X: numeric matrix or data.frame of predictors (rows = samples)
  # y: numeric vector of labels with values -1 or +1
  X <- as.matrix(X)
  n <- nrow(X); p <- ncol(X)
  if(length(y) != n) stop("nrow(X) must equal length(y)")
  if(!all(y %in% c(-1, 1))) stop("y must be -1 or +1")
  # add bias column
  Xb <- cbind(1, X)             # bias as first column
  w <- rep(0, ncol(Xb))         # initialize weights to zero
  for(epoch in 1:epochs) {
    errors <- 0
    for(i in 1:n) {
      xi <- Xb[i, ]
      pred <- ifelse(sum(w * xi) >= 0, 1, -1)
      if(pred != y[i]) {
        w <- w + lr * y[i] * xi  # perceptron update
        errors <- errors + 1
```

```
      }
    }
    if(verbose && epoch %% 50 == 0) cat("epoch", epoch, "errors", errors, "\n")
    if(errors == 0) break
  }
  list(weights = w, epochs = epoch)
}

# Prediction function
perceptron_predict <- function(model, X) {
  X <- as.matrix(X)
  Xb <- cbind(1, X)
  preds <- ifelse(Xb %*% model$weights >= 0, 1, -1)
  as.vector(preds)
}

# Example
set.seed(2)
n <- 100
X <- matrix(rnorm(n*2), ncol=2)
# linearly separable labels
y <- ifelse(2*X[,1] - X[,2] + 0.3 > 0, 1, -1)

# X1=c(-1,-1)
# X2=c(-2,2)
# X3=c(1.5,2)
# X4=c(-3,-2)
# X=rbind(X1,X2,X3,X4)
# y=c(1,-1,-1,1)

m <- perceptron_train(X, y, lr = 0.1, epochs = 1000, verbose = TRUE)
m$weights
preds <- perceptron_predict(m, X)
mean(preds == y)   # accuracy
```