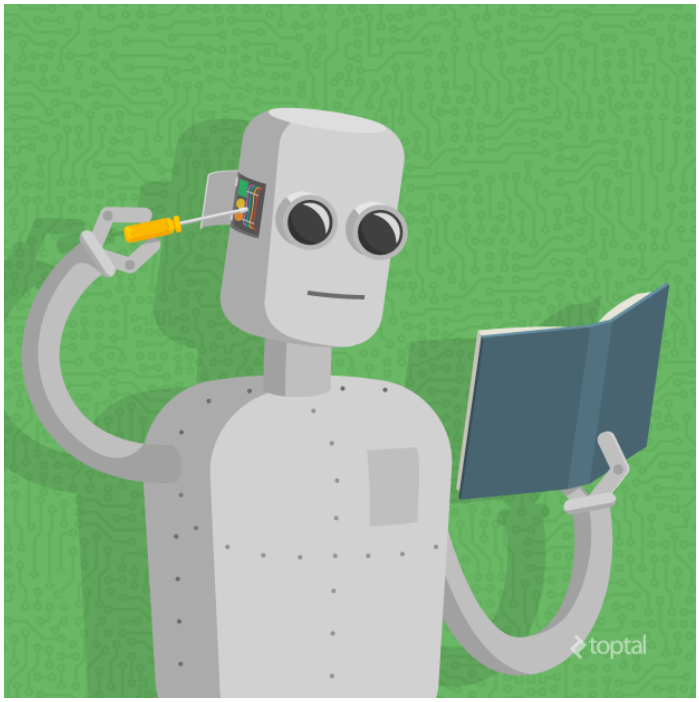
Practical: Machine Learning

BaselRBootcamp 2017



Source: https://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer (https://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer)

Slides

- Here are the introduction slides for this practical on machine learning! (https://therbootcamp.github.io/_sessions/D2S3_MachineLearning/MachineLearning.html)

Overview

In this practical you'll conduct machine learning analyses on a dataset on heart disease. You will see how well many different machine learning models can predict new data. By the end of this practical you will know how to:

1. Create separate training and test data
2. Fit a model to data
3. Make predictions from a model
4. Compare models in how well they can predict new data.

Glossary and packages

Here are the main functions and packages you'll be using. For more information about the specific models, click on the link in *Additional Details*.

Algorithm	Function	Package	Additional Details
Regression	`glm()`	Base R	https://bookdown.org/ndphillips/YaRrr/regression.html#the-linear-model (https://bookdown.org/ndphillips/YaRrr/regression.html#the-linear-model)
Fast-and-Frugal trees	`FFTrees()`	FFTrees	https://cran.r-project.org/web/packages/FFTrees/vignettes/guide.html (https://cran.r-project.org/web/packages/FFTrees/vignettes/guide.html)

Algorithm	Function	Package	Additional Details
Support Vector Machines	`svm()`	`e1071`	https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html (https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html)
Decision Trees	`rpart()`	`rpart`	https://statweb.stanford.edu/~lpekelis/talks/13_datafest_cart_talk.pdf (https://statweb.stanford.edu/~lpekelis/talks/13_datafest_cart_talk.pdf)
Random Forests	`randomForest()`	`randomForest`	http://www.blopig.com/blog/2017/04/a-very-basic-introduction-to-random-forests-using-r/ (http://www.blopig.com/blog/2017/04/a-very-basic-introduction-to-random-forests-using-r/)

Examples

- The following examples will take you through all steps of the machine learning process, from creating training and test data, to fitting models, to making predictions. Follow along and try to see how piece of code works!

```r
# ---------------------------------------------
# A step-by-step tutorial for conducting machine learning
# In this tutorial, we'll see how well 3 different models can
#  predict medical data
# ---------------------------------------------

# ----------------------
# Part A:
# Load libraries
# ----------------------

library(e1071)          # for svm()
library(randomForest)   # for randomForest()
library(rpart)          # for rpart()
library(yarrr)          # for pirateplot()
library(tidyverse)      # for datawrangling and ggplot2
library(FFTrees)        # for the heartdisease data

# ----------------------
# Part B: Create datasets
#  heart_train, heart_test
#  heart_train_fac, heart_test_fac
# ----------------------

heart <- heartdisease   # Save a copy of the heartdisease data as heart

set.seed(101)   # To fix the training / test randomization

# Randomly sort rows
heart <- heart %>%
  arrange(rnorm(nrow(heart)))

# Savew first 125 rows as heart_train and remaining as heart_test
heart_train <- heart %>% slice(1:100)
heart_test <- heart %>% slice(101:nrow(heart))


# Create heart_train_fact, heart_test_fact
#  Just heart_train and hear_test with factors
#  We're only doing this because the randomForest() function
#   requires factors!!!!

heart_train_fac <- heart_train
heart_test_fac <- heart_test

for(i in 1:ncol(heart_train_fac)) {  # Convert character columns and diagnosis to factor

  if(class(heart_train_fac[[i]]) == "character") {

    heart_train_fac[[i]] <- factor(heart_train_fac[[i]])
    heart_test_fac[[i]] <- factor(heart_test_fac[[i]])

  }}


# ------------------------------
# Part I: Build Models
# ------------------------------

# Build FFTrees_model
FFTrees_model <- FFTrees(formula = sex ~ .,
                         data = heart_train)
```

```r
# Build glm_model
glm_model <- glm(formula = factor(sex) ~ .,
                 data = heart_train,
                 family = "binomial")  # For predicting a binary variable

# Build randomForest model
randomForest_model <- randomForest(formula = factor(sex) ~ .,
                                   data = heart_train_fac)


# ------------------------------
# Part II: Explore Models
# ------------------------------

print(FFTrees_model)
summary(FFTrees_model)

print(glm_model)
summary(glm_model)

print(randomForest_model)
summary(randomForest_model)

# ------------------------------
# Part III: Training Accuracy
# ------------------------------

# FFTrees training decisions
FFTrees_fit <- predict(FFTrees_model, heart_train)

# Regression training decisions
#  Positive values are predicted to be 1, negative values are 0
glm_fit <- predict(glm_model, heart_train) > 0

# randomForest training decisions
randomForest_fit <- predict(randomForest_model, heart_train_fac)

# Now calculate fitting accuracies and put in dataframe

# Truth value for training data is heart_train$sex
train_truth <- heart_train$sex

# Put training results together
training_results <- data_frame(FFTrees = mean(FFTrees_fit == train_truth),
                               glm = mean(glm_fit == train_truth),
                               randomForest = mean(randomForest_fit == train_truth))

# Plot training results
barplot(height = unlist(training_results),
        main = "Training Results",
        ylim = c(0, 1),
        col = c("palegreen3", "wheat2", "orchid3"))

# ------------------------------
# Part IV: Prediction Accuacy!
# ------------------------------

# Calculate predictions for each model for heart_test

# FFTrees testing decisions
FFTrees_pred <- predict(FFTrees_model, heart_test)
```

```
# Regression testing decisions
#  Positive values are predicted to be 1, negative values are 0
glm_pred <- predict(glm_model, heart_test) >= 0

# randomForest testing decisions
randomForest_pred <- predict(randomForest_model, heart_test_fac)

# Now calculate testing accuracies and put in dataframe

# Truth value for test data is heart_test$sex
test_truth <- heart_test$sex

testing_results <- data_frame(FFTrees = mean(FFTrees_pred == test_truth),
                              glm = mean(glm_pred == test_truth),
                              randomForest = mean(randomForest_pred == test_truth))

# Plot testing results
barplot(height = unlist(testing_results),
        main = "Testing Results",
        ylim = c(0, 1),
        col = c("palegreen3", "wheat2", "orchid3"))
```

Tasks

- Note, most of this practical will be copying and pasting code from the Examples and only making small changes.
- You should start by copying and pasting all of the code in the examples into a new .R file.
- Try running pieces of the code line by line and understand what it's doing!

Part A: Load packages

A. Load all of the necessary packages. For this practical, we'll need `FFTrees`, `e1071`, `randomForest`, `rpart`, and `tidyverse`.

Part B: Create training and test data

B. Now run the code in Part B to save a copy of the data as a tibble called `heart`. Afterwards, print it to make sure it looks ok.

C. Create separate training dataframes `heart_train` (and `heart_train_fac`) for model training and `heart_test` (and `heart_test_fac`) for model testing. Print each of these dataframes to make sure they look ok.

Part I: Train models on `diagnosis`

1. In our analyses, we will try to predict each patient's diagnosis `diagnosis`. Look at the help menu for `heartdisease` to see what this, and the other variables, mean. Then, Create three new model objects `FFTrees_model`, `glm_model`, and `randomForest_model`, each predicting `diagnosis`.

Part II: Calculate fits for training data

2. Calculate fits for each model with `predict()`, then create `training_results` containing the fitting accuracy of each model in a dataframe. The code will be almost identical to what is in the Example. All you need to do is change the value of `truth_train` to the correct column in `heart_train`. Afterwards, plot the results using `barplot()`. Which model had the best training accuracy?

Part III: Explore models

3. Explore each of the three models by applying `print()` and `summary()`. Can you interpret any of them?

Part IV: Calculate predictions for test data

4. Calculate predictions for each model based on `heart_test`, and then calculate the prediction accuracies. Don't forget to change the value of `truth_test` to reflect the true value for the current analysis! Then plot the results. Which model predicted each patient's diagnosis the best?

Extras and Challenges

5. A fellow colleague things that support vector machines should perform better than the models you used. Is she right? Test her prediction by including support vector machines using the `svm()` function from the `e1071` package in all of your analyses. You'll need to add code for support vector machines at each stage of the machine learning process, model building, data fitting, and data prediction. Was she right?

6. You'll notice in Part C that we trained the models on 125 cases (out of the 303) in the full dataset. In other words, we trained the data on about half of the total cases. Try repeating the same machine learning process as above (for either cholesterol or resting heart rate), but instead of training the models on 125 cases, try training it on only 50 cases (about 15% of the data). How do you think having fewer cases in the training data will affect accuracy in fitting and prediction? When you are done, try training the models based on 250 cases (over 80% of the data) and then making predictions on the remaining cases.

7. In all of our machine learning, we have allowed all models to use all data in the `heartdisease` dataset. What do you think would happen if we only let the models use a single predictor like `age` ? Test your prediction by replicating the machine learning process, but *only* allow the models to make predictions based on `age` . Does one model substantially outperform the others? For example, what happens if you include three variables such as `age` , `cp` and `slope` ? (Hint: You can easily tell a model what specific variables to include using the `formula` argument. For example, `formula = y ~ a + b` will tell a model to model a variable y, but *only* using variables a and b.)

8. How do you think these algorithms would perform on a randomly generated dataset? Let's test this by creating a random training and test dataset, and then see how well the algorithms do. Run the code below to add a random column of data called `random` to `heart_train` and `heart_test` . Then, run your machine learning analysis, but now train and test the models on the new random data column. How well do the models do in training and testing?

```
# Add a new column random to heart_train and heart_test

heart_train$random <- sample(c(0, 1), size = nrow(heart_train), replace = TRUE)

heart_test$random <- sample(c(0, 1), size = nrow(heart_test), replace = TRUE)
```

9. So far we've only looked at the `heartdisease` data. Try conducting a similar analysis on the `ACTG175` data from the `speff2trial` package. For example, you could try to predict whether or not a patient was a intravenous drug user. Which of the different machine learning algorithms performs the best in predicting new data from this dataset? Do you discover any challenges in working with dataset that weren't present in the `heartdisease` data?

Additional reading

- For more advanced machine learning functionality in R, check out the `caret` package caret documentation link (http://topepo.github.io/caret/index.html) and the `mlr` package mlr documentation link (https://cran.r-project.org/web/packages/mlr/vignettes/mlr.html). Both of these packages contain functions that automate most aspects of the machine learning process.
- To read more about the fundamentals of machine learning and statistical prediction, check out An Introduction to Statistical Learning by James et al. (https://www.amazon.com/Introduction-Statistical-Learning-Applications-Statistics/dp/1461471370)