Déployez vos modèles R en production

Bruno Tremblay

2019-05-15

C'est qui le monsieur en avant?

Conseiller expert en actuariat équipe de recherche et modélisation @LaCapitale

Actuariat ULaval 2007, FCAS, CSPA

bruno.tremblay@lacapitale.com
github@meztez
linkedin@neoxone

R Community highlight :

> La fois où Max Kuhn a accepté mon pull request.

topepo commented on Mar 20 — Collaborator

Thanks!

Projet

Déployez une application pour utiliser un pipeline de modélisation prédictive

Ce qu'ils ont dit

> R c'est pas faite pour rouler en production
> Python > R

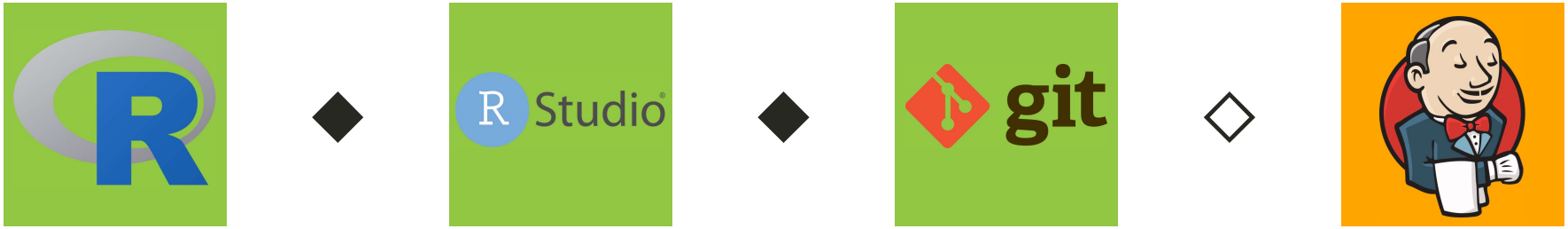
Ce que j'en pense

> [Qualité du code + Composition de l'équipe] > Choix de language[1]

[1] La plupart du temps

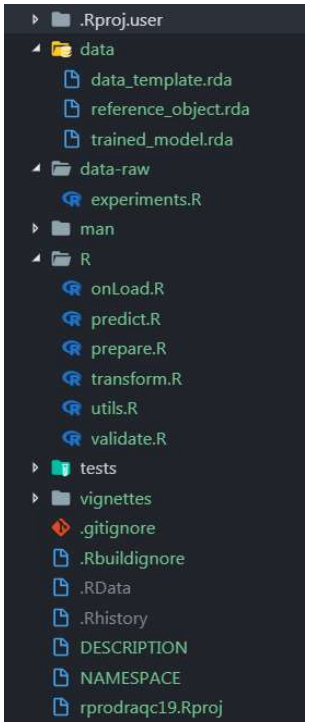
Architecture

Outils



Structure du package[1]

- `data` : Objets nécessaires au pipeline enregistré avec le package.
- `data-raw` : Entraînement des modèles et expérimentations.
- `man` : Documentation générée par roxygen2.
- `R/onLoad.R` : Exécuté au chargement du package.
- `R/predict.R` : Fonctions de prédictions.
- `R/prepare.R` : Ingénierie des fonctionnalités.
- `R/transform.R` : Adapter les données reçues par l'API.
- `R/utils.R` : Autres fonctions dont le warmup.
- `R/validate.R` : Validations pré traitement.
- `tests` : Tests unitaires.
- `vignettes` : Exemples d'appels ou d'utilisation.
- `.gitignore` : Fichiers ignorés git.
- `.Rbuildignore` : Fichiers ignorés R BUILD.
- `.RData` : ...
- `.RHistory` : ...
- `DESCRIPTION` : Description du package.
- `NAMESPACE` : Générée par roxygen2.
- `rprodraqc19.Rproj` : ...



[1] https://support.rstudio.com/hc/en-us/articles/200486488-Developing-Packages-with-RStudio

Pratico-pratique

```
library(profvis)
library(roxygen2)
library(plumber)
debugonce()
browser()
plumb("plumber.R")$run # ou le bouton dans RStudio
```

Documenter avec roxygen2[1]

Identifier les trous noirs avec profvis[2]

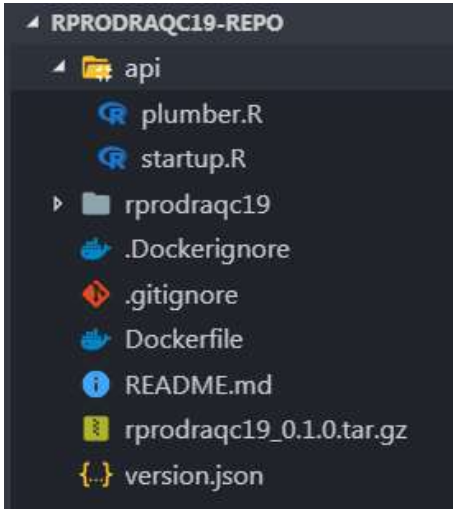
Utiliser debug pour inspecter l'exécution

Valider l'API du package localement

[1] `vignette("roxygen2", package = "roxygen2")`
[2] https://rstudio.github.io/profvis

Structure du repo

- `api/plumber.R` : Définir le routeur plumber.
- `api/startup.R` : Script d'exécution plumber run (avec logs).
- `rprodraqc19` : Les fichiers du package.
- `.Dockerignore` : Fichiers ignorés Docker.
- `Dockerfile` : Configuration image Docker.
- `README.md` : Lecture libre.
- `rprodraqc19_0.1.0.tar.gz` : R Source package built.
- `version.json` : Version récupérée par l'outil d'intégration.



Cycle de vie des modèles

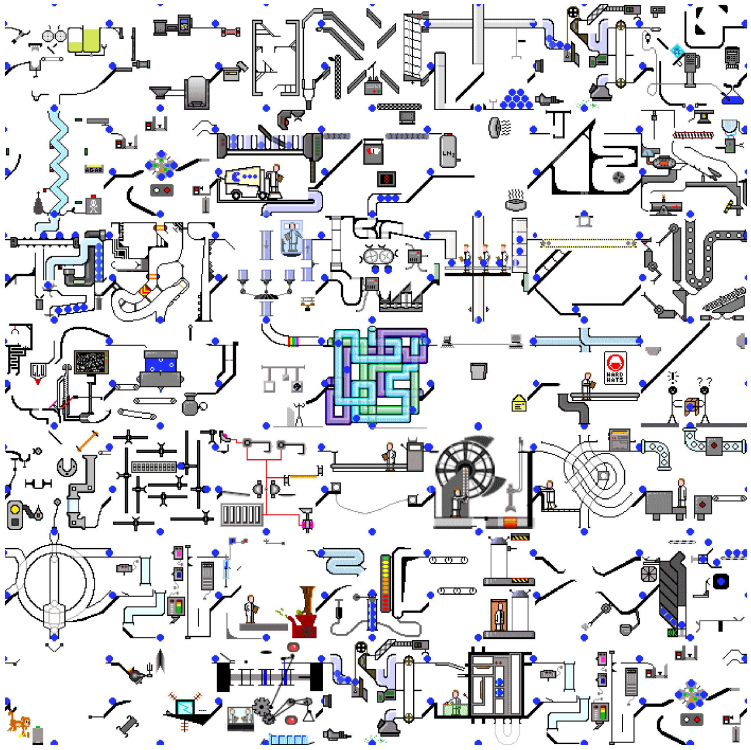
Versions de package

Environnements Dev / Test / Prod

Livraison continue

Intégration continue

Cohabitation des conteneurs



MacGyver (devops)

Appels API

Contrat d'appel

Tests de charges

Logging

Références devops

https://www.rplumber.io/docs/
https://github.com/trestletech/plumber
https://rviews.rstudio.com/2018/07/23/rest-apis-and-plumber/
https://opensource.t-mobile.com/blog/posts/r-tensorflow-api/
https://www.json.org/
https://github.com/docker-slim/docker-slim
https://cloud.google.com/kubernetes-engine/docs/
https://cloud.google.com/blog/products/ai-machine-learning/making-the-machine-the-machine-learning-lifecycle

Autres packages évalués

https://www.opencpu.org/
http://restrserve.org/
https://cran.r-project.org/web/views/ModelDeployment.html

experiments.R

```
library(odbc)
library(data.table)
library(xgboost)
library(Matrix)
library(rprodraqc19)
library(usethis)

con <- dbConnect(odbc(), "DSN", encoding = "latin1", bigint = "integer")
src <- dbGetQuery(con, "SELECT ... FROM ... WHERE ...")

setDT(src)
validate(src)
model_data <- prepare(copy(src))

n <- nrow(model_data)
set.seed(90210)
train_indices <- sort(sample(1:n,0.8*n))
valid_indices  <- (1:n)[-train_indices]

mtrx_train <- xgb.DMatrix(model_data[train_indices, ], label = ifelse(src[train_indices]$TARGET == "O", 1, 0))
mtrx_valid <- xgb.DMatrix(model_data[valid_indices, ], label = ifelse(src[valid_indices]$TARGET == "O", 1, 0))

set.seed(1234)
xgb.tree <- xgb.train(data = mtrx_train,
                      watchlist = list(eval = mtrx_valid, train = mtrx_train),
                      nrounds = 500,
                      objective = "binary:logistic",
                      booster = "gbtree",
                      print_every_n = 50,
                      max_depth = 8,
                      subsample = 0.7,
                      colsample_bytree = 1,
                      eta = 0.03)

xgb.tree$evaluation_log[which.min(xgb.tree$evaluation_log$eval_error)]

trained_model <- xgb.tree
data_template <- rprodraqc2019:::data_template
reference_object <- rprodraqc2019:::reference_object

use_data(trained_model, data_template, reference_object, internal = FALSE, overwrite = TRUE)
```

onLoad.R

```
.onLoad <- function(lib, pkg){
  utils::data(trained_model,
              data_template,
              reference_object,
              package = pkg, envir = parent.env(environment()))
}
```

utils.R

```
year_diff <- function(time1, time2) {
  time1 <- as.POSIXlt(time1, "GMT")
  time2 <- as.POSIXlt(time2, "GMT")
  yeardiff <- time2$year - time1$year
  yeardiff <- ifelse(time2$mon < time1$mon | (time2$mon == time1$mon & time2$mday < time1$mday),
                     yeardiff - 1,
                     yeardiff)

  return(as.numeric(yeardiff))
}

...

#' @importFrom ...
#' @export
warmup <- function() {

  input <- jsonlite::fromJSON('[{"..."}]')
  setDT(input)
  replicate(3, prepare(copy(input)))

  return(invisible())
}
```

transform.R

```r
#' @importFrom ...
transform <- function(input) {

  setDT(input)

  old = c("VAR1", ...)
  new = c("MODVAR1", ...)
  setnames(input, old, new, skip_absent = TRUE)

  input[, VAR4 := "*"]
  input[, VAR5 := as.numeric(VAR5)]
  input[, DATE := as.POSIXct(DATE, tz = "UTC")]

  invisible(input)
}
```

validate.R

```r
#' @export
validate <- function(input) {
  selected <-  c("MODVAR1", "DATE",  ...)
  stopifnot(all(selected %in% names(input)))
  removed <- names(input)[!names(input) %in% selected]
  input[, removed := NULL]
  invisible(x)
}
```

prepare.R

```r
#' @importFrom ...
#' @export
prepare <- function(input) {

  if (nrow(input) > 1) {
    if (all(is.na(input$MODVAR1))) {input[, MODVAR1 := as.character(MODVAR1)]}
  }

  input[, ':=' (JOURSEMAINE = wday(DATE), MOIS = month(DATE), ANNEE = year(DATE))]

  input[is.na(VAR4), VAR4 := "NR"]

  input[, AGE := 0]
  input[!is.na(DATE_ACHAT), AGE_ACHAT := year_diff(DATE_ACHAT, DATE)]
  input[AGE_ACHAT < -1, AGE_ACHAT := -1]
  input[AGE_ACHAT > 50, AGE_ACHAT := 50]

  input[, ':=' (DATE = NULL, DATE_ACHAT = NULL)]

  input[reference_object, on = "MODVAR1 == CP13", TERRITOIRE := TERRCP13]

  for (var in names(input)) {
    if (is.factor(reference_object[[var]])) {
      lvl <- levels(reference_object[[var]])
      fact <- factor(input[[var]], lvl)
      if (sum(is.na(fact))) {
        val <- sort(unique(input[[var]][is.na(fact)]))
        remp <- lvl[1]
        warning(paste("Valeur(s) (", paste0(val,collapse = ","),") du champ", var ,
                      "non comprise dans le domaine reconnu (",
                      paste(lvl, collapse = ", "),"). Le programme utilisera",
                      remp, "comme substitution."))
        fact[is.na(fact)] <- remp
      }
      input[[var]] <- fact
    }
  }

  smm <- fspmatrix(input[, names(reference_object), with = FALSE])

  return(smm)
```

predict.R

```
#' @export
predict_model_api <- function(input) {
  transform(input)
  predict_model(input)
}

#' @importFrom ...
#' @export
predict_model <- function(input) {

  validate(input)
  model_data <- prepare(copy(src))
  model_revision <- list("model revision" = as.character(packageVersion("rprodraqc19")))
  model_predictions <- predict(object = trained_model, newdata = xgb.DMatrix(model_data))

  return(list("model revision" = model_revision, "model predictions" = model_predictions))
}
```

DESCRIPTION

```
Package: rprodraqc19
Type: Package
Title: Pseudo code template
Version: 0.1.0
Author: Bruno Tremblay
Maintainer: Bruno Tremblay <bruno.tremblay@lacapitale.com>
Description: It is a placeholder
License: GPL-2
Encoding: UTF-8
LazyData: false
Depends: R (>= 2.10), xgboost, Matrix, data.table
Imports:
  jsonlite
```

.gitignore

```
.Rproj.user
.Rhistory
.RData
.Ruserdata
```

.Rbuildignore

```
^.*\.Rproj$
^\.Rproj\.user$
^\.RData$
^\.Rhistory$
^data-raw$
^vignettes$
^tests$
```

plumber.R

```
library(rprodraqc19)
warmup()

#* @apiTitle R en Production
#* @api

#* Predict total loss cgen
#* @param input Input for our model
#* @post /predict_model_api
#* @get /predict_model_api
#* @json
function(input) {
    return(predict_model_api(input))
}
```

startup.R

```
library(plumber)
pr <- plumb("/etc/plumber.R")
pr$registerHooks(list(
    postroute = function(req) {
        if (req$REQUEST_METHOD == "POST") {
            cat("[", req$REQUEST_METHOD, req$PATH_INFO, "] - REQUEST - ", req$postBody, "\n", sep = "")
        }
    },
    postserialize = function(req, res) {
        if (req$REQUEST_METHOD == "POST") {
            cat("[", req$REQUEST_METHOD, req$PATH_INFO, "] - RESPONSE - ", res$status, " - ", res$body, "\n", sep = "")
        }
    }
))
pr$run(host = '0.0.0.0', port = 80)
```

.Dockerignore

```
rprodraqc19
```

Dockerfile

```
FROM trestletech/plumber:latest

COPY ./rprodraqc19_0.1.0.tar.gz /tmp/rprodraqc19_0.1.0.tar.gz
COPY ./api /etc

RUN apt-get update && \
    apt-get upgrade -f -y && \
    apt-get install -y apt-utils
RUN R -e 'install.packages(c("xgboost", "data.table", "Matrix"))'
RUN R CMD INSTALL /tmp/rprodraqc19_0.1.0.tar.gz

EXPOSE 80
ENTRYPOINT ["R", "-f", "/etc/startup.R", "--slave"]
```

Useful docker command

```
docker run -d -p 8123:80 image
docker exec -i -t image /bin/bash
docker image ls -a
docker ps
docker cp image:src dest
docker kill image
docker image rm
docker container ls -a
```