# Set up and run a R project in MLflow

In this [story](https://medium.com/@dsml4real/tracking-machine-learning-models-in-r-with-mlflow-9ccce342ce91) I have described how to use [MLflow](https://mlflow.org/) to track machine learning model training. MLflow also comes with a [Projects](https://mlflow.org/docs/latest/projects.html) component that packs data, source code with commands, parameters and execution environment setup together as a self-contained specification. Once a MLproject is defined, users can run it everywhere. Currently MLproject can run Python code or shell command. It can also set up the Python environment for the project specified in the conda.yaml file defined by users.

For R users, it is common to load some packages in the R source codes. These packages need to installed for the R code to run. In the future, it could be a good enhancement for MLflow to add something similar to conda.yaml to set up R package dependencies. But we do not have to wait for it. I will show how to create a MLproject containing R source code and run it with mlflow run command.

First, create a directory and copy the data and R source codes to that directory. For example,

.

└── R

├── MLproject

├── prep.R

├── rpart-example.R

└── wine-quality.csv

In this example, the data to be learned is [wine-quality.csv](https://github.com/adrian555/DocsDump/raw/dev/files/mlflow-projects/R/wine-quality.csv). The example is to run the [rpart-example.R](https://github.com/adrian555/DocsDump/raw/dev/files/mlflow-projects/R/rpart-example.R) to fit a tree model:

# Source prep.R file to install the dependencies

source("prep.R")

# Import mlflow python package for tracking

library(reticulate)

mlflow <- import("mlflow")

# Load rpart to build a tree model

library(rpart)

# Read in data

wine <- read.csv("wine-quality.csv")

# Build the model

fit <- rpart(quality ~ ., wine)

# Save the model that can be loaded later

saveRDS(fit, "fit.rpart")

# Save the model to mlflow tracking server

mlflow$log\_artifact("fit.rpart")

# Plot

jpeg("rplot.jpg")

par(xpd=TRUE)

plot(fit)

text(fit, use.n=TRUE)

dev.off()

# Save the plot to mlflow tracking server

mlflow$log\_artifact("rplot.jpg")

The R code above includes three parts: the model training, the artifacts logging through MLflow, and the R package dependencies installation. In this example, these two R packages, reticulate and rpart, are required for the code to run. To pack these codes into a self-contained project, some sort of script should be run to automatically install these packages if the platform does not have them installed.

With our approach, any specific R package needed for the project is going to be installed through [prep.R](https://github.com/adrian555/DocsDump/raw/dev/files/mlflow-projects/R/prep.R) with these codes:

# Accept parameters, args[6] is the R package repo url

args <- commandArgs()

# All installed packages

pkgs <- installed.packages()

# List of required packages for this project

reqs <- c("reticulate", "rpart")

# Try to install the dependencies if not installed

sapply(reqs, function(x){

if (!x %in% rownames(pkgs)) {

install.packages(x, repos=c(args[6]))

}

})

Before packaging these into a MLproject, try to test by directly invoking Rscript command as follow:

Rscript rpart-example.R https://cran.r-project.org/

From the MLflow UI, you should see this run been tracked like this screen snapshot:

Now let’s write the spec and pack this project into a MLproject that MLflow knows to run. All needed to be done is creating the [MLproject](https://github.com/adrian555/DocsDump/raw/dev/files/mlflow-projects/R/MLproject) file in the same directory.

name: r\_example

entry\_points:

main:

parameters:

r-repo: {type: string, default: "https://cran.r-project.org/"}

command: "Rscript rpart-example.R {r-repo}"

In this file, it defines a r\_example project with a main entry point. The entry point specifies the command and parameters to be executed by the mlflow run. For this project, Rscript is the shell command to invoke the R source code. r-repo parameter provides the URL string where the dependent packages can be installed from. A default value is set. This parameter is passed to the command running the R source code.

Now that all are set so the project can be checked in and pushed to github repository. With following command, it can be run on any platform that has R installed.

mlflow run https://github.com/adrian555/DocsDump#files/mlflow-projects/R

The project can also be viewed from the MLflow tracking UI like this screen snapshot:

The differences between this view and previous run without Mlproject spec are the Run Command which captures the exact command to run the project, and the Parameters which automatically logs any parameters passed to entry points.

This is exactly what Projects component of MLflow is designed for, to define the project and make it easily to be rerun. R users can quickly set up their projects and enjoy the easiness of tracking and running projects with MLflow once going through this example.