reticulate is an R package that allows us to use Python modules from within RStudio. I recently found this functionality useful while trying to compare the results of different uplift models. Though I did have R’s uplift package producing [Qini](https://rdrr.io/cran/uplift/man/qini.html) charts and metrics, I also wanted to see how things looked with Wayfair’s promising pylift package. Since pylift is only available in python, reticulate made it easy for me to quickly use pylift from within RStudio.

Qini Code

[library](https://rdrr.io/r/base/library.html)(uplift)

*### simulate data for uplift modeling*

[set.seed](https://rdrr.io/r/base/Random.html)(123)

dd <- [sim\_pte](https://rdrr.io/cran/uplift/man/sim_pte.html)(n = 1000, p = 20, rho = 0, [sigma](https://rdrr.io/r/stats/sigma.html) = [sqrt](https://rdrr.io/r/base/MathFun.html)(2), beta.den = 4)

dd$treat <- [ifelse](https://rdrr.io/r/base/ifelse.html)(dd$treat == 1, 1, 0)

*### fit uplift random forest*

fit1 <- [upliftRF](https://rdrr.io/cran/uplift/man/upliftRF.html)(y ~ X1 + X2 + X3 + X4 + X5 + X6 + [trt](https://rdrr.io/cran/uplift/man/trt.html)(treat),

[data](https://rdrr.io/r/utils/data.html) = dd,

mtry = 3,

ntree = 100,

split\_method = "KL",

minsplit = 200, *# need small trees as there is strong uplift effects in the data*

verbose = [**TRUE**](https://rdrr.io/r/base/logical.html))

[print](https://rdrr.io/r/base/print.html)(fit1)

[summary](https://rdrr.io/r/base/summary.html)(fit1)

*### predict on new data*

dd\_new <- [sim\_pte](https://rdrr.io/cran/uplift/man/sim_pte.html)(n = 2000, p = 20, rho = 0, [sigma](https://rdrr.io/r/stats/sigma.html) = [sqrt](https://rdrr.io/r/base/MathFun.html)(2), beta.den = 4)

dd\_new$treat <- [ifelse](https://rdrr.io/r/base/ifelse.html)(dd\_new$treat == 1, 1, 0)

pred <- [predict](https://rdrr.io/r/stats/predict.html)(fit1, dd\_new)

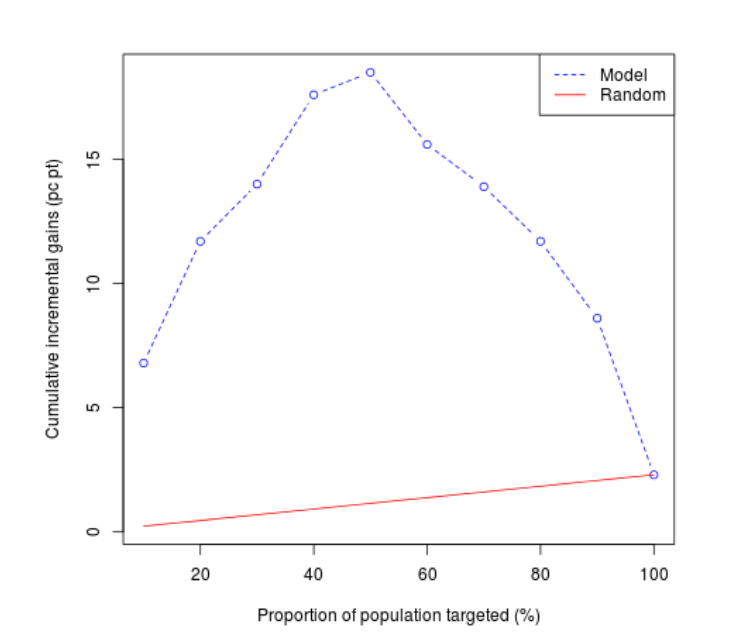
*### evaluate model performance*

perf <- [performance](https://rdrr.io/cran/uplift/man/performance.html)(pred[[](https://rdrr.io/r/base/Extract.html), 1], pred[[](https://rdrr.io/r/base/Extract.html), 2], dd\_new$y, dd\_new$treat, direction = 1)

*### compute Qini coefficient*

Q <- [qini](https://rdrr.io/cran/uplift/man/qini.html)(perf, plotit = [**TRUE**](https://rdrr.io/r/base/logical.html))

Q



In the article below, I’ll show how I worked through the following circumstances:

* Since pylift has only been tested on Python >= 3.6, and my system version of Python was 2.7, I needed to build and install Python 3.6 for myself, preferably within a self-contained virtual environment.
* I wanted to install pylift in the virtual environment and set up reticulate in my R Project to work within that environment.
* Finally, I needed to access pylift from an R Markdown document via the reticulate interface.

**Setting up Python, virtualenv, and RStudio**

Python 2.7 is the default on the systems I use (CentOS 6/7). Since I did not want to modify the system-level Python version, I installed Python 3.6.x at the user level in $HOME/opt and created a virtual environment using Python 3. I then activated the Python 3 environment and installed pylift. Finally, I ensured RStudio-Server 1.2 was installed, as it has advanced reticulate support like plotting graphs in line in R Markdown documents.

Below is a brief script that accomplishes the tasks in bash on CentOS 7:

cd ~

mkdir tmp

cd tmp

wget https://www.python.org/ftp/python/3.6.2/Python-3.6.2.tgz

tar -xzvf Python-3.6.2.tgz

cd Python-3.6.2

./configure --prefix=$HOME/opt/python-3.6.2 --enable-shared

make

make install

cd ~

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$HOME/opt/python-3.6.2/lib

virtualenv -p $HOME/opt/python-3.6.2/bin/python3 pylift

source pylift/bin/activate

cd pylift

git clone https://github.com/wayfair/pylift

cd pylift

pip install .

pip install -r requirements.txt

cd

wget https://s3.amazonaws.com/rstudio-ide-build/server/centos6/x86\_64/rstudio-server-rhel-1.2.1335-x86\_64.rpm

sudo yum install -y --nogpgcheck rstudio-server-rhel-1.2.1335-x86\_64.rpm

sudo rstudio-server start

Some notes:

* the --enable-shared option is required when building Python in order for reticulate to work
* the LD\_LIBRARY\_PATH library also needs to be set prior to creating the virtual environment
* we use virtualenv to create a virtual environment called “pylift” and then ensure that all Python packages are installed to that environment only (so as not to pollute any other environments we are working with)
* we then clone the pylift source and install pylift along with all of its requirements via pip install -r requirements.txt
* finally, we install the RStudio Server 1.2 Preview version in order to leverage its advanced reticulate features

**Using Python from within RStudio via reticulate**

Switching from bash to RStudio, we load reticulate and set it up to use the virtual environment we just created. Finally, and specific to pylift, we set matplotlib parameters so that we can plot directly in R.

library(reticulate)

Sys.setenv(LD\_LIBRARY\_PATH = paste0(Sys.getenv("HOME"),"/opt/python-3.6.2/lib"))

Sys.getenv("LD\_LIBRARY\_PATH")

use\_virtualenv("/home/rstevenson/pylift", required=TRUE)

py\_config()

# Currently this must be run in order for R-markdown plotting to work

matplotlib <- import("matplotlib")

matplotlib$use("Agg", force = TRUE)

**Test that it works**

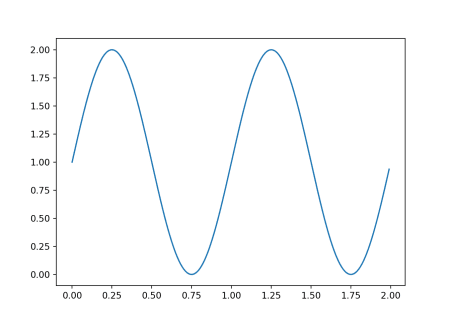
import matplotlib.pyplot as plt

import numpy as np

t = np.arange(0.0, 2.0, 0.01)

s = 1 + np.sin(2\*np.pi\*t)

plt.plot(t,s)



When run, the above code chunk should display a sinusoidal graph below it.

import numpy as np, matplotlib as mpl, matplotlib.pyplot as plt, pandas as pd

from pylift import TransformedOutcome

from pylift.generate\_data import dgp

# Generate some data.

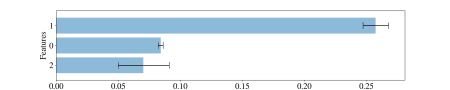
df = dgp(N=10000, discrete\_outcome=True)

# Specify your dataframe, treatment column, and outcome column.

up = TransformedOutcome(df, col\_treatment='Treatment', col\_outcome='Outcome', stratify=df['Treatment'])

# This function randomly shuffles your training data set and calculates net information value.

up.NIV()



The above Python chunk uses reticulate from within RStudio to interact with pylift in the context of a custom virtual environment, using a custom version of Python. This degree of customization and functionality should be useful to users who:

* want to use a different Python version than they typically use while not affecting their typical setup by way of a virtual environment
* want to install a Python module like pylift within a virtual environment so as not to affect any of their user- or system-level Python module installations
* want to use reticulate from RStudio to access a custom virtual environment, Python version, and Python modules
* wants to be able to delete the virtual environment and R-Project and have everything go back to the way it was
* wants to be able to reproduce or share the environment exactly so that the workflow can be shared with others