Architecture

This API has been built with Flask and it runs on the development Server supplied by Flask. The application is coded in python. In this way i was able to include some machine learning libraries that i use to classify users. This whole thing is wrapped into a docker image, so that it becomes portable (dependencies are already included in the image),easy to maintain (one container one service) and to set up (containers start in a few seconds).

Technology Involved:

Docker A software to build and deploy containers. A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

Flask-Restful Flask is a lightweight web application framework with the ability to scale up to complex applications.

Scikit-learn A machine learning library that provides you with many techniques to clean, standardise and process your data to build predictive models. I used SupportVectorClassification, RandomForestClassification and some other modules for preprocessing. To be able to install this package in a docker image, i had to include other required packages and compilers in the docker image (like numpy, scipy, cython, gcc, ...)

API Documentaion

The application API is organized around REST. This API accepts JSON-encoded requests in POST method, returning JSON-encoded responses. The payload of the request should be a JSON having websites as keys and numberOfVisits as values. The answer will either be a JSON with key='name' and value='prediction' (if the request was processed correctly) or a JSON containing an error message

CRUD

Operation	Http	Uri	Req Body	Res Body
Predict with SVC	POST	/api/svc	web-hystory	name
Predict with RandForest	POST	/api/forest	web-hystory	name

eg.

web-hystory = { "http://en.wikipedia.org/wiki/Category:Obscenity_controversies_in_music": 2, "http://en.wikipedia.org/wiki/Category:Cities_in_Dallas%E2%80%93Fort_Worth_metroplex": 4, "http://en.wikipedia.org/wiki/Category:Entertainment_companies_based_in_New_York": 2, "http://en.wikipedia.org/wiki/Category:Hospital_buildings_on_the_National_Register_of_Historic_Places": 1, "http://en.wikipedia.org/wiki/Category:County_seats_in_Wisconsin": 5 } name = { "name": "michelle" }

Try it

To test out this application you can run the docker image with

> sudo docker run --name webapp -d -p 80:5000 app-v1

In this way the server will start up and you will be able to contact the API at the following links*/api/svc /api/forest*

Then you can send a request like this

> curl --header "Content-Type: application/json" --request POST --data \
> '{"wiki.org/writer_month":10,"wiki.org/music_hiv":3}' http://localhost/api/svc

or you can visit http://localhost (the form won't send a JSON request, but the app should be able to parse it anyway)

Matching algorithms

Support Vector Classification

This algorithm tries to classify data by building an hyperplane that maximize the distance between 2 sets of points. Finding the best hyperplane is a convex optimization problem (that's solved by the library) prediction: is w.x + b >= 1 ? (where w is the vector perpendicular to the hyperplane, x is the vector input, and b is a constant) to optimize we try to minimize ||w||

Random Forest

This algorithm selects groups of random features and for each group builds a decision tree. To decide how to split a node you compute the gini coefficient so that you maximize the loss of entropy at each step. Probably many trees will make bad choices, but some of them will be good trees and these trees will decide the right choice (because the wrong ones cancels each other vote).