### Intro

This task serves as an example of a fairly common workflow in DS projects, where some data comes in that must be munged into a format edible by the classification algorithms, a model is trained on the data and gets deployed as a REST API for easy consumption by whomever requests the classifier.

### Task

The task comprises four parts:

* Data transformation
* Model training
* Deploying the model in a REST API
* Building a pipeline

#### Transformation

The data consists of multiple files. You must combine relevant files, filter out the necessary features, deal with inconsistencies, if any, and format the data for the classifier.

#### Model training

In the file *rating\_final.csv* there are three separate ratings. Your classification objective is the first of these, named simply *rating*. The performance of the classifier does not matter. Any will do.

#### Deployment

In this part you have to deploy the model created in the previous step as a REST API. The API can be very simple and just have a single endpoint to which the user sends a vector and gets back a predicted rating.

#### Pipeline

All the previous steps should finally be added to a pipeline so the whole process could be easily re-run when new data is added to the data folder.

The solution to the task should be presented as a git repository at your preferred service provider. Don’t forget clean code principles, commit early and often. Also, make a readme where you describe how you tackled the assignment, what components you used and if anything was challenging.

### Materials:

For this task you are given a dataset that originates from [UCI](https://archive.ics.uci.edu/ml/datasets/Restaurant+%26+consumer+data). There’re a lot of parameters in this data, but let’s assume that we already know that the features we want to use for this task are known and as follows:

* Price
* Parking
* Smoking Area
* Other Services
* Dress Code
* Accessibility

As a convenience you are also given a boilerplate for a Flask API just to save time.

### Notes:

* Model versioning can be disregarded. Let’s assume that the client always wants to use the latest model and does not care which version is being used.
* The pipeline should be built in python and most likely you will need to use libraries such as **pandas, scikit-learn, flask** and **luigi**. You are not limited to those and are completely free to even write your own csv parser if you will, but these are most likely to make your life simpler.
* It is safe to assume that new data replaces the old data and file names stay the same.