# Finding nearest neighbours using min-hashing

In this lab, we apply min-hashing to predict user ratings for the [Amazon book reviews dataset]

## Dataset(s)

Recall that your data is a `.json` file. Each line describes a review and has the following format:


```
{"reviewerID": "A14CK12J7C7JRK", "asin": "1223000893", "reviewerName":
"Consumer in NorCal", "helpful": [0, 0], "reviewText": "I purchased the Trilogy
with hoping my two cats, age 3 and 5 would be interested.  The 3 yr old cat was
fascinated for about 15 minutes but when the same pictures came on, she got
bored.  The 5 year old watched for about a few minutes but then walked away. It
is possible that because we have a wonderful courtyard full of greenery and
trees and one of my neighbors has a bird feeder, that there is enough going on
outside that they prefer real life versus a taped version.  I will more than
likely pass this on to a friend who has cats that don't have as much wildlife
to watch as mine do.", "overall": 3.0, "summary": "Nice Distraction for my cats
for about 15 minutes", "unixReviewTime": 1294790400, "reviewTime": "01 12,
2011"}
```


## Our analysis goal

The dataset is split into a a training and a test csv file, with the format

`ReviewerId,asin,rating,Timestamp`

Only this information from the original dataset is retained (`Timestamp` is not used for the moment). Given a pair `(u, i)` in the training set, our goal is estimating $r_u(i)$, i.e., $u$'s rating of item $i$. To this purpose, we use the following estimator:

$$\hat{r}_u(i) = \overline{r}_u + \frac{\sum_{v\in N(u,i)}(r_v(i) - \overline{r}_v)J(u,v)}{\sum_{v\in N(u,i)} J(u,v)}$$

where $\overline{r}_x$ is the average rating given by a user $x$ on the items she rated in the the training set, $J(x, y)$ is the Jaccard similarity between the items rated by $x$ and $y$ (in the training set) and $N(u, i)$ is the subset of $u$'s *neighbours* in the training set who also rated $i$. In particular, $v$ is considered a neighbour of $u$ whenever $J(u, v) \geq \theta$, with $\theta$ a suitable threshold. We could tentatively pick $\theta = 0.05$ for this dataset, being aware that this is a critical choice, which might i) have an impact on our design choices (e.g., the parameters of the LSH data structure) and ii) on the quality of results (e.g., too low a $\theta$ might negatively impact accuracy of the estimate, too high a value might result in too few similar users to the one under consideration).

## Implementing the idea

Given the size of the dataset, we use LSH for Jaccard similarity estimation. This means, in the above formula

- $J(x, y)$ will actually be an estimate of the actuall Jaccard similarity between $x$'s and $y$'s baskets

- $N(u,i)$ will be the subset of users that are near neighbours of $u$ according to LSH and who also happened to have rated $i$.

For development, we will resort to scikit surprise for the recommender system infrastructure and to perform validation, while we use the datasketch package's implementation LSH for Jaccard similarity.

Note that you are implementing your own prediction algorithm in *surprise*, so please first of all refer to the guidelines given in the documentation. To simplify things, I uploaded a stub of a python module you might implement. You will notice that implementing your own prediction algorithm entails three main steps: i) extending the `AlgoBase` class of *surprise*; ii) implementing the `fit` method (here is where you initialized the LSH data structure using your training set); iii) implementing the `estimate` method (this is invoked in the validation phase, one time for each record in the test set).

Please feel free to modify the code I gave you