67978: A Needle in a Data Haystack
Introduction to Data Science
Homework #1: Frequent Itemset, Recommender Systems
Due: **26 Nov, 11:59pm, on Moodle**

Read the instructions **carefully** (that's a good idea in general).

- There will be two submission links, for theoretical and practical parts.
- Each person submits their own theoretical part. The theoretical part should be a single file in pdf format only (no docx or jpg) named **ex1t_ID.pdf** (ID is your ID).
- If you are submitting handwritten answers, make sure they are crystal clear.
- Only **one** person from each group should submit the practical part code and answers. **All three people** should write the name and id of their partners in the theoretical part pdf.
- For the practical part, the person who submits should submit a single ZIP file named **ex1p_ID.zip** (where ID is the ID of the person submitting). The zip should contain a folder named **code** and a folder called **output**.
- Points may be reduced for submissions that fail to comply.

‘

Problem 1 (Frequent Itemsets, Association Rules).

(a) Consider Table 1. Suppose the sup threshold = 30%. Find all *frequent itemsets* using Apriori. Show the candidate and frequent itemsets for each iteration.

TID	Itemset
1	$\{a, b, d\}$
2	$\{a, b, c, d\}$
3	$\{a, c, d, e, f\}$
4	$\{b, c, d\}$
5	$\{b, c, d, e\}$
6	$\{b, d, e\}$
7	$\{c, d, e\}$
8	$\{b, c, e\}$
9	$\{a, d, e\}$
10	$\{b, c\}$

Table 1: Transactions

(b) When picking high-confidence association rules, often rules with 100% confidence are not interesting to the user (but rules with 99% confidence are). Explain why.

Problem 2 (Strange Baskets).

Imagine there are 100 baskets, numbered 1,2,...,100, and 100 items (also numbered 1,2,...,100). Item i is in basket j if and only if i divides j with no remainder. For example, basket 24 is the set of items {1,2,3,4,6,8,12,24}.

(a) Describe all the association rules that have 100% confidence. Give an example.

(b) If the support threshold is 5, which items are frequent? Which *pairs* of items are frequent?

(c) What is the confidence of the following rules? $\{5, 7\} \rightarrow 2, \{2, 3, 4\} \rightarrow 7$.

Problem 3 (Borrrrrrring). This question involves data from which nothing interesting can be learned about frequent itemsets, because there are no sets of items that are correlated. Suppose the items are numbered 1 to 10, and each basket is constructed by including item i with probability $\frac{1}{i}$, each decision being made independently of all other decisions. That is, all the baskets contain item 1, half contain item 2, a third contain item 3, and so on. Assume the number of baskets is sufficiently large that the baskets collectively behave as one would expect statistically. Prove that there are no interesting association rules; i.e., the interest of every association rule is 0.

Problem 4 (Recommender Systems). Consider the following utility matrix, representing the ratings, on a 1-5 star scale, of eight items, a through h, by four users A, B, C and D. Compute the following from the data of this matrix.

	a	b	c	d	e	f	g	h
A	4	5		5	1		2	3
B		4	4	3	1	2	3	
C	2		1	3		1	5	2
D	5	4		5	1		3	3

(a) Treating the utility matrix as boolean, compute the Jaccard similarity between each pair of users.

(b) Repeat Part (a), but use cosine similarity.

(c) Repeat Part (a), but use centered cosine similarity.

(d) Which of your answers would change if we add another item that nobody has rated? Explain.

(e) Predict user A's rating of item f using user-user collaborative filtering ($|N| = 2$, predicted score weighted by similarity).

(f) Show that the centered cosine similarity measure corresponds to the Pearson correlation.

Problem 5 (Creepy Crawlers).

In this question, you will use crawling to create a (tiny) dataset.
Crawling websites is impolite in general, so MAKE SURE you are following these rules:

- Start by downloading a single page to your computer. Only when you have it right (that is, you can open the local file, parse the html and save the output in the format you want), you can add a loop in and go for multiple pages.
- Make sure there is a significant delay between each request to the website (several seconds should do the trick).
- Be **really** careful. :)

Instructions: Pick a crawling task from the list below. For the sake of this homework, we only need **300 pages**, but your code should be able to crawl the entire site, if we remove that restriction. (In other words, do not hand-code URLs you want to crawl). You should submit three things:

(a) The code, inside a directory called **code**.

(b) Output should be **a JSON file** (indented for readability, please). This is the general format:

```
{
  "records": {
    "record": [
      {
        "id": "1",
        "url": "http://something",
        "field1": "value 1",
        "field2": "value 2"
      },
      {
        "id": "2",
        "url": "http://something",
        "field1": "value 1"
      },
      {
        "id": "3",
        "url": "http://something",
        "field1": "value 1",
        "field2": "value 2"
      }
    ]
  }
}
```

See task-specific fields.

(c) In addition, attach **one example**: a **screenshot** of one of the pages you crawled, plus the corresponding JSON.

You can use existing code for a web crawler/scraper in your favorite language (or write your own, if you so choose). A part of the assignment is experimenting with tools, so I will not recommend specific ones. I will, however, recommend considering some tool for extracting content from html, if html is too convoluted (e.g., Beautiful Soup, lxml).
Have fun. :)

Options:

- Kickstarter, Technology category: `https://www.kickstarter.com/discover/categories/technology`
  Fields: Creator, Title, Text (entire html), DollarsPledged, DollarsGoal, NumBackers, DaysToGo, AllOrNothing. Extra credit: have a sub JSON of rewards (Text, Price, NumBackers, TotalPossible-Backers).
- Indiegogo, Home category: `https://www.indiegogo.com/explore/home?project=all&project=all&sort=trending`
  Fields: Creators, Title, Text (entire html), DollarsPledged, DollarsGoal, NumBackers, DaysToGo, FlexibleGoal. Extra credit: have a sub JSON of perks (Text, Price, NumBackers, TotalPossible-Backers).
- Quirky `https://invent.quirky.com/` , you might need to create an account.
  Fields: ProductName, Category, ShortDescription, SubmittedBy, Summary, Features (array).
  Features example: `https://invent.quirky.com/invent/326566/features`
- TopCoder, previous challenges, `https://www.topcoder.com/challenges?bucket=past`.
  Features: Title, Tags, Text (entire html), Prize, Details, NumRegistrants, NumSubmissions, NumWinners, ChallengeLinks (on the right), FinalReview (on the right)

Easier (worth 75% of the points for this question):

- Innocentive, `https://www.innocentive.com/ar/challenge/browse`.
  Features: Title, Tags, Award, Active Solvers, Abstract, Overview, Status (under eval, open, awarded...)
- Instructables (home or technology categories, `https://www.instructables.com/home/` , `https://www.instructables.com/technology/`)
  Features: Title, Category, Creator, Views, Likes, NumComments, Fulltext
- Manuals of home appliances by Sony `https://www.manualslib.com/brand/sony/` with table of contents (if there aren't enough, extend to other companies)
  Features: Title, Category, TableOfContests (array of strings, no need for page number)
- Recipes (cake recipes: `https://www.allrecipes.com/recipes/276/desserts/cakes/`)
  Fields: Title, Creator, Rating, NumMadeIt, NumReviews, NumPhotos, Ingredients (array of strings), Directions (array of strings), PrepTime, CookTime, ReadyIn)

Problem 6 (Meta). How long (in hours) did this assignment take?