1. Goals and constraints

The goal is to display a ***list of 5 recommended products*** in order to increase the “back to school” time period sales.

<u>Constraints are :</u>

- Focus on recommending products based on customer’s overall ***profiles***.

- To reach a high ***diversity*** in recommended products :
    - "school supplies, consumable supplies, durable office equipment"
    - " both cheaper and more expensive products"

- To reach a high ***serendipity*** in recommended products :
    - "customer discovery of new products they likely couldn’t buy at a local store"

2. Data

<u>Available data:</u>

- I have a 2-dimensional ***ratings matrix***, filled for all users and for all items :
    - Ratings are integers between 1 and 5.
    - Only 7% of filled ratings.
    - There are 100 users and 200 items.

- I also have ***items description*** for all items :
    - Those are items content data.
    - Prices :
        - 11 items are without price.
        - 85% of all prices are under 20 dollars.
    - Brands :
        - There are 34 different brands.
    - Categories :
        - There are 72 different categories.
        - Each leaf category corresponds to a unique full path category.

3. Algorithms

3.1. Proposed single algorithms

- ***cbfDF*** : as I have item description, I can try a ***content-based*** recommendations, with the following features :
    - The <u>price</u> values can be used as they are.
    - <u>Brands</u> can be transformed in One-Hot encoding as there are only 34 different ones.
    - <u>Titles</u> could be represented thanks to TFIDF.
    - <u>Full path categories</u> are more accurate, so it will be preferred to leaf categories. But in this case, all words description would be important, so a TFIDF could be assigned.

    Remark : Maybe item title would be redundant with categories, so in a 2nd test, this feature could be removed in order to analyse whether there are improvements in prediction results.

- Because I have a ratings matrix, I can try :
    - ***userDF : User-user collaborative filtering***, as I have relatively few users compared to the number of products (I have only 14 ratings in average for each user).

    - ***itemDF : Item-item collaborative filtering***, even if I do not have lots of users.
      Products are seasonal, but my recommendations will be for those specific seasonal items.

    - ***mfDF : Matrix factorization*** in order to handle matrix with lower dimensions.

3.2. Proposed hybrids algorithms

The following hybrid algorithms could be used :

- ***Linear combination*** of previous predictions or lists :
    - Of the previous algorithms score.
    - Of the previous algorithms ranks.

- ***Algorithm's combination*** : an algorithm mainly based on Item-Item collaborative filtering, but where similarity is calculated thanks to content-based similarity between items.

- Matrix factorization with ***data hybridization***.

3.3. Additional rules

The following rules must be applied :

- We have to create a specific rule for ***item availability*** : in case where this availability score is null, the corresponding item must not be recommended.

- Not recommend any of the 11 items ***without price***.

- Note that it is possible to recommend items ***already purchased***, as one normally needs to buy such products regularly. But I consider that this rule must only be append to items with price under 20 dollars.

4. Metrics

4.1. Algorithm's comparison

4.1.1. During training

Cross validation must be used in order to correctly evaluate metrics, with the following parameters :

- 5 bulks have been created so that 5 experience will be performed in order to evaluate several times the accuracy of each algorithm :

Bulk0	**Bulk1**	**Bulk2**	**Bulk3**	**Bulk4**	
Train				Test	**Experience 0**
Train (1^{st} part)	Test	Train (2^{nd} part)			**Experience 1**
Train (1^{st} part)		Test	Train (2^{nd} part)		**Experience 2**
Train (1^{st} part)			Test	Train (2^{nd} part)	**Experience 3**
Train				Test	**Experience 4**

- 3 kinds of bulks have been calculated :
    - ***bulks_by_users*** : bulks are created according to users' identifiers :
        - Users' ratings are divided in 5 bulks for each user.
        - The train set is made up of 4 bulks among the 5 for each user, and ratings of all users are concatenated.
        - The test set is made up of the remaining bulk for each user, and ratings of all users are concatenated.

    - ***bulks_rand_lines*** : bulks are created by randomly chosen 1/5 items lines

    - ***bulks _rand_col*** : bulks are created by randomly chosen 1/5 users' columns

- Then error calculation is calculated for each experience :
    - Ratings corresponding of items and user of test bulk are retrieved.
        - Top-5 list of items in current test bulk are kept.
    - Predictions are performed on all those pair.
        - Top-5 list of items in current test bulk predictions are kept.
    - The two Top-5 lists are compared in order to calculate error.

- The 5 experience's errors are averaged in order to allow algorithms comparison.

Remarks :

- *Some bulks own less than 5 ratings for each user (as in average each user has 14 ratings and 1/5 part of those ratings are less than 5).*
  ⇨ *In that case items in Top-5 list of test bulk are the same than in Top-5 list predicted (but maybe ranked in another way).*

- *In real trainings, we should obtain 5 different "CBF" / "Item-Item", "User-User", "MF" and "PersBias" predictions, one for each of the 5 experience performed during cross validation.*

Note that metrics evaluations have also directly been calculated on all available ratings (and no more only on test set).

The following ***offline metrics*** will be used : all those metrics are dedicated to the ***Top-N accuracy*** and they all give more weight to the errors of the top ranks :

- ***MRR : Average reciprocal rank*** : the rank k_u of the 1^{st} recommended item considered relevant by the user is evaluated.

	MRR_by_users	MRR_rand_lines	MRR_rand_col	MRR
cbfDF	0.797267	0.755817	0.556167	0.556167
itemDF	0.785133	0.776490	0.554833	0.554833
userDF	0.797533	0.775928	0.493167	0.493167
mfDF	0.762433	0.713642	0.526500	0.526500

⇨ ***Content-based*** and ***Item-Item collaborative*** filtering are always accurate with this metric.

Note that in case where bulks are created with less lines, User-User collaborative filtering is one of the best algorithms, elsewhere, it is the worst. So, this result is considered as not significant.

- ***MAP : Medium accuracy*** : the number of relevant items among recommended items is successively calculated on all the 5 recommended and relevant items.

	MAP_by_users	MAP_rand_lines	MAP_rand_col	MAP
cbfDF	1.0	1.0	1.0	1.0
itemDF	1.0	1.0	1.0	1.0
userDF	1.0	1.0	1.0	1.0
mfDF	1.0	1.0	1.0	1.0

⇨ This metric gives no information on algorithms, except that Top-5 predictions of test items rated by users, are the same as the Top-5 test items (may ranked in another way).

- ***nDCG*** : importance value given by the user attenuated by the rank in the recommendation.

	nDCG_by_users	MAP_rand_lines	MAP_rand_col	MAP
cbfDF	0.996352	0.995268	0.997548	0.997548
itemDF	0.997133	0.995616	0.999004	0.999004
userDF	0.997584	0.996466	0.998043	0.998043
mfDF	0.993178	0.991986	0.996817	0.996817

⇨ ***All algorithms*** are good according to this metric.

4.1.2. Characterization of algorithms

In that case I used all the predicted ratings provided for each algorithm : the aim is to observe the behaviour of the algorithms, once trained with all are real users' ratings data.

The following ***user centric metrics*** will also be used, in order to be sure that algorithms reach their objectives :

- Recommendation's ***personalization*** : the personalisation has been calculated as follows :
  - For each user, the Top-5 items have been retrieved and transformed into a One-Hot representation : for each user (column) only 5 '1' appears, one for each item of the Top-5 items (lines).
  - The correlation matrix has been calculated.
  - The average of right upper triangular correlation matrix values has been calculated.
  - The personalization is 1 – this previous average.

	Perso
cbfDF	0.774518
itemDF	0.959790
userDF	0.986538
mfDF	0.606046

⇨ Algorithms with the best personalization are ***Item-Item*** and ***User-User*** collaborative filtering.

- Recommendation's ***diversity*** : the diversity as been calculated as follows :
  - A content representation of items as been created with prices and TFIDF representation of FullCat.
  - For each user the cosine similarity has been calculated between all possible pair inside Top-5 items.
  - The diversity is 1 - average of all those previous users' similarities.

	diver
cbfDF	0.899824
itemDF	0.890849
userDF	0.859362
mfDF	0.898591

⇨ ***All algorithms*** have a good diversity.

- Recommendation's ***serendipity*** : the serendipity has been calculated as follows :
  - For each user, the Top-5 items have been retrieved.
  - If the user has rated the item Si, then the relevance is 1 (else 0).
  - Pr(Si) is the predicted score provided in the algorithm sheet.

- Prim(Si) is the popularity of the item, which means ratings' percentage among the users' number.

	serend
cbfDF	0.487183
itemDF	1.106019
userDF	2.576557
mfDF	0.315983

⇨ ***User-User collaborative filtering*** is the best algorithm for this metric.

4.1.3. Conclusion

The following table summarizes algorithms which perform the best for each calculated metrics :

	MRR	MAP	nCDG	Personalization	Diversity	Serendipity	Number of "X"
CBF	X		X	X	X	X	5
Item-Item	X		X	X	X		4
User-User			X		X		2
MF			X		X		2

⇨ The algorithm which performs the best recommendations according to all metrics calculated is ***CBF***.

5. Production

A specific rule must be decided for the ***integration of new data*** : for new items bought by a user, the new user profile could be calculated with 95% of the old one and 5% represented by the new data.