CAZIN Romain
ESCUDIER Cloe
FERHANI Massil
GROLIER Paul
JULIEN Alexis

Rapport SYMA

Présentation des données:

Pour ce projet de prédiction nous disposons de 4 datasets :

- candidate_items.csv, contenant la liste de tous les items avec leur ID.
- item_features.csv, contenant, pour chaque ID d'item, l'ID de la catégorie de leurs features, ainsi que les valeurs de ces features.
- train_purchases.csv, contenant les différentes sessions d'achat, leur heure, et les ID des différents achats effectués.
- train_sessions.csv, contenant pour chaque ID de session, les ID des différents objets regardés ainsi que l'heure associée.

Exploration et nettoyage des datasets:

Premièrement nous nous sommes intéressés aux features de chaque items.

Notre premier travail était de redimensionner le dataset des features pour qu'il soit plus supportable pour nos ordinateurs. Nous avons réalisé ceci à l'aide de TruncatedSVD ce qui nous a permis de ne garder que 12 features.

Par la suite nous avons étudié les sessions. Après les avoir visualisées nous avons merge les sessions avec la matrice des features créée précédemment pour pouvoir afficher pour chaque session, la médiane des features des items observés.

Pareille que précédemment, nous allons gérer la table des achats en la mergeant avec notre matrice des features. Nous rajoutons des labels 1. Pour que l'algorithme apprenne il est important de rajouter des labels 0. Pour cela nous copions la table, mélangeons les colonnes (pour les features du produit ne correspondent plus du

tout au produit), puis les étiquettons d'un label 0. Nous avons ici notre deuxième table.

Premier modèle: Régression logistique

Après avoir préparé nos datasets, il était assez simple de les passer dans un modèle de régression linéaire et nous avons obtenu une mean square error de 0,2. Nous avons ensuite appliqué ce modèle de régression sur notre dataframe de test et avons obtenu les résultats ci-contre.

	item_id	rank	session_id
0	7034	1	3
1	21106	2	3
2	14070	3	3
3	0	4	3
4	18643	5	3
...	...	...	...
95	18515	96	3
96	18513	97	3
97	5676	98	3
98	18291	99	3
99	18250	100	3

Deuxième modèle: RNN

Pour ce modèle nous avons implémenté un modèle long short-term memory. Après le training nous avons obtenu un accuracy de 78,94%.

```
400/400 [==============================] - 1873s 5s/step - loss: 1.3337 - accuracy: 0.6476 - val_loss: 0.7229 -
val_accuracy: 0.7894
31250/31250 [==============================] - 308s 10ms/step - loss: 0.7231 - accuracy: 0.7894
Accuracy: 78.94
```

Pistes d'améliorations:

Nous aurions pu utiliser spark pour pouvoir répartir les jobs sur plusieurs machines car nous avons eu des problèmes de performances et de rapidité.

De même nous aurions pu faire plus de recherches sur nos modèles pour avoir des paramètres optimaux et donc de meilleurs résultats.