TP 2 : Fouille d'itemsets fréquents et de règles d'association sous `Python`

IMT Atlantique – FIL A3

Apprentissage Automatique

Objectifs :
`MLxtend` (pour machine learning extensions) est une bibliothèque logicielle développée par Sebastian Raschka, qui propose des méthodes d'extraction de itemsets fréquents et de règles d'association basée sur Apriori. D'autres algorithmes plus sophistiqués sont aussi proposés.

Ce TP se déroule en deux parties : découverte de `MLxtend` via des exemples [1] simples et application d'algorithmes de fouille de itemsets sur des données de vente au détail en ligne.

1 Partie 1 : `MLxtend` pour la fouille d'itemsets

1.1 Installation

`MLxtend` est une bibliothèque `Python` qui implémente différentes méthodes pour l'apprentissage machine. Parmi ces méthodes, on retrouve les algorithmes d'extraction de itemsets fréquents, de maximaux fréquents et de règles d'association.

L'installation de cette bibliothèque se fait via `canda` [2], avec la commande suivante : `conda install mlxtend --channel conda-forge`

L'utilisation de fonctionnalités de `MLxtend` commence par l'importation de ces trois librairies :

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

1.2 Importation et préparation des données

1.2.1 Données transactionnelle

Considérons le dataset ci-dessous décrivant les caddies de supermarché.

```
dataset = [ ['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
            ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
            ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
            ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
            ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']
          ]
```

1.2.2 Transformation en tableau binaire

La méthode `apriori` de la librairie `MLxtend` prend en entrée un tableau binaire où sont recensés la présence (codé True) ou l'absence (codée par False) des produits dans chaque caddie.

— L'objet `TransactionEncoder` permet de transformer un ensemble de données en un tableau binaire ;

1. Les différents exemples sont tirés du site de `MLxtend`.
2. Pour plus de détails voir `http://rasbt.github.io/mlxtend/installation/`

— Avec la méthode `fit`, le `TransactionEncoder` apprend les étiquettes uniques dans l'ensemble de données, et via la méthode de transformation, il transforme l'ensemble de données d'entrée (une liste de listes) en un tableau de booléens.

```
TB = TransactionEncoder()
TBA = TB.fit(dataset).transform(dataset)
```

Le résultat de cette étape est un tableau `NumPy` au format `ndarray`. Il est possible de transformer ce tableau en un dataframe grâce à `Pandas` :

```
df = pd.DataFrame (TBA, columns=TB.columns_)
```

1.3 Extraction d'itemsets fréquents et des maximaux

Pour extraire les itemsets fréquents il faut appliquer la fonction `apriori()` au dataframe et fixer la valeur du support minimum (dans notre cas, `min_support=0.5`). Il est aussi possible de contraindre la taille des itemsets retournés avec le paramètre `max_len`.

```
FI = apriori(df,min_support=0.5, use_colnames=True)
```

Les résultats sont stockés dans une structure de type "pandas/DataFrame".

```
type(FI)
```

Elle est composée de 2 colonnes : le support et la description des itemsets.

```
print(FI.columns)
```

Pour nos données et avec les paramètres ci-dessus, vous devez obtenir 11 itemsets fréquents suivants.

```
print(FI)
```

support	itemsets
0.8	(Eggs)
1.0	(Kidney Beans)
0.6	(Milk)
0.6	(Onion)
0.6	(Yogurt)
0.8	(Kidney Beans, Eggs)
0.6	(Onion, Eggs)
0.6	(Milk, Kidney Beans)
0.6	(Onion, Kidney Beans)
0.6	(Kidney Beans, Yogurt)
0.6	(Onion, Kidney Beans, Eggs)

On souhaite à présent mesurer le temps d'exécution de Apriori pour extraire les itemsets fréquents. Pour cela, nous allons utiliser la fonction magique ipython `%timit`, qui peut être utilisée pour chronométrer un morceau de code particulier (une seule instruction d'exécution ou une seule méthode).

> **Note**
> Usage, in line mode :
> `%timeit -n<N> -r<R> [-t|-c] -q -p<P> -o]` statement

```
%timeit -n 100 -r 10 apriori(df, min_support=0.5)
```

La librairie `MLxtend` propose d'autres algorithmes sophistiqués pour extraire les itemsets fréquents, comme l'algorithme `fpgrowth`.

> **Question**
> Appliquez l'algorithme `fpgrowth` pour extraire les itemsets fréquents, puis comparez les temps de calculs avec Apriori. Que pouvez-vous conclure ?

Pour extraire les itemsets fréquents maximaux, il faut utiliser la fonction `fpmax`.

```
MFI = fpmax(df,min_support=0.5, use_colnames=True)
print(MFI)
```

support	itemsets
0.6	(Milk, Kidney Beans)
0.6	(Kidney Beans, Yogurt)
0.6	(Onion, Kidney Beans, Eggs)

1.4 Extraction de règles d'association

La fonction `association_rules` de `MLxtend` prend les dataframes des itemsets fréquents produits par les fonctions `apriori()`, `fpgrowth()` ou `fpmax()`.

Pour limiter le nombres de règles extraites, la fonction permet de spécifier (1) la métrique d'intérêt (paramètre `metric`) et (2) le seuil correspondant (paramètre `min_threshold`). Les mesures actuellement mises en œuvre sont la confiance (metric = "confidence") et le lift (metric = "lift").

Le code ci-dessous illustre un exemple d'extraction de règles d'association à partir de itemsets fréquents avec un niveau de confidence d'au moins 80% (`min_threshold=0.8`).

```
AR = association_rules(FI,metric="confidence",min_threshold=0.8)
```

> **Question**
> Affichez pour chaque règle l'antécédent et son le conséquent.

2 Partie 2 : Application à des données de vente en détail

Nous allons maintenant appliquer les différents algorithmes vus précédemment sur des données de vente en détail en ligne issu de la base `retail`. Le fichier "`retail_dataset.csv` " est un fichier au format csv qui se présente sous la forme d'une base transactionnelle ou les transactions représentent des caddies de supermarché : chaque ligne correspond aux noms des produits.

2.1 Évaluation

Vous devez rendre un code python qui tourne sur netbook Jupyter + un mini rapport (format pdf) présentant le travail réalisé, les résultats obtenus et une analyse (intéressante) de ces résultats.

2.2 Travail à faire

1. Charger et transformer les données de façon à ce qu'elles soient reconnues comme des transactions. En pratique on construit un tableau de données binaires.

   Les instructions Python suivantes permettent de charger le jeu de données `retail` :

   ```
   import pandas as pd
   db = pd.read_csv('retail_dataset.csv',sep=',',header=0)
   ```

2. Utiliser l'algorithme Apriori pour extraire les itemsets fréquents et les maximaux. Vous choisirez un support minimum de 3%.
   - Que se passe t'il si on fait varier le seuil du support ?
   - Tracer une courbe montrant l'évolution du nombre de itemsets extraits en fonction du support minimum.

3. Nous souhaitons pouvoir filtrer les itemsets selon la présence d'items ou d'un ensemble d'items. Par exemple, quels sont les itemsets qui contiennent le produit `'Eggs'` ? les produits `{'Eggs','Meat'}` ?

> **Note**
> Plusieurs solutions s'offrent à vous pour la recherche d'itemsets répondant à des conditions de présence d'items. Vous pouvez par exemple utiliser les opérateurs de comparaison de `pandas.Series` (`https://pandas.pydata.org/pandas-docs/stable/reference/series.html`).

4. Utiliser l'algorithme Apriori pour extraire les règles d'association à partir des itemsets fréquents et des itemsets maximaux. Vous choisirez une confiance minimale de 75%. Extraire les règles ayant pour conséquents `'Chesse'`.

> **Note**
> La fonction `association_rules()` renvoie un objet de type pandas.dataframe contenant les différentes règles d'association, chacune décrite par différentes caractéristiques qui sont l'antécédent, le conséquent, et 7 indicateurs numériques d'évaluation des règles. Il faudra adapter l'affichage pour disposer que des informations liées aux mesures support, lift et la confidence.

5. Compléter l'analyse des différentes règles d'association extraites via des graphiques permettant d'étudier la corrélation entre les trois mesures (lift, confiance et support) d'évaluation des règles.

3 Références

1. `MLxtend` : machine learning extensions [3].
2. Introduction to Market Basket Analysis in Python [4].
3. Association Rule Mining via Apriori Algorithm in Python [5].
4. Documentation `Pandas` [6]
5. Documentation `MLxtend` [7]

3. `http://rasbt.github.io/mlxtend/`
4. `https://pbpython.com/market-basket-analysis.html`
5. `https://stackabuse.com/association-rule-mining-via-apriori-algorithm-in-python/`
6. `https://pandas.pydata.org/pandas-docs/stable/reference/index.html#api`
7. `http://rasbt.github.io/mlxtend`