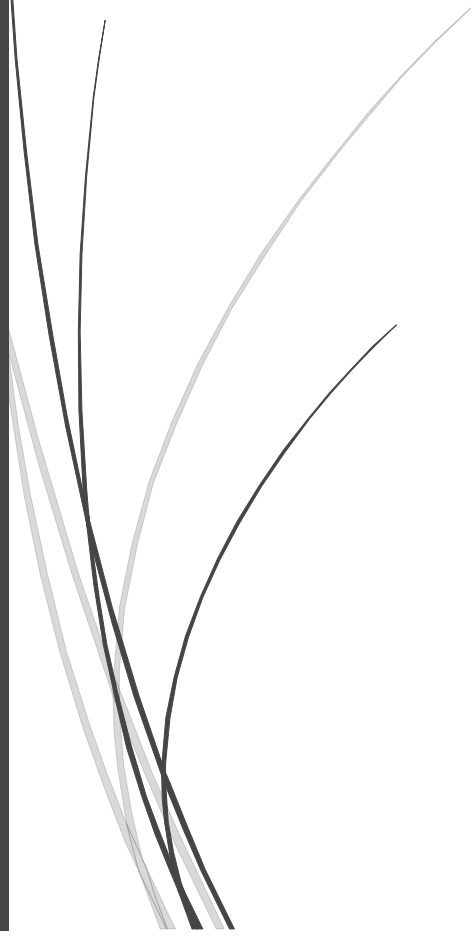
Semestre 3

Modélisation

Apprentissage machine et recommandation



Maxime Miquel - Ascencio Masao
IUT INFORMATIQUE

Table des matières

I – Contexte

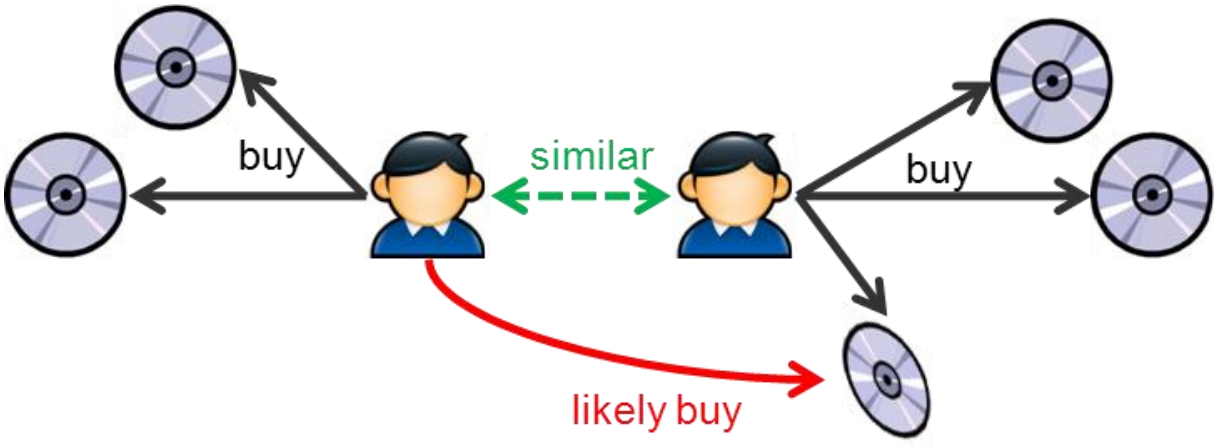
Nous avons choisi le sujet sur l'apprentissage machine et les systèmes de recommandation. Curieux de comprendre les méthodes de Netflix et Amazon, nous nous sommes lancés dans un premier temps dans une recherche plutôt générale sur le sujet.

Il s'agit de recommander un item à un utilisateur par des algorithmes selon les goûts de l'utilisateur. Nous possédons donc deux jeux de données comportant 1000 items pour 100 utilisateurs, l'un étant complet et l'autre incomplet. Notre objectif est donc de trouver les notes du tableau incomplet qui ne sont pas renseignées à l'aide des données renseignées.

A – Compréhension du Sujet

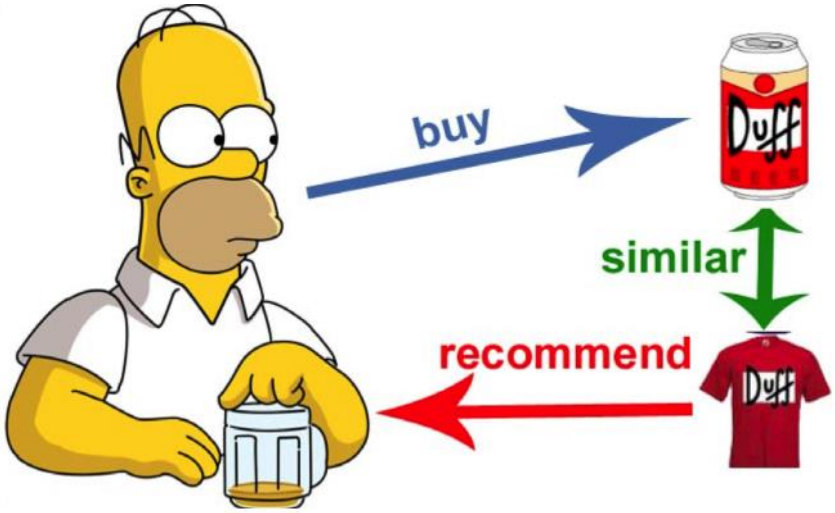
Tout d'abord afin de comprendre les systèmes de recommandation et le filtrage collaboratif, nous avons utilisé les ressources fournies par le sujet et celles que nous avons trouvé par nous-même.

Voici une image illustrant un filtrage basé sur les utilisateurs :



Si un utilisateur B similaire à A met une bonne note à un item. Le programme de recommandation propose alors cet item à A.

Voici une image illustrant un filtrage basé sur l'item :



Si un utilisateur met une bonne note à un item A qui est similaire à B, le programme de recommandation proposera B à l'utilisateur.

B – Comment procéder ?

Pour commencer, il faut utiliser une formule déterminant la similarité entre les utilisateurs ou les items afin de pouvoir prédire une note qui servira à la recommandation. Ensuite, il faut implémenter les filtres collaboratifs et prédire les notes qu'aurait attribué les utilisateurs et remplir les cases manquantes du tableau incomplet.
De plus, un paramètre rentre en compte, et il s'agit de la sévérité. En effet, un utilisateur peut être plus ou moins sévère, et des utilisateurs ayant des goûts similaires peuvent avoir des sévérités différentes. Pour finir, il faut comparer les performances entre les différentes méthodes utilisés que nous détaillerons ensuite.

II – Manipulation des données

A – Utilisation des Dataframes

Afin de manipuler les Dataframes nous avons dû importer les bibliothèques *pandas* et *numpy* de Python. *Pandas* permet l'utilisation de la classe *pandas.DataFrame* qui permet de manipuler un grands nombres de données facilement. Nous avons donc créé deux Dataframes pour les fichiers .csv *toy_complet* et *toy_incomplet*. Mais nous avons rencontré un problème d'implémentation du au séparateur que nous avons dû changer de ',' à ' \t '. De plus, la première colonne était remplie de valeurs non reconnues (NaN : Not a Number) nous avons donc décidé de supprimer cette colonne.

Pour commencer nous ne nous sommes pas servis du fichier *toy_complet*, car l'intérêt ici est de prédire les notes non renseignées dans le fichier *toy_incomplet* en utilisant seulement les notes présentes dans celui-ci.

B – Parcourir les données

Pour parcourir les données des Dataframes, afin de calculer une note moyenne d'un utilisateur par exemple, nous avons réutilisé les mêmes techniques durant tout le projet. Lorsque nous devions calculer une somme, nous avons systématiquement utilisé une boucle for. Voici un exemple du calcul d'une moyenne des notes attribués par l'utilisateur i :

```
for j in range(DataFrame.shape[1]):
        moyennei = moyennei + DataFrame.iloc[i,j]
        x = x + 1
   moyennei = moyennei / x
```

Ici, *DataFrame.shape[1]* vaut 1000, cette fonction de la bibliothèque *pandas* renvoie le nombre de colonnes du Dataframe lorsque la valeur entre crochets vaut 1, et le nombre de lignes lorsque cette valeur vaut 0. *DataFrame.iloc[i,j]* renvoie quant à lui la valeur présente à l'endroit du Dataframe où la ligne vaut i et la colonne vaut j.

Cependant, il subsiste un problème car nous ne devons pas prendre en compte les valeurs manquantes du Dataframe (modélisées par des -1). Nous avons donc dû ajouter une condition avant de faire notre somme "si la valeur de la cellule ne vaut pas -1 alors faire l'addition". En python cela donne :

```
if(DataFrame.iloc[i,j] != -1) :
        …
```

III – Application

A – Similarités

Le calcul des similarités entre les utilisateurs ou les items sont une partie essentielle à l'estimation des notes que nous allons attribuer au tableau incomplet.
Il existe ainsi deux formules.

La similarité Pearson :

$$Pearson\ Correlation : Sim(u_i, u_k) = \frac{\sum_j (r_{ij} - r_i)(r_{kj} - r_k)}{\sqrt{\sum_j (r_{ij} - r_i)^2 \sum_j (r_{kj} - r_k)^2}}$$

La similarité Cosinus :

$$Cosine\ Similarity : Sim(u_i, u_k) = \frac{r_i \cdot r_k}{|r_i||r_k|} = \frac{\sum_{j=1}^{m} r_{ij} r_{kj}}{\sqrt{\sum_{j=1}^{m} r_{ij}^2 \sum_{j=1}^{m} r_{kj}^2}}$$

Nous avons d'abord calculé les 10 premières similarités entre utilisateurs afin de les comparer aux valeurs présentes sur Moodle.

Les 10 premières similarités Cosinus :
[0.72986 ; 0.73075 ; 0.72325 ; 0.69877 ; 0.96328 ; 0.70884 ; 0.72731 ; 0.70965 ; 0.69541 ; 0.70710]

Les 10 premières similarités Pearson :
[-0.021661 ; 0.015390 ; -0.0031177 ; -0.036256 ; 0.87040 ; -0.017336 ; 0.042396 ; -0.041776 ; -0.040732 ; -0.050547]

Une fois les valeurs validées, nous avons décidé de mettre toutes les similarités dans des fichiers .csv afin de les conserver et de ne plus avoir à effectuer les calculs qui prennent du temps.

B – Agrégation (filtrage basé utilisateur)

Afin de procéder à l'agrégation, on détermine un ensemble d'utilisateurs ayant notés un objet puis on procède à celle-ci en utilisant la similarité (Pearson ou Cosinus) afin de déterminer le poids de chaque utilisateur. Ici, le poids représente la similarité entre deux utilisateurs, poids (i, k) = sim(i, k).

Formule de Prédiction :

$$N_{predite}(i,j) = \frac{\sum_{k \in K} poids(i,k) n(k,j)}{\sum_{k \in K} poids(i,k)}.$$

Fonction de prédiction dans Python :

```
def prediction(u1, i1) :
    num = 0
    den = 0
    for i in range(DataFrame.shape[0]) :
        if DataFrame.iloc[i, i1] != -1 :
            num = num + DFCOS.iloc[u1, i] * (DataFrame.iloc[i, i1])
            den = den + DFCOS.iloc[u1, i]
    return (num/den)
```

Ici DFCOS correspond au tableau contenant les similarités cosinus, nous avons ensuite effectué la même manipulation pour le tableau contenant les similarités Pearson.

Un autre facteur aurait également pu être pris en compte lors de la prédiction, c'est la sévérité de l'utilisateur. Nous avons réussi à coder celle-ci dans la fonction Prédiction mais elle prenait trop de temps à calculer, nous avons donc décidé de l'abandonner.

Maintenant que nous avons la fonction de prédiction pour un utilisateur et un item donné. Nous avons copié le Dataframe *toy_incomplet* dans *CopieIncomplet*, pour ensuite y remplacer les valeurs manquantes par nos prédictions.

```
CopieIncomplet = DataFrame.copy()
for i in range(DataFrame.shape[0]):
    for j in range(DataFrame.shape[1]):
        if DataFrame.iloc[i,j] == -1:
            CopieIncomplet.iloc[i, j] = prediction(i, j)
```

IV – Résultats

Après avoir remplacé les valeurs manquantes dans le Dataframe *toy_incomplet*, il est maintenant temps de comparer nos résultats avec le Dataframe *toy_complet*.
Pour cela nous allons utiliser deux indicateurs très efficaces pour valider ou non nos prédictions. Le biais et l'erreur moyenne. Voici leur formule :

Le biais :

$$\frac{1}{Q}\sum N_{predite}(i,j) - n_{vraie}(i,j)$$

L'erreur moyenne :

$$\frac{1}{Q}\sum \left| N_{predite}(i,j) - n_{vraie}(i,j) \right|$$

Le biais permet de rendre compte de la précision des résultats et de déterminer si nos prédictions ont tendances à surévaluer ou à sous-évaluer les notes attribuées. S'il est négatif, on a tendance à sous-évaluer, s'il est positif on a tendance à surévaluer. Plus

il est proche de 0, plus nos prédictions sont bonnes. Cependant il réside un inconvénient à cet indicateur, en effet si nous surévaluons et sous évaluons à la fois, la valeur sera proche de 0 mais notre prédiction n'en reste pas moins imprécise.

C'est pour cela que l'erreur moyenne est un indicateur qui est bon à prendre en compte en parallèle du biais. L'erreur moyenne nous montre "en moyenne" de combien nous nous trompons sur les prédictions sans nous indiquer la sur ou sous-évaluation.
Dans notre tableau *toy_incomplet* il y'a 50 330 valeurs à prédire, ce seras donc notre valeur Q. Nous effectuons alors le calcul pour nos prédictions avec la similarité Pearson et avec la similarité Cosinus. En python nous procédons comme à notre habitude par une boucle for pour les sommes. Nos valeurs auront deux chiffres significatifs pour améliorer leur lisibilité.
Commentons d'abord les résultats des biais :

$$biais_{cosinus} = -0{,}0093$$

$$biais_{pearson} = -0{,}0042$$

On remarque que les deux valeurs sont négatives, on a donc tendance à sous-évaluer nos notes. Cependant elles sont proches de 0, ce qui laisse à croire que cela reste de bonnes prédictions. Pearson reste quand même un petit peux plus performant que cosinus. Voyons maintenant l'erreur moyenne :

$$erreur_{cosinus} = 1{,}1$$

$$erreur_{pearson} = 0{,}53$$

On se trompe en moyenne de 1,1 sur les notes avec la similarité cosinus, et de 0,53 avec la similarité Pearson. L'erreur moyenne est plus proche de 0 pour la similarité Pearson que pour la similarité Cosinus. On peut donc en conclure que, sur un filtrage basé sur l'utilisateur et en agrégant les notes sans correction de la sévérité, la similarité Pearson est plus performante. Les deux prédictions restent néanmoins très performantes.

Sources et documentations

Collaboration - Wikipedia : https://en.wikipedia.org/wiki/Collaboration

Introduction to Recommender System | by Shuyu Luo | Towards Data Science : https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26

Comment valider un modèle de prédiction ? – Aspexit : https://www.aspexit.com/comment-valider-un-modele-de-prediction/

Python Recommender Systems: Content Based & Collaborative Filtering Recommendation Engines - DataCamp : https://www.datacamp.com/community/tutorials/recommender-systems-python

pandas - Python Data Analysis Library (pydata.org) : https://pandas.pydata.org/

Recommendation Engine Models - DZone Java : https://dzone.com/articles/recommendation-engine-models

Build Recommendation Engine Using Graph | by Handaru Sakti | tiket-dev | Medium : https://medium.com/tiket-com-dev-team/build-recommendation-engine-using-graph-cbd6d8732e46