TP3_Corrige

March 9, 2025

##

Polytechnique Montréal Département Génie Informatique et Génie Logiciel INF8008 – Prétraitement de données . TP3 - Échantillonnage et imputations Hiver 2025 . 17 février 2025

0.1 Introduction

Le TP3 porte principalement sur l'échantillonnage et l'imputation, mais nous aborderons tout de même l'agrégation, la tabulation, le remodelage, le pivotement et les statistiques descriptives. Nous survolons l'utilisation de fonctions de base de Pandas et de l'analyse de données numériques.

Dans ce travail, vous aurez à utiliser les données des deux fichiers .csv suivants :

- appointments.csv : Matrice de données de 100 000 rendez-vous faits par 943 patients et portant sur 1682 docteurs;
- patients.csv : Matrice de données sur les patients.

Ces données ont été générées synthétiquement par un grand modèle de langage pour approximer les interactions réelles dans le domaine des soins de santé tout en garantissant la confidentialité des données.

Voici les librairies python qui seront à utiliser pour ce TP : - matplotlib - numpy - pandas

[ ]:
```
import pandas as pd
import numpy as np
```

[ ]:
```
df_appointments = pd.read_csv('appointments.csv')
df_patients = pd.read_csv('patients.csv')
```

[ ]:
```
df_appointments
```

[ ]:

	patient.id	doctor.id	niveau.recommandation	date
0	938	588	4	2021-01-15
1	81	1083	4	2021-11-26
2	890	1144	2	2021-05-05
3	428	18	5	2021-06-09
4	483	1130	5	2021-01-26
…	…	…	…	…
96943	351	435	4	2021-01-12
96944	187	1246	3	2021-01-05
96945	748	560	5	2021-06-16

```
96946          285        1429                      3  2021-05-14
96947          817         923                      3  2021-10-15

[96948 rows x 4 columns]
```

0.2 Prédiction de recommendations par factorisation de matrice

Mise en contexte :

Un principe fondamental des systèmes de recommandations est de prédire les rendez-vous des patients aux docteurs qui n'ont pas encore de rendez-vous et sont présumés des candidats à recommander. On recommande alors les docteurs dont les rendez-vous estimés sont les plus élevés. Une méthode très facile à implémenter consiste à factoriser une matrice de rendez-vous dans un espace à dimensions réduites puis à effectuer le produit des matrices factorisées. Ce produit constitue la prédiction des rendez-vous sur la base de réduction de dimensions.

Le fichier appointments.csv contient 100 000 rendez-vous de 943 patients pour 1682 docteurs. Il faut, dans un premier temps, créer cette matrice et ensuite la factoriser en un produit de matrices dont le rang est plus petit. Mais la matrice originale contient un grand nombre de valeurs manquantes et il est nécessaire d'effectuer une imputation avant la factorisation par des méthodes analytiques, notamment la méthode de décomposition en valeurs singulières que nous utiliserons (SVD).

De plus, il faut aussi effectuer une validation croisée afin de déterminer le bon nombre dimensions pour la factorisation ainsi que s'il est préférable de faire une imputation de valeurs aléatoires, de la moyenne des lignes, ou de la moyenne des lignes et des colonnes.

0.2.1 A)

Mettez les données de "df_appointments" sous la forme d'une matrice de rendez-vous patients × docteurs. (1.5 points)

Pour ce faire: 1. Retirez la colonne "date" du DataFrame "df_appointments". 2. Triez les données par "patient.id" et "doctor.id". 3. Regroupez les données par "patient.id" et faites l'agrégation des "niveau.recommandation" pour chaque "doctor.id". 4. Convertissez en matrice.

Affichez les dimensions de la matrice, celle-ci devrait être de "shape" (943, 1682).

[ ]:
```
#TODO:

Rdf = df_appointments.drop(columns=['date'])
Rdf = Rdf.sort_values(by=['patient.id', 'doctor.id'])
Rdf = Rdf.groupby('patient.id').apply(lambda x: pd.Series(x['niveau.
 ↪recommandation'].values, index=x['doctor.id'])).unstack()

R = Rdf.to_numpy()

R.shape
```

```
C:\Users\abdoj\AppData\Local\Temp\ipykernel_2324\1652898773.py:5:
DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns.
This behavior is deprecated, and in a future version of pandas the grouping
```

```
columns will be excluded from the operation. Either pass `include_groups=False`
to exclude the groupings or explicitly select the grouping columns after groupby
to silence this warning.
  Rdf = Rdf.groupby('patient.id').apply(lambda x:
pd.Series(x['niveau.recommandation'].values, index=x['doctor.id'])).unstack()
```

[ ]:
```
(943, 1682)
```

[ ]:
```
R
```

[ ]:
```
array([[nan,  5., nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan, nan],
       [ 1., nan,  2., …, nan, nan, nan],
       …,
       [nan,  2., nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan,  5.],
       [nan, nan,  4., …, nan, nan, nan]])
```

0.2.2 B)

Pour factoriser une matrice, elle ne doit contenir aucune valeur manquante. Imputez (remplacez) les valeurs manquantes (valeurs non observées) dans la matrice créée précédemment selon les trois méthodes suivantes : (2.5 points)
1. Remplacer les “nan” par des valeurs aléatoires d’une distribution normale (0,1). 2. Remplacer les “nan” par la moyenne des lignes. 3. Remplacer les “nan” par la moyenne des lignes et colonnes.

Affichez la matrice obtenue pour chaque méthode.

[ ]:
```
nan_mask = np.isnan(R)
valeursObservees = ~np.isnan(R)

#TODO:

# Methode 1: Remplacer les "nan" par des valeurs aléatoires d'une distribution␣
 ↪normale (0,1)
random_values = np.random.randn(*R.shape)
R_filled_random = np.where(np.isnan(R), random_values, R)

print("Méthode 1: \n", R_filled_random)
print()

# Methode 2: Remplacer les "nan" par la moyenne des lignes
row_means = np.nanmean(R, axis=1, keepdims=True)
R_filled_row_mean = np.where(np.isnan(R), row_means, R)

print("Méthode 2: \n", R_filled_row_mean)
print()

# Method 3: Remplacer les "nan" par la moyenne des lignes et colonnes
```

```
row_means = np.nanmean(R, axis=1, keepdims=True)
col_means = np.nanmean(R, axis=0, keepdims=True)
mean_of_means = (row_means + col_means) / 2
R_filled_row_col_mean = np.where(np.isnan(R), mean_of_means, R)

print("Méthode 3: \n", R_filled_row_col_mean)
print()
```

```
Méthode 1:
 [[ 4.89443867e-03  5.00000000e+00 -1.64486478e+00 … -1.01491477e+00
  -2.89035348e-01 -4.79805672e-01]
 [-7.06530255e-01  1.24296921e+00 -9.88944537e-01 …  2.73580500e-01
   3.35941078e-01 -1.37274445e+00]
 [ 1.00000000e+00  1.92367575e-01  2.00000000e+00 … -2.16803563e-01
  -1.49362956e+00 -5.15637952e-01]
 …
 [ 1.02115536e-01  2.00000000e+00 -3.30888095e-02 …  2.82556362e-01
   1.02172107e+00 -4.68805362e-01]
 [ 1.48838930e-02  5.64425674e-01 -2.32836044e-01 …  1.26789424e+00
  -1.72121732e+00  5.00000000e+00]
 [ 1.20634562e+00 -6.90520174e-01  4.00000000e+00 …  1.14792472e+00
   8.57904050e-01 -7.86265151e-01]]

Méthode 2:
 [[3.35185185 5.         3.35185185 … 3.35185185 3.35185185 3.35185185]
 [2.78431373 2.78431373 2.78431373 … 2.78431373 2.78431373 2.78431373]
 [1.         2.70873786 2.         … 2.70873786 2.70873786 2.70873786]
 …
 [2.95614035 2.         2.95614035 … 2.95614035 2.95614035 2.95614035]
 [3.18691589 3.18691589 3.18691589 … 3.18691589 3.18691589 5.        ]
 [3.07526882 3.07526882 4.         … 3.07526882 3.07526882 3.07526882]]

Méthode 3:
 [[3.33956229 5.         3.02592593 … 3.21829881 3.11823362 3.05328442]
 [3.05579323 2.68215686 2.74215686 … 2.93452974 2.83446456 2.76951535]
 [1.         2.64436893 2.         … 2.89674181 2.79667662 2.73172742]
 …
 [3.14170654 2.         2.82807018 … 3.02044306 2.92037787 2.85542867]
 [3.25709431 2.88345794 2.94345794 … 3.13583083 3.03576564 5.        ]
 [3.20127077 2.82763441 4.         … 3.08000729 2.9799421  2.9149929 ]]
```

0.2.3 C)

Mise en contexte:

Il faut maintenant effectuer une validation croisée à 5 replis en échantillonnant aléatoirement 5 ensembles de rendez-vous parmi les 100 000 rendez-vous observés qui serviront tour à tour d'ensemble de test, les autres étant utilisées pour l'entraînement (estimation des matrices de factorisation).

Pour ce faire, utilisez la fonction validation_croisee_replis() qui effectue une validation croisée par replis de la factorisation SVD et retourne l'erreur quadratique moyenne. Elle utilise une matrice, echantillonReplis, dont chaque ligne contient l'indice des observations de test pour chaque "replis" et qui est créée plus loin.

Utilisez la variable valeurs_observees (obtenue à l'aide de la matrice créer dans la partie "A" ci-dessus) et la fonction random.choice() de numpy.

Affichez les dimensions de l'échantillon, vous devriez avoir (5, 19389). (1 point)

[ ]:
```
from validation_croisee import validation_croisee_replis, validation_croisee
```

[ ]:
```
n_dimensions = 14
nombre_replis = 5
valeurs_observees = ~np.isnan(R)

#TODO:

# Get the indices of observed values
observed_indices = np.where(valeurs_observees)

# Determine the sample size for each fold
sample_size = min(np.sum(valeurs_observees), np.sum(valeurs_observees) //␣
 ↪nombre_replis)

# Sample indices for each fold
echantillon_replis = np.random.choice(np.arange(np.sum(valeurs_observees)),␣
 ↪size=(nombre_replis, sample_size), replace=False)

# Output the dimensions of echantillonReplis
print("Dimensions of echantillonReplis:", echantillon_replis.shape)
```

```
Dimensions of echantillonReplis: (5, 19389)
```

0.2.4 D)

Pour chacune des 3 méthodes d'imputation utilisées précédemment, affichez le résultat de la validation croisée. (0.5 point)

Les résultats qui seront affichés pour chaque méthode devraient être similaire à ceux-ci: 1. Méthode 1 : array([3.07936543 2.97096366 3.00239858 2.98364498 3.07552508]).

2. Méthode 2 : array([1.35600187 1.36579327 1.37056779 1.35480193 1.3561739]).
3. Méthode 3 : array([1.34846446 1.36657222 1.3627661 1.3456932 1.34794998]).

[ ]:
```
#TODO: Méthode 1

method1_results = validation_croisee_replis(R, R_filled_random,␣
 ↪echantillon_replis, valeurs_observees, n_dimensions)
```

[ ]:
```
#TODO: Méthode 2

method2_results = validation_croisee_replis(R, R_filled_row_mean,␣
 ↪echantillon_replis, valeurs_observees, n_dimensions)
```

[ ]:
```
#TODO: Méthode 3

method3_results = validation_croisee_replis(R, R_filled_row_col_mean,␣
 ↪echantillon_replis, valeurs_observees, n_dimensions)
```

[ ]:
```
print ("Méthode 1 : ")
print(method1_results)
print ("Méthode 2 : ")
print(method2_results)
print ("Méthode 3 : ")
print(method3_results)
```

```
Méthode 1 :
[3.07936543 2.97096366 3.00239858 2.98364498 3.07552508]
Méthode 2 :
[1.35600187 1.36579327 1.37056779 1.35480193 1.3561739 ]
Méthode 3 :
[1.34846446 1.36657222 1.3627661  1.3456932  1.34794998]
```

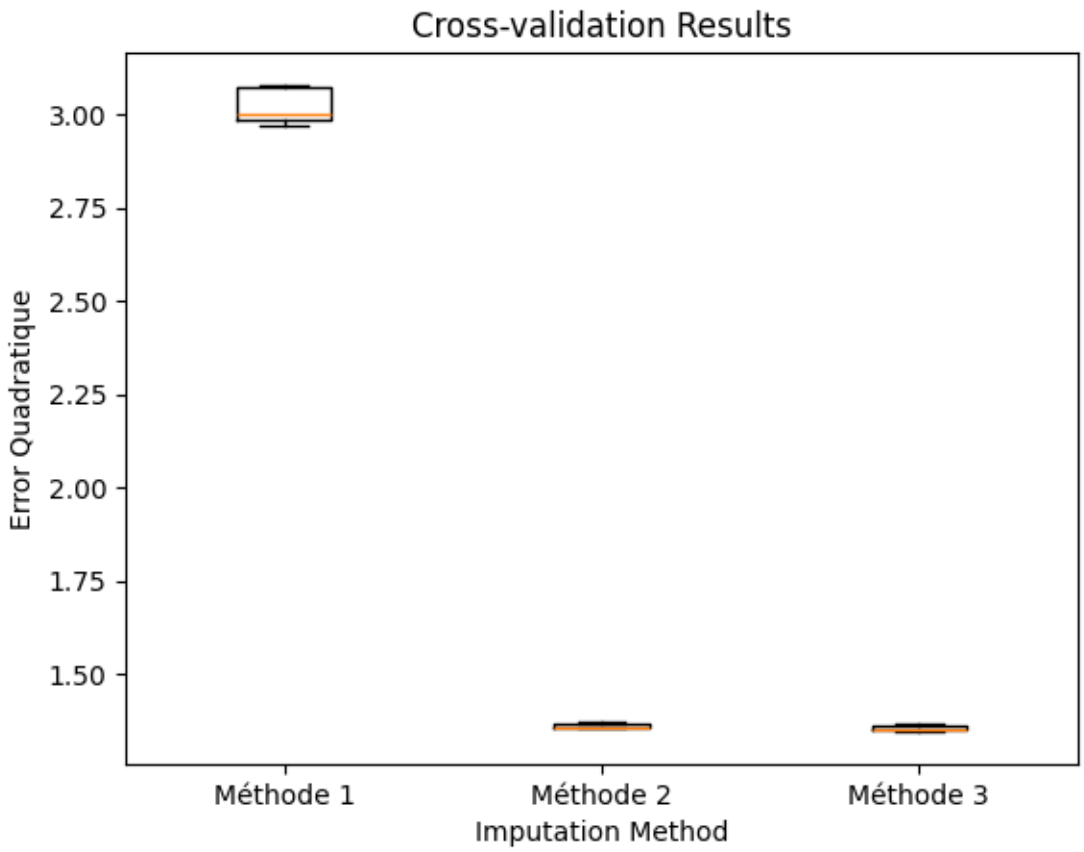
Affichez les résultats de la validation croisée pour chaque méthode (Méthode 1, Méthode 2, Méthode 3) sur un seul graphique en boîte à moustaches (boxplot). (2 points)

[ ]:
```
# TODO:
import matplotlib.pyplot as plt

# Combine results into a single list
all_results = [method1_results, method2_results, method3_results]

# Plotting the results
plt.boxplot(all_results, labels=["Méthode 1", "Méthode 2", "Méthode 3"])
plt.title("Cross-validation Results")
plt.xlabel("Imputation Method")
plt.ylabel("Error Quadratique")
plt.show()
```

```
C:\Users\abdoj\AppData\Local\Temp\ipykernel_2324\107265810.py:8:
MatplotlibDeprecationWarning: The 'labels' parameter of boxplot() has been
renamed 'tick_labels' since Matplotlib 3.9; support for the old name will be
dropped in 3.11.
  plt.boxplot(all_results, labels=["Méthode 1", "Méthode 2", "Méthode 3"])
```



0.2.5 E)

Explorez le nombre de dimensions qui est optimal par visualisation de la performance prédictive en fonction du nombre de dimensions. Prenez la moyenne par ligne et colonne comme imputation et cet ensemble de dimensions à explorer : (2, 4, 8, 10, 15, 20, 40). N'utilisez qu'un échantillon de données. (1 point)

Pour ce faire: 1. Calculez l'erreur quadratique pour chaque dimension. 2. Affichez dans un graphique l'erreur quadratique selon le nombre de dimensions.

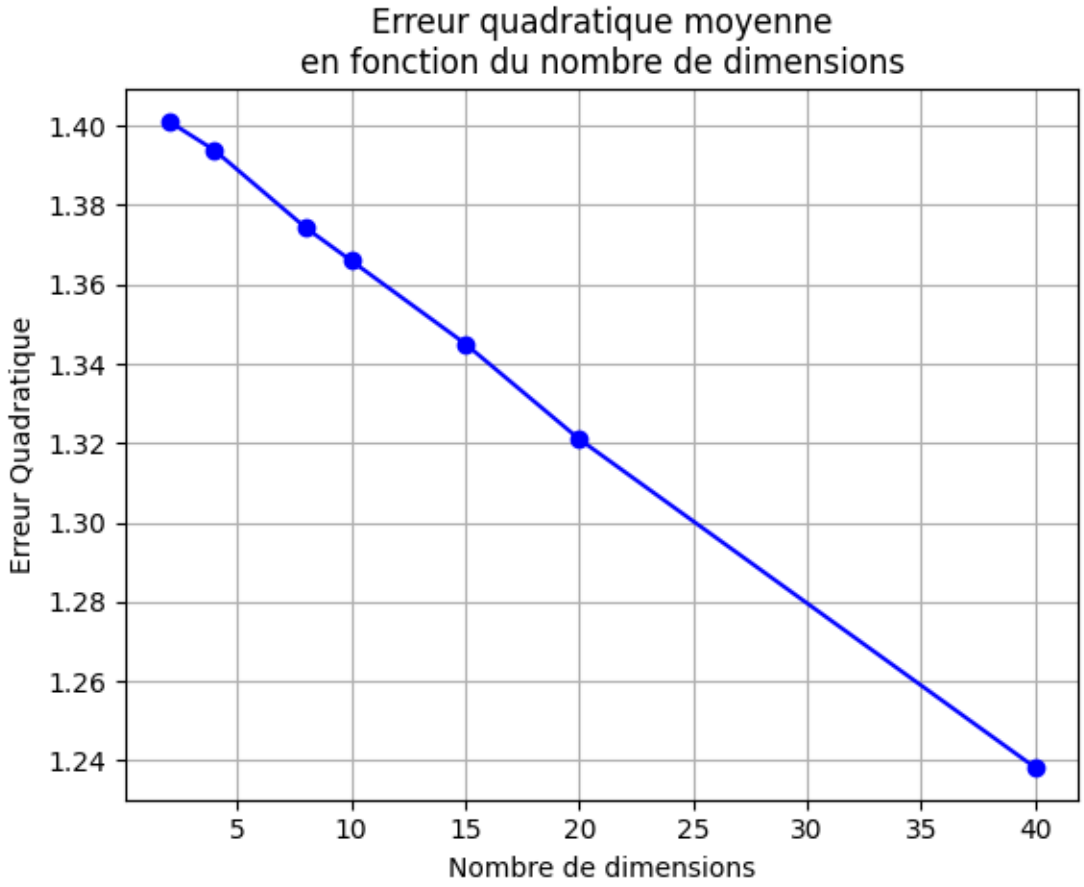
Note: Le graphique doit utiliser des marqueurs de type 'o' et un style de ligne '-'. N'oubliez pas de nommer les axes et de donner un titre.

[ ]:
```
import matplotlib.pyplot as plt
```

[ ]:
```
#TODO:

dimensions = [2, 4, 8, 10, 15, 20, 40]
err_quad = [validation_croisee(R, R_filled_row_col_mean, echantillon_replis[0],␣
 ↪valeurs_observees, ndim) for ndim in dimensions]
```

```
plt.plot(dimensions, err_quad, marker='o', linestyle='-', color='b')
plt.title("Erreur quadratique moyenne\nen fonction du nombre de dimensions")
plt.xlabel("Nombre de dimensions")
plt.ylabel("Erreur Quadratique")
plt.grid(True)
plt.show()
```



0.3 Probabilités bayésiennes

Mise en contexte :

La seconde approche de prédiction est basée sur les probabilités conditionnelles avec des ratios de vraisemblance. Ces calculs nécessitent des tableaux de contingences entre deux variables, la variable à prédire et le facteur de prédiction. Nous voudrons prédire si un individu est susceptible de recommender un docteur en fonction de trois variables, son sexe, son âge et sa condition de santé. Il nous faut donc créer les tableaux suivants.

0.3.1 F)

Créez un tableau des données de rendez-vous et des patients.

Pour ce faire: 1. Renommez la colonne “patients.id” du DataFrame appointments par “id”. 2. Faites un left-join de nouveau DataFrame créé avec le DataFrame des patients. 3. Enlevez les colonnes “date” et ” zip”. 4. Ajoutez une nouvelle colonne “recommande” qui est à True si le “niveau.recommandation” est supérieur à 4.

Affichez le nouveau tableau. (1 point)

[ ]:
```
# TODO:
# Attention aux espaces dans les noms de colonnes

# Renommez la colonne "patients.id" du DataFrame appointments par "id "
u_data_renamed = df_appointments.rename(columns={'patient.id': 'id'})

# Faites un left-join de nouveau DataFrame créé avec le DataFrame des patients
votes = pd.merge(u_data_renamed, df_patients, on='id', how='left')

# Enlevez les colonnes "date" et " zip"
votes = votes.drop(columns=['date', ' zip'])

# Ajoutez une nouvelle colonne "recommande" qui est à True si le "niveau.
↪recommandation" est supérieur à 4
votes['recommande'] = votes['niveau.recommandation'] > 4
votes
```

[ ]:
```
         id  doctor.id  niveau.recommandation  age gender     condition  \
0       938        588                      4   38      M        Asthma
1        81       1083                      4   70      M  Hypertension
2       890       1144                      2   62      F        Asthma
3       428         18                      5   56      F  Hypertension
4       483       1130                      5   27      M       Healthy
…     …         …                       …  …     …           …
96943  351        435                      4   52      F        Asthma
96944  187       1246                      3   71      F       Healthy
96945  748        560                      5   49      M       Healthy
96946  285       1429                      3   41      F        Asthma
96947  817        923                      3   27      F       Healthy

       recommande
0           False
1           False
2           False
3            True
4            True
…           …
96943       False
96944       False
96945        True
96946       False
```

```
96947       False

[96948 rows x 7 columns]
```

0.3.2 G)

Créez un nouveau tableau “recommande_gender” qui présente la proportion de patients qui recommande ou qui ne recommande pas un docteur.

Pour ce faire : 1. À partir du DataFrame “appointments”, regroupez les données par “doctor.id” et par “gender”. 2. Appliquez la fonction “recommande_tbl” sur la colonne “recommande” et réinitialisez l’index. 3. Créez un “pivot table” dont l’index est “doctor.id”, la colonne est ” gender “, les valeurs sont”recommande” et “recommandePas”, puis utilisez la sommation comme fonction d’agrégation. 4. Créez une colonne “LS” à l’aide de la fonction “ratio_vraisemblance” qui contient le résultat du ratio de vraisemblance. Voici les paramètres à donner à cette fonction : - EH: recommande_gender[(‘recommande’, ‘F’)] - EnH: recommande_gender[(‘recommandePas’, ‘F’)] - nEH: recommande_gender[(‘recommande’, ‘M’)] - nEnH: recommande_gender[(‘recommandePas’, ‘M’)]

Note: La valeur LS de votre 1ère ligne devrait être environ 1.147059.

Affichez le nouveau DataFrame. (1 point)

[ ]:
```python
def recommande_tbl(x):
    return pd.DataFrame({'recommande': [sum(x)], 'recommandePas': [sum(1-x)]})

def ratio_vraisemblance(EH, EnH, nEH, nEnH):
    return ((EH+1)/(EH+nEH+2))/((EnH+1)/(EnH+nEnH+2))
```

[ ]:
```python
# TODO:
# Solution des points 1, 2 et 3
recommande_gender = votes.groupby(['doctor.id', 'gender'])['recommande'] \
                    .apply(recommande_tbl) \
                    .reset_index() \
                    .pivot_table(index='doctor.id', columns='gender',␣
 ↪values=['recommande', 'recommandePas'], aggfunc='sum')

# Créez une colonne "LS" à l'aide de la fonction "ratio_vraisemblance" qui␣
 ↪contient le résultat du ratio de vraisemblance.
recommande_gender['LS'] = ratio_vraisemblance(recommande_gender[('recommande',␣
 ↪'F')], recommande_gender[('recommandePas', 'F')],␣
 ↪recommande_gender[('recommande', 'M')], recommande_gender[('recommandePas',␣
 ↪'M')])

# Reset index to get a flat DataFrame
recommande_gender = recommande_gender.reset_index()
recommande_gender
```

[ ]:

	doctor.id	recommande		recommandePas		LS
gender		F	M	F	M	
0	1	9	9	16	21	1.147059
1	2	5	4	16	25	1.379679
2	3	5	1	21	23	1.568182
3	4	10	8	30	26	1.029032
4	5	3	10	25	19	0.471795
…	…	…	..	…	..	…
1677	1678	4	1	18	22	1.578947
1678	1679	8	12	16	26	1.058824
1679	1680	10	5	23	21	1.240196
1680	1681	4	3	26	19	0.967078
1681	1682	4	4	20	25	1.119048

[1682 rows x 6 columns]

0.3.3 H)

Affichez dans un graphique à barres la valeur LS pour les dix premiers “doctor.id” en utilisant le tableau “recommande_gender”. (0.5 point)

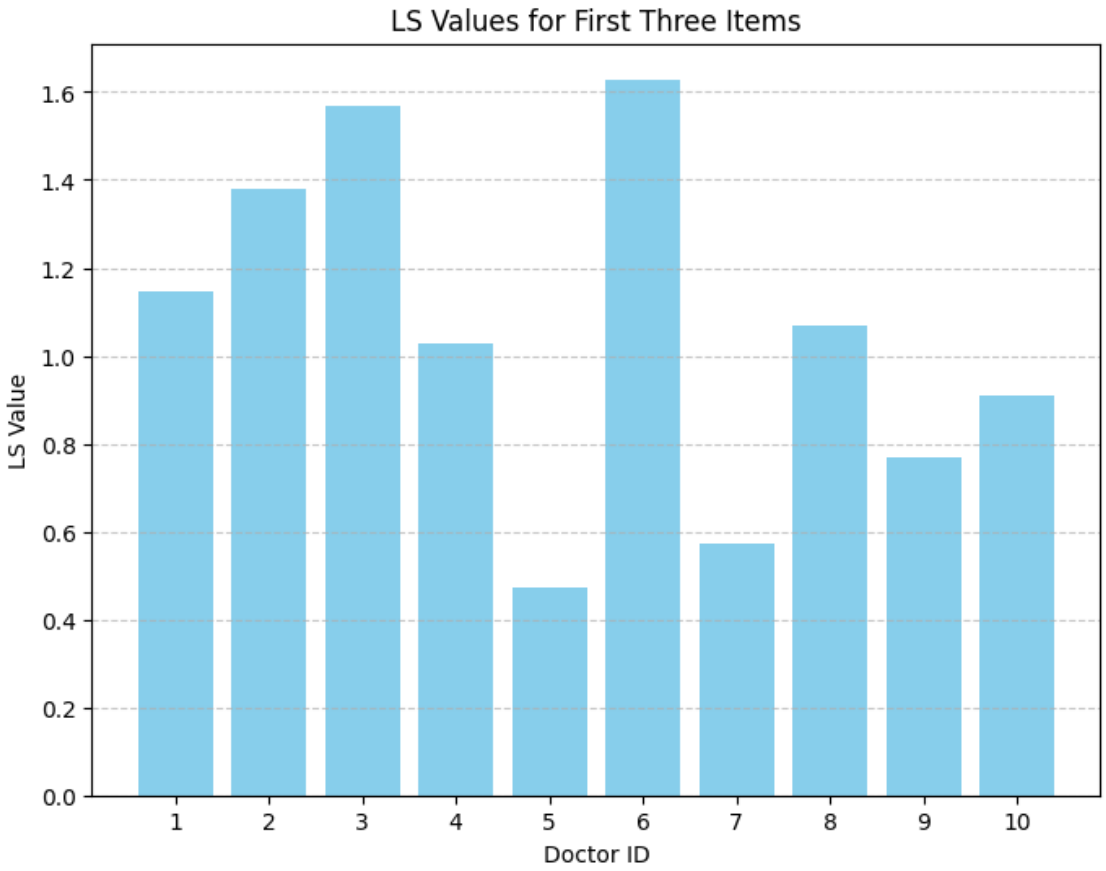
Note: N’oubliez pas de nommer vos axes et votre graphique.

[ ]:
```
import matplotlib.pyplot as plt

# TODO:

# Sous-ensemble du DataFrame recommande_gender pour n'inclure que les dix␣
 ↪premiers "doctor.id"
doctor_range = [i for i in range (11)]
recommande_gender_subset = recommande_gender[recommande_gender['doctor.id'].
 ↪isin(doctor_range)]

# Plot les valeurs LS pour les dix premiers "doctor.id"
plt.figure(figsize=(8, 6))
plt.bar(recommande_gender_subset['doctor.id'], recommande_gender_subset['LS'],␣
 ↪color='skyblue')
plt.title('LS Values for First Three Items')
plt.xlabel('Doctor ID')
plt.ylabel('LS Value')
plt.xticks(recommande_gender_subset['doctor.id'])
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



0.4 3. LIVRABLES

Vous devez remettre sur Moodle un fichier compressé .zip contenant :

1) Le code : Un Jupyter notebook en Python qui contient le code tel implanté avec les librairies minimales demandées pour ce TP (Python, Pandas, Matplotlib). Le code doit être exécutable sans erreur et accompagné des commentaires appropriés dans le notebook de manière. Tous vos résultats doivent être reproductibles avec le code dans le notebook. *Attention, en aucun cas votre code ne doit avoir été copié de d'ailleurs.*

2) Un fichier pdf représentant votre notebook complètement exécuté sous format pdf (obtenu via latex ou imprimé en pdf avec le navigateur). Assurez-vous que le PDF est entièrement lisible. Tutoriel youtube

ATTENTION: assurez-vous que votre fichier compressé .zip ne dépasse pas la taille limite acceptée sur Moodle.

ÉVALUATION Votre TP sera évalué sur les points suivants :

Critères : 1. Implantation correcte et efficace 2. Qualité du code (noms significatifs, structure, performance, gestion d'exception, etc.) (1 point) 3. Réponses correctes/sensées aux questions de réflexion ou d'analyse

CODE D'HONNEUR - Règle 1: Le plagiat de code est bien évidemment interdit. Toute utilisation de code doit être référencée adéquatement. Vous **ne pouvez pas** soumettre un code, écrit par quelqu'un d'autre. Dans le cas contraire, cela sera considéré comme du plagiat. - **Règle 2**: Vous êtes libres de discuter avec d'autres équipes. Cependant, vous ne pouvez en aucun cas incorporer leur code dans votre TP. - **Règle 3**: Vous ne pouvez pas partager votre code publiquement.

0.4.1 Conversion en PDF sur Google Colab

[ ]:
```
%%capture
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic
```

Assurez vous d'avoir téléchargé le TP complété en notebook sur votre ordinateur, puis importé ce fichier dans le répertoire "content" avant de rouler la ligne suivante.

[ ]:
```
!jupyter nbconvert --to pdf /content/TP1.ipynb
```