solution de la feuille feuille du 26.01.23

Exercice

Quel est dans `ventes` le numéro de l'enregistrement classé au 7ème rang si on trie par ordre décroissant en fonction de la colonne "TV" ?
Classez `big_df` en fonction de la colonne "mobility", quelle est la valeur de la variable "stop" pour le premier enregistrement ?
Utilisez `sort_index` sur les colonnes de la table `big_df`, quel est l'ordre des colonnes ?

La commande `rank` remplie la table en fonction du classement

```
>>> import pandas as pd
>>> ventes = pd.read_csv('sales.csv')
>>> ventes.head()
      TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3   12.0
3  151.5   41.3       58.5   16.5
4  180.8   10.8       58.4   17.9
>>> ventes.Radio.rank().head()
0    157.0
1    162.0
2    187.0
3    169.0
4     54.5
Name: Radio, dtype: float64
>>> ventes.Radio.rank(method='first').head()
0    157.0
1    162.0
2    187.0
3    169.0
4     54.0
Name: Radio, dtype: float64
>>> ventes.rank().head()
      TV  Radio  Newspaper  Sales
0  162.0  157.0      188.0  182.0
1   34.0  162.0      150.5   40.5
2   13.5  187.0      189.0   67.0
3  102.0  169.0      177.0  107.5
4  117.0   54.5      176.0  139.5
>>> ventes.rank(method='first').head()
```

```
      TV  Radio  Newspaper  Sales
0  162.0  157.0      188.0  182.0
1   34.0  162.0      150.0   40.0
2   13.0  187.0      189.0   66.0
3  102.0  169.0      177.0  107.0
4  117.0   54.0      176.0  139.0
```

Mais ce n'est pas ce qui est demandé, on vous demande l'index associé à la 7ème ligne après un tri décroissant

```
>>> ventes.sort_values(by="TV", ascending=False).head(10)
        TV  Radio  Newspaper  Sales
101  296.4   36.3      100.9   23.8
42   293.6   27.7        1.8   20.7
30   292.9   28.3       43.2   21.4
35   290.7    4.1        8.5   17.8
98   289.7   42.3       51.2   25.4
183  287.6   43.0       71.8   26.2
188  286.0   13.9        3.7   20.9
169  284.3   10.6        6.4   20.0
198  283.6   42.0       66.2   25.5
17   281.4   39.6       55.8   24.4
>>> ventes.sort_values(by="TV",
ascending=False).rank(method='first').head(10)
        TV  Radio  Newspaper  Sales
101  200.0  149.0      199.0  190.0
42   199.0  119.0        5.0  169.0
30   198.0  122.0      147.0  178.0
35   197.0   25.0       31.0  138.0
98   196.0  175.0      166.0  195.0
183  195.0  178.0      190.0  199.0
188  194.0   64.0       12.0  174.0
169  193.0   53.0       23.0  161.0
198  192.0  173.0      187.0  198.0
17   191.0  163.0      173.0  193.0
>>>
```

On cherche donc simplement l'enregistrement **188**

```
>>> ventes.sort_values(by="TV", ascending=False).index[6]
188
```

On trie `big_df` selon un critère et on récupère la valeur de la colonne 'stop'

```
print("big_df sorted, get 1st stop")
print(big_df.sort_values(by="mobility").head())
print("1st stop value is", end=' -> ')
print(big_df.sort_values(by="mobility").stop.iloc[0:1]) # slice en ligne
```

```
# accès direct par colonne
>>> big_df.sort_values(by='mobility').stop.T.iloc[0] # pas de slice
```

Tri sur les colonnes

```
>>> big_df.columns
Index(['mobility', 'start', 'stop', 'delay', 'date', 'target'],
dtype='object')
>>> big_df.sort_index(axis=1).columns # ascendant
Index(['date', 'delay', 'mobility', 'start', 'stop', 'target'],
dtype='object')
>>> big_df.sort_index(axis=1, ascending=False).columns # descendant
Index(['target', 'stop', 'start', 'mobility', 'delay', 'date'],
dtype='object')
```

Exercice

Dans le fichier `ventes`, ajoutez une colonne "pub" qui a pour valeur la moyenne des 3 médias. Construisez alors une colonne "cout" qui vaudra "gros" si pub est dans le dernier quartile, "micro" si pub est dans le premier quartile et "moyen" sinon

Vous aviez bien entendu corrigé, il s'agissait de la *table* `ventes`

```
ventes['pub'] = ventes[['TV', 'Radio', 'Newspaper']].mean(axis=1)
ventes.head()
```

La détection des valeurs qui nous intéresse se fait par

```
ventes.pub < ventes.pub.quantile(.25) # premier quartile
ventes.pub > ventes.pub.quantile(.75) # dernier quartile
```

L'affectation par

```
table.loc[mask, 'colonne'] = valeur
```

D'où

```
ventes['cout'] = "moyen" # valeur par défaut
ventes.loc[ventes.pub < ventes.pub.quantile(.25), 'cout'] = 'micro'
ventes.loc[ventes.pub > ventes.pub.quantile(.75), 'cout'] = 'gros'
ventes.cout.value_counts()
```

Exercice

Vous devez sortir le classement d'une promotion, déterminer quels sont ceux qui valident ou pas leur année. Pour valider son année, il faut avoir la moyenne sur l'ensemble des matières. On suppose qu'une personne inscrite en "ecoge" n'est pas inscrite en "scico" et réciproquement, par ailleurs toutes les autres matières sont obligatoires, une note manquante est un **0**. Les reports, sont les notes attribuées à une session antérieure.

```
# on va insérer les notes de report dans les bonnes tables
# insertion par concatenation
# step 1 on filtre par matière on enlève la colonne matiere, on renomme note
report_math=\
report[report.matiere=="maths"].drop('matiere',axis=1).rename({'note':'eval'
},
                                                              axis=1)
# step 2 on colle dans maths par concat et on renomme en accord
# avec l'index dans coeff
maths = pd.concat([maths,report_math]).rename({'eval':'math'}, axis=1)

# info même combat
report_info=\
report[report.matiere=="info"].drop('matiere',axis=1).rename({'note':'eval'}
,
                                                              axis=1)
# step 2 on colle dans info par concat et on renomme en accord
# avec l'index dans coeff
info = pd.concat([info,report_info]).rename({'eval':'info'}, axis=1)

## renommage pour les autres tables
ecoge.rename({'eval':'ecoge'}, axis=1, inplace=True)
scico.rename({'eval':'scico'}, axis=1, inplace=True)
option.rename({'eval':'option'}, axis=1, inplace=True)

## jointure sur userID
table = pd.merge(info,maths,on='userID',how='outer')
table = pd.merge(table,ecoge,on='userID',how='outer')
table = pd.merge(table,scico,on='userID',how='outer')
table = pd.merge(table,option,on='userID',how='outer')

## fillna(0)
table.fillna(0, inplace=True)

### truc de folie on va ordonner la table en fonction des coeff
print(table[coeff.index])
### et le top du top on peut grâce au produit de matrices obtenir
### les notes coefficientées
print(table[coeff.index] * coeff)
# Plus qu'à produire la moyenne, dommage que le calcul direct
```

```
# ne soit pas possible simplement
table['total'] = (table[coeff.index]*coeff).sum(axis=1)
# pour la moyenne il faut enlever l'un des coeff ecoge ou scico
table['moyenne'] = table.total/(coeff.sum()-coeff.ecoge)
# on attribue le meilleur classement aux ex aequo (par min)
table['classement'] = table.total.rank(ascending=False, method='min')
# si la moyenne est >= 10, on valide, sinon on ne valide pas
table['validation'] = np.where(table.moyenne>=10, 'Oui', 'Non')
print(table)
```

Quelques ressources

- petite initiation: kaggle
- video youtube youtube **1:00:26**
- la doc pands-doc