Nom	Prénom	matricule	SG
HAMAIDI	assim	202031044454	2
Meddad	Mohamed el mehdi		2

TP1 : Initiation et prise en main du langage Python (Modules : NumPy, matplotlib, Pandas, fichiers CSV, etc)

I. But du tp :

Le but de ce Tp numéro 1 c'est d'apprendre a utiliser le langage de programmation python ainsi que comment traiter un fichier CSV (Comma-separated values) en utilisant des différent fonction et library (pandas ……)

II. Manipulation :

1) Phase de prétraitement :

❖ afficher l'entête de l'ensemble de donnée :

```
In [1]: import pandas as pd
        df=pd.read_csv('titanic-passengers.csv', delimiter=';')

In [2]: df
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	343	No	2	Collander, Mr. Erik Gustaf	male	28.0	0	0	248740	13.0000	NaN	S
1	76	No	3	Moen, Mr. Sigurd Hansen	male	25.0	0	0	348123	7.6500	F G73	S
2	641	No	3	Jensen, Mr. Hans Peder	male	20.0	0	0	350050	7.8542	NaN	S
3	568	No	3	Palsson, Mrs. Nils (Alma Cornelia Berglund)	female	29.0	0	4	349909	21.0750	NaN	S
4	672	No	1	Davidson, Mr. Thornton	male	31.0	1	0	F.C. 12750	52.0000	B71	S
5	105	No	3	Gustafsson, Mr. Anders Vilhelm	male	37.0	2	0	3101276	7.9250	NaN	S
888	535	No	3	Cacic, Miss. Marija	female	30.0	0	0	315084	8.6625	NaN	S
889	102	No	3	Petroff, Mr. Pastcho ("Pentcho")	male	NaN	0	0	349215	7.8958	NaN	S
890	428	Yes	2	Phillips, Miss. Kate Florence ("Mrs Kate Louis...	female	19.0	0	0	250655	26.0000	NaN	S

891 rows × 12 columns

en analysant les données de ce tableau on a 891 lignes et 12 colonnes (passengerid , survived , pclasse , name …..) d'autre part en remarque que il y'a des valeurs manquant associé par le nom NAN (Not a Number) .

- ou bien en utilisant la fonction df.shape
- pour définir la type de données des colonnes :

```
In [13]: df.dtypes
Out[13]: PassengerId      int64
         Survived        object
         Pclass           int64
         Name            object
         Sex             object
         Age            float64
         SibSp            int64
         Parch            int64
         Ticket          object
         Fare           float64
         Cabin           object
         Embarked        object
         dtype: object
```

- la fonction df.tail() nous permet d'afficher les derniers lignes du tableau par exemple si on veut afficher la dernier ligne :

```
In [15]: df.tail(1)
```

Out[15]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
890	428	Yes	2	Phillips, Miss. Kate Florence ("Mrs Kate Louis...	female	19.0	0	0	250655	26.0	NaN	S

- la fonction df.info () nous permet d'afficher les valeurs NaN ainsi que les types de donnée :

```
In [16]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null object
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(4), object(6)
memory usage: 83.6+ KB
```

- **<u>Recherchez les informations manquantes et remplacez-les par les valeurs appropriées :</u>**

  pour afficher les informations manquantes en a plusieurs fonctions qui nous permet de la réaliser tel que :

- df.isnul()Il reviendra True pour les composants manquants et False pour les cellules non manquantes. Cependant, lorsque la dimension d'un ensemble de données est grande, il peut être difficile de déterminer l'existence de valeurs manquantes

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	True	False
4	False	False	False	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False	False	True	False

- donc pour simplifier on vas tout simplement déterminer le nombre de valeurs manquants en utlisant df.isnull().sum() ou bien df.isnull().sum().sum() qui nous faite la somme des NaN :

```
In [25]: df.isnull().sum()

Out[25]: PassengerId      0
         Survived         0
         Pclass           0
         Name             0
         Sex              0
         Age            177
         SibSp            0
         Parch            0
         Ticket           0
         Fare             0
         Cabin          687
         Embarked         2
         dtype: int64
```

```
In [26]: df.isnull().sum().sum()

Out[26]: 866
```

- pour afficher les positions des valeurs manquants spécifier a une colonne on utilise :

```
In [28]: print(df['Cabin'].head().isnull())

0     True
1    False
2     True
3     True
4    False
Name: Cabin, dtype: bool
```

True = valeur manquant ; false=valeur non manquant

- **<u>remplacez-les par les valeurs appropriées</u>**

  en premier lieu en utlise la fonction df["Age"].value_counts() qui nous permettra d'afficher une series qui contient le nombre de valeurs uniques :

```
number_of_elements=len(df["Cabin"])
print("number of element:",number_of_elements)
print(df["Age"].value_counts())

number of element: 891
24.00    30
22.00    27
18.00    26
28.00    25
19.00    25
30.00    25
```

Et pour les remplacer par les valeurs appropriées , on peut utiliser la fonction fillna() qui sert a remplacer les valeurs manquantes de colonne 'Age' par la moyenne de tout les valeurs :

```
df["Age"].fillna('24',inplace=True)
df.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	10	Yes	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708	G6	C
887	61	No	3	Sirayanian, Mr. Orsen	male	22	0	0	2669	7.2292	G6	C
888	535	No	3	Cacic, Miss. Marija	female	30	0	0	315084	8.6625	G6	S
889	102	No	3	Petroff, Mr. Pastcho ("Pentcho")	male	24	0	0	349215	7.8958	G6	S
890	428	Yes	2	Phillips, Miss. Kate Florence ("Mrs Kate Louis...	female	19	0	0	250655	26.0000	G6	S

Si on veut vérifier que les NaN sont disparus on utillise :

```
: df.isnull().sum()
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       2
dtype: int64
```

Et on remarque bien que Les NaN POUR la colonne ‘Age’ =0 (probléme Résolue)

- la même chose pour des valeurs str , on peut les remplacer par la valeurs la plus fréquenté en utilisant fillna() (elle est faite pour la catégorie ‘Cabin ‘) .

2) Phase de visualisation :

- Pour cela en se basent sur barplot seaborn et histograms
  D’abord il faut importer les bibliothèques seaborn et plt en utilisant import seaborn as sns et import matplotlib. pyplot as plt .

❖ barplot seaborn

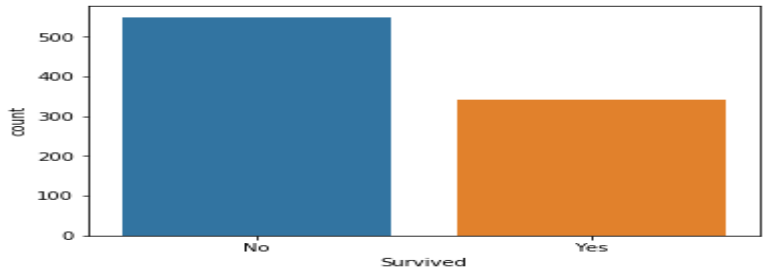
- en se basant sur :

```
seaborn.countplot(data=None, *, x=None, y=None, hue=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, fill=True, hue_norm=None, stat='count', width=0.8, dodge='auto', gap=0, log_scale=None, native_scale=False, formatter=None, legend='auto', ax=None, **kwargs)
```

on peut tracer notre visualisation désirée , pour notre cas data=df et x=’Survived’
plt.xticks est utilisée pour la rotation des x.

```
import seaborn as sns
import matplotlib. pyplot as  plt
sns.countplot(x='Survived',data=df)
plt.xticks(rotation=0)
```

```
(array([0, 1]), <a list of 2 Text xticklabel object
```

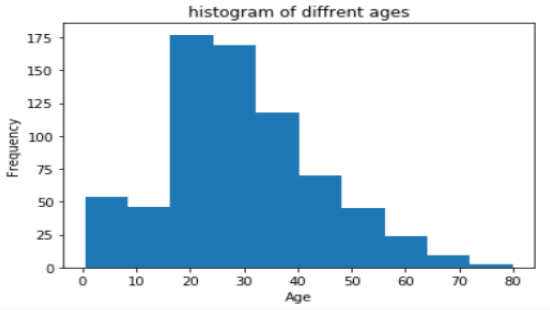


Donc le barplot nous a permet d’englober tous les données dans une seule figure d’où la facilité de la lecture et on ramarque que le nombre de morts est supérieur par rapport au Survivants .

❖ **Histograms interpretation :**

en se basant sur la commande df['Age'].plot.hist() en peut tracer l'histogram de la colonne Age :

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('titanic-passengers.csv', delimiter=';')
plt.title('histogram of diffrent ages')
plt.xlabel('Age')
df['Age'].plot.hist()
df
```



Le bareplot et l' **histogramme** sont les deux moyens d'afficher des données sous forme de diagramme tel que le bareplot affiche des barres séparer par contre l'histogramme il les a affiches attachées .

Analyse : le diagramme montre que l'age le plus dominant est entre 18ans et 25ans ainsi que dans le titanic l'intervalle d'age est entre 2ans et 80ans .

❖ **La suppression d'une colonne :**

Pour cela on utilise la fonction df.drop('colonne désiré') , par exemple pour supprimer la colonne Nom :

```
df.drop(columns='Name')
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	343	No	2	Collander, Mr. Erik Gustaf	male	28.0	0	0	248740	13.0000	NaN	S
1	76	No	3	Moen, Mr. Sigurd Hansen	male	25.0	0	0	348123	7.6500	F G73	S
2	641	No	3	Jensen, Mr. Hans Peder	male	20.0	0	0	350050	7.8542	NaN	S
3	568	No	3	Palsson, Mrs. Nils (Alma Cornelia Berglund)	female	29.0	0	4	349909	21.0750	NaN	S
4	672	No	1	Davidson, Mr. Thornton	male	31.0	1	0	F.C. 12750	52.0000	B71	S

❖ **Corrélation**

Pour afficher la corrélation il suffit juste d'utliser df.corr() :

```
df.corr()
```

	PassengerId	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Pclass	-0.035144	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	-0.549500	0.096067	0.159651	0.216225	1.000000

❖ **Programme a exécuté**

D'abord il faut se débarasser de tout les NAN en utilisant fillna() :

```
In [22]: df['Cabin'].fillna('G6' , inplace=True)

In [23]: df['Age'].fillna(df['Age'].mean() , inplace=True)

In [24]: df['Embarked'].fillna(df['Embarked'].mode(),inplace=True)

In [25]: df.isnull().sum()
```

```
In [26]: df=df.drop(['Embarked'] , axis=1)

In [27]: df
```

Après avoir éliminer tout les NANS et en vérifiant avec isnull().sum()

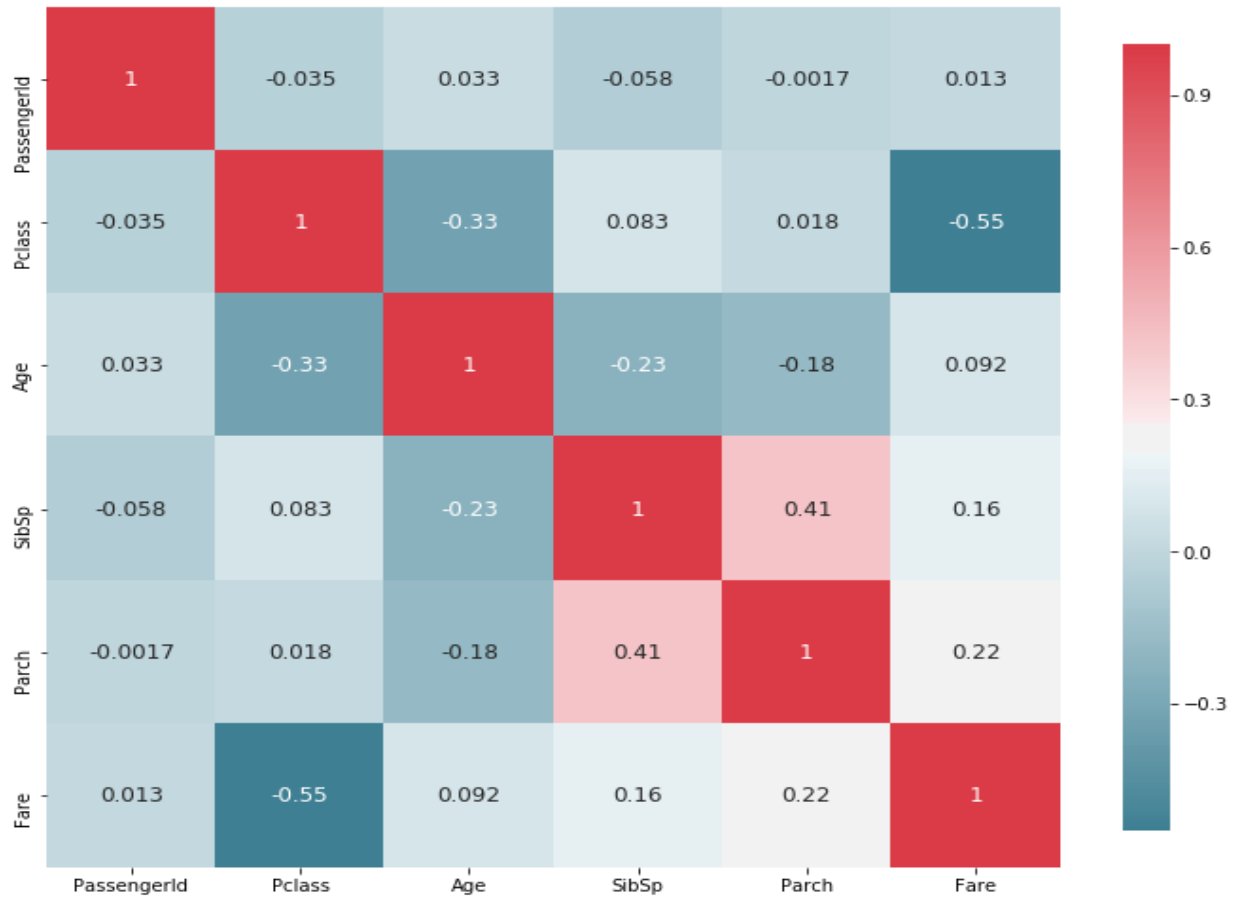
```
In [28]: df.isnull().sum()
ut[28]: PassengerId    0
        Survived       0
        Pclass         0
        Name           0
        Sex            0
        Age            0
        SibSp          0
        Parch          0
        Ticket         0
        Fare           0
        Cabin          0
        dtype: int64
```

Maintenant nous pouvons éxucuté le programme de corrélation :

```
def plot_correlation_map( df ):
  corr = df.corr()
  s , ax = plt.subplots( figsize =( 12 , 10 ) )
  cmap = sns.diverging_palette( 220 , 10 , as_cmap = True )
  s = sns.heatmap(corr,cmap = cmap,square=True,cbar_kws={ 'shrink' : .9 },ax=ax,annot = True,annot_kws = { 'fontsize' : 12 })
```

Pour afficher la figure de corrélation en utilise la commande :

```
plot_correlation_map( df )
```



Analyse : en remarque que pour x=y la corrélation =1 ce qui signifie que c'est une bonne corrélation cad 100 % de ressemblance , d'autre part les autres corrélation sont different , plus que la corrélation tend vers 0 cad la corrélation est mauvaise .

III. Conclusion :

Ce tp nous a permet de comprendre et maitriser le langage python ainsi comment utiliser jupyter et traiter un fichier CSV en se débarrassant des valeurs non nulles et apprendre comment tracer le barplot et l'histogramme ainsi de tracer la corrélation .