

Livre blanc sur l'analyse des données et la BI

Sarah Bouchikh et Emma Gadea

Sommaire

Introduction

Visualtion de la base de données
Définition des données du fichier

I- Chapitre 1 : Description des données

A-Typologie des données
B- Catégorie de données
C-Transformation des données
D-Les mesures de tendances centrale
E-Les mesures de dispersion et de position
F-Représentation graphique de la distribution des données

II- Chapitre 2 : Exploration des données

A-Comparer des variables
B-Représenter graphiquement les relations entre les variables

III- Chapitre 3 : Modelisation des données

A-Quel est le profil des collaborateurs qui vont quitter l'entreprise ?
B-Pourquoi les collaborateurs quittent l'entreprise ?

IV- Conclusion

Entrée [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from PIL import Image, ImageTk
import tkinter as tk

from matplotlib import image
%matplotlib inline

bidata = pd.read_csv('HR_training.csv', encoding='unicode_escape', engine='python', delimit
```

Entrée [2]:

bidata

Out[2]:

	id_colab	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company
0	4	0,72	0,87	5	223	10
1	5	0,37	0,52	2	159	10
2	7	0,1	0,77	6	247	10
3	9	0,89	1	5	224	10
4	10	0,42	0,53	2	142	10
...
10046	14993	0,4	0,48	2	155	10
10047	14995	0,4	0,57	2	151	10
10048	14996	0,37	0,48	2	160	10
10049	14997	0,37	0,53	2	143	10
10050	14998	0,11	0,96	6	280	10

10051 rows × 11 columns

DEFINITION DES DONNEES DU FICHIER

- Id : Numéro du collaborateur
- Satisfaction level : Niveau de satisfaction des collaborateurs
- Last evaluation : Niveau de satisfaction des collaborateurs lors de l'évaluation précédente
- Number project : Nombre de projets portés par le collaborateur
- Average monthly hours : Nombre d'heures mensuel moyen d'un collaborateur
- Time spend company : Nombre d'années passées au sein de l'entreprise
- Work accident : Information qui précise si le collaborateur a subi un accident du travail
- Promotion last 5 years : Information qui précise si le collaborateur a été sujet à une promotion durant les 5 dernières années
- Job : Secteur dans lequel travaille le collaborateur
- Salary : Salaire du collaborateur (bas, moyen, haut)
- Left : Si le collaborateur a finalement quitté l'entreprise ou non

I- Chapitre 1 : Description des données

A-Typologie des données

Il existe deux grandes catégories de données :

Les données qualitatives et les données quantitative elle même divisés en sous catégorie en voici la définition :

Les données quantitatives

Continues : Les données quantitatives continues sont des données subjectives mesurées sur une échelle continue, comme un degré de satisfaction ou une intensité d'émotion.

Discrètes : les données quantitatives discrètes sont à l'inverse non mesurables. Par exemple, ce sont le nombre d'enfants dans une famille, le nombre de produits achetés, le nombre de personnes dans une pièce.

Les données qualitatives

Ordinal: Une donnée qualitative ordinaire est une mesure numérique qui permet de classer les observations selon un ordre, mais sans échelle de mesure précise.

Nominale: Une donnée qualitative nominale est une donnée qui est utilisée pour étiqueter ou catégoriser les observations, sans aucun ordre ou signification numérique.

B- Catégorie de données

Variables à expliquer (dépendantes)

On considère les variables à expliquer comme étant des variables que l'on peu prédire à l'aide des autres variables. On peut dire qu'elles sont corrélées. Par exemple on peut utiliser la droite de régression linéaire.

Variables explicatives (indépendantes)

On considère les variables explicatives comme étant des variables qui sont indépendantes les unes des autres.

Voici un tableau résumant la typologie de nos variables, ainsi que leur catégorie :

Entrée [3]:

```
data = {'Nom des données':['id_colab','satisfaction_level','last_evaluation','number_project']
tableau_variables = pd.DataFrame(data)
tableau_variables
```

Out[3]:

	Nom des données	Typologie des données	Catégorie de données
0	id_colab	Qualitative nominale	Variable explicative
1	satisfaction_level	Quantitative continue	Variable explicative
2	last_evaluation	Quantitative continue	Variable explicative
3	number_project	Quantitative continue	Variable explicative
4	average_montly_hours	Quantitative continue	Variable explicative
5	time_spend_company	Quantitative continue	Variable explicative
6	work_accident	Qualitative nominale	Variable explicative
7	promotion_last_5years	Qualitative nominale	Variable explicative
8	job	Qualitative nominale	Variable explicative
9	salary	Qualitative ordinaire	Variable explicative
10	left	Qualitative nominale	Variable à expliquer

C-Transformation des données

Données qualitatives en données quantitatives

Nous allons transformer la variable 'left', que nous considérons comme une donnée qualitative continue, en donnée quantitative.

Pour cela, nous avons d'abord dupliqué la colonne 'left', puis nous avons changé les données par : 1='yes' et 0='no'.

Entrée [4]:

```
bidata = bidata.assign(left_quanti=bidata['left'])
bidata['left_quanti'] = bidata['left_quanti'].map({1 : 'yes', 0 : 'no'})
bidata
```

Out[4]:

	id_colab	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_compa
0	4	0,72	0,87	5	223	
1	5	0,37	0,52	2	159	
2	7	0,1	0,77	6	247	
3	9	0,89	1	5	224	
4	10	0,42	0,53	2	142	
...
10046	14993	0,4	0,48	2	155	
10047	14995	0,4	0,57	2	151	
10048	14996	0,37	0,48	2	160	
10049	14997	0,37	0,53	2	142	

Données quantitatives en données qualitatives

Nous allons transformer la variable 'average_monthly_hours', que nous considérons comme quantitative nominale, en donnée qualitative.

Pour cela, nous avons d'abord dupliqué la colonne 'average_monthly_hours', puis nous avons défini les contraintes relative à la dénomination des différentes données. Nous avons défini 3 catégories : 'low' pour les personnes travaillant moins de 110h/mois,'medium' pour les personnes travaillant entre 110h et 160h/mois et enfin 'high' pour les personnes travaillant plus de 160h/mois.

Entrée [5]:

```
bidata = bidata.assign(average_montly_hours_quali=bidata['average_montly_hours'])
bins = [0, 110, 160, float('inf')]
labels = ['low', 'medium', 'high']
bidata['average_montly_hours_quali'] = pd.cut(bidata['average_montly_hours'], bins=bins, labels=labels)
```

ib	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	work_accident
4	0,72	0,87	5	223	5	
5	0,37	0,52	2	159	3	
7	0,1	0,77	6	247	4	
9	0,89	1	5	224	5	
10	0,42	0,53	2	142	3	
...	
33	0,4	0,48	2	155	3	
35	0,4	0,57	2	151	3	
36	0,37	0,48	2	160	3	
37	0,37	0,53	2	142	3	

Entrée [6]:

```
bidata.salary.unique().tolist()
```

Out[6]:

```
['low', 'medium', 'high']
```

D-Les mesures de tendances centrale

Il existe 3 mesures de tendance centrale utilisées pour décrire les données d'une distribution statistique :

La médiane

C'est la valeur qui sépare la moitié inférieure de la moitié supérieure d'un ensemble (deux parties d'effectifs égaux). La médiane est utilisée pour mesurer la tendance centrale d'un ensemble de données.

La moyenne

La moyenne est utilisée pour mesurer la tendance centrale d'un ensemble de données, elle peut également être utilisée pour identifier des tendances ou des changements dans les données au fil du temps en comparant les moyennes de différentes périodes ou groupes.

C'est la somme des données divisée par l'effectif global. C'est l'indicateur le plus simple pour résumer l'information fournie par un ensemble de données.

La mode

La mode est une mesure de tendance centrale utilisée pour décrire les données d'une distribution statistique. C'est la valeur qui apparaît le plus fréquemment dans un ensemble de données. Elle est souvent utilisée pour décrire les données qualitatives ou pour identifier les éléments les plus fréquents dans un ensemble de données.

Afin de rendre notre livrable plus concis nous avons décidé de ne pas interpréter toutes les variables.

Entrée [7]:

```
#Mediane  
mediane_bidata = bidata.median()  
mediane_bidata  
##on ne tient pas compte de id_colab et left car insignifiant
```

C:\Users\Emma\AppData\Local\Temp\ipykernel_23652\3573655911.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

```
mediane_bidata = bidata.median()
```

Out[7]:

```
id_colab          7500.0  
number_project    4.0  
average_montly_hours 199.0  
time_spend_company 3.0  
work_accident     0.0  
promotion_last_5years 0.0  
left              0.0  
dtype: float64
```

D'après le résultat, nous obtenons une médiane de 4 pour "number_project" et une médiane de 199 pour "average_montly_hours". Nous pouvons déduire que 50% des employés interrogés ont été affectés à plus de 4 projets et 50% ont été affectés à moins de 4 projets. Nous pouvons également déduire que plus de moitié ont travaillé plus de 199h et l'autre moitié moins de 199h.

Entrée [8]:

```
#Moyenne  
average_bidata = bidata.mean()  
average_bidata  
##on ne tient pas compte de id_colab et left car insignifiant
```

C:\Users\Emma\AppData\Local\Temp\ipykernel_23652\3805715750.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

```
average_bidata = bidata.mean()
```

Out[8]:

```
id_colab          7492.237489  
number_project    3.803303  
average_montly_hours 200.812655  
time_spend_company 3.507213  
work_accident     0.147050  
promotion_last_5years 0.020098  
left              0.237688  
dtype: float64
```

D'après le résultat, nous obtenons une moyenne de 3.8 pour "number_project" et une moyenne de 200.8 pour "average_monthly_hours". Nous pouvons donc déduire que en moyenne les employés ont environ 4 projets et qu'ils travaillent en moyenne 200h/mois.

Entrée [9]:

```
#Mode
from scipy import stats

mode_bidata = bidata.mode()
print(mode_bidata.loc[0])
##on ne tient pas compte de id_colab car insignifiant
```

```
id_colab                      4
satisfaction_level             0,1
last_evaluation                 0,54
number_project                  4.0
average_monthly_hours          156.0
time_spend_company              3.0
work_accident                   0.0
promotion_last_5years          0.0
job                            sales
salary                          low
left                           0.0
left_quanti                     no
average_monthly_hours_quali    high
Name: 0, dtype: object
```

D'après le résultat, le mode pour number_project est de 4 et 156 pour average_monthly_hours. Ce qui signifie que les employés ont tendance à être affecté à 4 projets et qu'ils ont tendance à travailler 156h/mois.

E-Les mesures de dispersion et de position

Les mesures de dispersion décrivent la variabilité ou la répartition des valeurs d'une variable statistique dans un ensemble de données. Les mesures les plus courantes de dispersion sont :

L'écart-type

L'écart-type est une mesure de dispersion des données autour de la moyenne. Plus l'écart-type est faible, plus la population est homogène et à l'inverse plus il est élevé plus elle est hétérogène. Il est particulièrement utile pour identifier les données aberrantes, qui peuvent avoir un impact important sur les résultats globaux. Il est également utilisé dans les calculs statistiques tels que la construction d'intervalles de confiance, le test d'hypothèse et l'analyse de la variance.

La variance

La variance mesure la moyenne des distances au carré des valeurs de la variable par rapport à la moyenne de cette variable. Elle est également utilisée dans les calculs statistiques tels que la construction d'intervalles de confiance, le test d'hypothèse et l'analyse de la variance. Elle est utilisée pour mesurer le risque dans l'investissement, pour identifier la volatilité d'un actif financier.

La fréquence

La fréquence désigne le nombre de fois où une valeur ou un événement se produit dans un ensemble de données. Elle peut être exprimée sous forme de nombre absolu ou de pourcentage (c'est-à-dire en rapport à la taille totale de l'ensemble de données).

C'est un outil clé pour l'analyse de données descriptives, qui permet de comprendre la répartition des valeurs dans un ensemble de données.

L'étendue (minimum,maximum)

L'étendue statistique est un concept utilisé pour mesurer la variabilité ou la dispersion des données dans un ensemble de données. Elle est généralement utilisée pour décrire la dispersion des données autour de la moyenne et peut être utilisée pour comparer la variabilité entre différents ensembles de données.

Le percentile

Un percentile est un concept statistique qui décrit la valeur en dessous de laquelle se trouve un certain pourcentage de valeurs dans un ensemble de données. Par exemple, si un élève a obtenu un score de 75ème percentile sur un test, cela signifie qu'il a obtenu un score supérieur à 75% des autres élèves. Les percentiles peuvent être utilisés pour donner une idée de la distribution des données et pour comparer les performances d'une personne à celles des autres dans un groupe.

Entrée [10]:

```
#Ecart-type  
std_all = bidata.std()  
std_all  
##on ne tient pas compte de id_colab et left car insignifiant
```

C:\Users\Emma\AppData\Local\Temp\ipykernel_23652\3234764835.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

```
std_all = bidata.std()
```

Out[10]:

id_colab	4331.355967
number_project	1.230464
average_montly_hours	49.775706
time_spend_company	1.480360
work_accident	0.354173
promotion_last_5years	0.140341
left	0.425688

dtype: float64

D'après le résultat, l'écart-type de "number_project" est de 1 donc plutôt homogène et est de 49 pour "average_montly_hours" donc plus hétérogène. Nous pouvons donc déduire que les employés sont affectés en général au même nombre de projet, mais que leurs temps passé au travail est généralement différents pour chacun d'entre eux.

Entrée [11]:

```
#Variance
var_all = bidata.var()
var_all
##on ne tient pas compte de id_colab et left car insignifiant
```

C:\Users\Emma\AppData\Local\Temp\ipykernel_23652\3545053567.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=y=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

```
var_all = bidata.var()
```

Out[11]:

```
id_colab          1.876064e+07
number_project    1.514043e+00
average_monthly_hours 2.477621e+03
time_spend_company 2.191465e+00
work_accident     1.254388e-01
promotion_last_5years 1.969555e-02
left              1.812103e-01
dtype: float64
```

D'après le résultat, la variance de number_project est de 1.5 c'est à dire que la dispersion est proche de la moyenne alors que la variance de average_monthly_hours est de 2.4 donc plus écarté de la moyenne.

Entrée [12]:

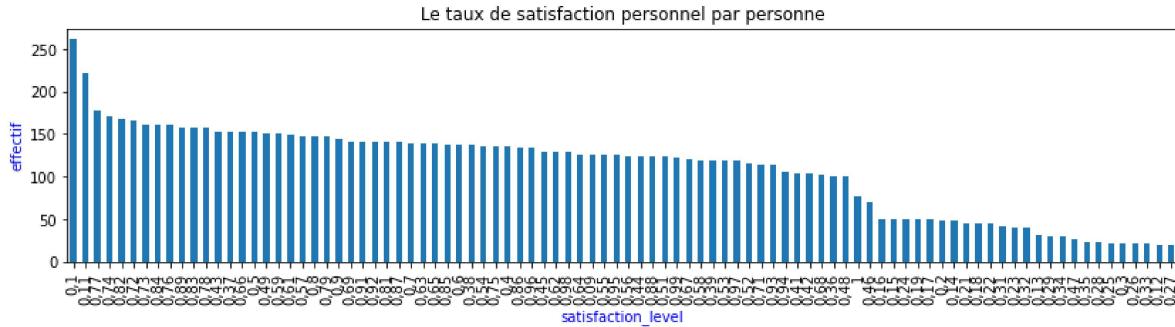
```
# Fréquence
##on ne tient pas compte de id_colab car insignifiant donc nous n'avons pas calculé sa fréq
freq_satisfaction_level = bidata['satisfaction_level'].value_counts()
freq_satisfaction_level
```

Out[12]:

```
0,1      261
0,11     222
0,77     177
0,74     171
0,82     167
...
0,3      22
0,26     22
0,33     21
0,12     20
0,27     19
Name: satisfaction_level, Length: 92, dtype: int64
```

Entrée [13]:

```
freq_satisfaction_level.plot(kind='bar')
plt.title("Le taux de satisfaction personnel par personne")
plt.xlabel('satisfaction_level', color = 'blue')
plt.ylabel('effectif', color = 'blue')
plt.gcf().set_size_inches(14, 3)
plt.show()
```



D'après le résultat, nous pouvons constater que 261 personnes ont répondu 0,1 de taux de satisfaction.

Entrée [14]:

```
freq_last_evaluation = bidata['last_evaluation'].value_counts()
freq_last_evaluation
```

Out[14]:

```
0,54    248
0,55    236
0,5     233
0,51    227
0,53    225
...
0,38    33
0,42    32
0,43    31
0,44    29
0,36    16
Name: last_evaluation, Length: 65, dtype: int64
```

Entrée [15]:

```
freq_last_evaluation.plot(kind='bar')
plt.title("Le taux de satisfaction personnel par personne durant la dernière évaluation annuelle")
plt.xlabel('last_evaluation', color = 'blue')
plt.ylabel('effectif', color = 'blue')
plt.gcf().set_size_inches(12, 2)

plt.show()
```



D'après le résultat, nous pouvons constater que l'année dernière 248 personnes (effectif le plus élevé) ont répondu 0.54 de taux de satisfaction. Seulement 16 personnes ont répondu 0.36.

Entrée [16]:

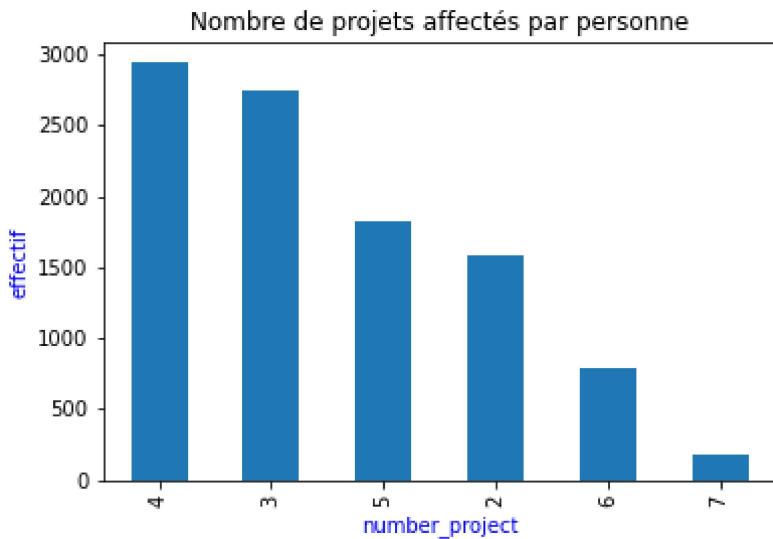
```
freq_number_project = bidata['number_project'].value_counts()
freq_number_project
```

Out[16]:

```
4    2938
3    2739
5    1830
2    1582
6     790
7     172
Name: number_project, dtype: int64
```

Entrée [17]:

```
freq_number_project.plot(kind='bar')
plt.title("Nombre de projets affectés par personne")
plt.xlabel('number_project', color = 'blue')
plt.ylabel('effectif', color = 'blue')
plt.show()
```



D'après le résultat, 2938 employés ont été affectés à 4 projets et 172 à 7 projets.

Entrée [18]:

```
freq_average_montly_hours = bidata['average_montly_hours'].value_counts()
freq_average_montly_hours
```

Out[18]:

```
156    102
149    101
140     99
151     98
159     93
...
288      5
297      5
96       5
299      4
303      3
Name: average_montly_hours, Length: 215, dtype: int64
```

Entrée [19]:

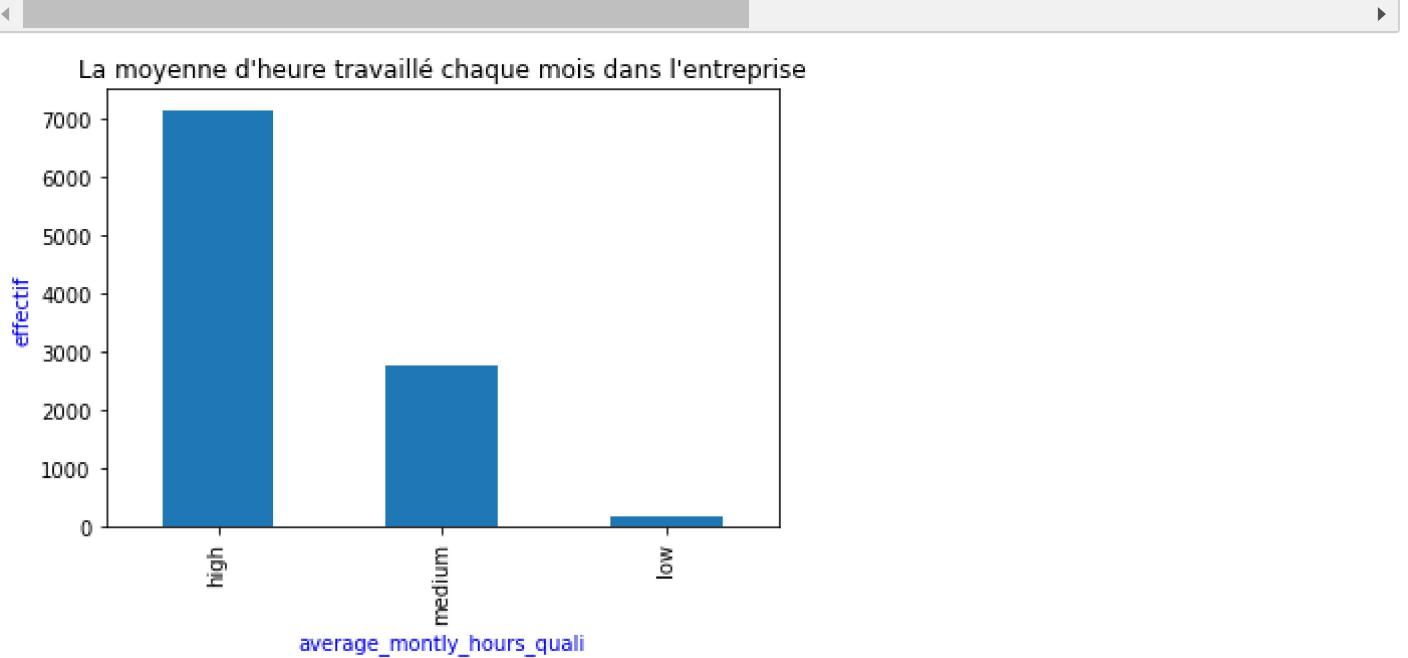
```
freq_average_montly_hours_quali = bidata['average_montly_hours_quali'].value_counts()
freq_average_montly_hours_quali
```

Out[19]:

```
high     7138
medium   2751
low      162
Name: average_montly_hours_quali, dtype: int64
```

Entrée [20]:

```
#Les colonnes "average_monthly_hours" et "average_monthly_hours_quali" étant les mêmes, il est
#Il y a moins de données, donc le graphique est plus visible et plus compréhensible. Mais c
freq_average_monthly_hours_quali.plot(kind='bar')
plt.title("La moyenne d'heure travaillé chaque mois dans l'entreprise")
plt.xlabel('average_monthly_hours_quali', color = 'blue')
plt.ylabel('effectif', color = 'blue')
plt.show()
```



D'après le résultat, 7238 employés ont travaillés en moyenne plus de 160h/mois et seulement 162 ont travaillés moins de 110h/mois

Entrée [21]:

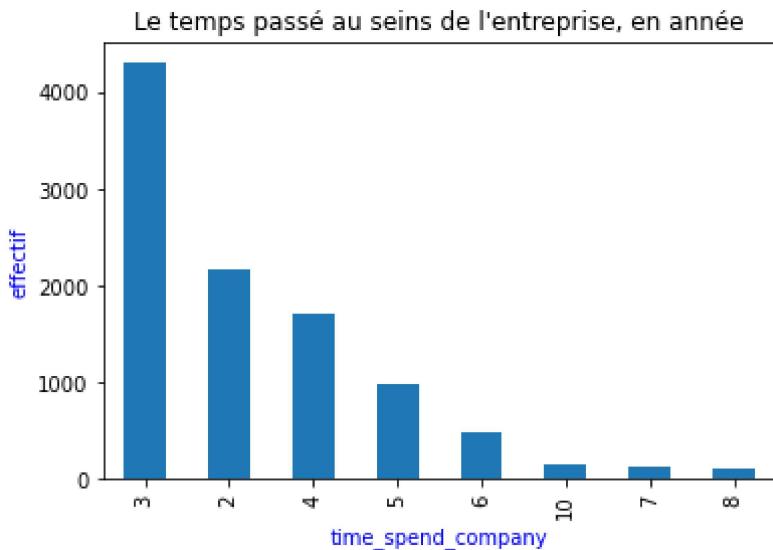
```
freq_time_spend_company = bidata['time_spend_company'].value_counts()
freq_time_spend_company
```

Out[21]:

```
3      4311
2      2174
4      1713
5      976
6      483
10     155
7      122
8      117
Name: time_spend_company, dtype: int64
```

Entrée [22]:

```
freq_time_spend_company.plot(kind='bar')
plt.title("Le temps passé au seins de l'entreprise, en année")
plt.xlabel('time_spend_company', color = 'blue')
plt.ylabel('effectif', color = 'blue')
plt.show()
```



D'après le résultat, 4311 employés sont là depuis 3 ans. Pour la minorité d'entre eux soit 117 employés, depuis 8 ans.

Entrée [23]:

```
freq_work_accident = bidata['work_accident'].value_counts()
freq_work_accident
```

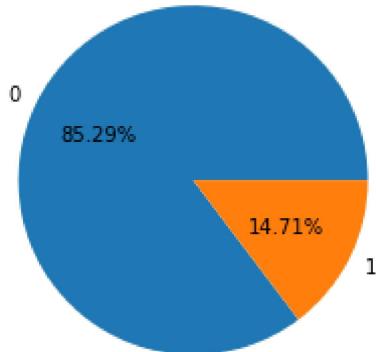
Out[23]:

```
0    8573
1    1478
Name: work_accident, dtype: int64
```

Entrée [24]:

```
freq_work_accident.plot(kind='pie', autopct='%1.2f%%')
plt.title('Pourcentage de personnes ayant subi ou non un accident du travail')
plt.ylabel('')
plt.show()
```

Pourcentage de personnes ayant subi ou non un accident du travail



D'après le résultat, 8573 ou 85% des employés n'ont pas subi d'accident de travail et 1478 ou 14% ont subi un accident du travail.

Entrée [25]:

```
freq_promotion_last_5years = bidata['promotion_last_5years'].value_counts()
freq_promotion_last_5years
```

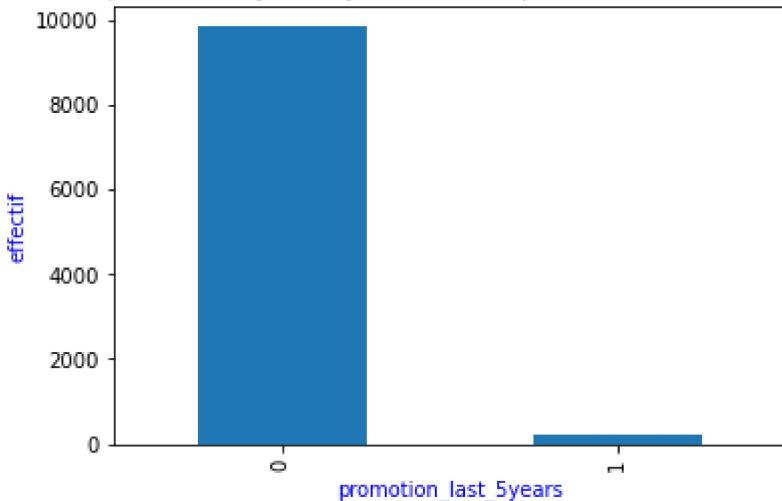
Out[25]:

```
0    9849
1    202
Name: promotion_last_5years, dtype: int64
```

Entrée [26]:

```
freq_promotion_last_5years.plot(kind='bar')
plt.title('Nombre de personnes ayant reçu ou non une promotion ces 5 dernières années')
plt.xlabel('promotion_last_5years', color = 'blue')
plt.ylabel('effectif', color = 'blue')
plt.show()
```

Nombre de personnes ayant reçu ou non une promotion ces 5 dernières années



D'après le résultat, seulement 202 employés ont eu une promotion lors de ces 5 dernières années.

Entrée [27]:

```
freq_job = bidata['job'].value_counts()
freq_job
```

Out[27]:

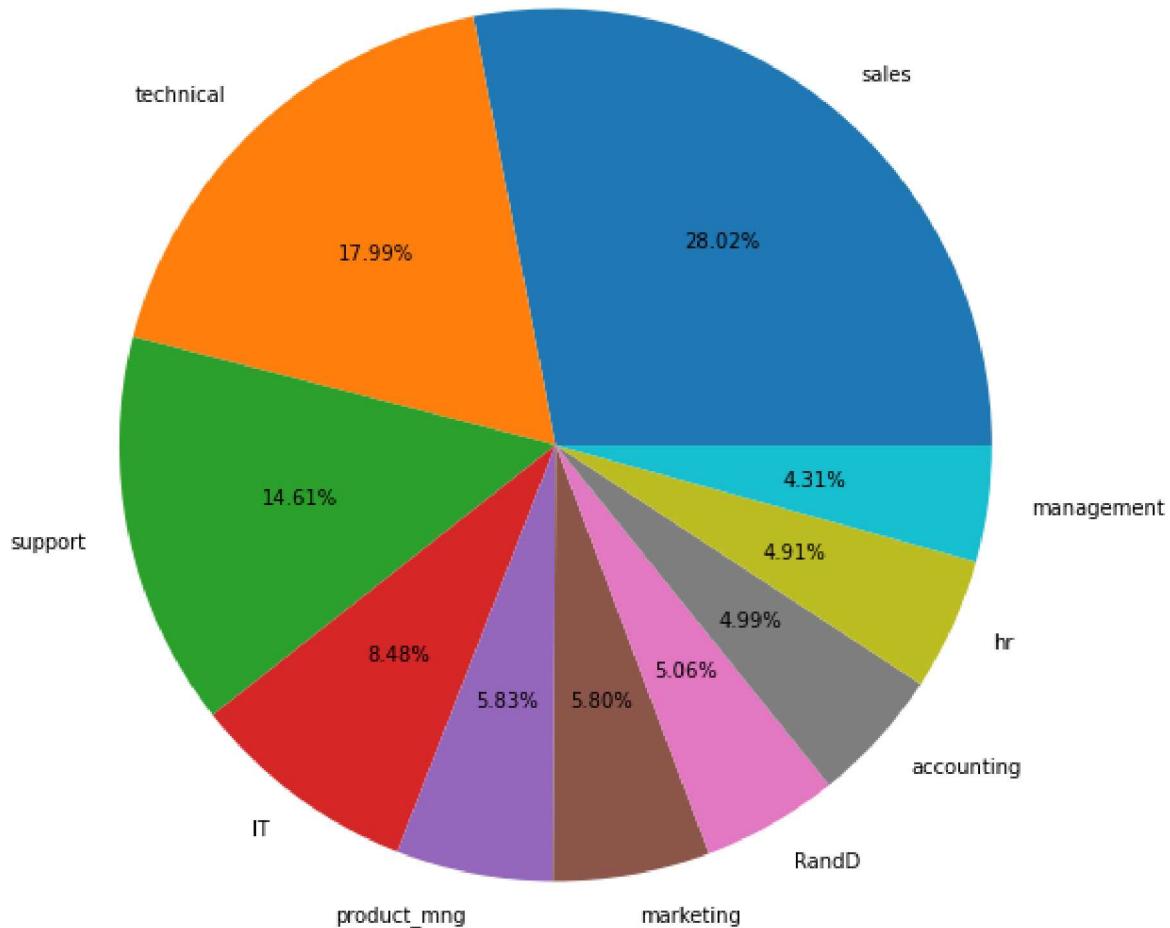
```
sales          2816
technical      1808
support        1468
IT             852
product_mng    586
marketing       583
RandD           509
accounting      502
hr              494
management      433
Name: job, dtype: int64
```

Entrée [28]:

```
freq_job.plot(kind='pie', autopct='%1.2f%%')
plt.title('Les différents jobs des personnes dans notre base de donnée')
plt.ylabel('')
plt.gcf().set_size_inches(10, 10)

plt.show()
```

Les différents jobs des personnes dans notre base de donnée



D'après le résultat, 2816 employés soit 28% ont un poste en vente, et seulement 433 soit 4.3% ont un poste dans le domaine du management.

Entrée [29]:

```
freq_salary = bidata['salary'].value_counts()
freq_salary
```

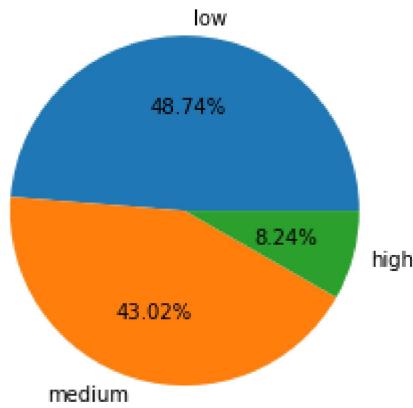
Out[29]:

```
low      4899
medium   4324
high     828
Name: salary, dtype: int64
```

Entrée [30]:

```
freq_salary.plot(kind='pie', autopct='%1.2f%%')
plt.title('La fréquence des différents niveau de salaire au sein de notre base de donnée, e
plt.ylabel('')
plt.show()
```

La fréquence des différents niveau de salaire au sein de notre base de donnée, en %



D'après le résultat, 4899 soit 48.7% des employés ont un salaire faible et 828 soit 8.2% ont un salaire élevé.

Entrée [31]:

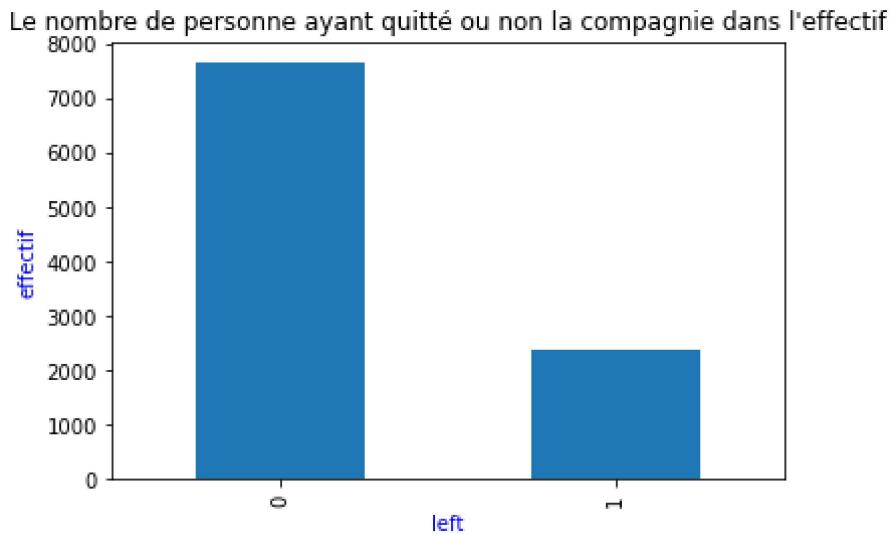
```
freq_left = bidata['left'].value_counts()
freq_left
#Les colonnes "left_quanti" et "left" ont les mêmes données. La fréquence sera donc la même.
```

Out[31]:

```
0    7662
1    2389
Name: left, dtype: int64
```

Entrée [32]:

```
freq_left.plot(kind='bar')
plt.title("Le nombre de personne ayant quitté ou non la compagnie dans l'effectif")
plt.xlabel('left', color = 'blue')
plt.ylabel('effectif', color = 'blue')
plt.show()
```



D'après le résultat, 7662 employés sont restés dans l'entreprise et 2389 sont partis.

Entrée [33]:

```
#Etendue min
#id_colab, job, salary, left_quanti, average_montly_hours_quali sont insignifiant.
min_all = bidata.min()
min_all
```

Out[33]:

id_colab	4
satisfaction_level	0,09
last_evaluation	0,36
number_project	2
average_montly_hours	96
time_spend_company	2
work_accident	0
promotion_last_5years	0
job	IT
salary	high
left	0
left_quanti	no
average_montly_hours_quali	low

dtype: object

D'après le résultat, le minimum de projets affecté est de 2 et le minimum d'heures effectuées par mois est de 96h.

Entrée [34]:

```
#Etendue max  
#id_colab, job, salary, left_quanti, average_monthly_hours_quali sont insignifiant.  
max_all = bidata.max()  
max_all
```

Out[34]:

```
id_colab                      14998  
satisfaction_level             1  
last_evaluation                 1  
number_project                  7  
average_monthly_hours           310  
time_spend_company              10  
work_accident                   1  
promotion_last_5years           1  
job                           technical  
salary                         medium  
left                           1  
left_quanti                     yes  
average_monthly_hours_quali     high  
dtype: object
```

D'après le résultat, le nombre maximum projets affecté est de 7 et le maximum d'heures effectués par mois est de 310h.

Entrée [35]:

```
#Percentile  
#id_colab est insignifiant  
percentiles_all = bidata.quantile([0.25, 0.5, 0.75])  
percentiles_all
```

Out[35]:

	id_colab	number_project	average_monthly_hours	time_spend_company	work_accident	pr
0.25	3750.5	3.0	156.0	3.0	0.0	
0.50	7500.0	4.0	199.0	3.0	0.0	
0.75	11249.0	5.0	245.0	4.0	0.0	

D'après le résultat, 25% des employés ont été affectés à 3 projets et 75% à 5 projets. Concernant les heures effectués par mois 25% des employés travaillent 156h/mois et 75% travaillent 245h/mois.

Voici un tableau regroupant l'ensemble des KPIs ci dessus:

Entrée [36]:

```
kpi = bidata.describe().applymap('{:.2f}'.format)  
kpi  
#On ne tient pas compte de id_colab et left car insignifiant.
```

Out[36]:

	id_colab	number_project	average_monthly_hours	time_spend_company	work_accident
count	10051.00	10051.00	10051.00	10051.00	10051.00
mean	7492.24	3.80	200.81	3.51	0.15
std	4331.36	1.23	49.78	1.48	0.35
min	4.00	2.00	96.00	2.00	0.00
25%	3750.50	3.00	156.00	3.00	0.00
50%	7500.00	4.00	199.00	3.00	0.00
75%	11249.00	5.00	245.00	4.00	0.00
max	14998.00	7.00	310.00	10.00	1.00

Entrée [37]:

```
bidata.describe(include=[object])
```

Out[37]:

	satisfaction_level	last_evaluation	job	salary	left_quanti
count	10051	10051	10051	10051	10051
unique	92	65	10	3	2
top	0,1	0,54	sales	low	no
freq	261	248	2816	4899	7662

F-Représentation graphique de la distribution des données

Entrée [38]:

```
#Rappelons ici à quoi notre dataframe "bidata" ressemble en affichant seulement les 5 premières lignes
```

Out[38]:

	<code>id_colab</code>	<code>satisfaction_level</code>	<code>last_evaluation</code>	<code>number_project</code>	<code>average_montly_hours</code>	<code>time_spe</code>
0	4	0,72	0,87	5	223	
1	5	0,37	0,52	2	159	
2	7	0,1	0,77	6	247	
3	9	0,89	1	5	224	
4	10	0,42	0,53	2	142	

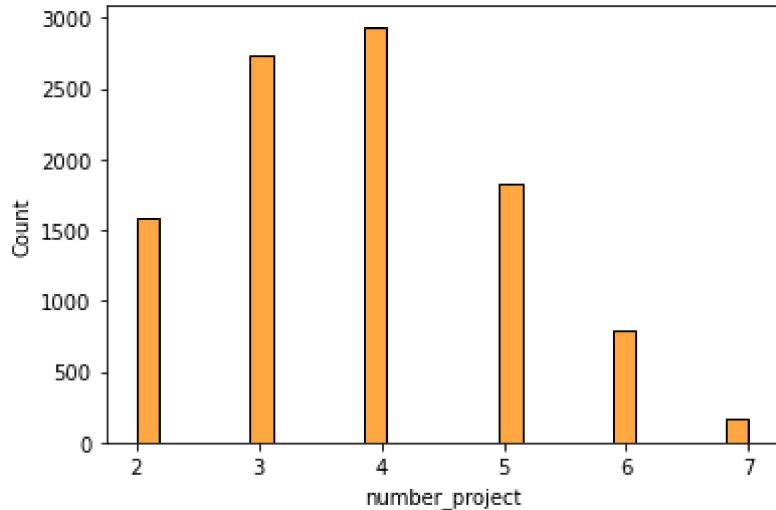
Nous allons maintenant créer graphiquement ces données afin de donner un aperçu plus clair de celles-ci.

Entrée [39]:

```
sns.histplot(bidata["number_project"], color= '#ff8800')
```

Out[39]:

```
<AxesSubplot: xlabel='number_project', ylabel='Count'>
```



Sur ce graphique nous pouvons visualiser le nombre de projets attribués à chaque salarié.

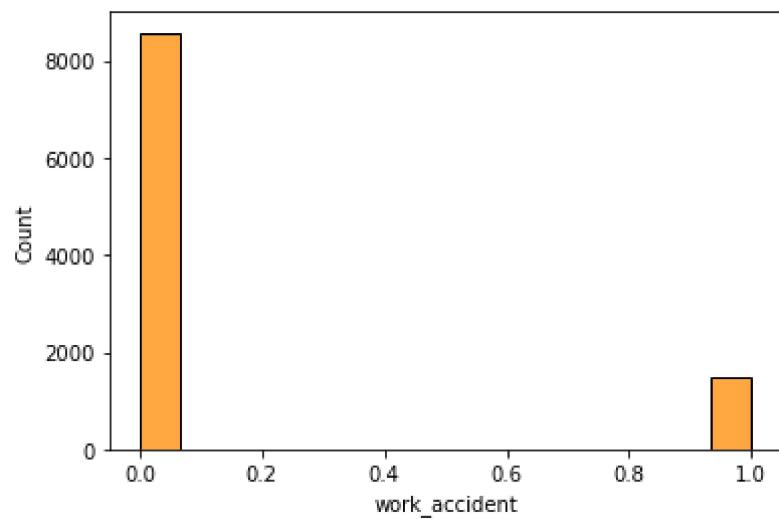
Nous pouvons constater que en général les salariés sont affectés à 4 projets.

Entrée [40]:

```
sns.histplot(bidata["work_accident"],color= '#ff8800')
```

Out[40]:

```
<AxesSubplot:xlabel='work_accident', ylabel='Count'>
```



Sur ce graphique, nous pouvons visualiser le nombre d'accidents de travail sur l'effectif de l'entreprise.

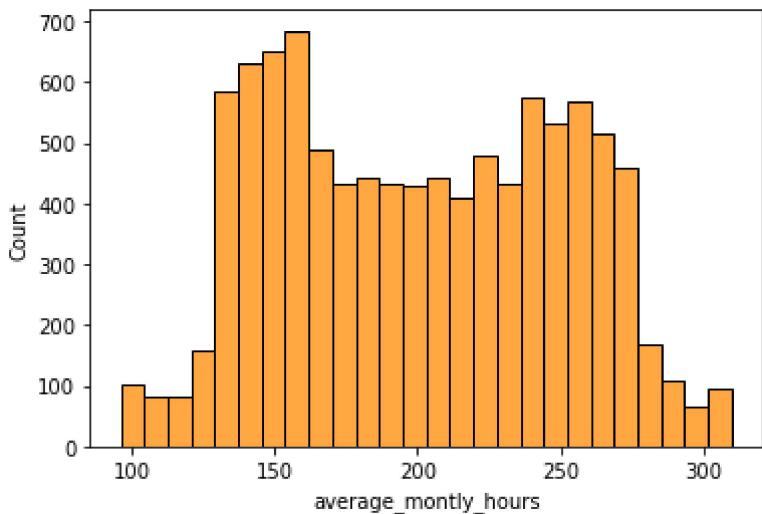
Nous pouvons constater sur ce graphique qu'il y a peu d'accidents de travail

Entrée [41]:

```
sns.histplot(bidata["average_monthly_hours"],color= '#ff8800')
```

Out[41]:

```
<AxesSubplot:xlabel='average_monthly_hours', ylabel='Count'>
```



Sur ce graphique, nous pouvons visualiser le nombre d'heure travaillés moyen par mois des salariés.

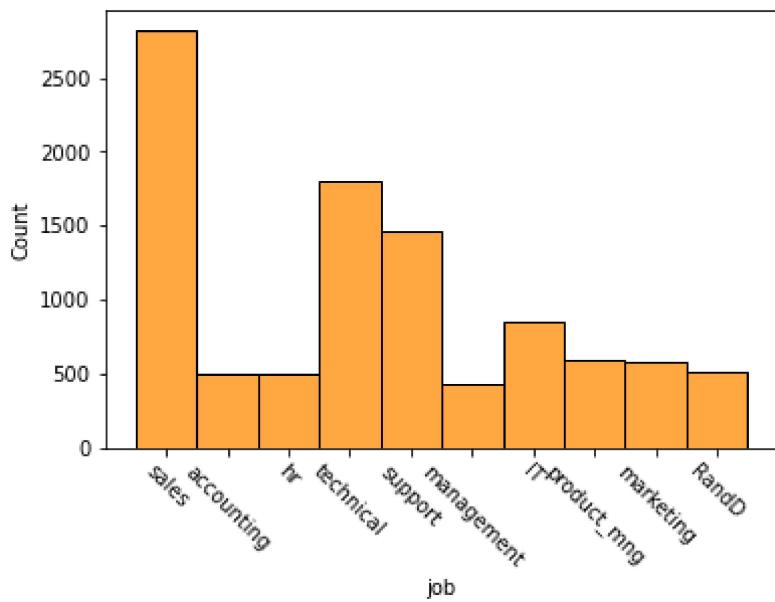
Nous pouvons constater que beaucoup travaillent environ 150h/mois et 250h/mois.

Entrée [42]:

```
sns.histplot(bidata["job"],color= '#ff8800')
plt.xticks(rotation=-45)
```

Out[42]:

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 [Text(0, 0, ''),
  Text(0, 0, '')])
```



Ce graphique nous montre la répartition des métiers dans l'entreprise

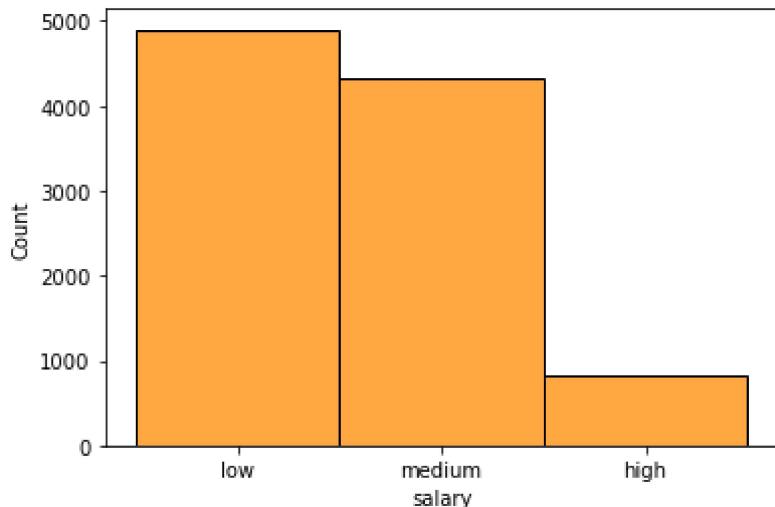
Nous pouvons constater que le domaine des ventes est le plus représenté.

Entrée [43]:

```
sns.histplot(bidata["salary"],color= '#ff8800')
```

Out[43]:

```
<AxesSubplot:xlabel='salary', ylabel='Count'>
```



Ici le graphique nous montre la variation des salaires dans l'entreprise.

Nous pouvons constater que la plupart des salariés ont un salaire bas et moyen et très peu ont un salaire élevé.

Voici un tableau de bord Power bi qui nous montre le profil des salariés de notre entreprise (sans prendre en compte les départs)

Entrée [44]:

```
imdashboardglo = Image.open('Dashboard effectif global.png')
imdashboardglo
```

Out[44]:



Ces graphiques sont un avant goût du prochain chapitre dans lequel nous allons explorer les données grâce à des analyses multivariées.

II- Chapitre 2 : Explorer les données

Dans ce chapitre nous allons comparer les variables entre elles afin de pouvoir interpréter des résultats. Cela va nous permettre de mieux comprendre nos données et de comprendre ce qui pousse les employés à quitter l'entreprise.

A-Comparer des variables

Comparaison de nos variables numériques

Voici un tableau de corrélation de nos variables numériques. Le tableau de corrélation est un outil statistique qui montre la force et la direction de la relation entre deux ou plusieurs variables quantitatives. Il peut être utilisé pour identifier les variables qui sont fortement corrélées et qui peuvent influencer les résultats d'une analyse.

Entrée []:

Voici une représentation graphique via une heatmap de ce tableau de corrélation qui nous permet d'avoir une meilleure visualisation des résultats :

Entrée [45]:

```
#Pour avoir une meilleure visibilité sur la corrélation entre la variable "left" et toutes les autres nous avons décidé de transformer les variables "salary" et "job".
bidata = bidata.assign(salary1=bidata['salary'])
bidata['salary1'] = bidata['salary1'].map({'low' : 1, 'medium' : 2, 'high' : 3 })
bidata = bidata.assign(job1=bidata['job'])
bidata['job1'] = bidata['job1'].map({'sales' : 1, 'technical' : 2, 'support' : 3, 'IT' : 4})
bidata.head()
```

Out[45]:

	id_colab	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spe
0	4	0,72	0,87	5		223
1	5	0,37	0,52	2		159
2	7	0,1	0,77	6		247
3	9	0,89	1	5		224
4	10	0,42	0,53	2		142

Entrée [46]:

```
bidata_numericvalues = bidata[bidata.columns[1:]]  
corr_matrix = bidata_numericvalues.corr(method='spearman')  
corr_matrix
```

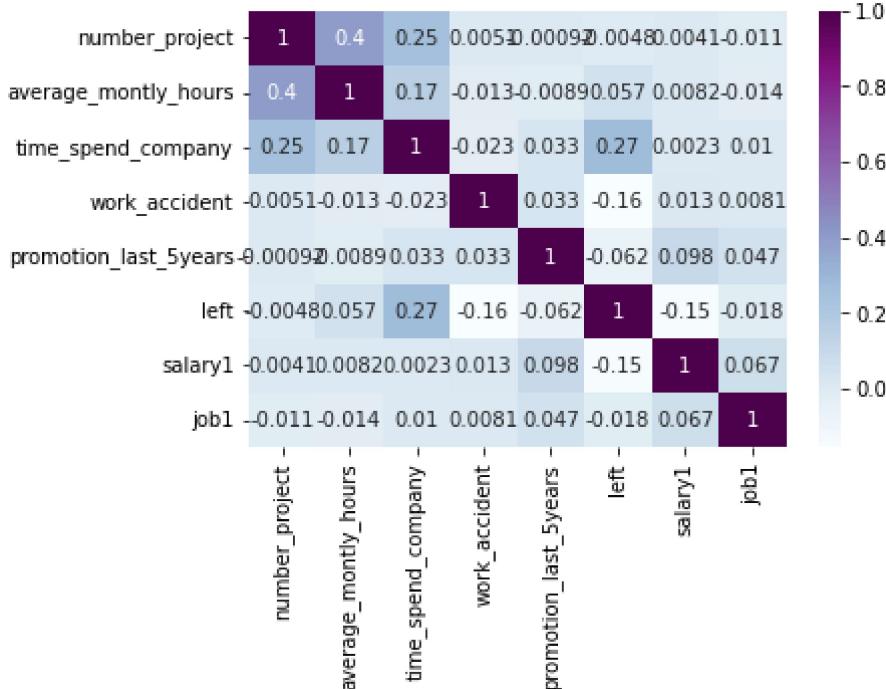
Out[46]:

	number_project	average_monthly_hours	time_spend_company	work_accident	promotion_last_5years	left	salary1	job1
number_project	1.000000		0.400247		0.250878	0.		
average_monthly_hours		0.400247		1.000000		0.168228	-0.	
time_spend_company		0.250878		0.168228		1.000000	-0.	
work_accident		0.005150		-0.013169		-0.022982	1.	
promotion_last_5years		-0.000919		-0.008891		0.033373	0.	
left		-0.004792		0.056638		0.270494	-0.	
salary1		0.004128		0.008204		0.002318	0.	
job1		-0.010735		-0.013683		0.009990	0.	

Entrée [47]:

```
sns.heatmap(corr_matrix, annot=True, cmap='BuPu')  
print(sns.heatmap)
```

<function heatmap at 0x0000022665D85B80>



Interprétation du tableau de corrélation :

Comparaison de nos variables qualitatives et test Chi2

Afin de pouvoir comparer nos variables qualitatives nous allons utiliser la méthode du Chi2. En effet, cette méthode permet d'évaluer si une différence observée entre les fréquences observées et les fréquences théoriques est statistiquement significative.

Le test du chi2 est souvent utilisé dans les sciences sociales et la médecine pour déterminer si une association statistique existe entre deux variables catégoriques, telles que le sexe et une maladie, ou l'âge et un comportement de consommation.

L'intérêt du test du Khi² est de mesurer l'indépendance entre deux variables qualitatives à partir du tableau de contingence.

Pour notre étude, nous allons comparer la variable "left" avec le reste des variables qualitatives.

ces deux liens nous ont aidé pour la compréhension du Test Chi-2

- <https://asardell.github.io/statistique-python/> (<https://asardell.github.io/statistique-python/>)
- <https://www.pythondatascience.org/chi-square-test-of-independence-python/> (<https://www.pythondatascience.org/chi-square-test-of-independence-python/>)

On pose les hypothèses de départ :

- H0 : Variables indépendantes si p-value > 5%
- H1 : Variables non indépendantes si p-value < 5%

Entrée [48]:

```
df_ls = pd.crosstab(bidata['salary'], bidata['left'], normalize='index')
df_ls
```

Out[48]:

salary	left	0	1
high	0.929952	0.070048	
low	0.704634	0.295366	
medium	0.795560	0.204440	

Entrée [49]:

```
#calcule de la contingence
from scipy.stats import chi2_contingency
Khi2_obs, p_value, ddl, effectif_theorique = chi2_contingency(df_ls)
p_value
```

Out[49]:

0.9198829974112606

Ici p-value > 5% donc les variables sont indépendantes.

Entrée [50]:

```
df_lwa = pd.crosstab(bidata['work_accident'], bidata['left'], normalize='index')
df_lwa
```

Out[50]:

left	0	1
work_accident		
0	0.734865	0.265135
1	0.921516	0.078484

Entrée [51]:

```
Khi2_obs, p_value, ddl, effectif_theorique = chi2_contingency(df_lwa)
p_value
```

Out[51]:

1.0

Ici p-value > 5% donc les variables sont indépendantes.

Entrée [52]:

```
df_lp = pd.crosstab(bidata['promotion_last_5years'], bidata['left'], normalize='index')
df_lp
```

Out[52]:

left	0	1
promotion_last_5years		
0	0.758554	0.241446
1	0.945545	0.054455

Entrée [53]:

```
Khi2_obs, p_value, ddl, effectif_theorique = chi2_contingency(df_lp)
p_value
```

Out[53]:

1.0

Ici p-value > 5% donc les variables sont indépendantes.

Entrée [54]:

```
df_lj = pd.crosstab(bidata['job'], bidata['left'], normalize='index')
df_lj
```

Out[54]:

left	0	1
job		
IT	0.773474	0.226526
RandD	0.852652	0.147348
accounting	0.701195	0.298805
hr	0.708502	0.291498
management	0.859122	0.140878
marketing	0.766724	0.233276
product_mng	0.781570	0.218430
sales	0.758878	0.241122
support	0.746594	0.253406
technical	0.750553	0.249447

Entrée [55]:

```
Khi2_obs, p_value, ddl, effectif_theorique = chi2_contingency(df_lj)
p_value
```

Out[55]:

0.9999998914774012

Ici p-value > 5% donc les variables sont idépendantes.

Comparaison de nos variables quantitatives et qualitatives avec la méthode ANOVA

Afin de comparer des variables quantitatives et qualitatives entre elles nous allons utiliser la méthode ANOVA. Cette méthode sert à évaluer l'effet d'une ou plusieurs variables explicatives sur une variable réponse. Elle permet de déterminer si les différences observées dans la variable réponse peuvent être attribuées à des différences dans les valeurs des variables explicatives ou sont le résultat d'une variation aléatoire. Cette méthode est souvent utilisée en analyse statistique pour des expériences scientifiques, en marketing pour des tests d'hypothèses sur les produits ou les segments de marché, et en économie pour des études de marché.

Comment interpréter un tableau ANOVA :

Le tableau ANOVA est le résultat final d'une succession de formules de calcul complexes. Il présente trois types de données numériques exploitables :

Les degrés de liberté ou ddl.

Le résultat noté F.

La signification notée p : cette valeur, obtenue grâce aux données ddl et F, constitue le rapport de variance qui confirme ou qui infirme l'hypothèse testée. Si la valeur de p est inférieure à 0,05, l'hypothèse nulle, selon laquelle les moyennes sont égales, peut être vraisemblablement rejetée. C'est-à-dire que les variables qualitatives ont un effet significatif sur la variable quantitative : une moyenne au moins se distingue dans une large mesure au sein de l'échantillon.

Nous avons ici pris des variables qu'il nous semblais logique d'étudier ensembles :

Entrée [56] :

```
#Tableau méthode anova en fonction de satisfaction_level en corrélation avec job et salary  
imganova1 = Image.open('anova.png')  
imganova1
```

Out[56] :

ANOVA - satisfaction_level

	Somme des carrés	ddl	Carrés moyens	F	p
job	0.841	9	0.0935	1.51	0.136
salary	0.427	2	0.2134	3.46	0.032
job * salary	1.692	18	0.0940	1.52	0.072
Résidus	618.647	10021	0.0617		

Ici nous pouvons constater, que p est supérieur à 0.05 ce qui signifie que le métiers ainsi que le salaire qui lui est attribués à une importance sur la satisfaction des employés.

Entrée [57] :

```
#Tableau méthode anova en fonction de average_monthly_hours en corrélation avec job et sala  
imganova2 = Image.open('anova2.png')  
imganova2
```

Out[57] :

ANOVA

ANOVA - average_monthly_hours

	Somme des carrés	ddl	Carrés moyens	F	p
job	39279.6	9	4364.4	1.7626	0.070
salary	50.2	2	25.1	0.0101	0.990
job * salary	62519.3	18	3473.3	1.4027	0.119
Résidus	2.48e+7	10021	2476.2		

[3]

Ici nous pouvons constater, que p est largement supérieur à 0.05 nous pouvons donc déduire que le métier et le salaire qui lui est attribué à un impacte sur le temp travaillé dans l'entreprise par mois. On constate également que le salaire impacte plus que le métiers en lui même.

Entrée [58]:

```
#Tableau méthode anova en fonction de last_evaluation en corrélation avec job et work accident
imganova3 = Image.open('anova3.png')
imganova3
```

Out[58]:

ANOVA

ANOVA - last_evaluation

	Somme des carrés	ddl	Carrés moyens	F	p
work_accident	0.0553	1	0.0553	1.879	0.170
job	0.3020	9	0.0336	1.140	0.330
work_accident * job	0.1846	9	0.0205	0.697	0.712
Résidus	295.3070	10031	0.0294		

[3]

Ici nous pouvons constater, que p est également supérieur à 0.05 et que par conclusion les accidents de travail en fonction des métiers ont impacté la dernière évaluation de satisfaction dans l'entreprise

B-Représenter graphiquement les relations entre les variables

Dans cette partie, nous allons representer graphiquement les relations entre les variables que nous jugeons pertinentes pour notre analyse.

Entrée [59]:

```
#Rappelons ici à quoi notre dataframe "bidata" ressemble en affichant seulement les 5 premières lignes
bidata.head()
```

Out[59]:

	id_colab	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company
0	4	0,72	0,87	5	223	
1	5	0,37	0,52	2	159	
2	7	0,1	0,77	6	247	
3	9	0,89	1	5	224	
4	10	0,42	0,53	2	142	

◀ ▶

Entrée [60]:

```
#tableau de contingence entre "number_project" et "time_spend_company"  
df_nt= pd.crosstab(bidata.number_project, bidata.time_spend_company)  
df_nt
```

Out[60]:

time_spend_company	2	3	4	5	6	7	8	10
number_project								
2	153	1223	87	56	36	10	9	8
3	860	1179	360	92	97	35	49	67
4	753	1230	380	296	151	43	32	53
5	362	585	277	393	148	25	22	18
6	45	89	463	119	51	9	5	9
7	1	5	146	20	0	0	0	0

Entrée []:

Entrée [61]:

```
#tableau de contingence entre "salary" et "time_spend_company"  
df_st= pd.crosstab(bidata.salary, bidata.time_spend_company)  
df_st
```

Out[61]:

time_spend_company	2	3	4	5	6	7	8	10
salary								
high	200	350	123	43	27	24	13	48
low	1008	2169	876	516	230	21	42	37
medium	966	1792	714	417	226	77	62	70

Entrée [62]:

```
#tableau de contingence entre "satisfaction_level" et "time_spend_company"
df_slt= pd.crosstab(bidata.satisfaction_level, bidata.time_spend_company)
df_slt
```

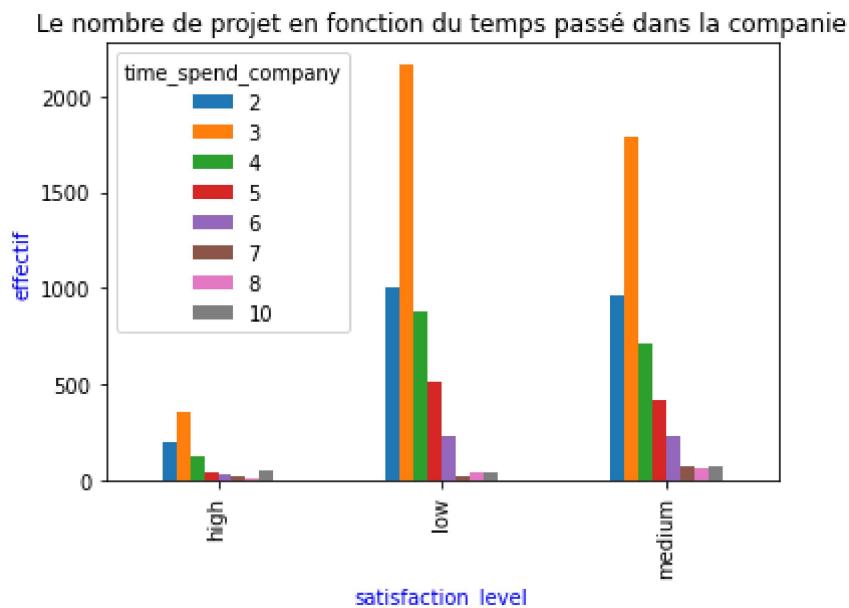
Out[62]:

time_spend_company	2	3	4	5	6	7	8	10
satisfaction_level								
0,09	1	0	111	14	0	0	0	0
0,1	0	1	236	24	0	0	0	0
0,11	0	2	202	18	0	0	0	0
0,12	2	4	7	4	3	0	0	0
0,13	3	2	8	13	5	0	0	0
...
0,96	42	55	16	6	6	3	2	3
0,97	47	44	11	2	6	2	4	2
0,98	37	59	18	1	7	0	0	6
0,99	43	48	21	2	3	0	2	3
1	14	40	17	3	1	2	0	0

92 rows × 8 columns

Entrée [63]:

```
df_st.plot(kind='bar')
plt.title("Le nombre de projet en fonction du temps passé dans la compagnie")
plt.xlabel('satisfaction_level', color = 'blue')
plt.ylabel('effectif', color = 'blue')
plt.show()
```



Entrée [64]:

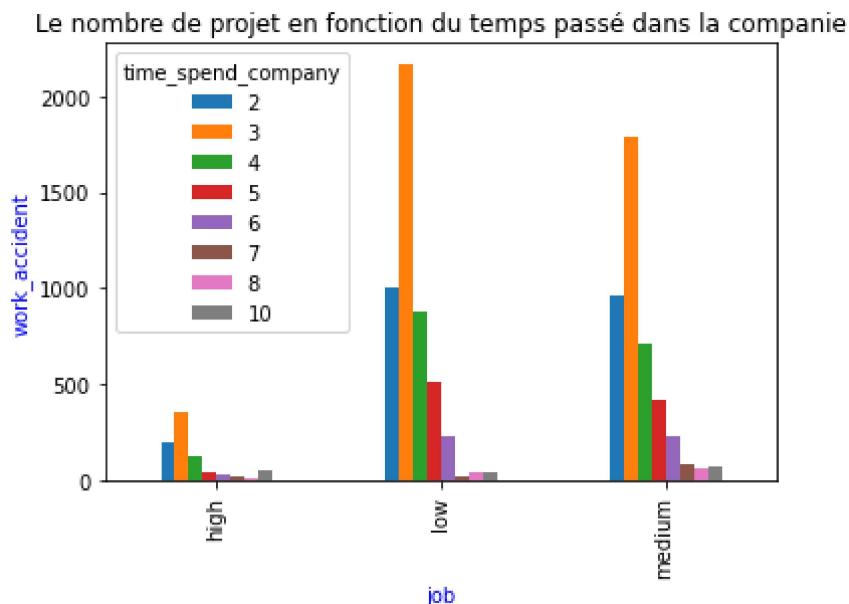
```
#tableau de contingence entre "work_accident" et "job"
df_wj= pd.crosstab(bidata.work_accident, bidata.job)
df_wj
```

Out[64]:

	job	IT	RandD	accounting	hr	management	marketing	product_mng	sales	su		
work_accident	0	741	422		441	426		359	494		497	2418
	1	111	87		61	68		74	89		89	398

Entrée [65]:

```
df_st.plot(kind='bar')
plt.title("Le nombre de projet en fonction du temps passé dans la compagnie")
plt.xlabel('job', color = 'blue')
plt.ylabel('work_accident', color = 'blue')
plt.show()
```



Entrée [66]:

```
#tableau de contingence entre "promotion_Last_5years" et "time_spend_company"
df_pt= pd.crosstab(bidata.promotion_last_5years, bidata.time_spend_company)
df_pt
```

Out[66]:

time_spend_company	2	3	4	5	6	7	8	10	
promotion_last_5years	0	2139	4225	1694	968	472	100	109	142
	1	35	86	19	8	11	22	8	13

Entrée []:

Entrée [67]:

```
#tableau de contingence entre "salary" et "job"
df_sj= pd.crosstab(bidata.salary, bidata.job)
df_sj
```

Out[67]:

job	IT	RandD	accounting	hr	management	marketing	product_mng	sales	support
salary									
high	60	36		46	28		158	53	
low	423	243		241	225		122	279	
medium	369	230		215	241		153	251	



Entrée []:

III-Chapitre 3 : Modélisation des données

Maintenant que l'interprétation de nos données est plus claire nous allons les utiliser afin de répondre à la problématique : "Pourquoi les employés quittent-ils l'entreprise ?"

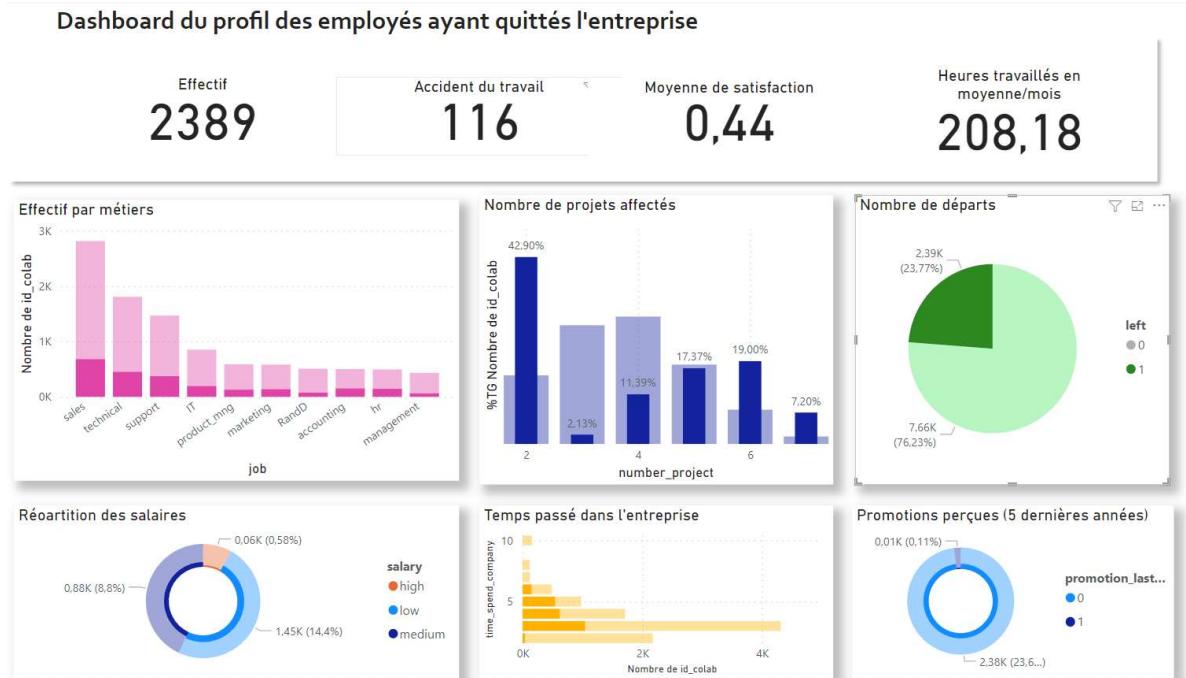
A-Quel est le profil des collaborateurs qui vont quitter l'entreprise ?

Afin de déterminer le profil des employés qui partent de l'entreprise, nous allons créer un tableau de bord sur Power BI avec toutes les variables que nous allons comparer à la variable "left". Voici le tableau de bord Power BI que nous obtenons :

Entrée [68]:

```
imdashboardglo = Image.open('Dashboard effectif left.png')  
imdashboardglo
```

Out[68]:



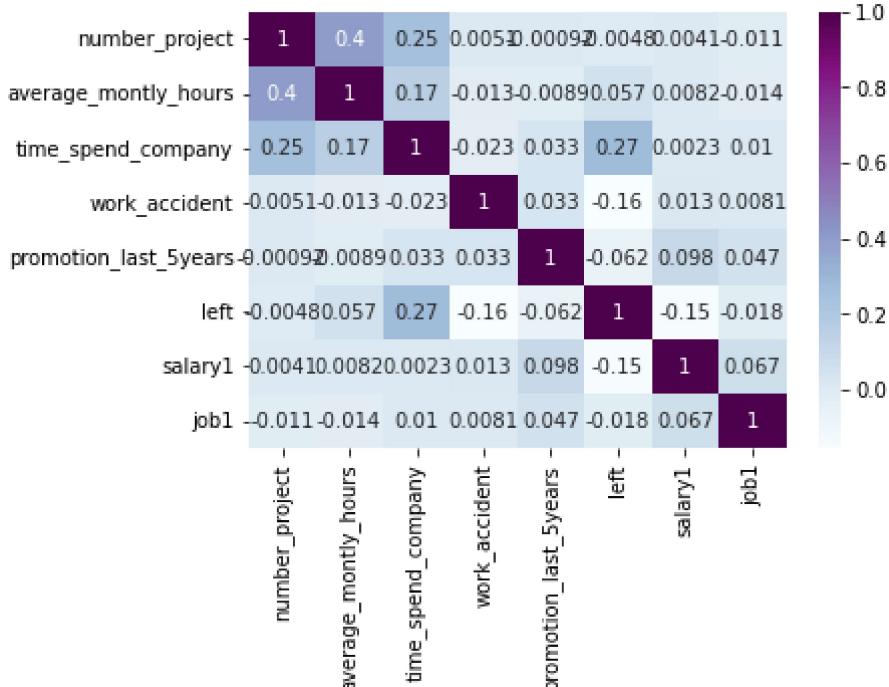
Afin de rendre le profil des employés quittant l'entreprise plus claire nous avons réalisé un persona sur canva, le voici :

Ensuite pour classer des différents critères qui jouent sur le départ des collaborateurs, nous avons repris la heatmap créée un peu plus haut.

Entrée [69]:

```
sns.heatmap(corr_matrix, annot=True, cmap='BuPu')
print(sns.heatmap)
```

```
<function heatmap at 0x0000022665D85B80>
```



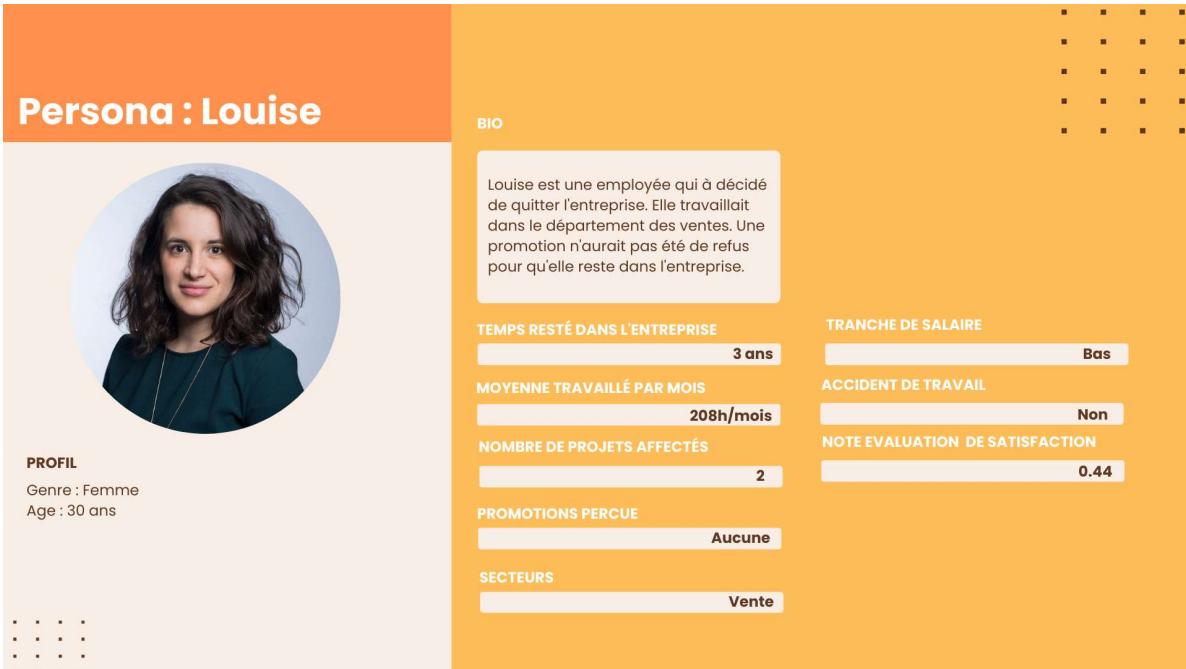
Maintenant nous pouvons faire notre classement :

- time_spend_company (0.27)
- average_monthly_hours
- number_project
- promotion_last_5years
- job
- salary
- work_accident (-0,16)

Entrée [70]:

```
persona = Image.open('persona.png')
persona
```

Out[70]:



B- Quelle est la différence entre une régression linéaire et logistique

La régression logistique et la régression linéaire sont deux méthodes différentes de modélisation statistique utilisées pour prédire des valeurs continues ou discrètes.

La régression linéaire est une méthode de prédiction qui suppose une relation linéaire entre les variables indépendantes et la variable dépendante. Elle est utilisée pour prédire une variable continue en utilisant une fonction linéaire.

La régression logistique, d'autre part, est une méthode de classification utilisée pour prédire une variable dépendante binaire (par exemple, oui/non ou vrai/faux). La régression logistique utilise une fonction logistique pour modéliser la probabilité d'appartenance à une classe donnée. Cette fonction logistique transforme les prédictions continues de la régression linéaire en une probabilité comprise entre 0 et 1.

En résumé, la régression linéaire est utilisée pour les prédictions continues, tandis que la régression logistique est utilisée pour les prédictions de variables binaires.

C-Pourquoi les collaborateurs quittent l'entreprise ?

Il est maintenant essentiel de déterminer qu'elles sont les variables qui sont le plus liés aux départs des salariés, afin de définir une stratégie pour les garder. Nous allons ici utiliser une fonction de régression logistique.

Nous allons donc l'utiliser dans notre cas afin de hiérarchiser nos variables et savoir lesquelles sont les plus amenées à être les causes des départs.

Voici la régrression logistique de notre dataframe :

Pour pouvoir faire notre regression logistique, nous devons d'abord importer les packages que nous allons utiliser, puis définir les features et les cibles

Entrée [71]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
```

Entrée [72]:

```
newdata = bidata[bidata.columns[1:-4]]
newdata.head()
```

Out[72]:

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_compa
0	0,72	0,87	5	223	
1	0,37	0,52	2	159	
2	0,1	0,77	6	247	
3	0,89	1	5	224	
4	0,42	0,53	2	142	

Entrée [73]:

```
#Pour que cela marche, nous devons changer les "," en "."
newdata["satisfaction_level"] = newdata["satisfaction_level"].str.replace(",",".")
newdata["satisfaction_level"] = newdata["satisfaction_level"].astype(float)

newdata["last_evaluation"]      = newdata["last_evaluation"].str.replace(",",".")
newdata["last_evaluation"]      = newdata["last_evaluation"].astype(float)
```

C:\Users\Emma\AppData\Local\Temp\ipykernel_23652\2208048325.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
newdata["satisfaction_level"] = newdata["satisfaction_level"].str.replace(",",".")
```

C:\Users\Emma\AppData\Local\Temp\ipykernel_23652\2208048325.py:3: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
newdata["satisfaction_level"] = newdata["satisfaction_level"].astype(float)
```

C:\Users\Emma\AppData\Local\Temp\ipykernel_23652\2208048325.py:5: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
newdata["last_evaluation"]      = newdata["last_evaluation"].str.replace(",",".")
```

C:\Users\Emma\AppData\Local\Temp\ipykernel_23652\2208048325.py:6: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
newdata["last_evaluation"]      = newdata["last_evaluation"].astype(float)
```

Entrée [74]:

```
newdata.head()
```

Out[74]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_compa
0	0.72	0.87	5	223	
1	0.37	0.52	2	159	
2	0.10	0.77	6	247	
3	0.89	1.00	5	224	
4	0.42	0.53	2	142	

Entrée [75]:

```
#nos features seront les X, et notre cible est y = "left" car c'est la variable à expliquer
X = newdata[["satisfaction_level","last_evaluation","number_project","average_montly_hours"]
y = newdata.left
```

Une fois cela fait, nous pouvons faire notre regression logistique (Pour défnir les paramètres, nous nous sommes aidés de :

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html))

Entrée [76]:

```
logistic = LogisticRegression()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0
#Définie les hyperparamètres à régler
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
              'penalty': ['l1', 'l2'],
              'max_iter' : [100, 200, 500, 1000],
              'multi_class' : ['auto', 'ovr', 'multinomial']}

#Créer l'objet de recherche de grille.
grid_search = GridSearchCV(logistic, param_grid)

#Ajuster la recherche de grille aux données.
grid_search.fit(X_train, y_train)

# Print le meilleur paramètre et le meilleur score
print("Best parameters: ", grid_search.best_params_)
print("Best score: ", grid_search.best_score_)
```

C:\Users\Emma\anaconda3\lib\site-packages\sklearn\linear_model_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html> (<https://scikit-learn.org/stable/modules/preprocessing.html>)
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
n_iter_i = _check_optimize_result(
C:\Users\Emma\anaconda3\lib\site-packages\sklearn\linear_model_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

Entrée [77]:

```
best_settings = grid_search.best_params_
print(best_settings)

{'C': 0.1, 'max_iter': 100, 'multi_class': 'auto', 'penalty': 'l2'}
```

Entrée [78]:

```
## Créer un modèle de régression logistique et entraîner le modèle sur les données d'entraînement
logistic = LogisticRegression(C = best_settings["C"], penalty = best_settings["penalty"], max_iter=1000)
logistic.fit(X_train, y_train)
```

```
C:\Users\Emma\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1): STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (`max_iter`) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html> (<https://scikit-learn.org/stable/modules/preprocessing.html>)

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

```
n_iter_i = _check_optimize_result()
```

Out[78]:

```
LogisticRegression(C=0.1)
```

Entrée [79]:

```
# Afficher les coefficients
coef = logistic.coef_
print(coef)
```

```
[[ -3.49593656  0.45895285 -0.25148551  0.00485783  0.20834317 -1.23029887
 -0.93407786]]
```

Entrée [80]:

```
preds = logistic.predict(X_test)
accuracy_score(preds, y_test)
```

Out[80]:

```
0.7707608155146694
```

Le score de précision de 0,77 dans la régression logistique est une mesure de la qualité des prédictions faites par le modèle. Il représente le pourcentage de prédictions correctes par rapport au nombre total de prédictions. Plus le score de précision est élevé, plus les prédictions du modèle sont précises. Une valeur de 0,77 signifie que 77% des prédictions faites par notre modèle sont correctes, ce qui peut être considéré comme une précision relativement élevée.

Cependant, il est important de noter que le score de précision n'est qu'un indicateur de la qualité du modèle et ne donne pas une vue complète de son ajustement aux données. Il peut être nécessaire de considérer d'autres mesures telles que le score R² ou les erreurs type pour évaluer complètement le modèle.

D-Estimer la validité de votre modèle (R², p-value,...)

Il existe plusieurs mesures pour évaluer la validité d'un modèle de régression. Certaines des mesures les plus couramment utilisées comprennent :

R^2 (coefficient de détermination) : Cette mesure quantifie la proportion de la variance totale de la variable dépendante qui est expliquée par le modèle. Plus R^2 est proche de 1, plus le modèle explique bien les variations de la variable dépendante.

Erreur quadratique moyenne (MSE) : Cette mesure quantifie la moyenne des erreurs de prédition pour chaque point de données. Plus MSE est proche de zéro, plus le modèle est précis.

Test statistique F : Ce test statistique vérifie si les coefficients de régression sont significativement différents de zéro. Si le test est statistiquement significatif, cela signifie que le modèle a un effet significatif sur la variable dépendante.

Valeur p : Ce terme fait référence à la significativité statistique de chaque coefficient de régression. Si la valeur p est inférieure à un niveau de significativité (généralement 0,05), nous pouvons rejeter l'hypothèse nulle selon laquelle le coefficient est égal à zéro.

Courbe ROC (Receiver Operating Characteristic) : Cette courbe est utilisée pour évaluer les modèles de classification binaire (tels que la régression logistique). Elle quantifie la relation entre le taux de vrais positifs et le taux de faux positifs pour différents seuils de décision.

Il est important de noter que ces mesures ne peuvent pas être utilisées seules pour évaluer la validité d'un modèle. Il est souvent nécessaire de considérer un certain nombre de mesures pour obtenir une image complète de la qualité du modèle.

Le R^2 (coefficient de détermination) est un indicateur statistique utilisé pour mesurer la qualité de l'ajustement d'un modèle de régression à des données. Il représente la proportion de la variation totale de la variable cible qui peut être expliquée par les prédictions du modèle. Plus précisément, le R^2 varie entre 0 et 1, où un score de 1 indique un ajustement parfait et un score de 0 indique que le modèle ne peut pas expliquer la variation de la variable cible.

En général, plus le score R^2 est élevé, meilleur est l'ajustement du modèle aux données. Toutefois, un score R^2 élevé ne garantit pas nécessairement que le modèle est généralisable à de nouvelles données ou qu'il prévoit avec une grande précision.

Il se peut que le score R^2 soit négatif. Cela indique que le modèle de régression est en fait moins performant que les prédictions basées sur la moyenne simple de la variable cible. En d'autres termes, il signifie que les prédictions du modèle sont moins précises que les prédictions basées sur la moyenne de la variable cible.

Entrée [81] :

```
#calcule de la mesure R2 score, ou R^2
from sklearn.metrics import r2_score
```

Entrée [82] :

```
r2_score(y_test, preds)
```

Out[82] :

```
-0.25793578641424864
```

Le score de précision de 0,77 est une mesure de la proportion de prédictions correctes faites par notre modèle de régression logistique. Une valeur de 0,77 signifie que 77% des prédictions faites par notre modèle sont correctes, ce qui peut être considéré comme une précision relativement élevée.

Cependant, le score R² de -0,25 est une mesure de l'ajustement de votre modèle de régression logistique. Un score R² négatif indique que le modèle est peu adapté aux données et est pire que de simplement prédire la valeur moyenne de la variable cible.

Ainsi, bien que le score de précision de 0,77 puisse indiquer que le modèle fait des prédictions relativement précises, le score R² négatif suggère que le modèle de régression logistique peut ne pas être le meilleur modèle pour ajuster les données. Nous pouvons donc considérer d'essayer d'autres modèles ou d'ajuster l'ensemble de caractéristiques pour voir si nous pouvons améliorer le score R² et obtenir un meilleur ajustement global du modèle aux données.

Entrée []: