PTS : Regression Linéaire

Sommaire

Partie 1 : Nettoyage et quantification du dataset

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re # Module pour les expression régulières
```

Ouverture

In [ ]:
```
laptop_data = pd.read_csv('laptops_final.csv', sep=",", encoding='latin1')
```

In [ ]:
```
laptop_data.head()
```

Out[ ]:

	Manufacturer	Model Name	Category	ScreenSize	Screen	CPU	RAM	SSD	HI
0	Apple	Macbook Air	Ultrabook	13.3"	1440x900	Intel Core i5 1.8GHz	8GB	0	
1	Apple	MacBook Air	Ultrabook	13.3"	1440x900	Intel Core i5 1.6GHz	8GB	0	
2	Acer	TravelMate B	Notebook	11.6"	1366x768	Intel Pentium Quad Core N3710 1.6GHz	4GB	0	
3	Acer	Swift SF114-31-P5HY	Notebook	14.0"	1366x768	Intel Pentium Quad Core N3710 1.6GHz	4GB	0	
4	Asus	R558UA-DM966T (i5-7200U/8GB/128GB/FHD/W10)	Notebook	15.6"	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	0	1

In [ ]:
```
laptop_data.shape
```

Out[ ]: (1303, 16)

In [ ]:
```
laptop_data.dtypes
```

Out[ ]:
```
Manufacturer                 object
Model Name                   object
Category                     object
ScreenSize                   object
Screen                       object
CPU                          object
RAM                          object
SSD                           int64
HDD                           int64
FlashStorage                  int64
Hybrid                        int64
GPU                          object
Operating System             object
Operating System Version     object
Weight                       object
Price (Euros)               float64
dtype: object
```

In [ ]:
```
laptop_data.loc[0]
```

Out[ ]:
```
Manufacturer                                    Apple
Model Name                                Macbook Air
Category                                    Ultrabook
ScreenSize                                      13.3"
Screen                                       1440x900
CPU                              Intel Core i5 1.8GHz
RAM                                               8GB
SSD                                                 0
HDD                                                 0
FlashStorage                                      128
Hybrid                                              0
GPU                            Intel HD Graphics 6000
Operating System                                macOS
Operating System Version                          NaN
Weight                                         1.34kg
Price (Euros)                                  898.94
Name: 0, dtype: object
```

Suppression des colones inexploitables

In [ ]:
```
#Suppression des colonnes inexploitables
laptop_data = laptop_data.drop('Model Name', axis=1)
laptop_data = laptop_data.drop('Operating System Version', axis=1)
laptop_data = laptop_data.drop('Operating System', axis=1)

laptop_data['Price (Euros)'] = laptop_data['Price (Euros)'].astype(str) # Transform
laptop_data['Price (Euros)'] = laptop_data['Price (Euros)'].str.replace(',', '.').a
```

Quantification du dataset

In [ ]:
```
laptop_data_quantified = laptop_data.copy()
```

On attribue un nombre à chaque ligne de certaines colonnes

In [ ]:
```
print((laptop_data_quantified['Category'].unique()))

laptop_data_quantified.Category[laptop_data_quantified.Category == 'Ultrabook'] = 1
laptop_data_quantified.Category[laptop_data_quantified.Category == 'Notebook'] = 2
laptop_data_quantified.Category[laptop_data_quantified.Category == 'Netbook'] = 3
laptop_data_quantified.Category[laptop_data_quantified.Category == 'Gaming'] = 4
laptop_data_quantified.Category[laptop_data_quantified.Category == '2 in 1 Converti
laptop_data_quantified.Category[laptop_data_quantified.Category == 'Workstation'] =

print((laptop_data_quantified['Manufacturer'].unique()))

laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Apple']
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'HP'] = 2
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Acer'] =
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Asus'] =
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Dell'] =
```

```
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Lenovo']
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Chuwi']
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'MSI'] =
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Microsof
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Toshiba'
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Huawei']
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Xiaomi']
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Vero'] =
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Razer']
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Mediacom
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Samsung'
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Google']
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Fujitsu'
laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'LG'] = 1
```

```
['Ultrabook' 'Notebook' 'Netbook' '2 in 1 Convertible' 'Gaming'
 'Workstation']
['Apple' 'Acer' 'Asus' 'Dell' 'Lenovo' 'HP' 'Microsoft' 'Toshiba' 'Google'
 'MSI' 'Samsung' 'Fujitsu' 'Razer' 'Huawei' 'Xiaomi' 'Vero' 'Mediacom'
 'LG' 'Chuwi']
```

```
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:3: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Category[laptop_data_quantified.Category == 'Ultrabook'] =
1
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:4: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Category[laptop_data_quantified.Category == 'Notebook'] = 2
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:5: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Category[laptop_data_quantified.Category == 'Netbook'] = 3
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:6: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Category[laptop_data_quantified.Category == 'Gaming'] = 4
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:7: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Category[laptop_data_quantified.Category == '2 in 1 Convert
ible'] = 5
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:8: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Category[laptop_data_quantified.Category == 'Workstation']
= 6
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:12: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Apple']
= 1
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:13: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'HP'] =
2
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:14: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Acer']
= 3
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:15: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Asus']
= 4
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:16: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Dell']
= 5
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:17: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Lenovo
'] = 6
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:18: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Chuwi']
= 7
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:19: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'MSI'] =
8
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:20: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Microso
ft'] = 9
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:21: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Toshiba
'] = 10
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:22: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Huawei
'] = 11
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:23: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Xiaomi
'] = 12
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:24: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Vero']
= 13
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:25: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Razer']
= 14
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:26: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Mediaco
m'] = 15
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:27: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Samsung
'] = 16
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:28: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Google
'] = 17
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:29: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'Fujitsu
'] = 18
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1188909230.py:30: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  laptop_data_quantified.Manufacturer[laptop_data_quantified.Manufacturer== 'LG'] =
19
```

Méthode moins laborieuse pour la suite de la quantification (plus de 100 CPU et GPU différents)

In [ ]:

```python
# En gros on met la lists de tous les CPU et les GPU dans une variable et on crée u
# Ensuite, la boucle va remplacer chaque CPU et GPU par un nombre dans le dictionna
# Après on remplace les colonnes CPU et GPU dans notre dataset par leur nouvelle va

gpu_mapping = {}
gpu = laptop_data_quantified['GPU'].unique()
cpu_mapping = {}
cpu = laptop_data_quantified['CPU'].unique()

for i, gpu in enumerate(gpu, start=1):
    #print(gpu, "=", i) # Pour le test : afficher quel GPU correspond à quel nombre
    gpu_mapping[gpu] = i

for i, cpu in enumerate(cpu, start=1):
    #print(cpu, "=", i)
    cpu_mapping[cpu] = i

laptop_data_quantified['GPU'] = laptop_data_quantified['GPU'].map(gpu_mapping)

laptop_data_quantified['CPU'] = laptop_data_quantified['CPU'].map(cpu_mapping)
```

Transformation de certaines colones

In [ ]:
```
#On supprime les unités à la fin de la colone Weight, Screen Size et RAM
laptop_data_quantified['Weight'] = laptop_data_quantified['Weight'].str.replace('kg
laptop_data_quantified['ScreenSize'] = laptop_data_quantified['ScreenSize'].str.rep
laptop_data_quantified['RAM'] = laptop_data_quantified['RAM'].str.replace('GB', '')

#Pour la colone ScreenSize, on supprime tout ce qu'il y a avant la taille avec re (
laptop_data_quantified['Screen'] = laptop_data_quantified['Screen'].apply(lambda x:
#Ensuite on sépare la première taille de la deuxième
laptop_data_quantified['Screen'] = laptop_data_quantified['Screen'].str.split('x',
#Ensuite on fait la multiplication afin d'avoir un seul chiffre
laptop_data_quantified['Screen'] = laptop_data_quantified['Screen'].apply(lambda x:
```

In [ ]:
```
laptop_data_quantified.head()
```

Out[ ]:

	Manufacturer	Category	ScreenSize	Screen	CPU	RAM	SSD	HDD	FlashStorage	Hy
0	1	1	13.3	1296000	1	8.0	0	0	128	
1	1	1	13.3	1296000	2	8.0	0	0	128	
2	3	2	11.6	1049088	3	4.0	0	0	128	
3	3	2	14.0	1049088	3	4.0	0	0	128	
4	4	2	15.6	2073600	4	8.0	0	128	0	

Partie 2 : Analyse exploratoire de données

In [ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

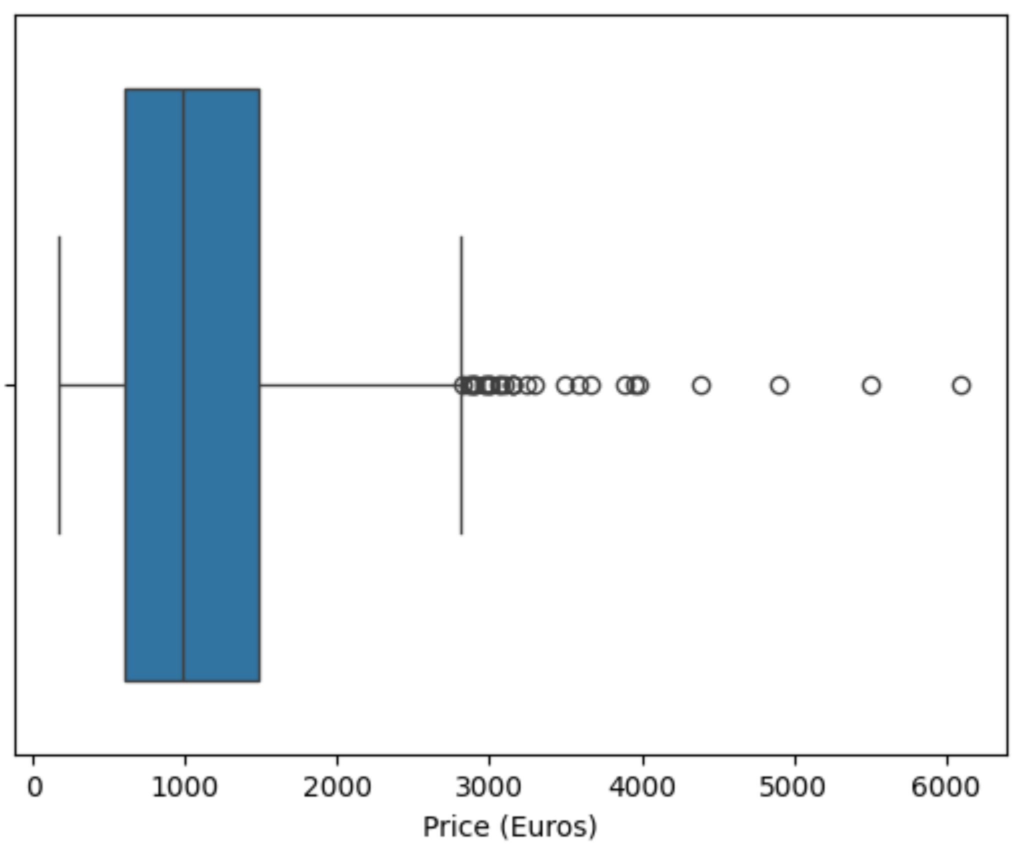
In [ ]:
```
laptop_data_quantified.describe()
```

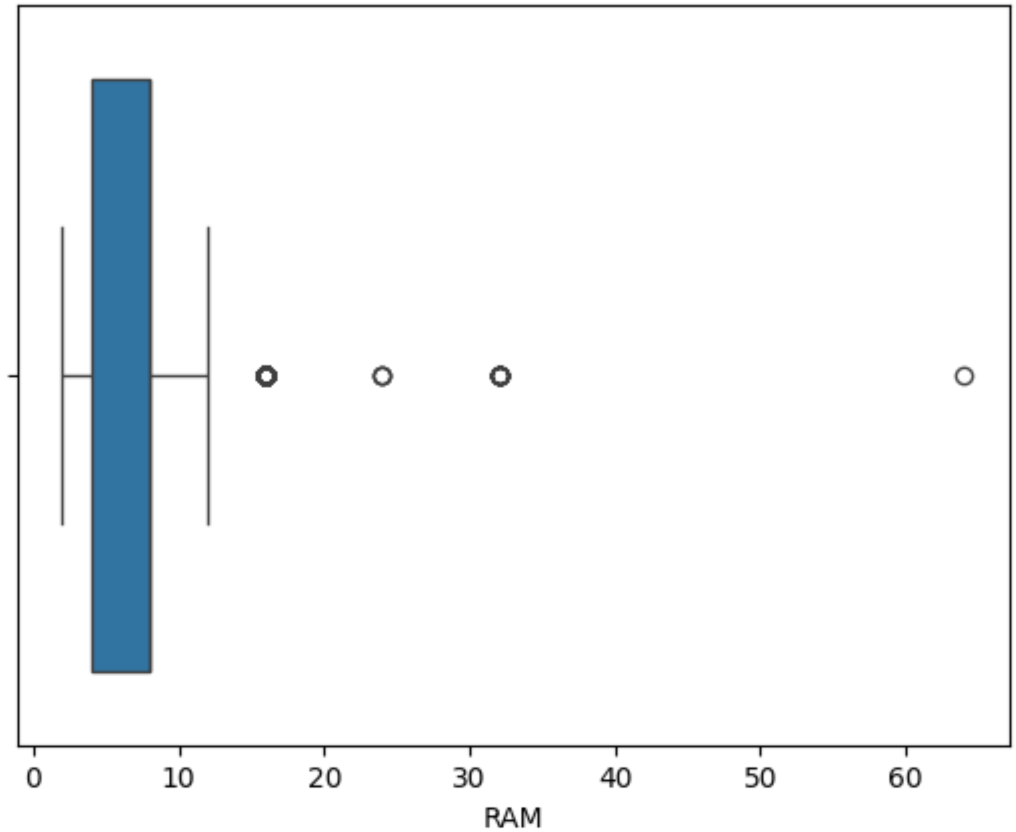
Out[ ]:

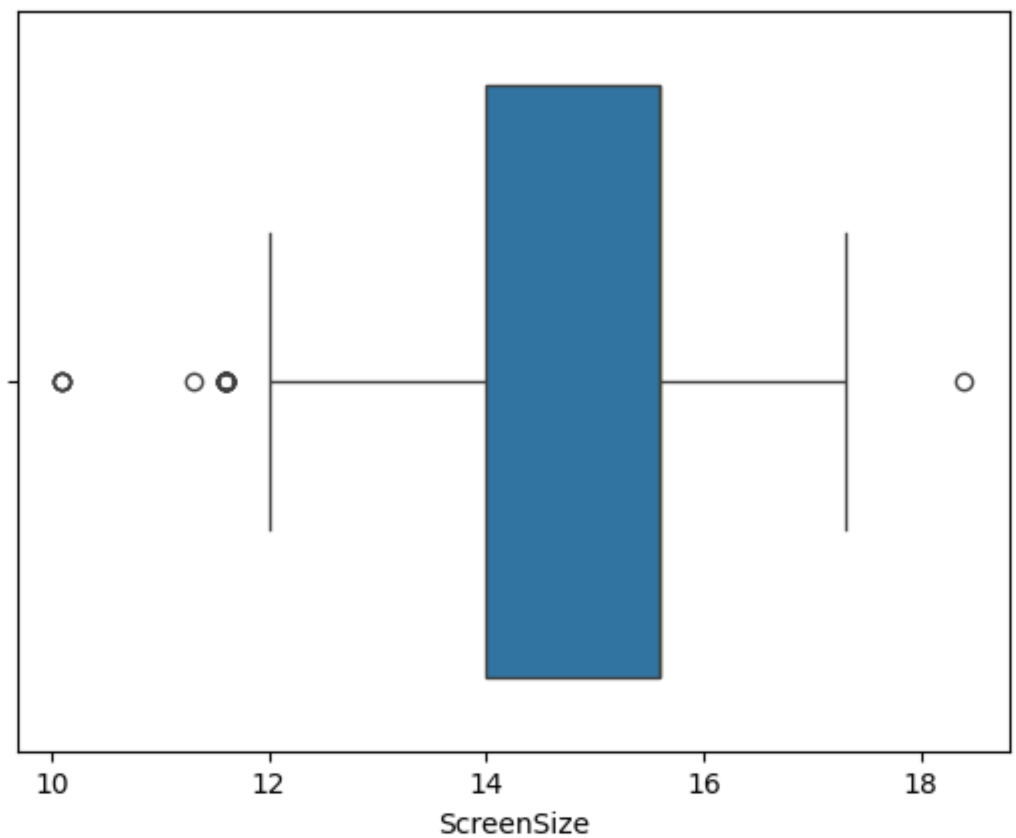
	ScreenSize	Screen	CPU	RAM	SSD	HDD	Flas
count	1303.000000	1.303000e+03	1303.000000	1303.000000	1303.000000	1303.000000	13
mean	15.017191	2.168807e+06	28.185725	8.382195	183.046048	413.783576	
std	1.426304	1.391292e+06	23.966135	5.084665	187.308110	515.818779	
min	10.100000	1.049088e+06	1.000000	2.000000	0.000000	0.000000	
25%	14.000000	1.440000e+06	7.000000	4.000000	0.000000	0.000000	
50%	15.600000	2.073600e+06	30.000000	8.000000	256.000000	0.000000	
75%	15.600000	2.073600e+06	33.000000	8.000000	256.000000	1000.000000	
max	18.400000	8.294400e+06	118.000000	64.000000	1024.000000	2000.000000	5

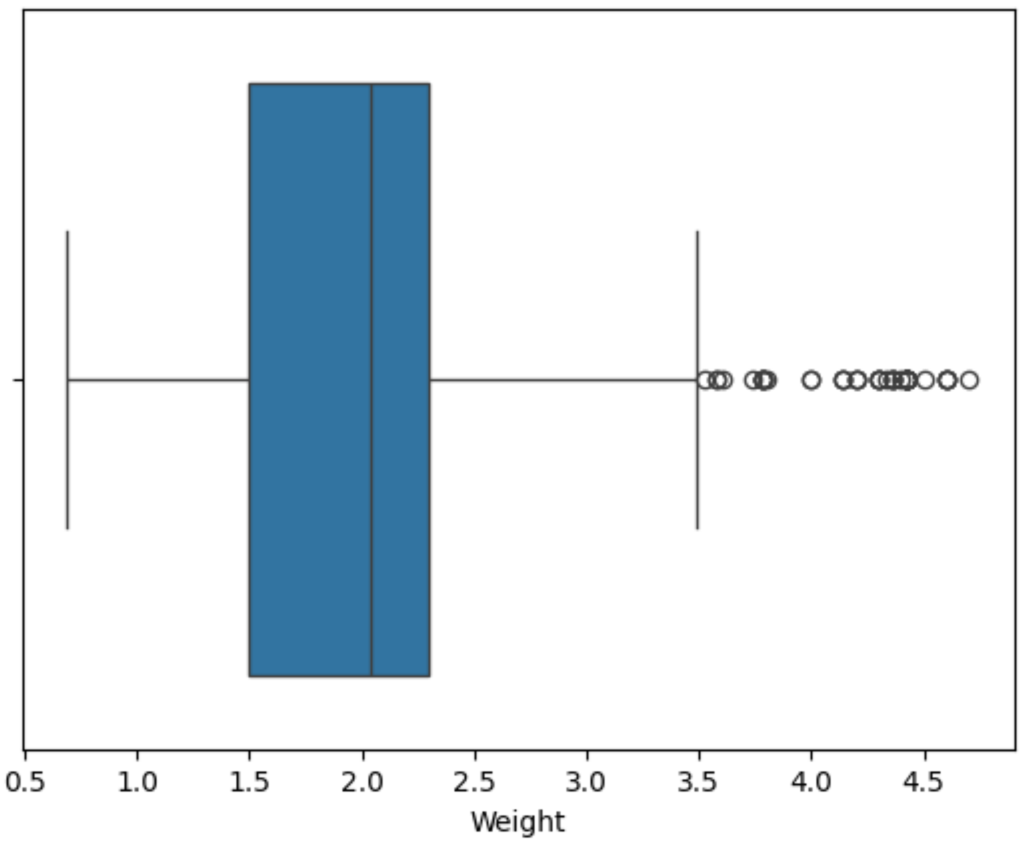
In [ ]:

```
#Boite à moustache de chaque colonne (pour les colonnes cohérentes)
sns.boxplot(x=laptop_data_quantified['Price (Euros)'])
plt.show()
sns.boxplot(x=laptop_data_quantified['RAM'])
plt.show()
sns.boxplot(x=laptop_data_quantified['ScreenSize'])
plt.show()
sns.boxplot(x=laptop_data_quantified['Weight'])
plt.show()
```



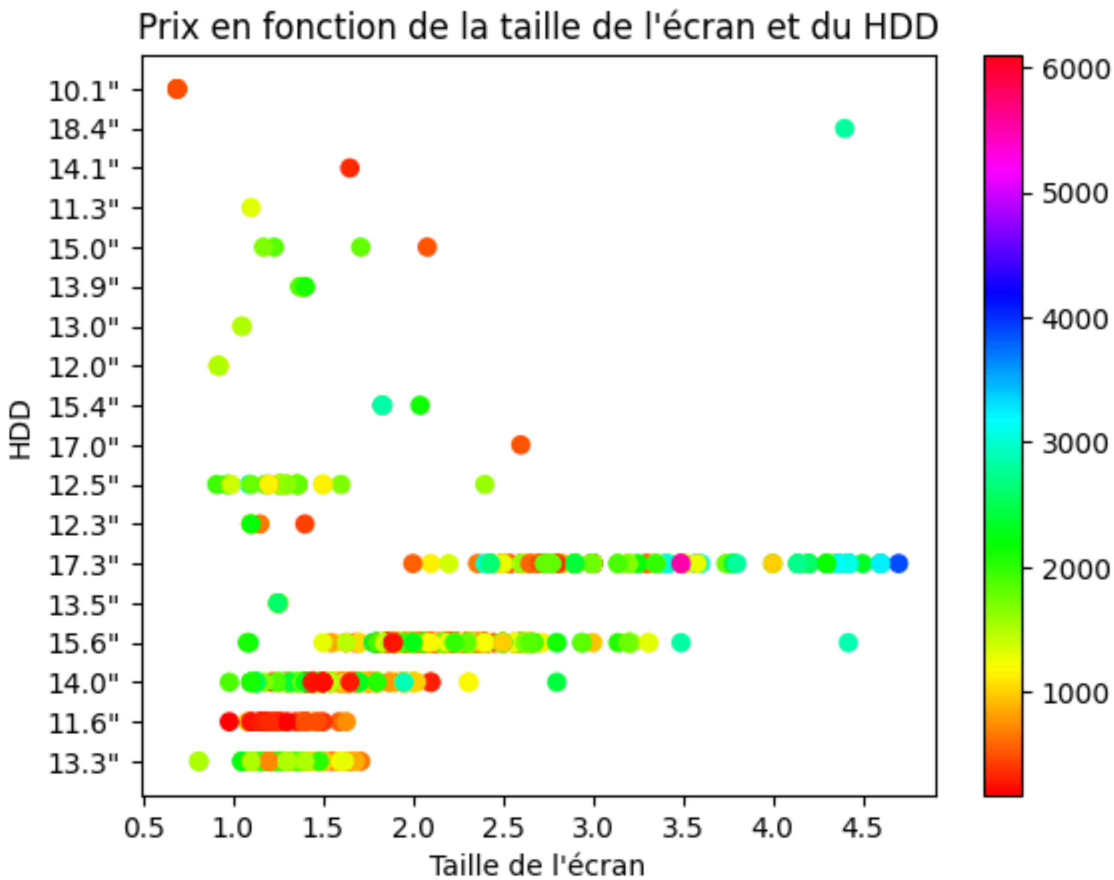

0
10
20
30
40
50
60
RAM


10
12
14
16
18
ScreenSize



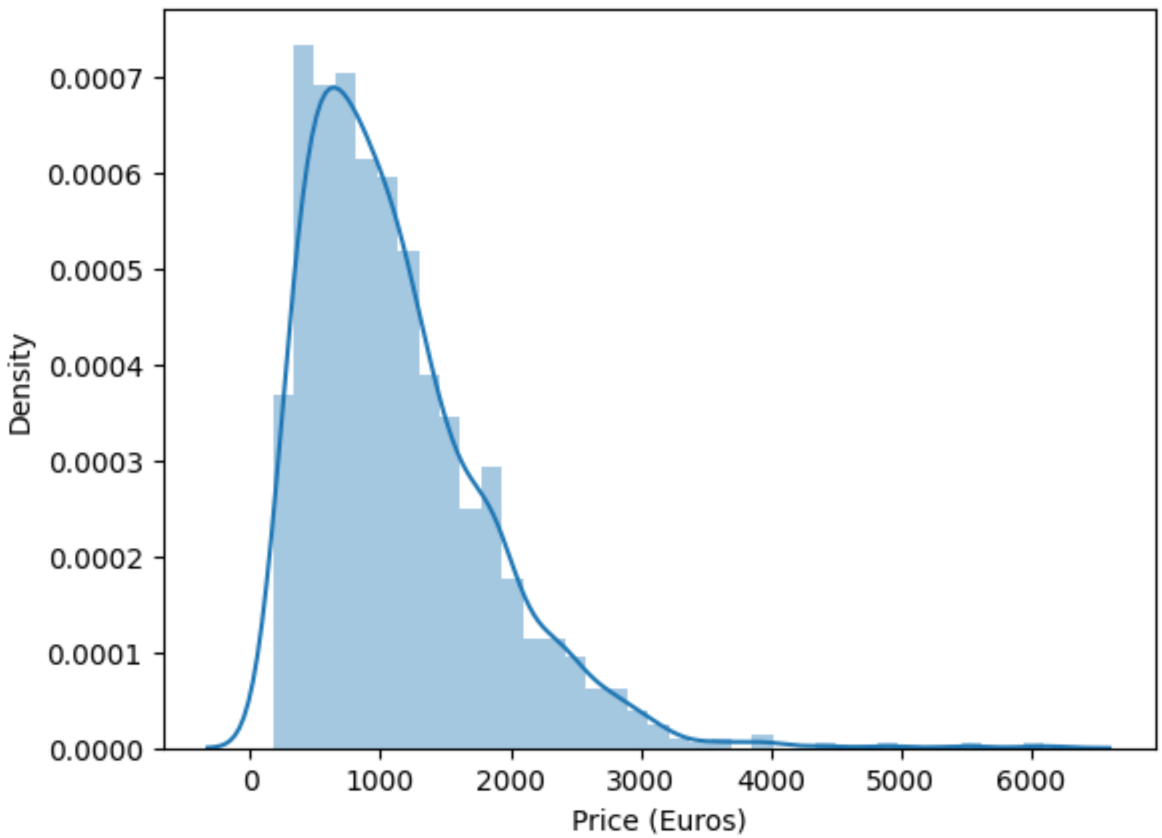
In [ ]:
```
# Pairplot complet (très long à afficher)

#sns.pairplot(laptop_data_quantified, hue="Price (Euros)")
plt.show()
```

In [ ]:
```
plt.scatter(laptop_data_quantified['Weight'].values,laptop_data["ScreenSize"].value
plt.xlabel('Taille de l\'écran')
plt.ylabel('HDD')
plt.title('Prix en fonction de la taille de l\'écran et du HDD')
plt.colorbar()
plt.show()
```



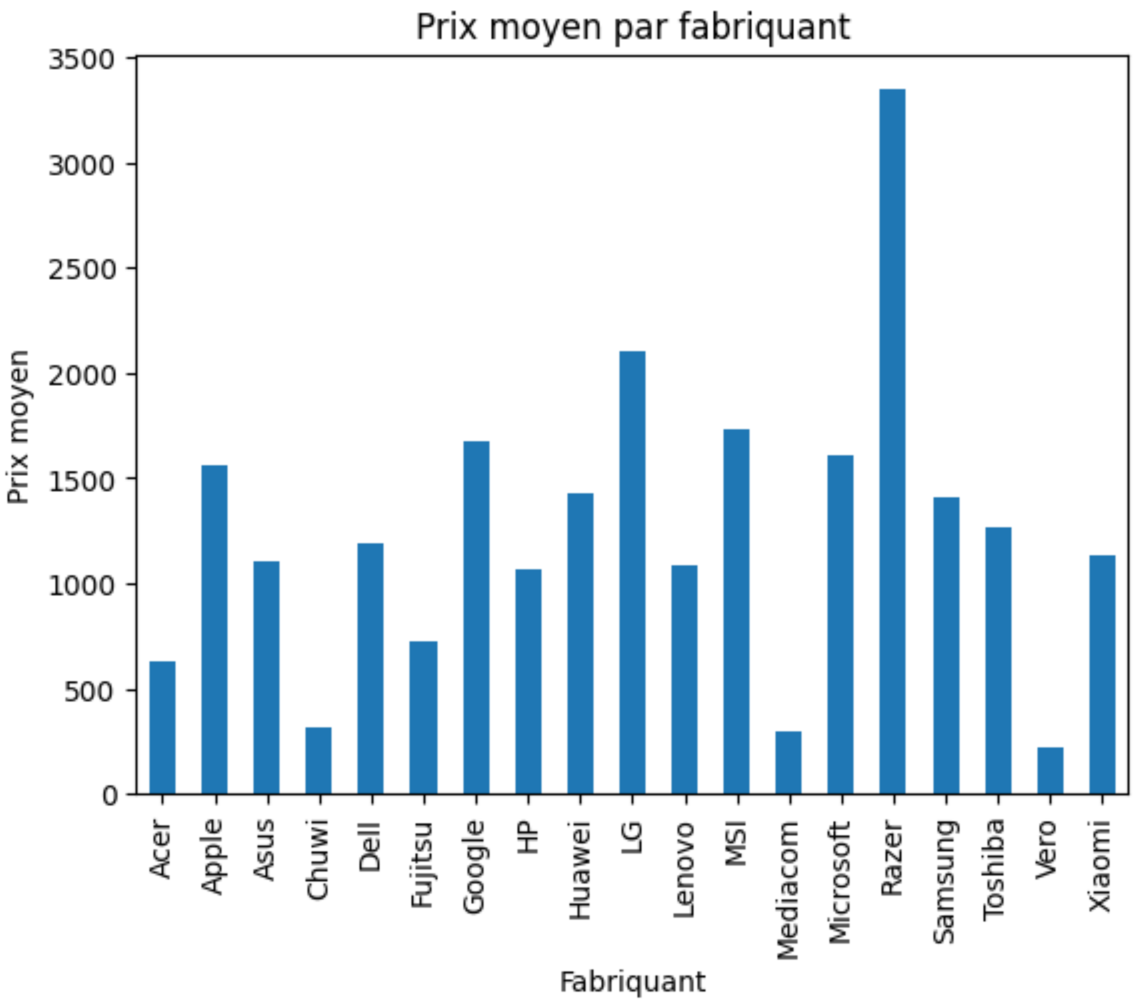
In [ ]:
```
sns.distplot(laptop_data_quantified['Price (Euros)'])
plt.show()
```

```
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\3571809138.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(laptop_data_quantified['Price (Euros)'])
```

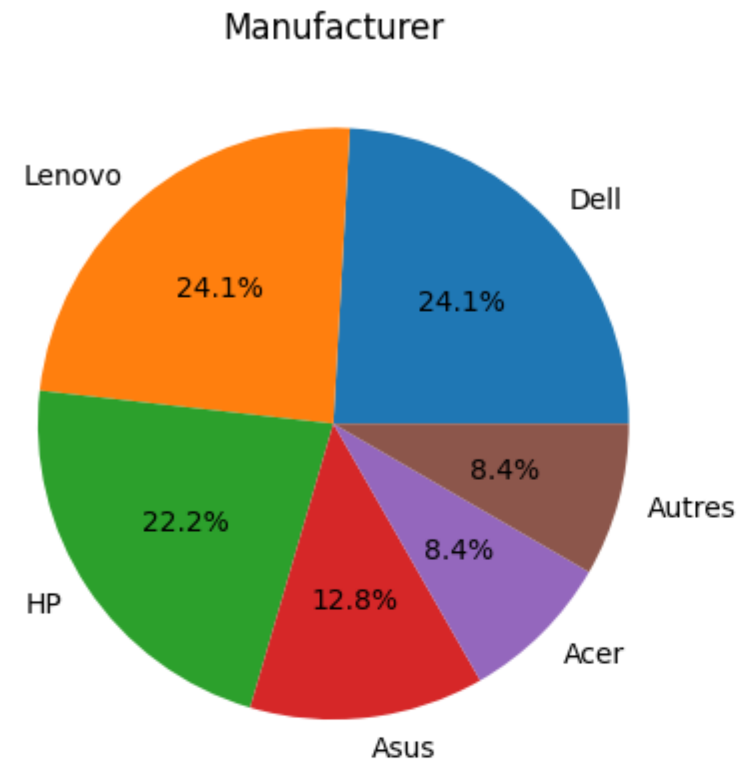


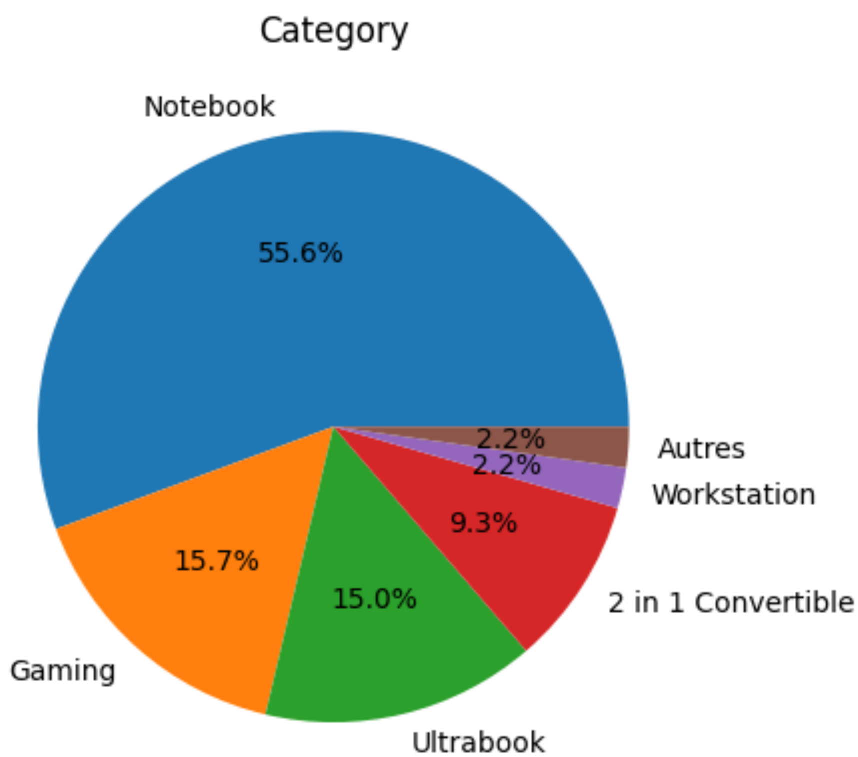
In [ ]:

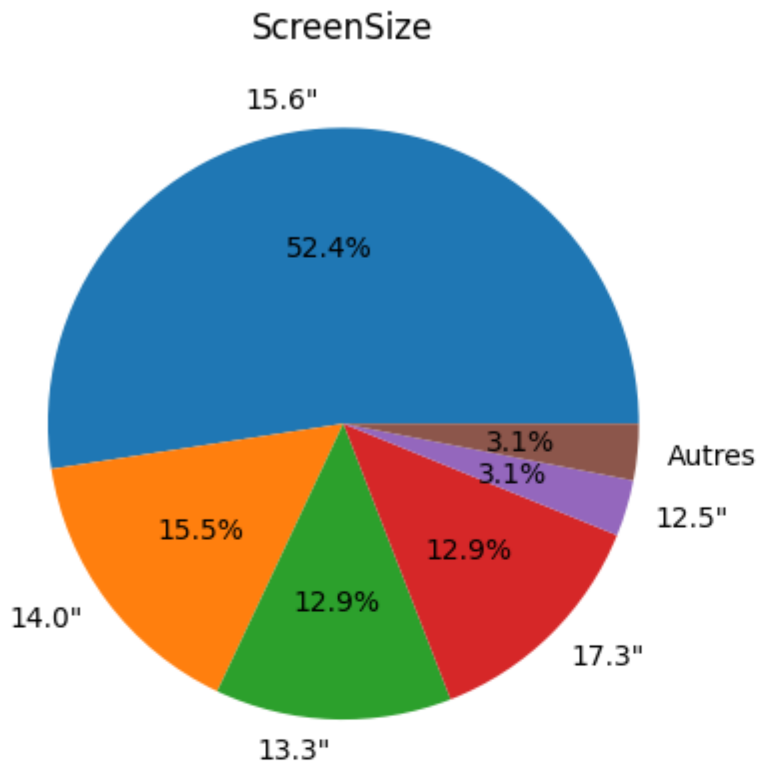
```
# Regroupement des données par fabriquant et calcul du prix moyen pour chacun
manufacturer_mean_price = laptop_data.groupby('Manufacturer')['Price (Euros)'].mean

manufacturer_mean_price.plot(kind='bar')
plt.xlabel('Fabriquant')
plt.ylabel('Prix moyen')
plt.title('Prix moyen par fabriquant')
plt.show()
```

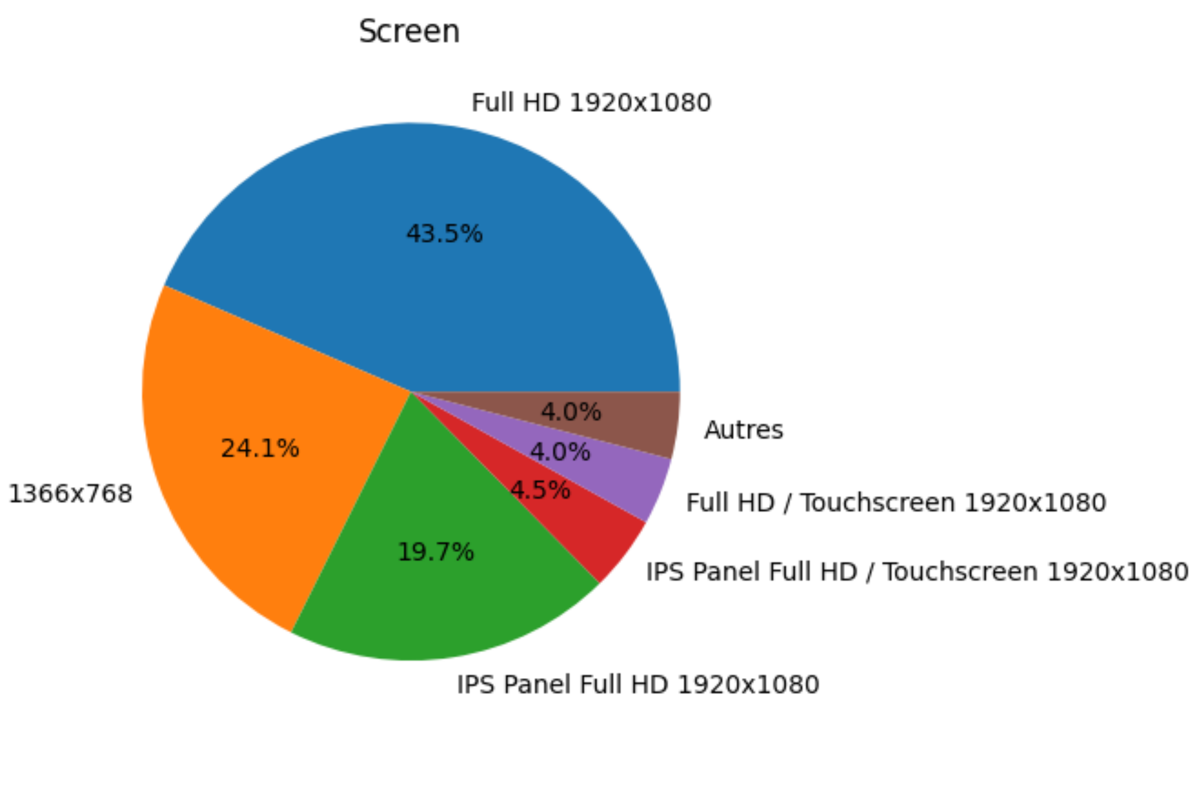


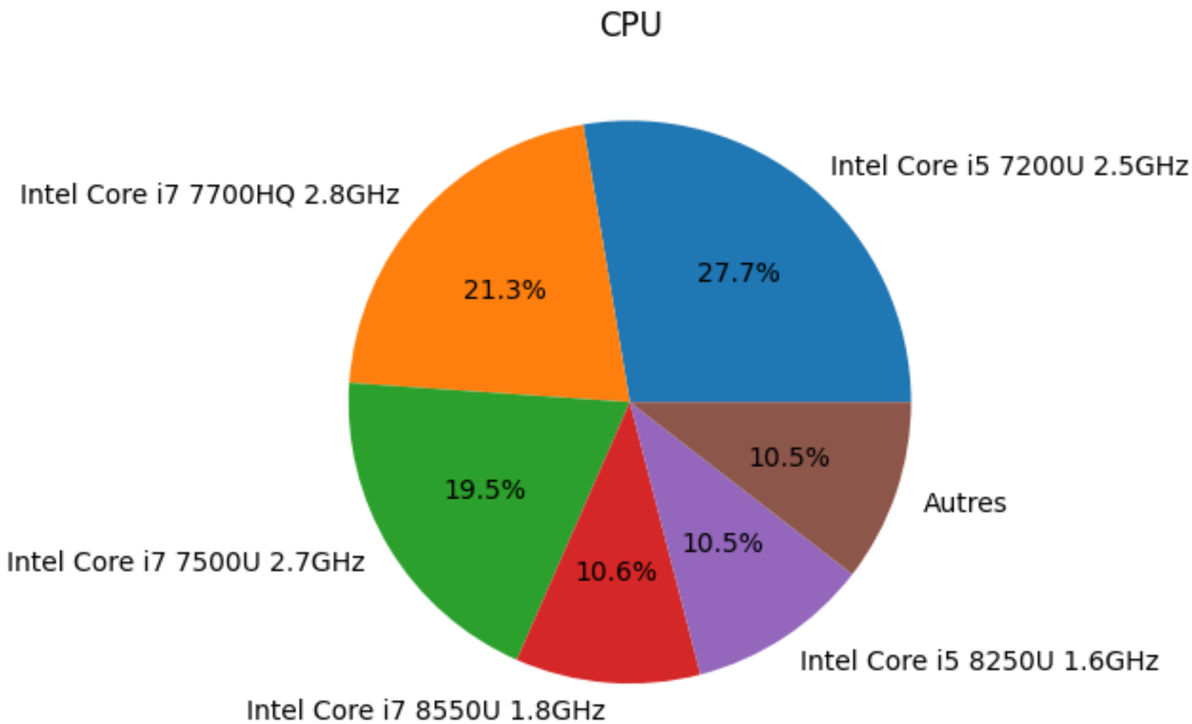
In [ ]:
```
for column in laptop_data.columns:
  if column != 'Price (Euros)' and column != 'ID':
      counts = laptop_data[column].value_counts().head(5)
      others = counts.sum() - counts.iloc[:4].sum()
      counts = pd.concat([counts, pd.Series([others], index=["Autres"])])
      counts.plot(kind='pie', autopct='%1.1f%%')
      plt.title(column)
      plt.show()
```

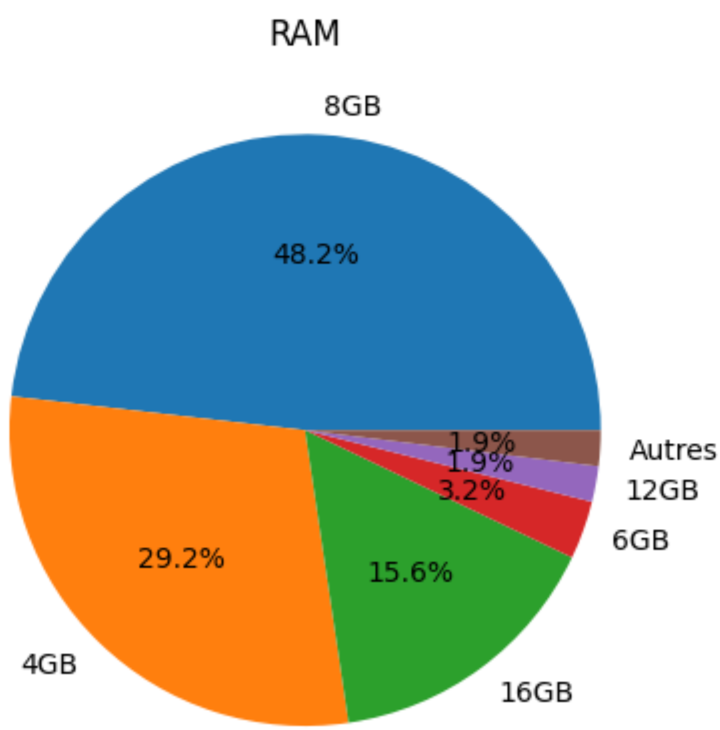

Manufacturer
Lenovo
Dell
24.1%
24.1%
8.4%
Autres
22.2%
8.4%
HP
12.8%
Acer
Asus

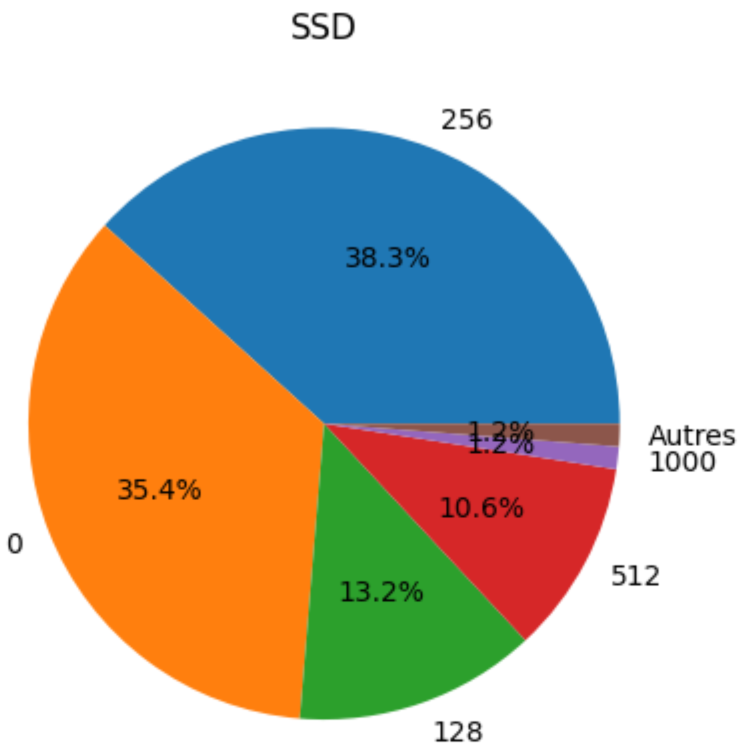

Category
Notebook
55.6%
2.2%
Autres
2.2%
Workstation
9.3%
15.7%
15.0%
2 in 1 Convertible
Gaming
Ultrabook

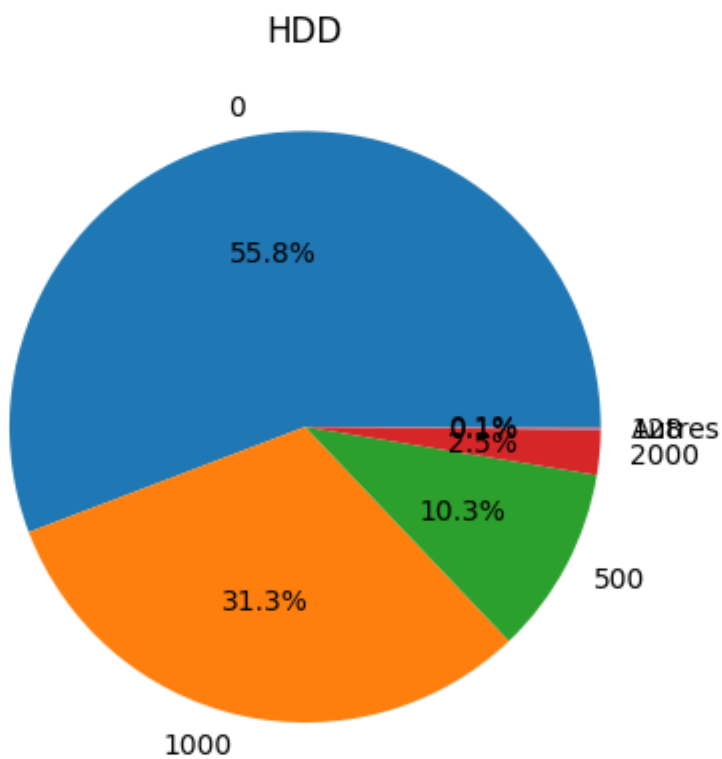

ScreenSize
15.6"
52.4%
14.0"
15.5%
13.3"
12.9%
17.3"
12.9%
12.5"
3.1%
Autres
3.1%

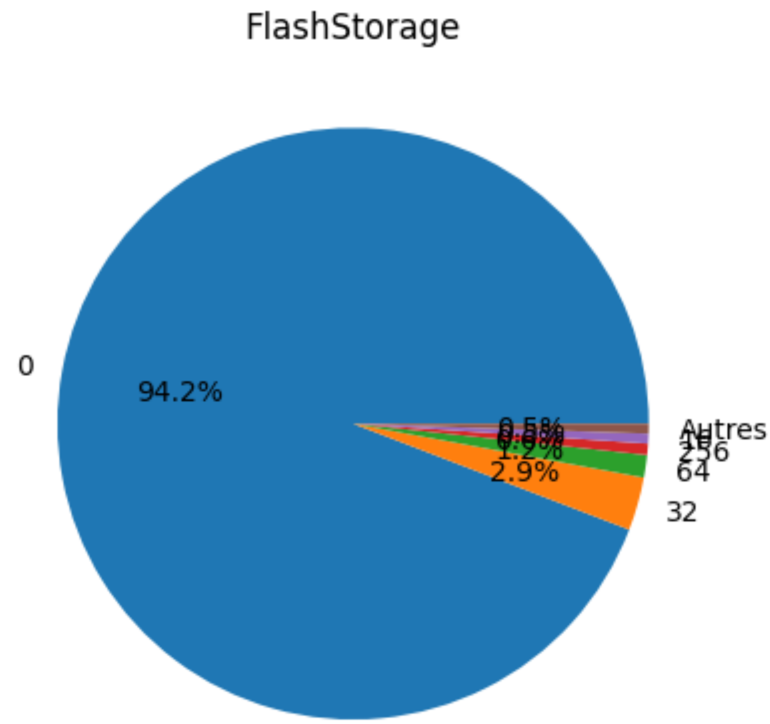

Screen
Full HD 1920x1080
43.5%
1366x768
24.1%
IPS Panel Full HD 1920x1080
19.7%
IPS Panel Full HD / Touchscreen 1920x1080
4.5%
Full HD / Touchscreen 1920x1080
4.0%
Autres
4.0%


CPU
Intel Core i5 7200U 2.5GHz
27.7%
Intel Core i7 7700HQ 2.8GHz
21.3%
Intel Core i7 7500U 2.7GHz
19.5%
Intel Core i7 8550U 1.8GHz
10.6%
Intel Core i5 8250U 1.6GHz
10.5%
Autres
10.5%

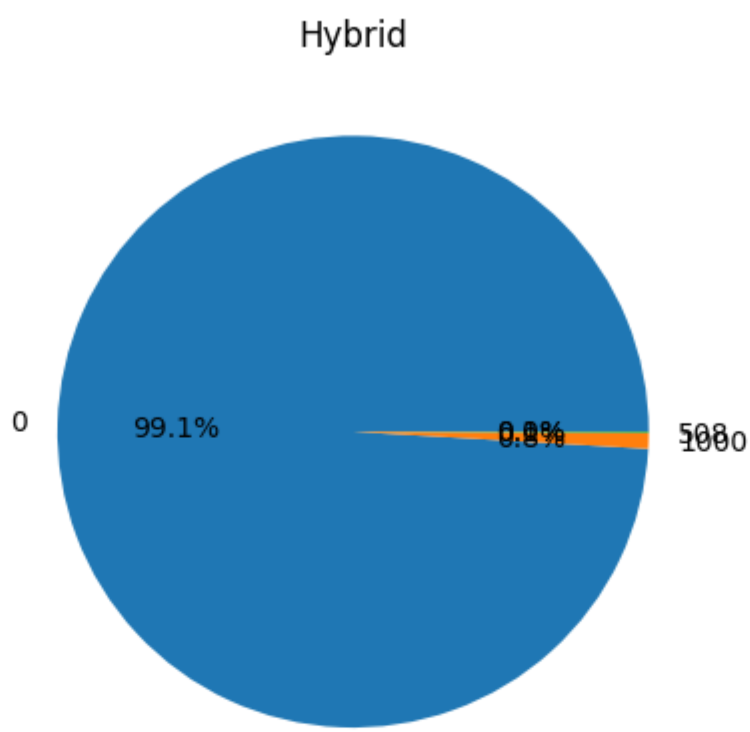

RAM
8GB
48.2%
4GB
29.2%
16GB
15.6%
6GB
3.2%
12GB
1.9%
Autres
1.9%


SSD
256
38.3%
0
35.4%
128
13.2%
512
10.6%
1000
1.2%
Autres
1.2%

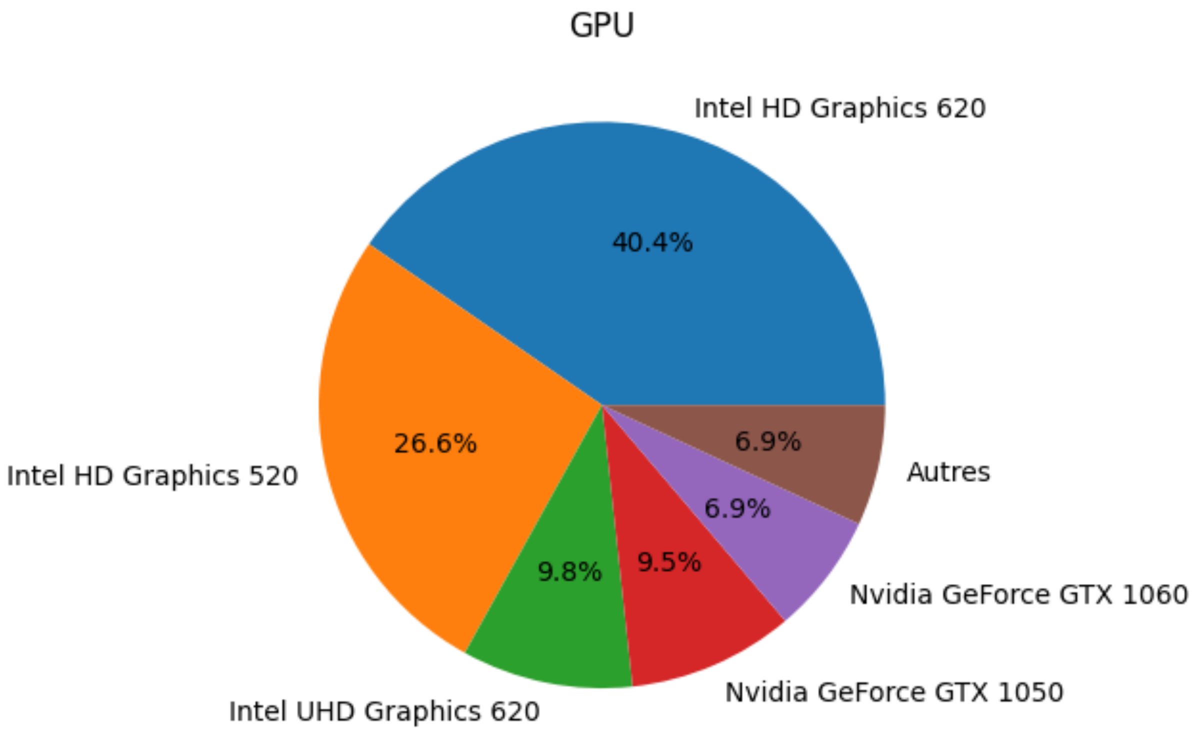

HDD
0
55.8%
1000
31.3%
500
10.3%
2000
2.5%
Autres

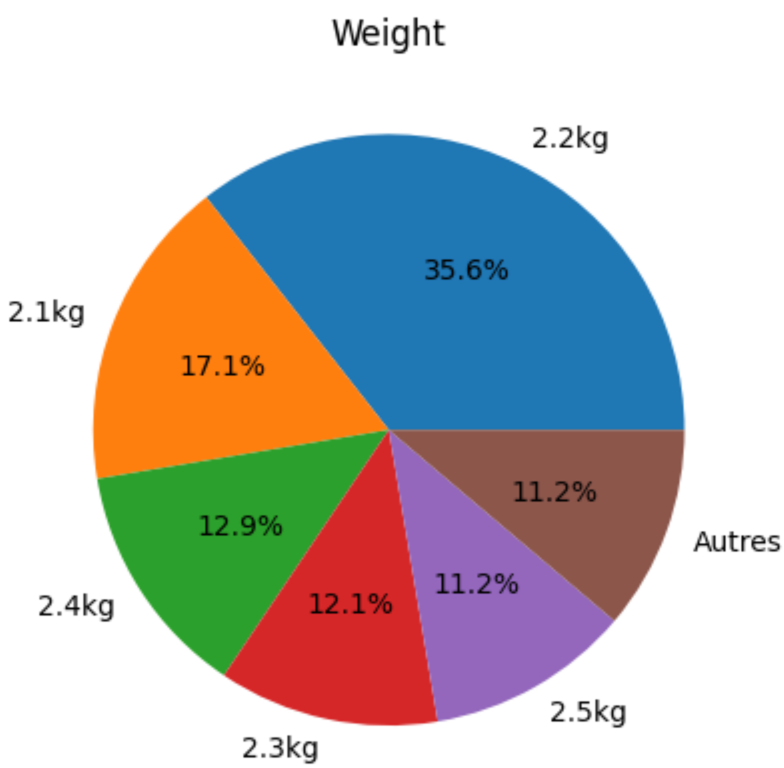


FlashStorage
0
94.2%
0.5%
1.2%
2.9%
Autres
256
64
32



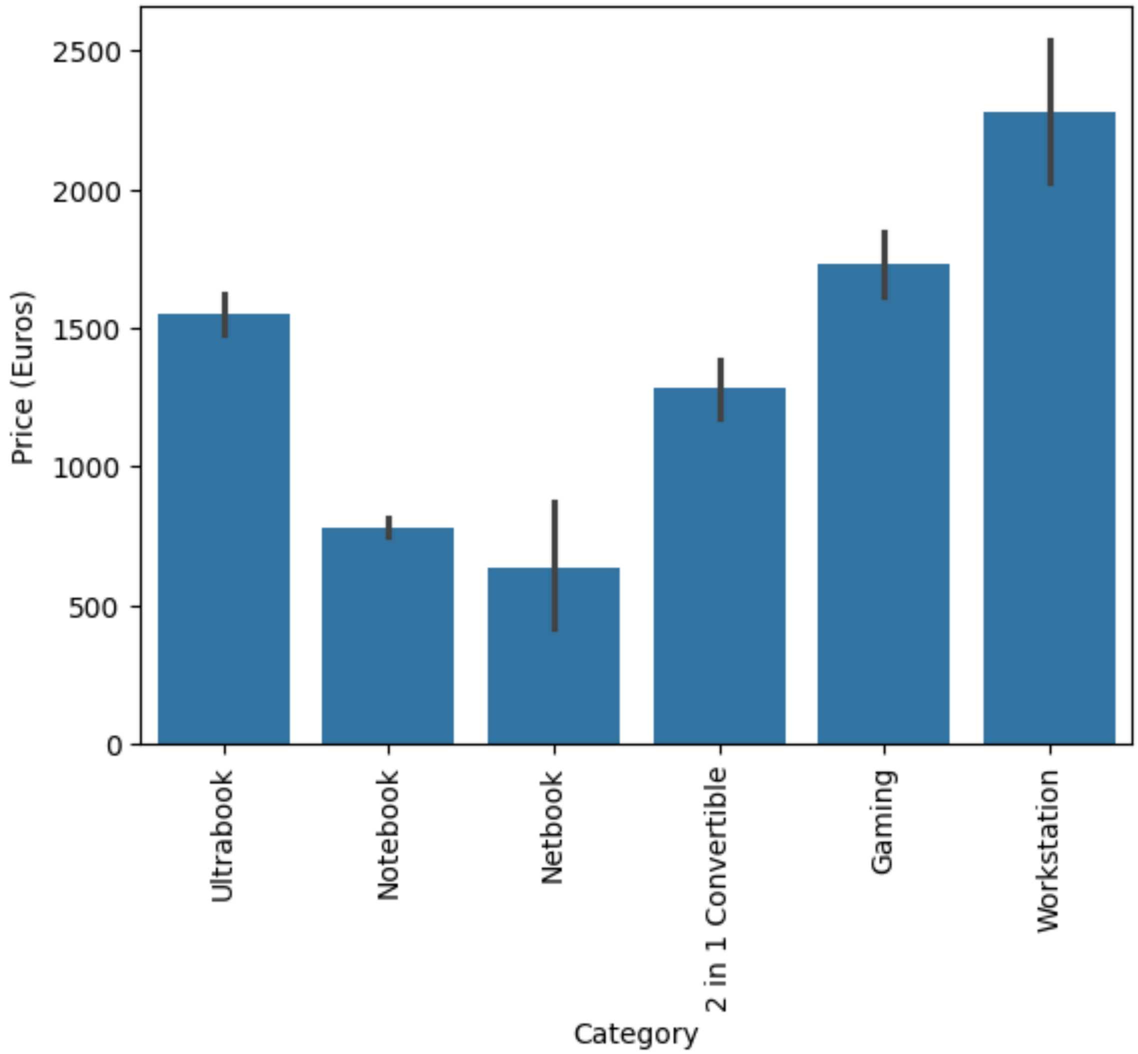
Hybrid
0
99.1%
508
1000





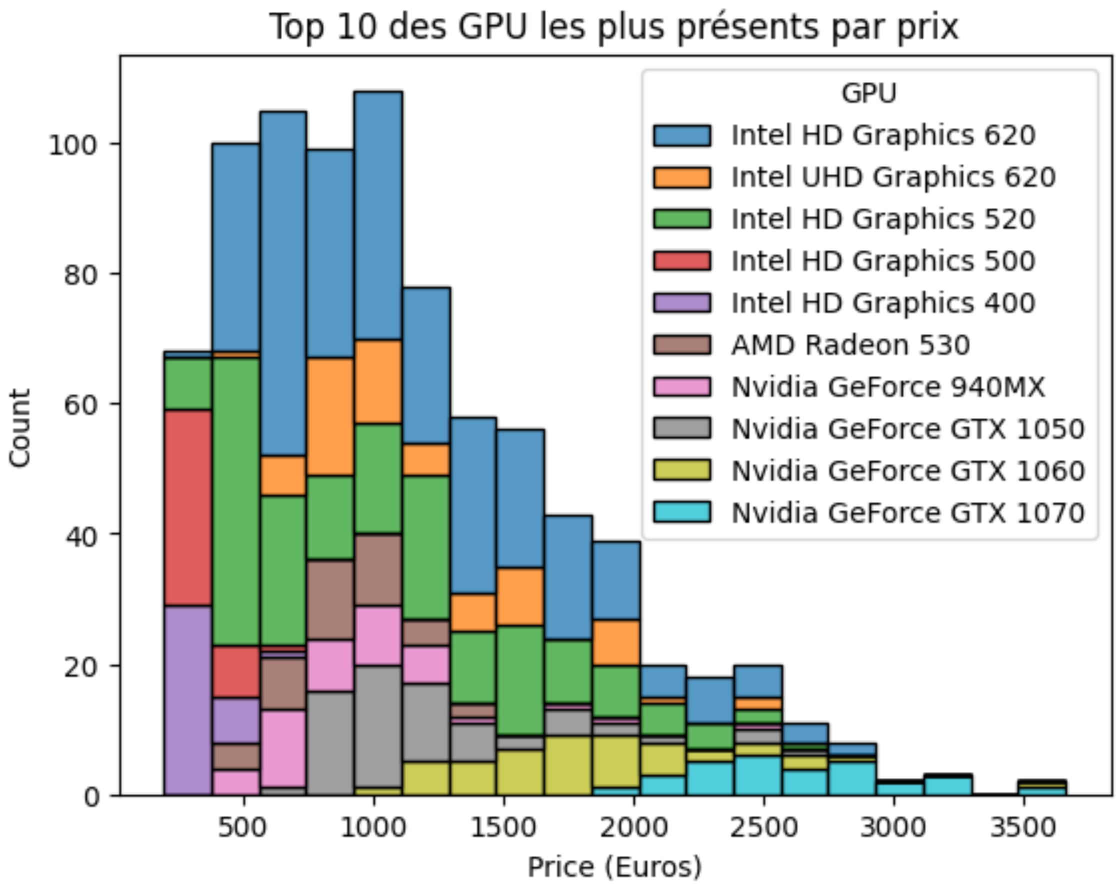
In [ ]:

```
#data['TypeName'].value_counts().plot(kind='bar')
sns.barplot(x=laptop_data['Category'], y=laptop_data_quantified['Price (Euros)'])
plt.xticks(rotation="vertical")
plt.show()
```



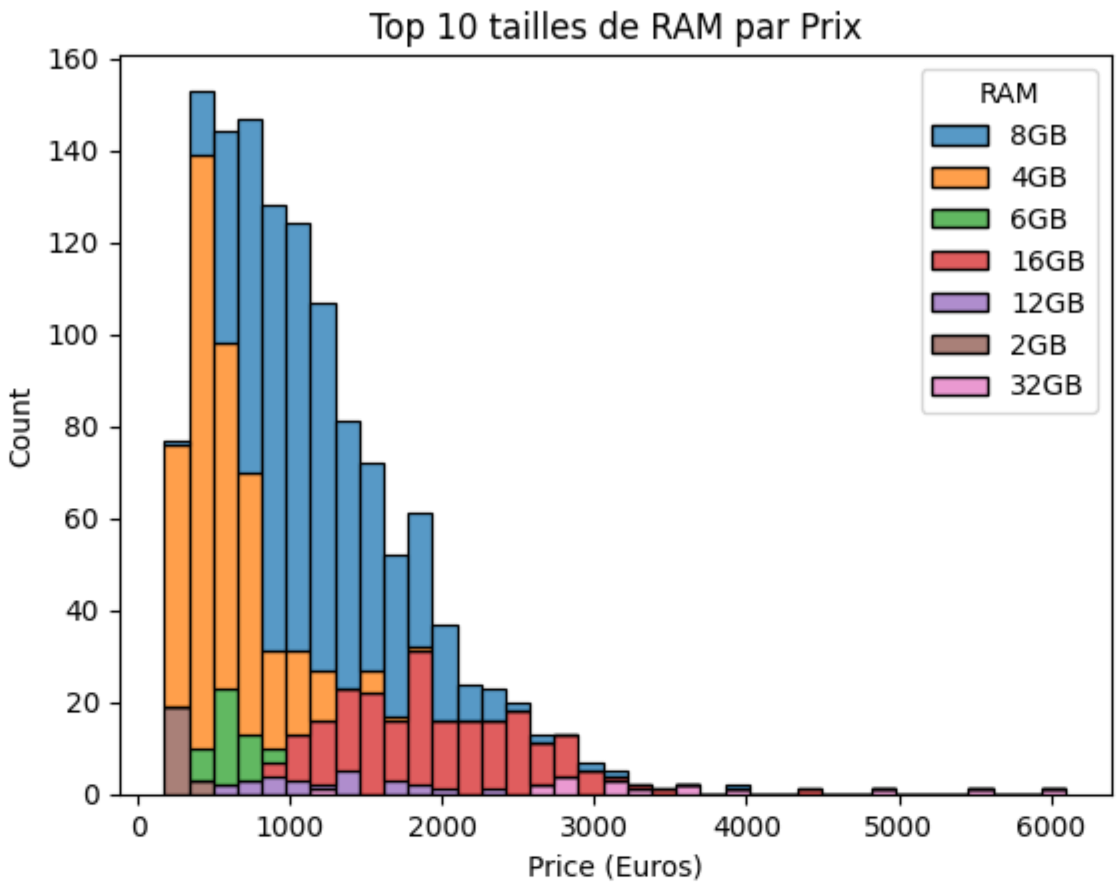
In [ ]:

```
# Séparation du top 10 des GPU (sinon le graphique est illisible)
top_10_gpus = laptop_data['GPU'].value_counts().head(10).index.tolist()

# Nouveau dataframe avec uniquement les 10 GPU les plus fréquents
laptop_data_top_10_gpus = laptop_data[laptop_data['GPU'].isin(top_10_gpus)]

sns.histplot(data=laptop_data_top_10_gpus, x='Price (Euros)', hue='GPU', multiple='
plt.title('Top 10 des GPU les plus présents par prix')
plt.show()
```



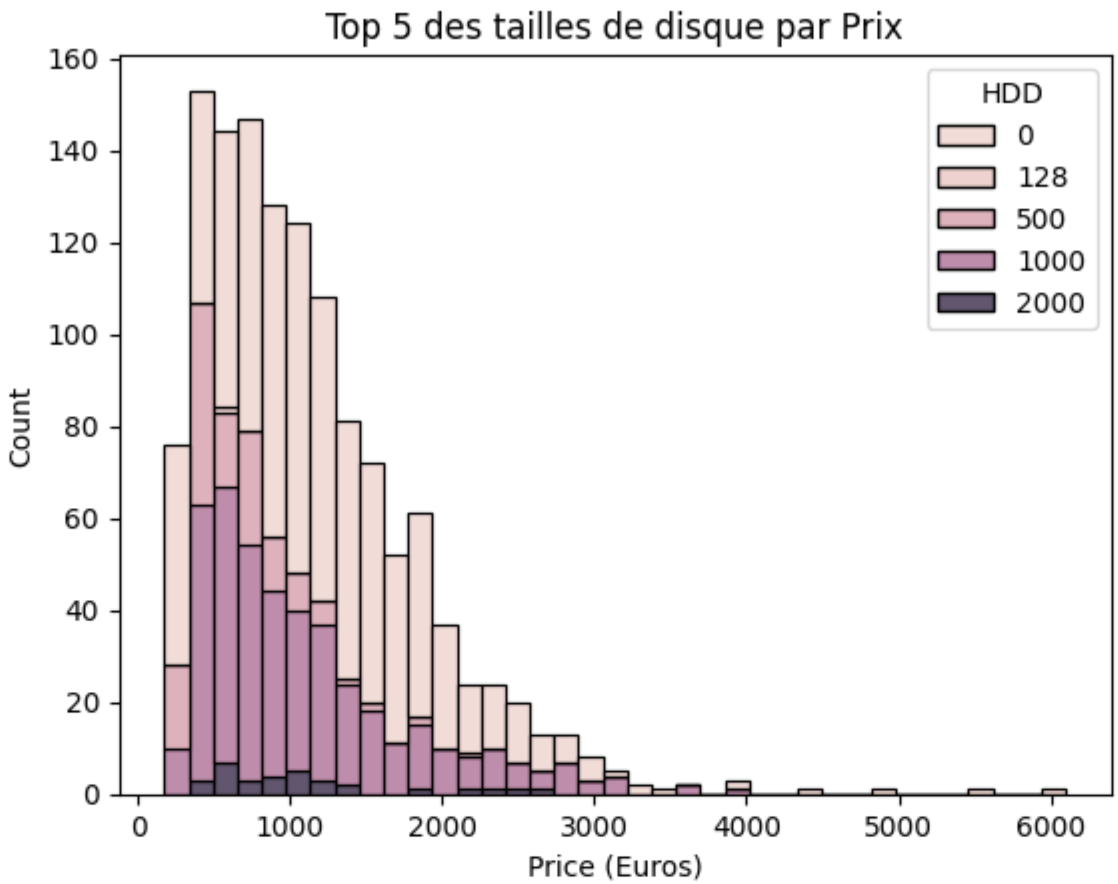
In [ ]:

```
#Pareil pour la RAM
ram_counts = laptop_data['RAM'].value_counts()

laptop_data_filtered_ram = laptop_data[laptop_data['RAM'].isin(ram_counts[ram_count

sns.histplot(data=laptop_data_filtered_ram, x='Price (Euros)', hue='RAM', multiple=
plt.title('Top 10 tailles de RAM par Prix')
plt.show()
```

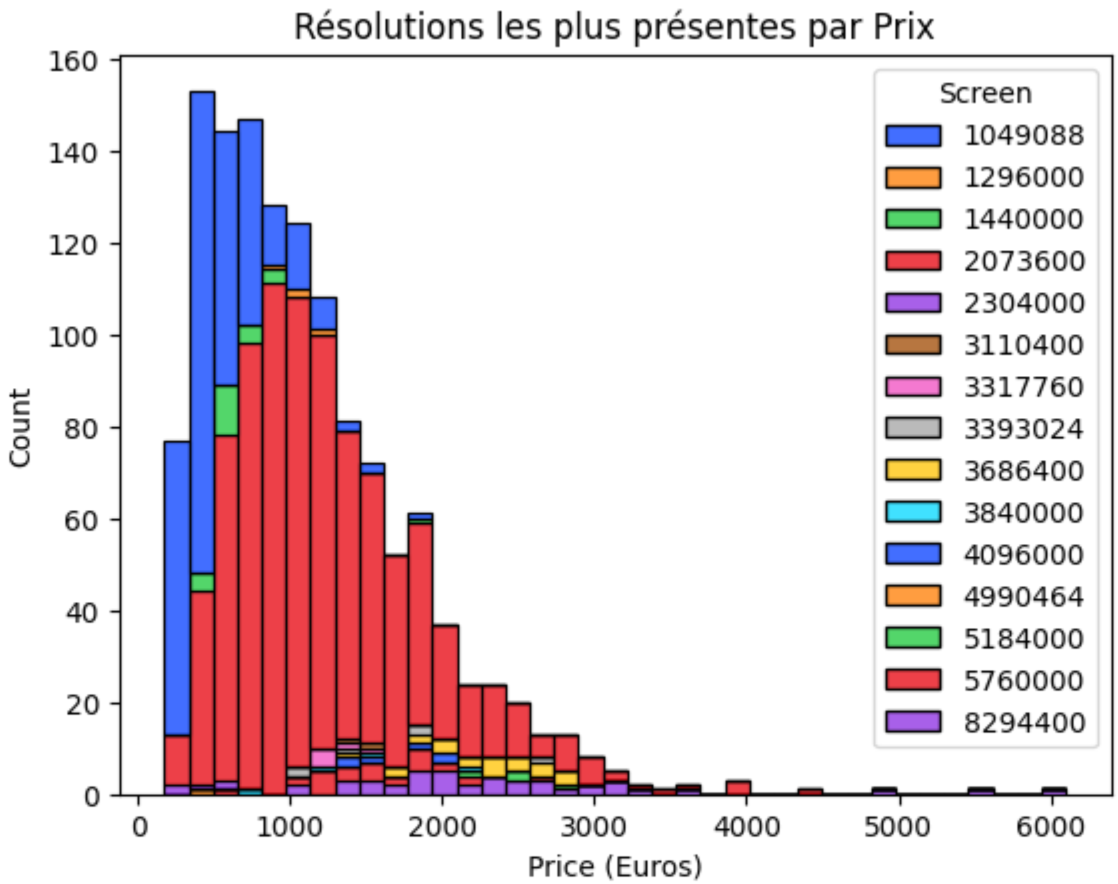


In [ ]:

```
# Pareil pour le stockage
storage_counts = laptop_data['HDD'].value_counts()

top_5_storage = storage_counts.head(5).index.tolist()
laptop_data_filtered_storage = laptop_data[laptop_data['HDD'].isin(top_5_storage)]

sns.histplot(data=laptop_data_filtered_storage, x='Price (Euros)', hue='HDD', multi
plt.title('Top 5 des tailles de disque par Prix')
plt.show()
```



In [ ]:
```
sns.histplot(data=laptop_data_quantified, x='Price (Euros)', hue='Screen', multiple
plt.title('Résolutions les plus présentes par Prix')
plt.show()
```



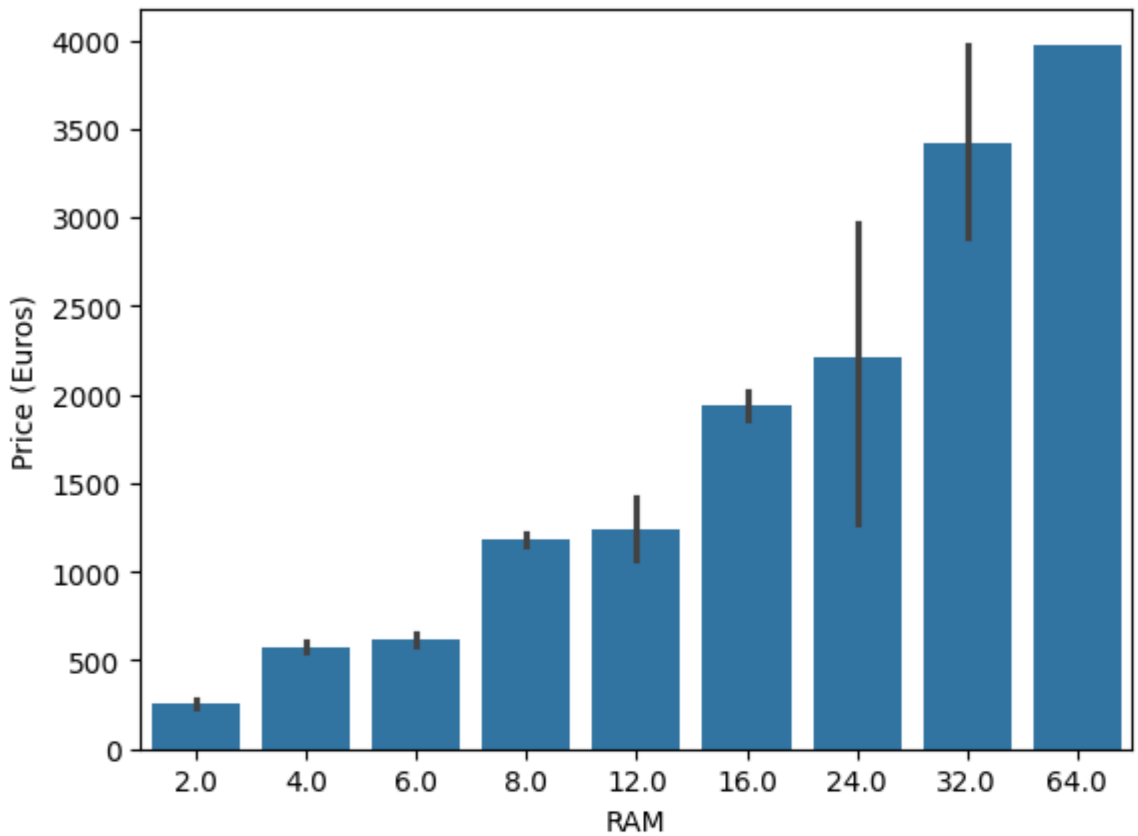
Rouge : écran 1920x1080, bleu : écran 1366x768.

In [ ]:
```python
laptop_data_quantified.corr()['Price (Euros)'].sort_values(ascending=False)
```

Out[ ]:
```
Price (Euros)     1.000000
RAM               0.743007
SSD               0.669709
Screen            0.515486
Category          0.292191
CPU               0.263000
GPU               0.234405
Weight            0.210370
Manufacturer      0.149303
ScreenSize        0.068197
Hybrid            0.007989
FlashStorage     -0.040511
HDD              -0.096441
Name: Price (Euros), dtype: float64
```

In [ ]:
```python
order = laptop_data_quantified['RAM'].sort_values(ascending=True).unique()
sns.barplot(x=laptop_data_quantified['RAM'], y=laptop_data_quantified['Price (Euros
```

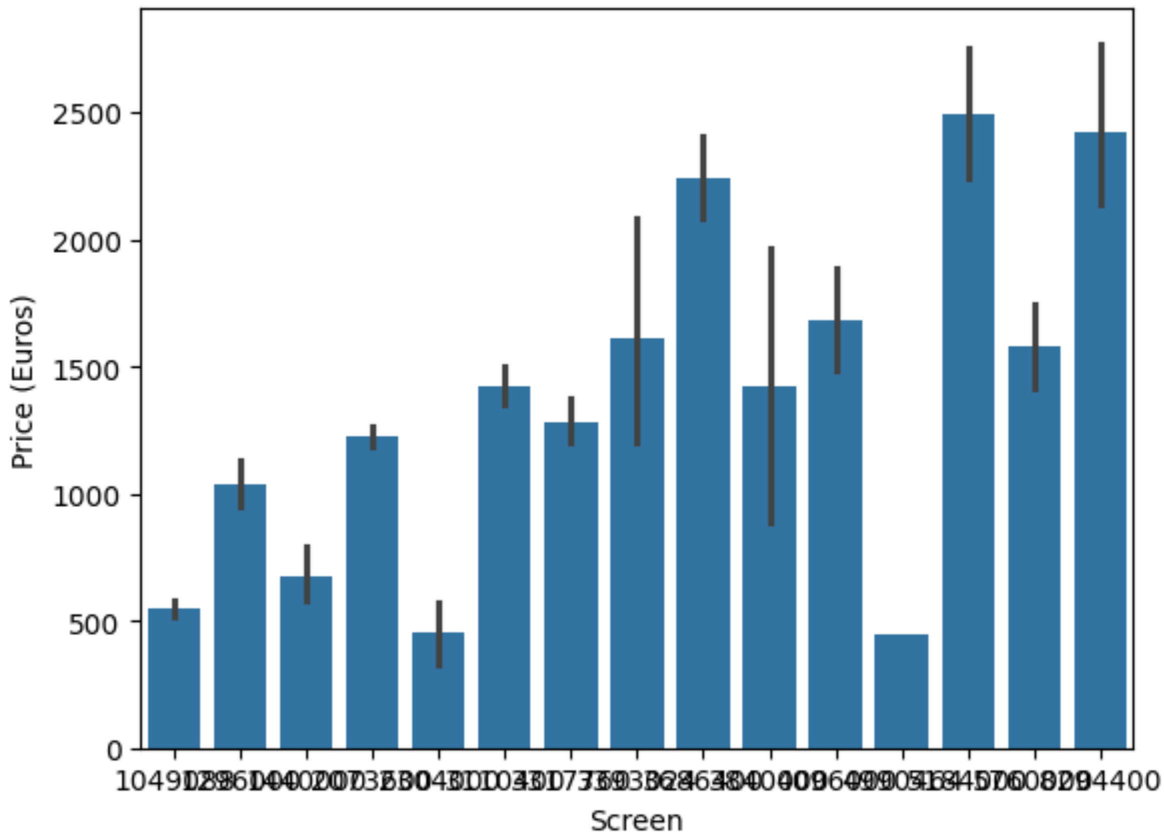
Out[ ]:
```
<Axes: xlabel='RAM', ylabel='Price (Euros)'>
```



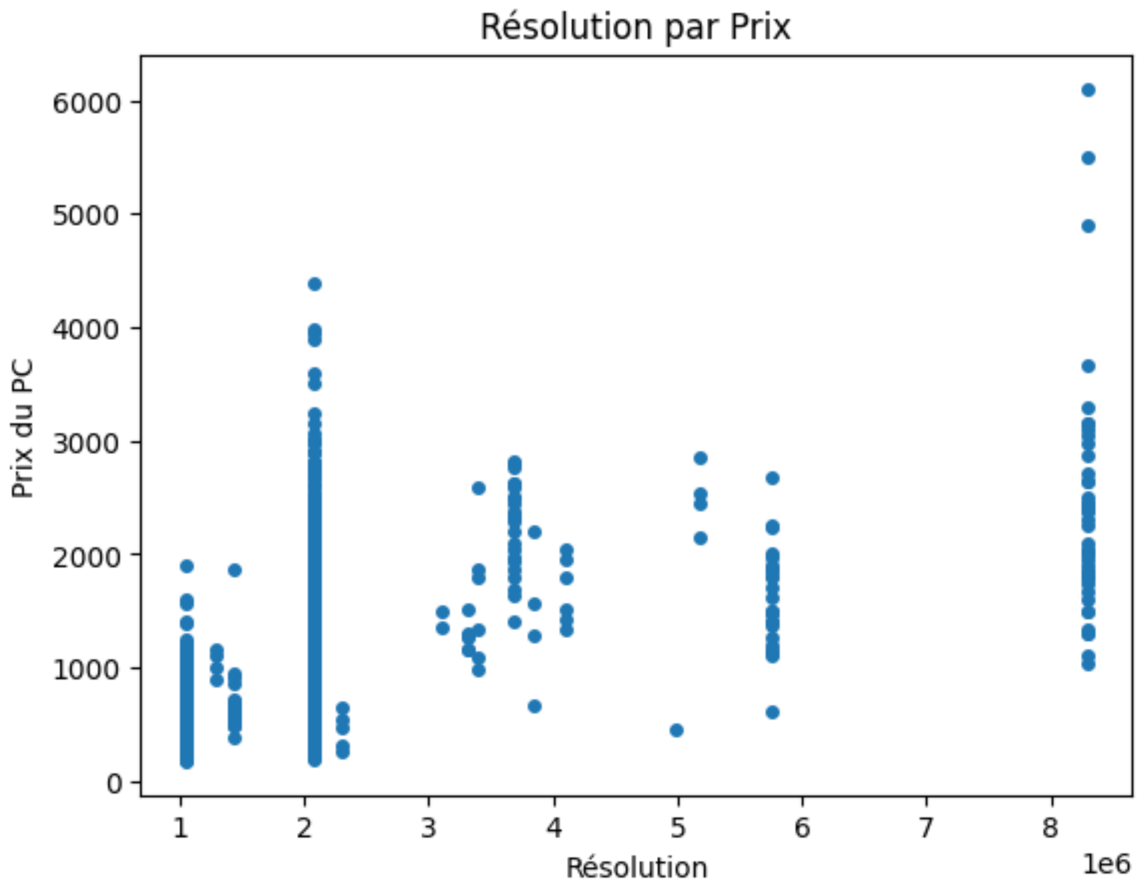
4000
3500
3000
2500
2000
1500
1000
500
0
Price (Euros)
2.0
4.0
6.0
8.0
12.0
16.0
24.0
32.0
64.0
RAM

In [ ]:
```
order_screen = laptop_data_quantified['Screen'].sort_values(ascending=True).unique(
sns.barplot(x=laptop_data_quantified['Screen'], y=laptop_data_quantified['Price (Eu
```

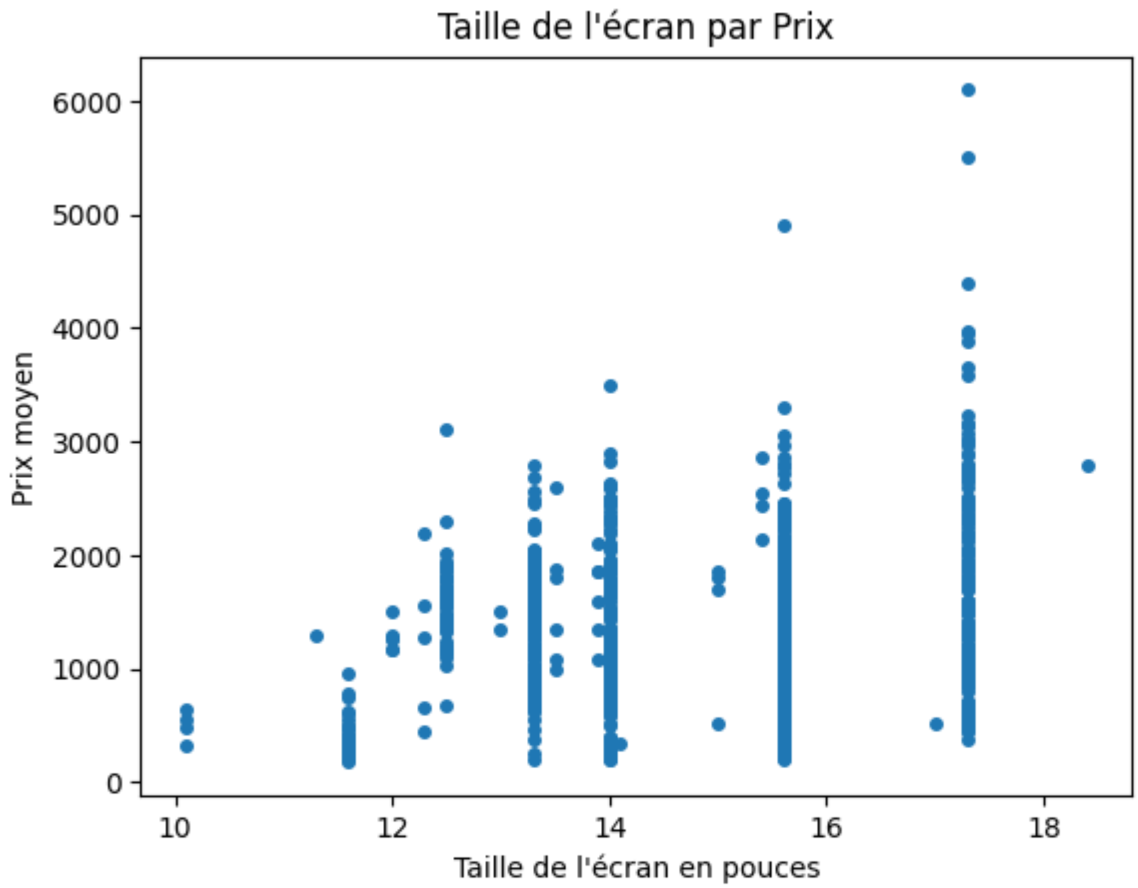
Out[ ]: <Axes: xlabel='Screen', ylabel='Price (Euros)'>



In [ ]:
```
plt.plot(laptop_data_quantified['Screen'], laptop_data_quantified['Price (Euros)'],
plt.xlabel('Résolution')
plt.ylabel('Prix du PC')
plt.title('Résolution par Prix')
plt.show()
```



In [ ]:
```
plt.plot(laptop_data_quantified['ScreenSize'], laptop_data_quantified['Price (Euros
plt.xlabel('Taille de l\'écran en pouces')
plt.ylabel('Prix moyen')
plt.title('Taille de l\'écran par Prix')
plt.show()
```



Corrélation, covariance

In [ ]:
```
# Matrice de covariance

cov_mat = laptop_data_quantified.cov()
cov_mat
```

Out[ ]:

	Manufacturer	Category	ScreenSize	Screen	CPU
Manufacturer	7.572157	0.049251	-0.222225	2.754218e+05	4.177640e+00
Category	0.049251	1.688446	0.262409	2.072867e+05	4.423694e+00
ScreenSize	-0.222225	0.262409	2.034343	-1.714510e+05	-2.898587e+00
Screen	275421.811559	207286.676970	-171451.022029	1.935695e+12	6.491415e+06
CPU	4.177640	4.423694	-2.898587	6.491415e+06	5.743756e+02
RAM	2.082844	1.818421	1.725990	2.803941e+06	2.588288e+01
SSD	67.707009	29.095958	-27.727367	1.265684e+08	8.485260e+02
HDD	-89.211959	91.323650	390.496197	-8.299558e+07	-1.067302e+03
FlashStorage	-7.358927	-3.028380	-9.922923	2.367227e+04	1.028215e+02
Hybrid	-1.099976	1.273945	7.138375	-1.241237e+06	4.701281e+01
GPU	-1.825615	6.179814	8.915456	4.569355e+06	2.141632e+02
Weight	-0.108541	0.278430	0.785562	-4.076955e+04	-3.827416e-01
Price (Euros)	287.185314	265.395461	67.992037	5.013239e+08	4.405915e+03

In [ ]:
```python
# Matrice de corrélation

cor_mat = laptop_data_quantified.cov()
cor_mat
```

Out[ ]:

	Manufacturer	Category	ScreenSize	Screen	CPU
Manufacturer	7.572157	0.049251	-0.222225	2.754218e+05	4.177640e+00
Category	0.049251	1.688446	0.262409	2.072867e+05	4.423694e+00
ScreenSize	-0.222225	0.262409	2.034343	-1.714510e+05	-2.898587e+00
Screen	275421.811559	207286.676970	-171451.022029	1.935695e+12	6.491415e+06
CPU	4.177640	4.423694	-2.898587	6.491415e+06	5.743756e+02
RAM	2.082844	1.818421	1.725990	2.803941e+06	2.588288e+01
SSD	67.707009	29.095958	-27.727367	1.265684e+08	8.485260e+02
HDD	-89.211959	91.323650	390.496197	-8.299558e+07	-1.067302e+03
FlashStorage	-7.358927	-3.028380	-9.922923	2.367227e+04	1.028215e+02
Hybrid	-1.099976	1.273945	7.138375	-1.241237e+06	4.701281e+01
GPU	-1.825615	6.179814	8.915456	4.569355e+06	2.141632e+02
Weight	-0.108541	0.278430	0.785562	-4.076955e+04	-3.827416e-01
Price (Euros)	287.185314	265.395461	67.992037	5.013239e+08	4.405915e+03

In [ ]:
```python
# Matrice de détermination

det_mat = cor_mat**2
det_mat
```

Out[ ]:

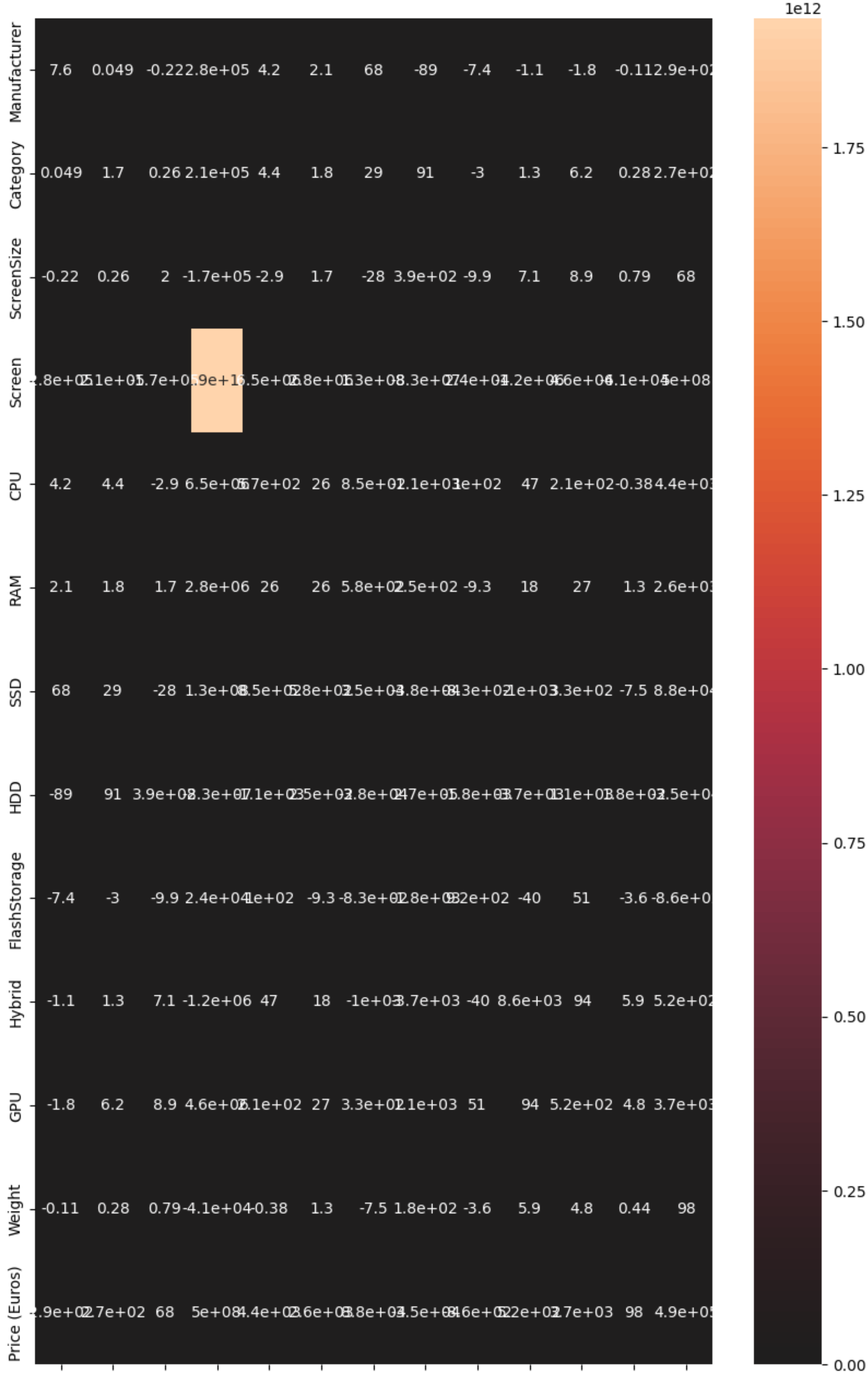
	Manufacturer	Category	ScreenSize	Screen	CPU	
Manufacturer	5.733756e+01	2.425625e-03	4.938399e-02	7.585717e+10	1.745267e+01	4.33
Category	2.425625e-03	2.850848e+00	6.885859e-02	4.296777e+10	1.956907e+01	3.30
ScreenSize	4.938399e-02	6.885859e-02	4.138552e+00	2.939545e+10	8.401806e+00	2.97
Screen	7.585717e+10	4.296777e+10	2.939545e+10	3.746913e+24	4.213847e+13	7.86
CPU	1.745267e+01	1.956907e+01	8.401806e+00	4.213847e+13	3.299074e+05	6.69
RAM	4.338241e+00	3.306655e+00	2.979041e+00	7.862086e+12	6.699234e+02	6.68
SSD	4.584239e+03	8.465748e+02	7.688069e+02	1.601955e+16	7.199964e+05	3.30
HDD	7.958774e+03	8.340009e+03	1.524873e+05	6.888267e+15	1.139134e+06	6.37
FlashStorage	5.415381e+01	9.171082e+00	9.846441e+01	5.603763e+08	1.057225e+04	8.58
Hybrid	1.209947e+00	1.622937e+00	5.095640e+01	1.540669e+12	2.210204e+03	3.27
GPU	3.332868e+00	3.819010e+01	7.948536e+01	2.087901e+13	4.586588e+04	7.47
Weight	1.178124e-02	7.752341e-02	6.171075e-01	1.662156e+09	1.464911e-01	1.68
Price (Euros)	8.247540e+04	7.043475e+04	4.622917e+03	2.513257e+17	1.941208e+07	6.97

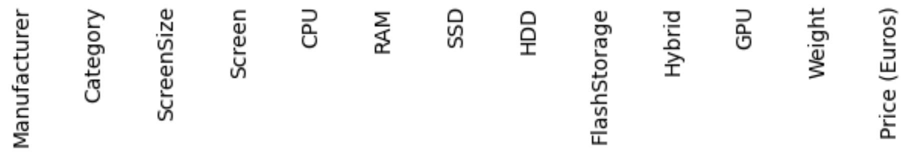
In [ ]:

```python
# Heatmap matrice de corrélation

plt.figure(figsize=(10, 16))
sns.heatmap(cor_mat, annot=True, center=0)
```

Out[ ]:

```
<Axes: >
```


Manufacturer
Category
ScreenSize
Screen
CPU
RAM
SSD
HDD
FlashStorage
Hybrid
GPU
Weight
Price (Euros)
1e12
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

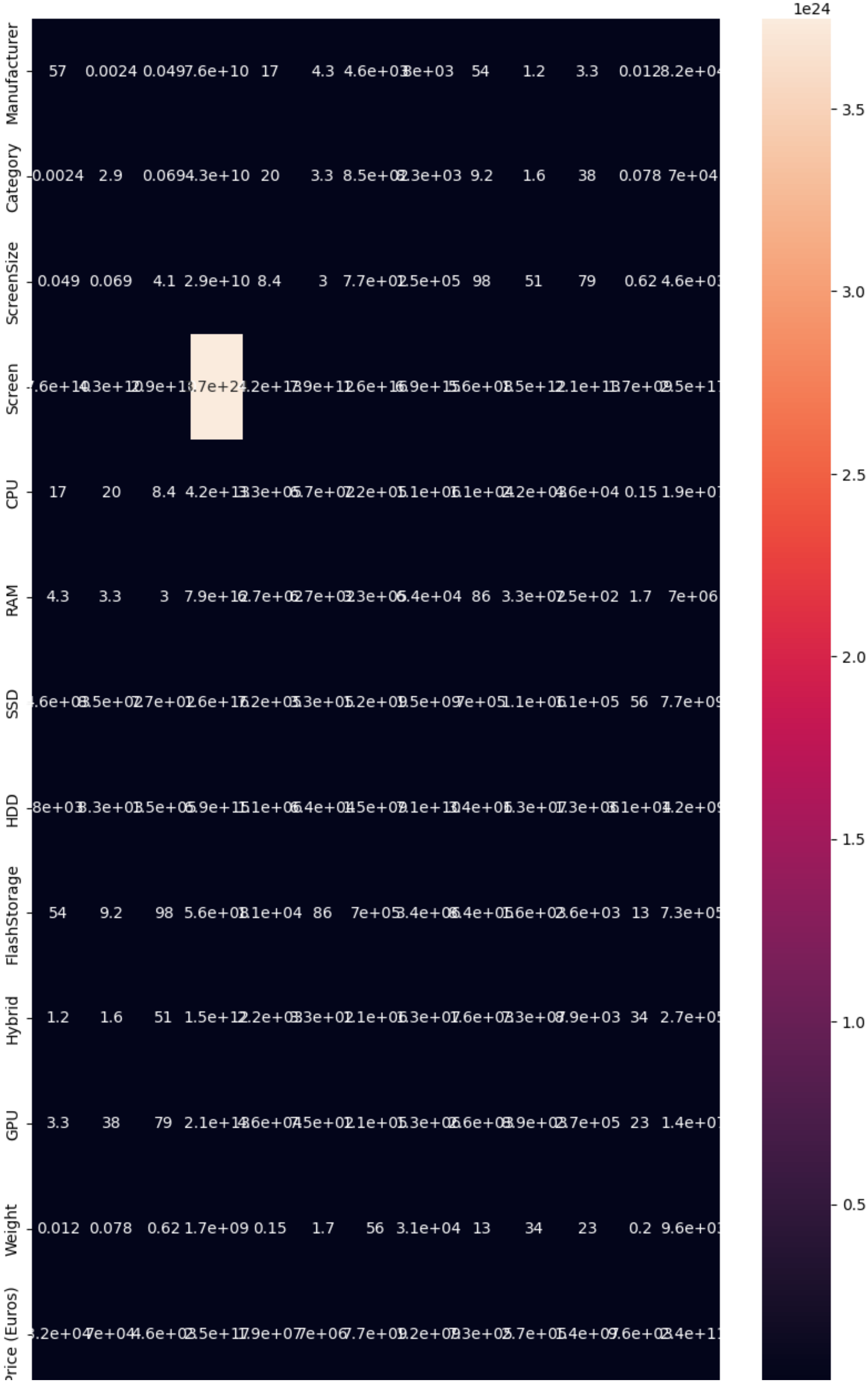


In [ ]:

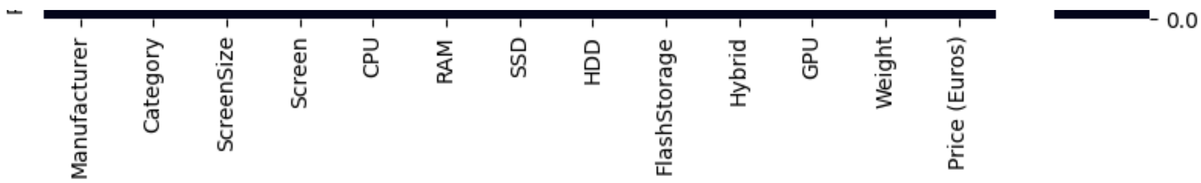
```python
# Heatmap détermination

plt.figure(figsize=(10, 16))
sns.heatmap(det_mat,annot=True)
```

Out[ ]:

```
<Axes: >
```



Manufacturer
Category
ScreenSize
Screen
CPU
RAM
SSD
HDD
FlashStorage
Hybrid
GPU
Weight
Price (Euros)
1e24
3.5
3.0
2.5
2.0
1.5
1.0
0.5



ACP et cercle de corrélation

In [ ]:
```
#On définit le nombre de lignes et de colonnes
nlignes, ncol = laptop_data_quantified.shape

# On définit la liste des colonnes en retirant le prix
feat_names=laptop_data_quantified.columns.tolist()
feat_names.remove('Price (Euros)')
feat_names
```

Out[ ]:
```
['Manufacturer',
 'Category',
 'ScreenSize',
 'Screen',
 'CPU',
 'RAM',
 'SSD',
 'HDD',
 'FlashStorage',
 'Hybrid',
 'GPU',
 'Weight']
```

In [ ]:
```
#on duplique les données
laptop_data_quantified_cr = laptop_data_quantified.copy()

#on créée le scaler et on l'applique pour centrer et réduire sur les features chois
scaler = StandardScaler()
laptop_data_quantified_cr[feat_names] = scaler.fit_transform(laptop_data_quantified
```

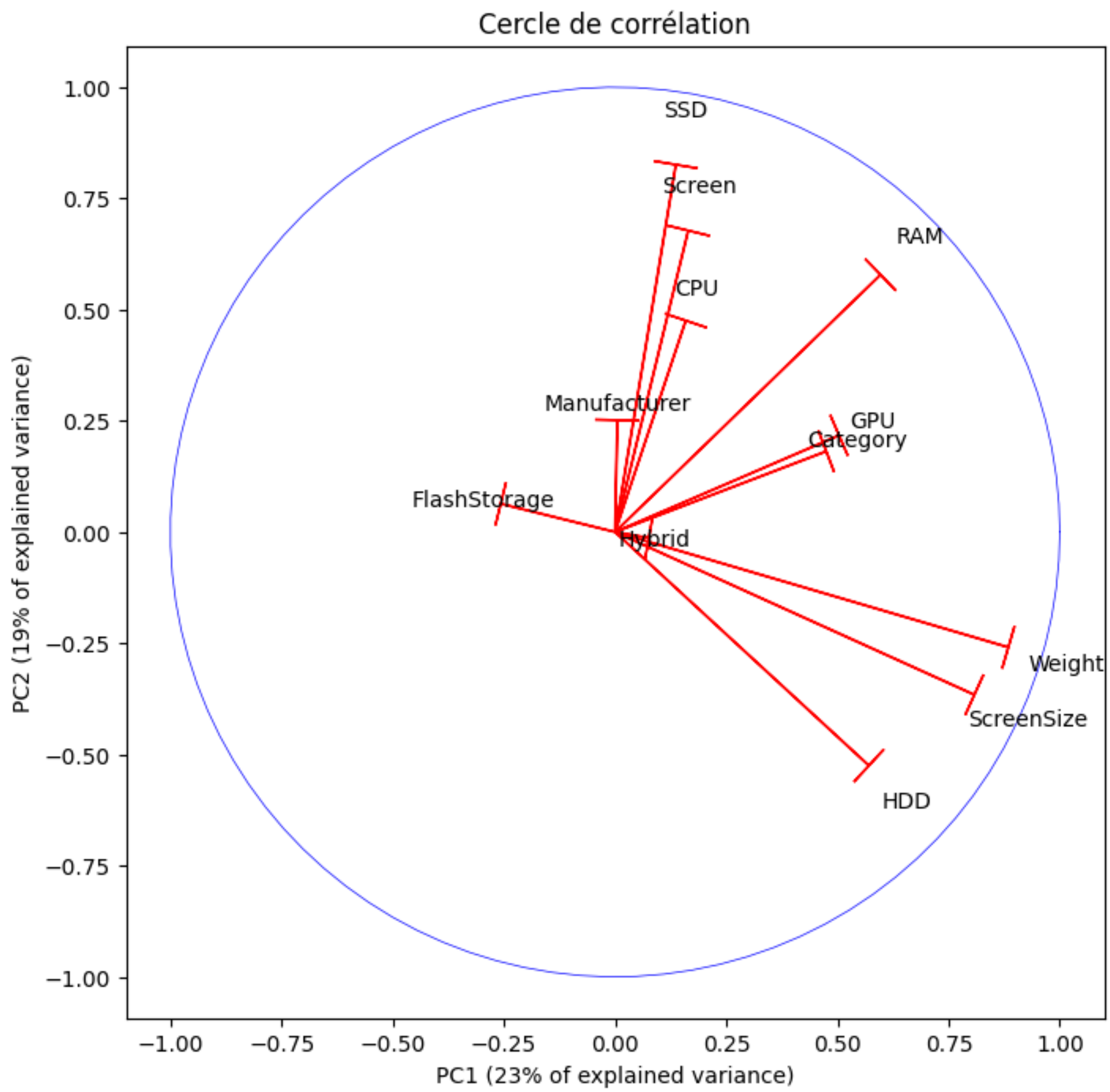
In [ ]:
```
#on créée un object PCA :
pca = PCA(n_components=12)

#on l'applique sur les données normalisées et on stocke le résultat dans la variabl
laptop_data_quantified_pca = pca.fit_transform(laptop_data_quantified_cr[feat_names

# PC1 et PC2
pc1 = int(round(pca.explained_variance_ratio_[0] * 100))
pc2 = int(round(pca.explained_variance_ratio_[1] * 100))
```

In [ ]:
```
#On affiche la variance expliquée
pca.explained_variance_ratio_
```

Out[ ]:
```
array([0.22808777, 0.19363632, 0.11233359, 0.08576497, 0.08029108,
       0.07039976, 0.06675371, 0.05472196, 0.04703769, 0.03308915,
       0.01714305, 0.01074096])
```

In [ ]:
```
# Code cercle de corrélation
```

```python
#on aura besoin de connaitre les nombre de ligne et de colonnes des données.
#Pour les colonnes, on fait -1 parce que la colonne des codes postaux ne compte pas
nlignes, ncol = laptop_data_quantified.shape
ncol = min(ncol - 1, pca.components_.shape[0])

#on définit les valeurs propres à partir de la variance expliquée
vpropres = pca.explained_variance_
sqrt_vpropres = np.sqrt(vpropres)

#on créée une matrice de corrélation
corvar = np.zeros((ncol, ncol))

#on remplit la matrice
for k in range(ncol):
    corvar[:, k] = pca.components_[k, :] * sqrt_vpropres[k]


#on créée une figure vide pour le cercle de corrélation
fig, ax = plt.subplots(figsize=(8, 8))

#on prépare un cercle unitaire
an = np.linspace(0, 2 * np.pi, 100)
ax.plot(np.cos(an), np.sin(an), 'b', linewidth=0.5)

#on boucle sur les différences variables d'origine
for i in range(0, corvar.shape[0]):
    ax.arrow(0, 0,  # on trace les flèches depuis l'origine [0,0]
             corvar[i, 0],  #PC1
             corvar[i, 1], #PC2
             head_width=0.1,
             head_length=0,
             color = 'r')
    #nom des variables au bout des flèches
    ax.text(corvar[i, 0]* 1.15, corvar[i, 1] * 1.15,
    feat_names[i], color = 'k', ha = 'center',
    va = 'center')

#on rajoute titres et légendes
ax.axis('equal')
ax.set_xlabel("PC1 ({0}% of explained variance)".format(pc1, fontsize=12));
ax.set_ylabel("PC2 ({0}% of explained variance)".format(pc2, fontsize=12));
ax.set_title('Cercle de corrélation');
```



Partie 3 : regression linéaire pour prédire le prix du PC

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
import statistics
```

Séparation du dataset d'entrainement et du dataset de test, avec 20% des données alloués au test

In [ ]:
```
xtrain, xtest, ytrain, ytest = train_test_split(laptop_data_quantified[["Manufactur
```

In [ ]:
```
regression = LinearRegression()
```

```
regression.fit(xtrain,ytrain)
```

Out[ ]:

▾ LinearRegression

LinearRegression()

In [ ]:
```
xtrain.head()
```

Out[ ]:

	Manufacturer	Category	ScreenSize	Screen	CPU	RAM	SSD	HDD	FlashStorage
1275	5	4	17.3	3686400	35	16.0	512	1000	0
584	2	1	14.0	2073600	33	8.0	256	0	0
160	4	4	17.3	2073600	34	8.0	128	1000	0
871	2	4	17.3	2073600	32	16.0	256	1000	0
113	4	4	15.6	2073600	32	8.0	128	1000	0

In [ ]:
```
ytrain.iloc[0]
```

Out[ ]:
```
Price (Euros)    2758.0
Name: 1275, dtype: float64
```

In [ ]:
```
# Faire une prédiction
Manufacturer = [[1]]
Category = [[2]]
ScreenSize = [[10.7]]
Screen = [[2073600]]
CPU = [[10]]
RAM= [[16.0]]
SSD= [[256]]
HDD= [[2000]]
FlashStorage= [[0]]
GPU = [[3]]
Weight= [[2.4]]

X_pred = np.column_stack((Manufacturer, Category, ScreenSize, Screen, CPU, RAM, SSD

prix_predit = regression.predict(X_pred)
```

```
C:\Users\Louis\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfr
a8p0\LocalCache\local-packages\Python311\site-packages\sklearn\base.py:465: UserWarn
ing: X does not have valid feature names, but LinearRegression was fitted with featu
re names
  warnings.warn(
```

In [ ]:
```
print (prix_predit)
```

```
[[1713.78862386]]
```

In [ ]:
```
# Coefficients de la régression
results = {
 "label":["constante"]+list(xtrain.columns)+["Score"],
  "coeff": np.concatenate((regression.intercept_.reshape(1,-1), regression.coef_,re
```

```
}

pd.DataFrame(results)
```

Out[ ]:

	label	coeff
0	constante	412.225127
1	Manufacturer	9.195995
2	Category	45.767271
3	ScreenSize	-43.556460
4	Screen	0.000088
5	CPU	1.309082
6	RAM	58.615127
7	SSD	1.094481
8	HDD	-0.034522
9	FlashStorage	0.661689
10	GPU	1.171058
11	Weight	133.150362
12	Score	0.677576

In [ ]:

```
final_pred = regression.predict(xtest)
r2_score(final_pred, ytest["Price (Euros)"])
```

Out[ ]: 0.21152740298978456

Caractère signicatif conjoint des coefficients

On calcule les caractéristiques en terme de SCE, SCR et SCT de cette régression. Le premier test d'hypothèse consiste à savoir le caractère significatif total de la régression.

In [ ]:

```
# Je n'ai pas réussi à régler l'erreur ici

#calcul de SCR
SCR =((prix_predit-regression.predict(X_pred))**2).sum()
#calcul de SCT
SCT =((prix_predit-statistics.mean(prix_predit))**2).sum()
#calcul de SCE
SCE =((regression.predict(X_pred)-statistics.mean(prix_predit))**2).sum()
n = len(prix_predit)
p = X_pred.shape[1]
R = SCE/SCT;
F = (SCE/p)/(SCR/(n-p-1))

results = {
```

```python
 "label":["SCR"]+["SCE"]+["SCT"]+["Score R2"] + ["Fisher"],
  "coeff": np.concatenate((SCR, SCE,SCT, R, F),axis=None),
}

pd.DataFrame(results)
```

```
C:\Users\Louis\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfr
a8p0\LocalCache\local-packages\Python311\site-packages\sklearn\base.py:465: UserWarn
ing: X does not have valid feature names, but LinearRegression was fitted with featu
re names
  warnings.warn(
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
File C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2032.0_x
64__qbz5n2kfra8p0\Lib\statistics.py:327, in _exact_ratio(x)
    325 try:
    326     # x may be an Integral ABC.
--> 327     return (x.numerator, x.denominator)
    328 except AttributeError:

AttributeError: 'numpy.ndarray' object has no attribute 'numerator'

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
Cell In[50], line 6
      4 SCR =((prix_predit-regression.predict(X_pred))**2).sum()
      5 #calcul de SCT
----> 6 SCT =((prix_predit-statistics.mean(prix_predit))**2).sum()
      7 #calcul de SCE
      8 SCE =((regression.predict(X_pred)-statistics.mean(prix_predit))**2).sum()

File C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2032.0_x
64__qbz5n2kfra8p0\Lib\statistics.py:430, in mean(data)
    414 def mean(data):
    415     """Return the sample arithmetic mean of data.
    416 
    417     >>> mean([1, 2, 3, 4, 4])
   (...)
    428     If ``data`` is empty, StatisticsError will be raised.
    429     """
--> 430     T, total, n = _sum(data)
    431     if n < 1:
    432         raise StatisticsError('mean requires at least one data point')

File C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2032.0_x
64__qbz5n2kfra8p0\Lib\statistics.py:193, in _sum(data)
    191 for typ, values in groupby(data, type):
    192     types_add(typ)
--> 193     for n, d in map(_exact_ratio, values):
    194         count += 1
    195         partials[d] = partials_get(d, 0) + n

File C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2032.0_x
64__qbz5n2kfra8p0\Lib\statistics.py:330, in _exact_ratio(x)
    328 except AttributeError:
    329     msg = f"can't convert type '{type(x).__name__}' to numerator/denominato
r"
--> 330     raise TypeError(msg)

TypeError: can't convert type 'ndarray' to numerator/denominator
```

In [ ]:

```
# Pour retrouver la matrice de covariance

X = laptop_data_quantified[['Manufacturer', 'Category','ScreenSize',    'Screen',

scaler = StandardScaler()
```

```
# suppression de la colonne Annual
scale = scaler.set_output(transform="pandas")

# Il faudra toujours trouver les variables centrées et réduites
scaled_db = scale.fit_transform(X)
```

In [ ]:

```
# Affichage de la matrice de covariance

n = scaled_db.shape[0]
p = scaled_db.shape[1]
S = np.zeros((n,p));
n = scaled_db.shape[0]
p = scaled_db.shape[1]
S = np.zeros((n,p))

S[:,0] = np.array(scaled_db['Manufacturer'])
S[:,1] = np.array(scaled_db['Category'])
S[:,2] = np.array(scaled_db['ScreenSize'])
S[:,3] = np.array(scaled_db['Screen'])
S[:,4] = np.array(scaled_db['CPU'])
S[:,5] = np.array(scaled_db['RAM'])
S[:,6] = np.array(scaled_db['SSD'])
S[:,7] = np.array(scaled_db['HDD'])
S[:,8] = np.array(scaled_db['FlashStorage'])
S[:,9] = np.array(scaled_db['GPU'])
S[:,10] = np.array(scaled_db['Weight'])

Rxx = np.dot(S.T,S)
print('La matrice de covariance est : \n', Rxx)
Rxx = np.dot(S.T,S)
print('La matrice de covariance est : \n', Rxx)
```

```
La matrice de covariance est :
 [[ 1.30300000e+03  1.79474571e+01 -7.37760812e+01  9.37377498e+01
   8.25405963e+01  1.93967565e+02  1.71163695e+02 -8.18955935e+01
  -1.15100713e+02 -3.79524073e+01 -7.72322539e+01]
 [ 1.79474571e+01  1.30300000e+03  1.84487882e+02  1.49401123e+02
   1.85092125e+02  3.58618746e+02  1.55767544e+02  1.77536232e+02
  -1.00309043e+02  2.72064544e+02  4.19552051e+02]
 [-7.37760812e+01  1.84487882e+02  1.30300000e+03 -1.12578113e+02
  -1.10489406e+02  3.10104614e+02 -1.35233517e+02  6.91595180e+02
  -2.99433816e+02  3.57578591e+02  1.07840332e+03]
 [ 9.37377498e+01  1.49401123e+02 -1.12578113e+02  1.30300000e+03
   2.53668982e+02  5.16455106e+02  6.32841049e+02 -1.50689790e+02
   7.32309693e-01  1.87878281e+02 -5.73760245e+01]
 [ 8.25405963e+01  1.85092125e+02 -1.10489406e+02  2.53668982e+02
   1.30300000e+03  2.76755772e+02  2.46294658e+02 -1.12495907e+02
   1.84654148e+02  5.11195353e+02 -3.12694753e+01]
 [ 1.93967565e+02  3.58618746e+02  3.10104614e+02  5.16455106e+02
   2.76755772e+02  1.30300000e+03  7.86966742e+02  1.25463110e+02
  -7.84102542e+01  3.07570071e+02  5.00187944e+02]
 [ 1.71163695e+02  1.55767544e+02 -1.35233517e+02  6.32841049e+02
   2.46294658e+02  7.86966742e+02  1.30300000e+03 -5.16279412e+02
  -1.91760874e+02  1.01595667e+02 -7.82211276e+01]
 [-8.18955935e+01  1.77536232e+02  6.91595180e+02 -1.50689790e+02
  -1.12495907e+02  1.25463110e+02 -5.16279412e+02  1.30300000e+03
  -1.53308480e+02  1.25075888e+02  6.70381838e+02]
 [-1.15100713e+02 -1.00309043e+02 -2.99433816e+02  7.32309693e-01
   1.84654148e+02 -7.84102542e+01 -1.91760874e+02 -1.53308480e+02
   1.30300000e+03  9.56362925e+01 -2.34430575e+02]
 [-3.79524073e+01  2.72064544e+02  3.57578591e+02  1.87878281e+02
   5.11195353e+02  3.07570071e+02  1.01595667e+02  1.25075888e+02
   9.56362925e+01  1.30300000e+03  4.08366294e+02]
 [-7.72322539e+01  4.19552051e+02  1.07840332e+03 -5.73760245e+01
  -3.12694753e+01  5.00187944e+02 -7.82211276e+01  6.70381838e+02
  -2.34430575e+02  4.08366294e+02  1.30300000e+03]]
La matrice de covariance est :
 [[ 1.30300000e+03  1.79474571e+01 -7.37760812e+01  9.37377498e+01
   8.25405963e+01  1.93967565e+02  1.71163695e+02 -8.18955935e+01
  -1.15100713e+02 -3.79524073e+01 -7.72322539e+01]
 [ 1.79474571e+01  1.30300000e+03  1.84487882e+02  1.49401123e+02
   1.85092125e+02  3.58618746e+02  1.55767544e+02  1.77536232e+02
  -1.00309043e+02  2.72064544e+02  4.19552051e+02]
 [-7.37760812e+01  1.84487882e+02  1.30300000e+03 -1.12578113e+02
  -1.10489406e+02  3.10104614e+02 -1.35233517e+02  6.91595180e+02
  -2.99433816e+02  3.57578591e+02  1.07840332e+03]
 [ 9.37377498e+01  1.49401123e+02 -1.12578113e+02  1.30300000e+03
   2.53668982e+02  5.16455106e+02  6.32841049e+02 -1.50689790e+02
   7.32309693e-01  1.87878281e+02 -5.73760245e+01]
 [ 8.25405963e+01  1.85092125e+02 -1.10489406e+02  2.53668982e+02
   1.30300000e+03  2.76755772e+02  2.46294658e+02 -1.12495907e+02
   1.84654148e+02  5.11195353e+02 -3.12694753e+01]
 [ 1.93967565e+02  3.58618746e+02  3.10104614e+02  5.16455106e+02
   2.76755772e+02  1.30300000e+03  7.86966742e+02  1.25463110e+02
  -7.84102542e+01  3.07570071e+02  5.00187944e+02]
 [ 1.71163695e+02  1.55767544e+02 -1.35233517e+02  6.32841049e+02
   2.46294658e+02  7.86966742e+02  1.30300000e+03 -5.16279412e+02
  -1.91760874e+02  1.01595667e+02 -7.82211276e+01]
```

```
[-8.18955935e+01  1.77536232e+02  6.91595180e+02 -1.50689790e+02
 -1.12495907e+02  1.25463110e+02 -5.16279412e+02  1.30300000e+03
 -1.53308480e+02  1.25075888e+02  6.70381838e+02]
[-1.15100713e+02 -1.00309043e+02 -2.99433816e+02  7.32309693e-01
  1.84654148e+02 -7.84102542e+01 -1.91760874e+02 -1.53308480e+02
  1.30300000e+03  9.56362925e+01 -2.34430575e+02]
[-3.79524073e+01  2.72064544e+02  3.57578591e+02  1.87878281e+02
  5.11195353e+02  3.07570071e+02  1.01595667e+02  1.25075888e+02
  9.56362925e+01  1.30300000e+03  4.08366294e+02]
[-7.72322539e+01  4.19552051e+02  1.07840332e+03 -5.73760245e+01
 -3.12694753e+01  5.00187944e+02 -7.82211276e+01  6.70381838e+02
 -2.34430575e+02  4.08366294e+02  1.30300000e+03]]
```

In [ ]:
```
# Affichage de l'inverse de la matrice de covariance

invRxx= np.linalg.inv(Rxx);
print('La matrice inverse de covariance est : \n',invRxx )
```

```
La matrice inverse de covariance est :
 [[ 8.10293977e-04 -2.22113484e-06 -2.05565731e-05  7.04753235e-06
  -4.34308032e-05 -1.91858921e-04  4.56555627e-05  3.77626168e-05
   9.25594734e-05  3.72468954e-05  1.26966278e-04]
 [-2.22113484e-06  9.52252432e-04  3.76237622e-04 -3.34189798e-05
  -6.83569089e-05 -5.77041874e-05 -5.31887105e-05 -6.66177997e-05
   6.08866959e-05 -8.45313955e-05 -5.30565265e-04]
 [-2.05565731e-05  3.76237622e-04  2.87754594e-03  6.65272203e-05
   1.32750324e-04  3.07715249e-04 -2.64800412e-04 -4.78356308e-04
   2.01053815e-04 -2.38396192e-04 -2.27481720e-03]
 [ 7.04753235e-06 -3.34189798e-05  6.65272203e-05  1.07314236e-03
  -4.12899115e-05 -1.78439508e-04 -4.37489537e-04 -1.31183309e-04
  -4.00397502e-05 -1.00607360e-04  1.36436714e-04]
 [-4.34308032e-05 -6.83569089e-05  1.32750324e-04 -4.12899115e-05
   1.00483017e-03 -1.13709388e-04 -5.43564343e-05 -8.98974885e-06
  -9.21217085e-05 -3.99811036e-04  8.56040740e-05]
 [-1.91858921e-04 -5.77041874e-05  3.07715249e-04 -1.78439508e-04
  -1.13709388e-04  2.01095483e-03 -1.31373364e-03 -4.75564640e-04
  -2.20350425e-04 -3.85934321e-05 -8.91747142e-04]
 [ 4.56555627e-05 -5.31887105e-05 -2.64800412e-04 -4.37489537e-04
  -5.43564343e-05 -1.31373364e-03  2.19051302e-03  9.02186609e-04
   3.71245929e-04  5.74289668e-05  4.38856488e-04]
 [ 3.77626168e-05 -6.66177997e-05 -4.78356308e-04 -1.31183309e-04
  -8.98974885e-06 -4.75564640e-04  9.02186609e-04  1.48955591e-03
   1.40356160e-04  9.51542937e-05 -1.20617215e-04]
 [ 9.25594734e-05  6.08866959e-05  2.01053815e-04 -4.00397502e-05
  -9.21217085e-05 -2.20350425e-04  3.71245929e-04  1.40356160e-04
   9.12320150e-04 -9.44022787e-05  4.38975787e-05]
 [ 3.72468954e-05 -8.45313955e-05 -2.38396192e-04 -1.00607360e-04
  -3.99811036e-04 -3.85934321e-05  5.74289668e-05  9.51542937e-05
  -9.44022787e-05  1.07973952e-03 -1.73368147e-04]
 [ 1.26966278e-04 -5.30565265e-04 -2.27481720e-03  1.36436714e-04
   8.56040740e-05 -8.91747142e-04  4.38856488e-04 -1.20617215e-04
   4.38975787e-05 -1.73368147e-04  3.32954589e-03]]
```

Partie 4 : Classification supervisée

Nous allons maintenant faire une classification supervisée avec du clustering. Pour l'analyse en composantes principales, on va prendre un dataset composé uniquement de 15 ordinateurs tirés aléatoirement à des fins de lisibilité.

In [ ]:
```
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
import matplotlib.cm as cm
```

In [ ]:
```
# Nouveau dataset avec les 15 premiers ordinateurs (à des fins de lisibilité)
laptop_data_quantified15=laptop_data_quantified.head(15)
```

In [ ]:
```
# Analyses en composantes principales (PCA)



scaler = StandardScaler()
# suppression des colonnes inutiles
db = laptop_data_quantified15
scale = scaler.set_output(transform="pandas")

# Il faudra toujours trouver les variables centrées et réduites
scaled_db = scale.fit_transform(db)

pca = PCA()
pca.fit(scaled_db)
```

Out[ ]:

▾ PCA

PCA()

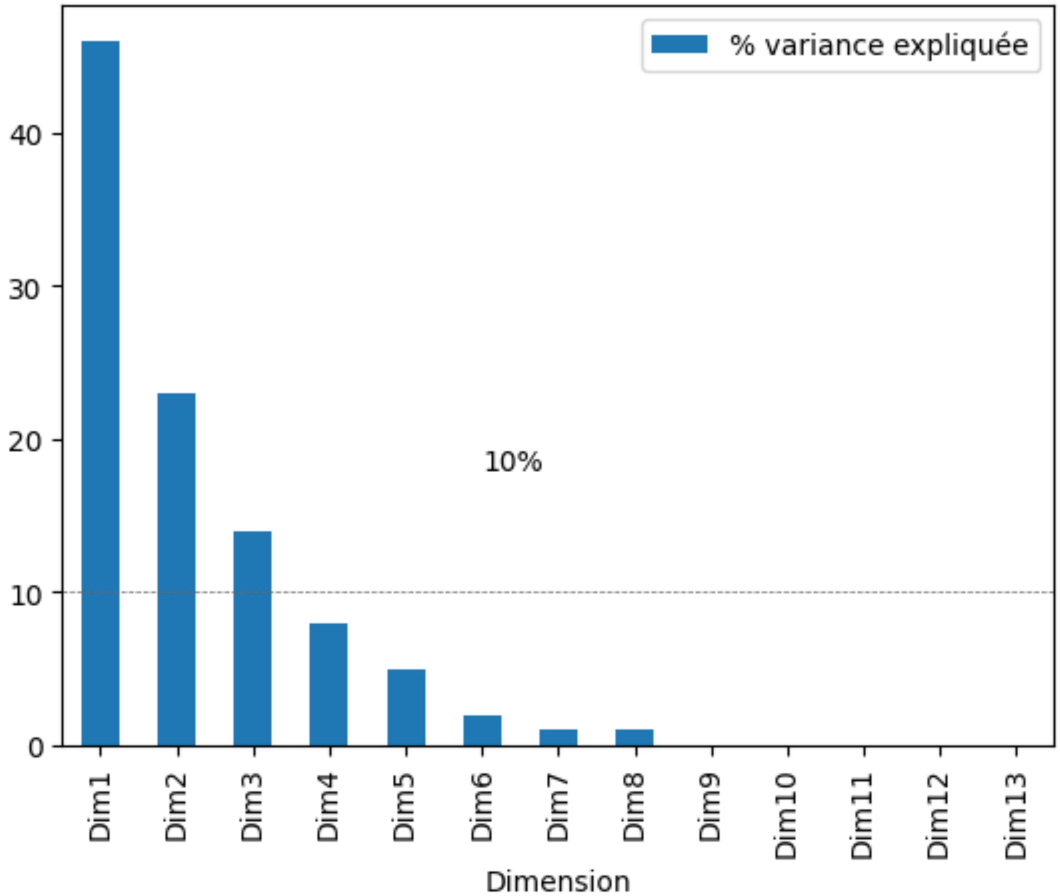
In [ ]:
```
# On imprime les valeurs des variances expliquées, des proportions de variance expl

eig = pd.DataFrame(
    {
        "Dimension" : ["Dim" + str(x + 1) for x in range(13)],
        "Variance expliquée" : pca.explained_variance_,
        "% variance expliquée" : np.round(pca.explained_variance_ratio_ * 100),
        "% cum. var. expliquée" : np.round(np.cumsum(pca.explained_variance_ratio_)
    }
)
eig
```

Out[ ]:

	Dimension	Variance expliquée	% variance expliquée	% cum. var. expliquée
0	Dim1	5.870215e+00	46.0	46.0
1	Dim2	2.925737e+00	23.0	68.0
2	Dim3	1.752880e+00	14.0	82.0
3	Dim4	1.019402e+00	8.0	90.0
4	Dim5	6.455039e-01	5.0	95.0
5	Dim6	2.855878e-01	2.0	97.0
6	Dim7	1.510294e-01	1.0	98.0
7	Dim8	1.124081e-01	1.0	99.0
8	Dim9	5.853067e-02	0.0	100.0
9	Dim10	3.212417e-02	0.0	100.0
10	Dim11	3.724986e-03	0.0	100.0
11	Dim12	2.419074e-32	0.0	100.0
12	Dim13	1.616512e-35	0.0	100.0

In [ ]:

```
# On visualise la variance expliquée un graphique

eig.plot.bar(x = "Dimension", y = "% variance expliquée") # permet un diagramme en
plt.text(5, 18, "10%") # ajout de texte
plt.axhline(y = 10, linewidth = .5, color = "dimgray", linestyle = "--") # ligne 17
plt.show()
```



In [ ]:
```
db_pca = pca.transform(scaled_db)
```

In [ ]:
```
# Nouveau dataset : laptops_data_quantified + la colonne "Model Name" utile pour la
laptop_data_original = pd.read_csv('laptops_final.csv', sep=",", encoding='latin1')
laptop_data_quantified15_models = laptop_data_quantified15.copy()
laptop_data_quantified15_models['Model Name'] = laptop_data_original['Model Name']

# Transformation en DataFrame pandas
data_pca_df = pd.DataFrame({
    "Dim1" : db_pca[:,0],
    "Dim2" : db_pca[:,1],
    "Modèle" : laptop_data_quantified15_models["Model Name"],
})

# Résultat (premières lignes)
data_pca_df.head()
```

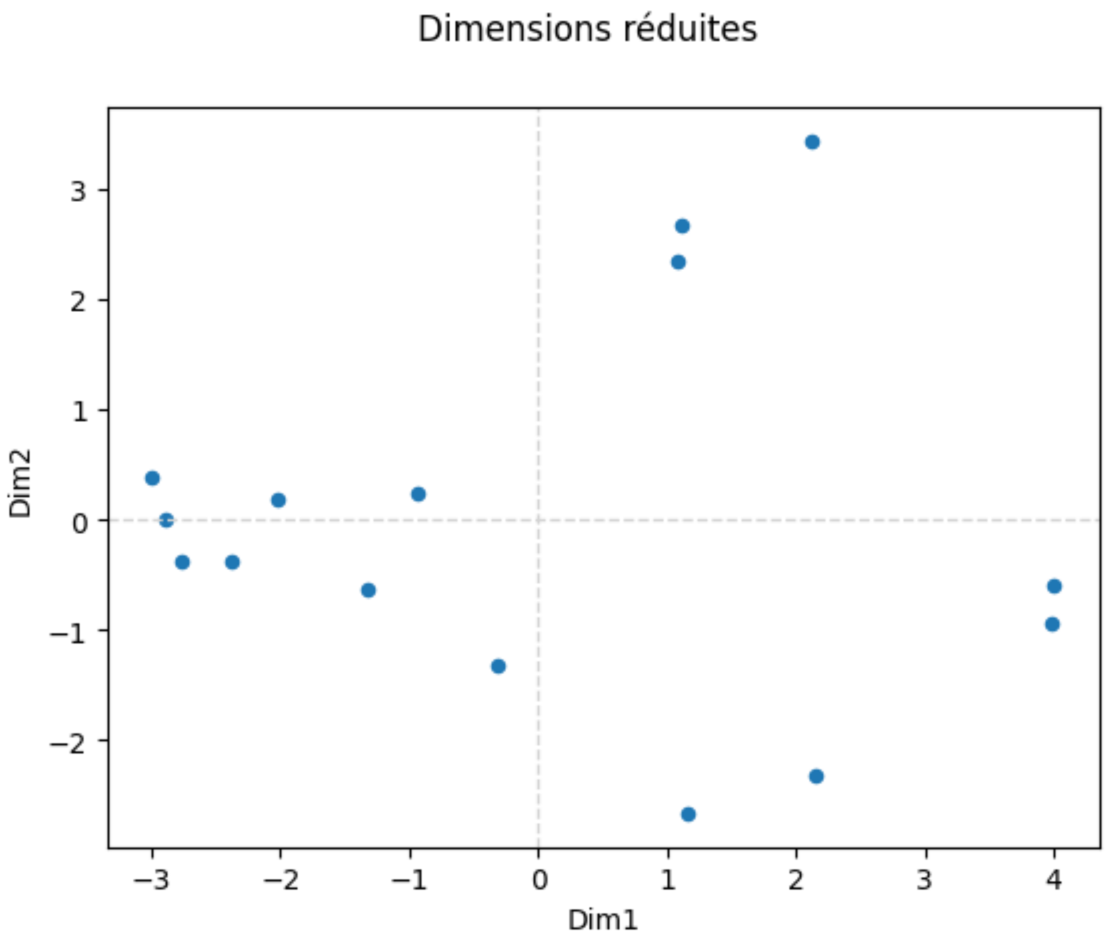
Out[ ]:

	Dim1	Dim2	Modèle
0	3.986369	-0.948248	Macbook Air
1	4.000815	-0.602577	MacBook Air
2	2.151037	-2.324067	TravelMate B
3	1.162120	-2.665722	Swift SF114-31-P5HY
4	-0.309715	-1.325844	R558UA-DM966T (i5-7200U/8GB/128GB/FHD/W10)

In [ ]:
```
# utilisation de subplots nécessaire car annotation du graphique
fig, ax = plt.subplots()
fig.suptitle("Dimensions réduites")
data_pca_df.plot.scatter("Dim1", "Dim2", ax = ax) # l'option ax permet de placer le
# Ajout des axes
ax.axvline(x = 0, color = 'lightgray', linestyle = '--', linewidth = 1)
ax.axhline(y = 0, color = 'lightgray', linestyle = '--', linewidth = 1)
# boucle sur chaque pays
for k in data_pca_df.iterrows():
    # annotation uniquement si valeur absolue sur une de 2 dimensions importantes (
        ax.annotate(k[1]["Model Name"], (k[1]['Dim1'], k[1]['Dim2']), fontsize = 9)
plt.xlabel("Dimension 1 (90%)")
plt.ylabel("Dimension 2 (9%)")
plt.suptitle("Premier plan factoriel (99%)")
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\Loc
alCache\local-packages\Python311\site-packages\pandas\core\indexes\base.py:3652, in
Index.get_loc(self, key)
   3651 try:
-> 3652     return self._engine.get_loc(casted_key)
   3653 except KeyError as err:

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\Loc
alCache\local-packages\Python311\site-packages\pandas\_libs\index.pyx:147, in panda
s._libs.index.IndexEngine.get_loc()

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\Loc
alCache\local-packages\Python311\site-packages\pandas\_libs\index.pyx:176, in panda
s._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.PyObjec
tHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.PyObjec
tHashTable.get_item()

KeyError: 'Model Name'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[61], line 11
      8 # boucle sur chaque pays
      9 for k in data_pca_df.iterrows():
     10     # annotation uniquement si valeur absolue sur une de 2 dimensions import
antes (valeurs choisies empiriquement)
---> 11         ax.annotate(k[1]["Model Name"], (k[1]['Dim1'], k[1]['Dim2']), fontsi
ze = 9)
     12 plt.xlabel("Dimension 1 (90%)")
     13 plt.ylabel("Dimension 2 (9%)")

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\Loc
alCache\local-packages\Python311\site-packages\pandas\core\series.py:1007, in Serie
s.__getitem__(self, key)
   1004     return self._values[key]
   1006 elif key_is_scalar:
-> 1007     return self._get_value(key)
   1009 if is_hashable(key):
   1010     # Otherwise index.get_value will raise InvalidIndexError
   1011     try:
   1012         # For labels that don't resolve as scalars like tuples and frozenset
s

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\Loc
alCache\local-packages\Python311\site-packages\pandas\core\series.py:1116, in Serie
s._get_value(self, label, takeable)
   1113     return self._values[label]
   1115 # Similar to Index.get_value, but we do not fall back to positional
-> 1116 loc = self.index.get_loc(label)
```

```
   1118 if is_integer(loc):
   1119     return self._values[loc]

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\pandas\core\indexes\base.py:3654, in Index.get_loc(self, key)
   3652     return self._engine.get_loc(casted_key)
   3653 except KeyError as err:
-> 3654     raise KeyError(key) from err
   3655 except TypeError:
   3656     # If we have a listlike key, _check_indexing_error will raise
   3657     #  InvalidIndexError. Otherwise we fall through and re-raise
   3658     #  the TypeError.
   3659     self._check_indexing_error(key)

KeyError: 'Model Name'
```



In [ ]:
```
n = db.shape[0] # nb individus
p = db.shape[1] # nb variables

print(n)
eigval = (n-1) / n * pca.explained_variance_ # valeurs propres
sqrt_eigval = np.sqrt(eigval) # racine carrée des valeurs propres
corvar = np.zeros((p,p)) # matrice vide pour avoir les coordonnées
for k in range(p):
    corvar[:,k] = pca.components_[k,:] * sqrt_eigval[k]
# on modifie pour avoir un dataframe
coordvar = pd.DataFrame({'id': db.columns, 'COR_1': corvar[:,0], 'COR_2': corvar[:,
```

```
coordvar
```

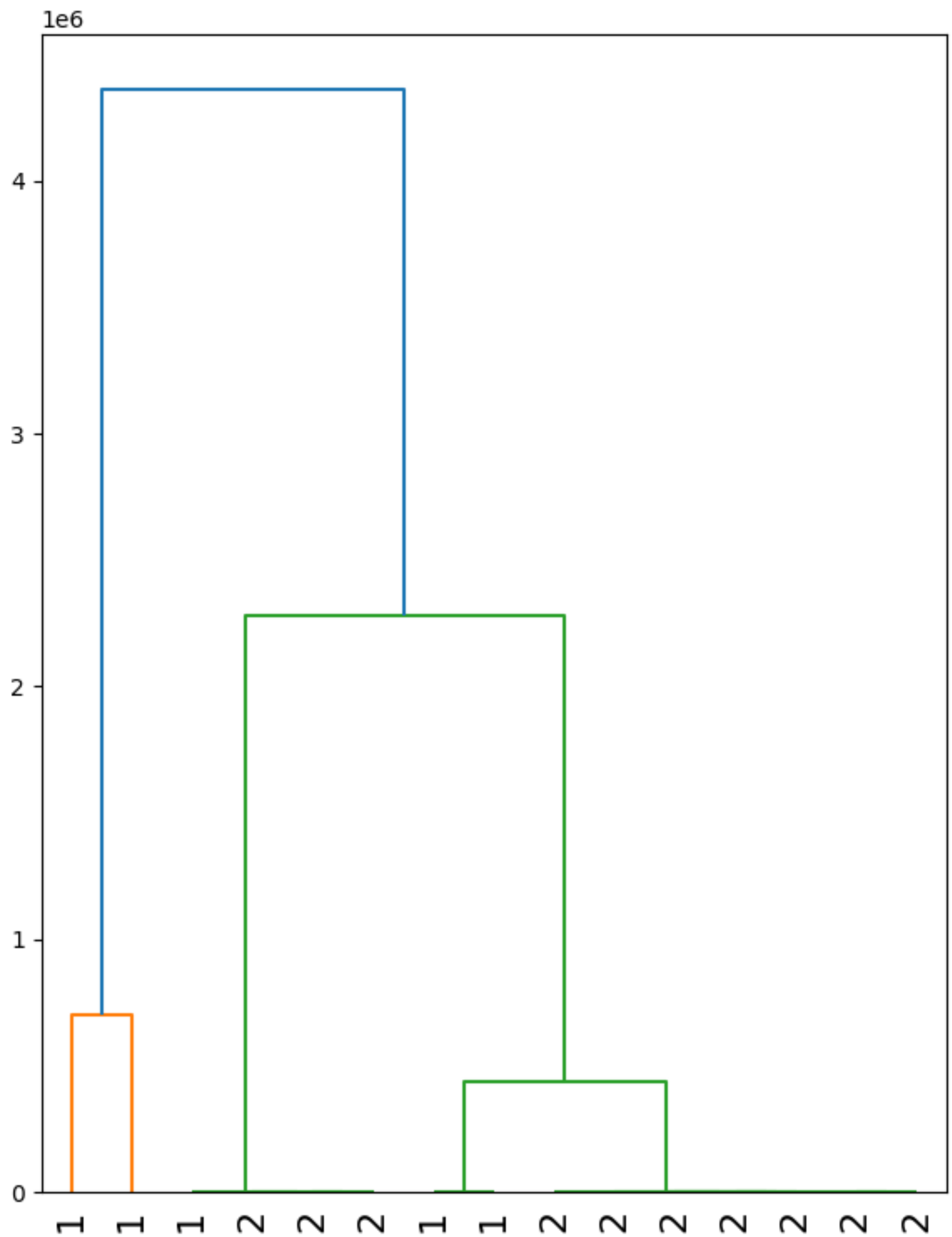
15

Out[ ]:

	id	COR_1	COR_2
0	Manufacturer	-0.486637	0.227522
1	Category	-0.743042	-0.591577
2	ScreenSize	-0.848470	-0.056187
3	Screen	0.209322	0.796945
4	CPU	-0.852505	0.305627
5	RAM	0.577853	0.258998
6	SSD	-0.664037	0.673222
7	HDD	-0.035363	-0.214433
8	FlashStorage	0.727813	-0.596700
9	Hybrid	-0.000000	-0.000000
10	GPU	-0.844290	0.334784
11	Weight	-0.849922	-0.306141
12	Price (Euros)	0.677083	0.684930

In [ ]:

```
# Dendogramme en fonction de la catégorie

labelList = list(laptop_data_quantified15["Category"])
Z2 = linkage(db, method='ward', metric='euclidean')
plt.figure(figsize=(7, 9))
dendrogram(
    Z2,
    leaf_rotation= 90,
    labels=labelList,
    distance_sort='ascending',
    show_leaf_counts=False,
    leaf_font_size=16
)
plt.show()
```



In [ ]:
```python
# Affichage des k-means

kmeansn = KMeans(n_clusters = 2)
kmeansn.fit(db)
kmeansn.cluster_centers_
```

```
C:\Users\Louis\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfr
a8p0\LocalCache\local-packages\Python311\site-packages\sklearn\cluster\_kmeans.py:14
16: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.
4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

Out[ ]:

```
array([[4.30769231e+00, 1.76923077e+00, 1.46384615e+01, 1.40230892e+06,
        5.00000000e+00, 5.53846154e+00, 7.87692308e+01, 9.84615385e+00,
        3.93846154e+01, 0.00000000e+00, 3.92307692e+00, 1.81692308e+00,
        6.03149231e+02],
       [5.00000000e+00, 1.00000000e+00, 1.34000000e+01, 3.74451200e+06,
        4.50000000e+00, 6.00000000e+00, 1.28000000e+02, 0.00000000e+00,
        0.00000000e+00, 0.00000000e+00, 3.50000000e+00, 1.31100000e+00,
        1.21434500e+03]])
```

In [ ]:

```python
db_kn = db.assign(classe = kmeansn.labels_)
db_kn.groupby("classe").mean()
```

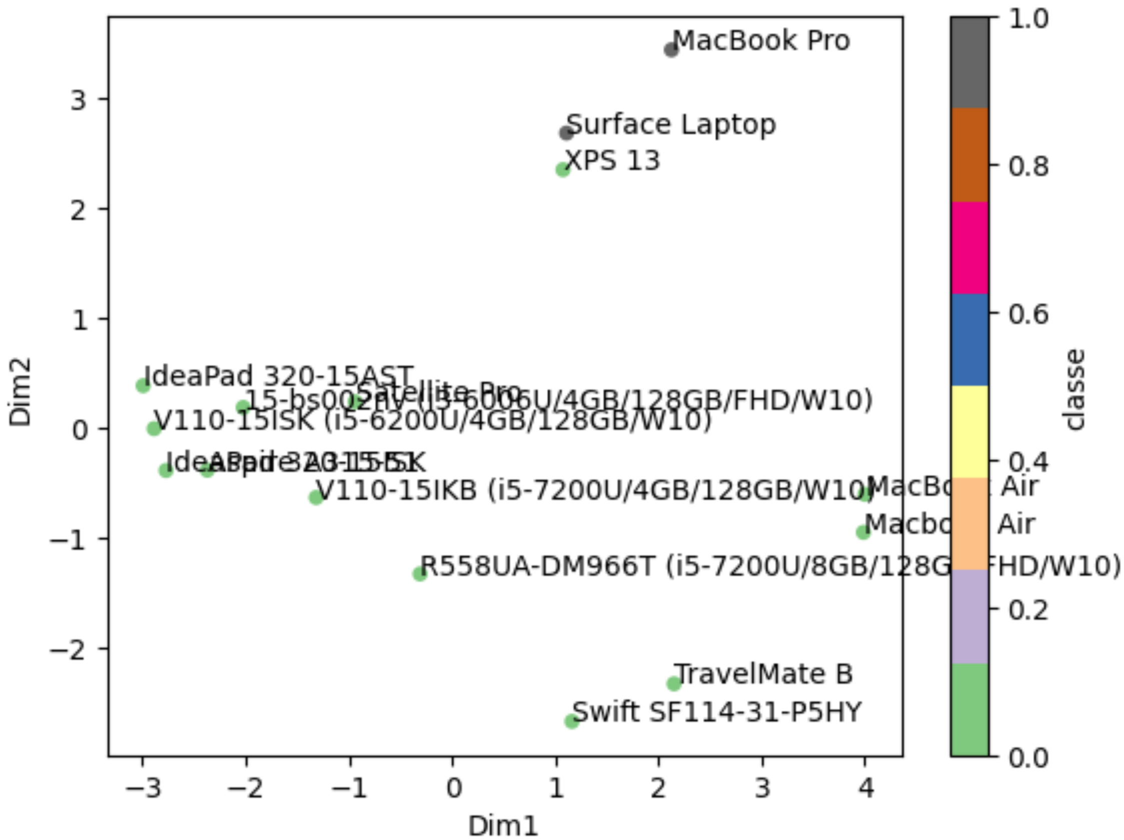
Out[ ]:

	Manufacturer	Category	ScreenSize	Screen	CPU	RAM	SSD	
classe								
0	4.307692	1.769231	14.638462	1.402309e+06	5.0	5.538462	78.769231	9.84
1	5.0	1.0	13.400000	3.744512e+06	4.5	6.000000	128.000000	0.00

In [ ]:

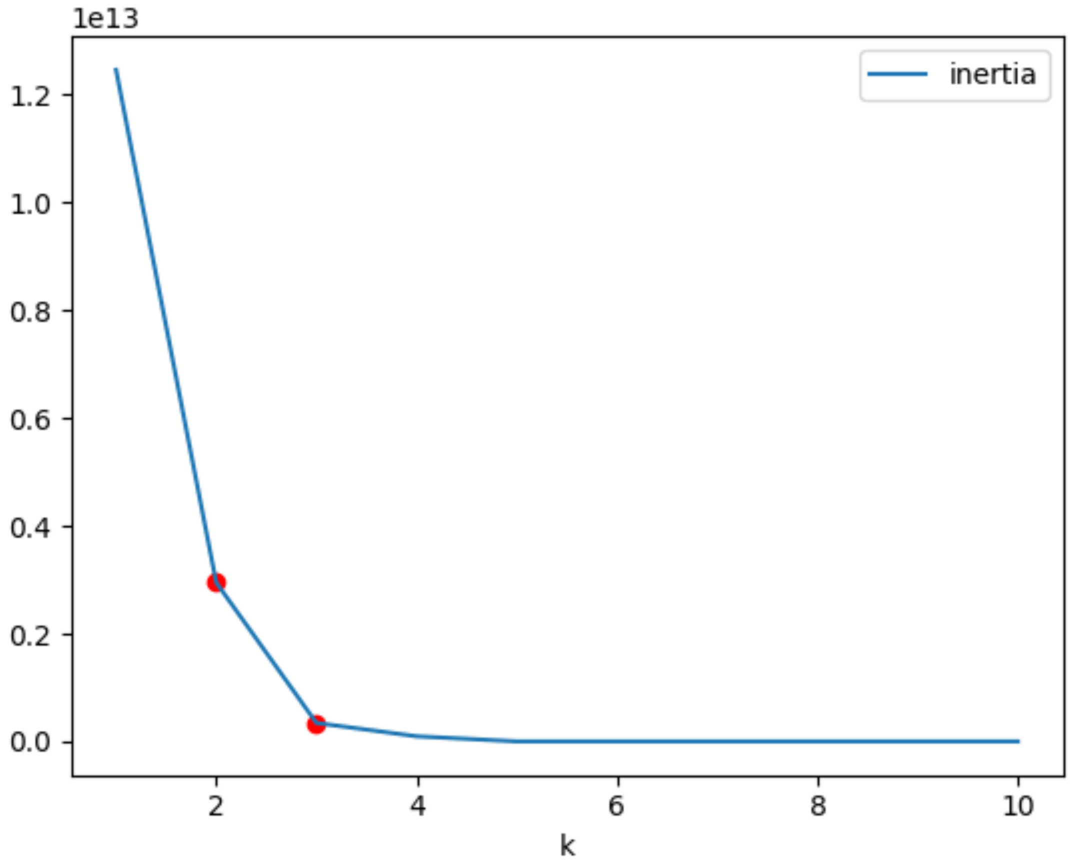
```python
# Affichage du clustering

db_pca_kn = data_pca_df.assign(classe = kmeansn.labels_, modele = laptop_data_quant

num_classes = len(db_pca_kn["classe"].unique())
coul_kmeans = dict(zip(db_pca_kn["classe"].unique(), cm.get_cmap('Accent', num_clas


fig, ax = plt.subplots()

db_pca_kn.plot('Dim1', 'Dim2', kind='scatter', c = "classe",cmap = "Accent", ax=ax)

print(db_pca_kn.iterrows())
for index, row in db_pca_kn.iterrows():
    ax.annotate(row['modele'], (row['Dim1'], row['Dim2']))

plt.show()
```

```
C:\Users\Louis\AppData\Local\Temp\ipykernel_16320\1296275014.py:6: MatplotlibDepreca
tionWarning: The get_cmap function was deprecated in Matplotlib 3.7 and will be remo
ved two minor releases later. Use ``matplotlib.colormaps[name]`` or ``matplotlib.col
ormaps.get_cmap(obj)`` instead.
  coul_kmeans = dict(zip(db_pca_kn["classe"].unique(), cm.get_cmap('Accent', num_cla
sses)(range(num_classes))))
<generator object DataFrame.iterrows at 0x0000022464AFBDE0>
```



In [ ]:

```
# Visualisation de l'inertie

inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters = k, init = "random", n_init = 20).fit(db)
    inertia = inertia + [kmeans.inertia_]
inertia = pd.DataFrame({"k": range(1, 11), "inertia": inertia})
inertia.plot.line(x = "k", y = "inertia")
plt.scatter(2, inertia.query('k == 2')["inertia"], c = "red")
plt.scatter(3, inertia.query('k == 3')["inertia"], c = "red")
plt.show()
```



1e13
inertia
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10
k