Examen 1 - Exploration de donnes

Author: Ricardo Vallejo

Etape 1

1.1. Load Data

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_excel('USA_cars_dataset.xlsx')
df.head(5)
```

Out[1]:

	Unnamed: 0	price	brand	model	year	title_status	mileage	color	vin	lot	state	country	condition
0	0	6300	toyota	cruiser	2008	clean vehicle	274117.0	black	jtezu11f88k007763	159348797	new jersey	usa	10 days left
1	1	2899	ford	se	2011	clean vehicle	190552.0	silver	2fmdk3gc4bbb02217	166951262	tennessee	usa	6 days left
2	2	5350	dodge	mpv	2018	clean vehicle	39590.0	silver	3c4pdcgg5jt346413	167655728	georgia	usa	2 days left
3	3	25000	ford	door	2014	clean vehicle	64146.0	blue	1ftfw1et4efc23745	167753855	virginia	usa	22 hours left
4	4	27700	chevrolet	1500	2018	clean vehicle	6654.0	red	3gcpcrec2jg473991	167763266	florida	usa	22 hours left

1.2. Identifiez les différentes variables et leurs types. Résumez le tout dans un tableau.

In [2]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2499 entries, 0 to 2498
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    2499 non-null   int64  
 1   price         2499 non-null   int64  
 2   brand         2499 non-null   object 
 3   model         2499 non-null   object 
 4   year          2499 non-null   int64  
 5   title_status  2499 non-null   object 
 6   mileage       2499 non-null   float64
 7   color         2499 non-null   object 
 8   vin           2499 non-null   object 
 9   lot           2499 non-null   int64  
 10  state         2499 non-null   object 
 11  country       2499 non-null   object 
 12  condition     2499 non-null   object 
dtypes: float64(1), int64(4), object(8)
memory usage: 253.9+ KB
```

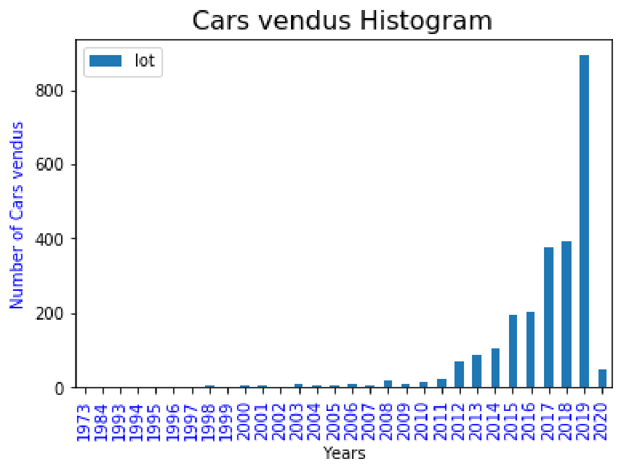
TYPES VARIABLES == Integers variables: price year lot Float ou point dec variables: mileage Object structure: brand model year tittle status color vin state country condition

1.3. Déterminer le nombre de voitures vendues annuellement. Interprétez les résultats

```
In [3]: df[['lot', 'year']].groupby('year').count()
```

```
Out[3]:
        lot
 year
 1973     1
 1984     1
 1993     1
 1994     2
 1995     1
 1996     2
 1997     2
 1998     4
 1999     1
 2000     4
 2001     5
 2002     2
 2003     9
 2004     6
 2005     6
 2006     8
 2007     6
 2008    18
 2009    11
 2010    13
 2011    23
 2012    72
 2013    86
 2014   104
 2015   196
 2016   203
 2017   377
 2018   395
 2019   892
 2020    48
```

```
In [4]: df[['lot', 'year']].groupby('year').count().plot(kind='bar')
        plt.title('Cars vendus Histogram', color = 'black', fontsize = 16)
        plt.xlabel('Years',color = 'black')
        plt.ylabel('Number of Cars vendus', color = 'blue')
        plt.xticks(color = 'blue')
        plt.yticks(color = 'black')
        plt.show()
```



En el anne 2019 sont vendus 892 voitures, cest le anne de plus ventes entre 1973 et 2020. La plupart de vehicles sont vendus entre 2012 et 2019.

Étape 2 : Analyse des prix de voiture

1. À partir du fichier USA_cars_dataset.xlsx, créez une nouvelle structure df2010 qui contient les observations

entre 2010 (inclus) et 2020 (exlus).

```
In [5]: yearscond = list(range(2009, 2020))
```

In [6]:
```
is2010 =  df['year'].isin(yearscond)
```

In [7]:
```
df2010 = df[is2010]

df2010.head(100)
```

Out[7]:

	Unnamed: 0	price	brand	model	year	title_status	mileage	color	vin	lot	state	country	condition
1	1	2899	ford	se	2011	clean vehicle	190552.0	silver	2fmdk3gc4bbb02217	166951262	tennessee	usa	6 days left
2	2	5350	dodge	mpv	2018	clean vehicle	39590.0	silver	3c4pdcgg5jt346413	167655728	georgia	usa	2 days left
3	3	25000	ford	door	2014	clean vehicle	64146.0	blue	1ftfw1et4efc23745	167753855	virginia	usa	22 hours left
4	4	27700	chevrolet	1500	2018	clean vehicle	6654.0	red	3gcpcrec2jg473991	167763266	florida	usa	22 hours left
5	5	5700	dodge	mpv	2018	clean vehicle	45561.0	white	2c4rdgeg9jr237989	167655771	texas	usa	2 days left
...	...	...	...	...	...	...	...	...	...	...	...	...	...
102	102	10780	ford	mpv	2017	clean vehicle	40455.0	white	1fm5k8ht0hga07252	167656360	texas	usa	2 days left
103	103	13800	ford	focus	2018	clean vehicle	23164.0	white	1fadp3j2xjl279400	167755491	south carolina	usa	21 hours left
104	104	25201	cadillac	door	2017	clean vehicle	19011.0	no_color	1gyknbrs8hz257399	167765111	michigan	usa	2 days left
105	105	7070	ford	mpv	2017	clean vehicle	45191.0	white	1fm5k7d82hgb39148	167656361	texas	usa	2 days left
106	106	8700	ford	focus	2018	clean vehicle	21405.0	white	1fadp3k23jl219764	167755494	south carolina	usa	21 hours left

100 rows × 13 columns

2. On s’intéresse uniquement à la variable price des voitures. Résumez dans un tableau deux indicateurs descriptifs de tendances centrales, deux de dispersions et deux de formes de cette variable.

Deux tendence central.

In [8]:
```
price = df2010['price']
price.head(5)
```

Out[8]:
```
1     2899
2     5350
3    25000
4    27700
5     5700
Name: price, dtype: int64
```

In [9]:
```
price.mode()
```

Out[9]:
```
0    16500
dtype: int64
```

In [10]:
```
price.mean()
```

Out[10]:
```
19189.131956155143
```

Deux dispersion

In [11]:
```
import statistics

statistics.stdev(price)
```

Out[11]:
```
11844.723105701421
```

In [12]:
```
statistics.variance(price)
```

Out[12]:
```
140297465.45073712
```

Deux de forme

In [13]:
```
price.skew()
```

Out[13]:
```
0.9656648298546364
```

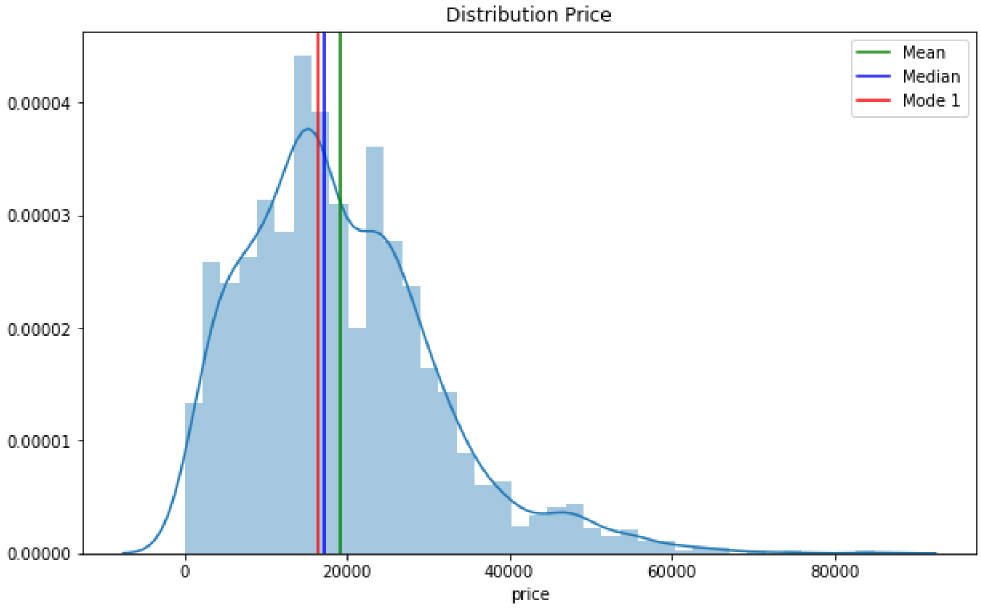
3. Visualisez graphiquement un indicateur de tendance centrale et un indicateur de forme.

In [14]:

```
import seaborn as sns

mean=price.mean();
median=price.median();
mode=price.mode();
fig, ax = plt.subplots(figsize=(10,6));

sns.distplot(price);
plt.title('Distribution Price');
plt.axvline(mean,color='green',label='Mean');
plt.axvline(median,color='blue',label='Median');
plt.axvline(mode[0],color='red',label='Mode 1')

plt.legend()
```

Out[14]:

```
<matplotlib.legend.Legend at 0x1ea6d804e88>
```



Faites le lien entre les représentations graphiques et les indicateurs et commentez les résultats.

Le skew positive 0,96 nous confirme une distribution non simetrique. mean = 19189 Mode es 16500, ca veut dire que bcp de voitures sont vendus a cette price.

Étape 3 : Analyse des ventes de voiture

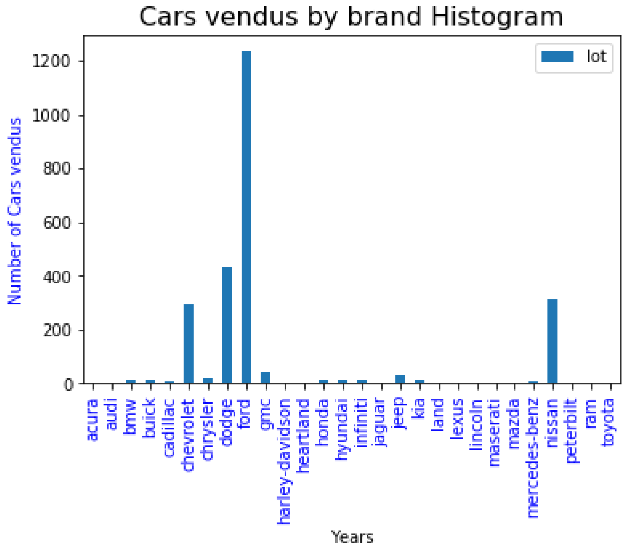
In [15]:

```
dfventes = df2010[['lot', 'brand']].groupby('brand').count()
dfventes

dfventes.sort_values(by=['lot'], inplace=True, ascending=False)
dfventes2 = dfventes.head(6) #First 5 after order
dfventes2
```

Out[15]:

	lot
brand	
ford	1189
dodge	424
nissan	298
chevrolet	262
gmc	39
jeep	28

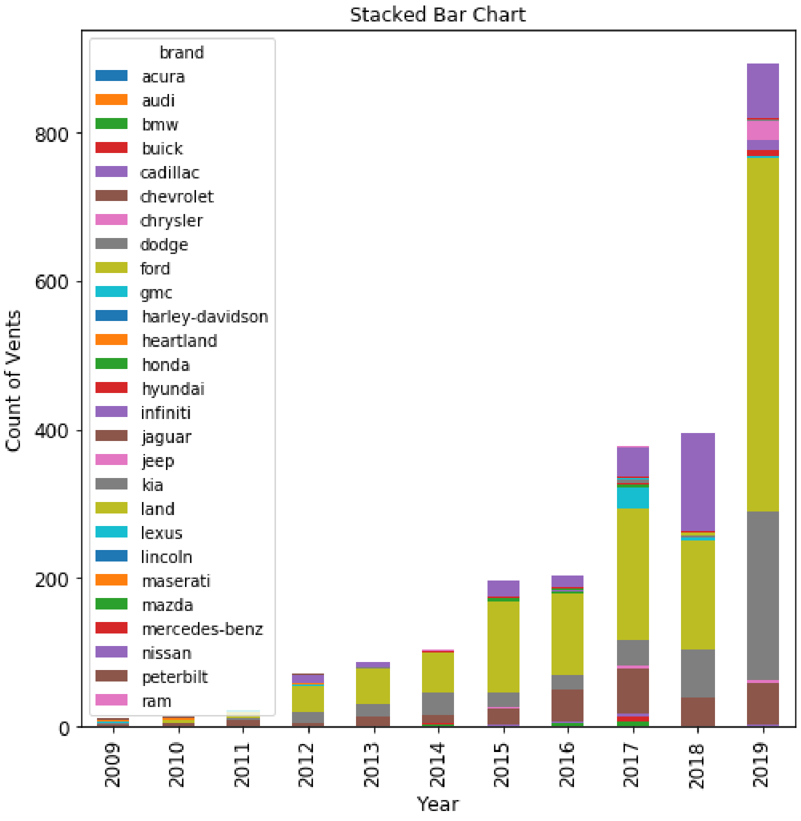
```
In [16]: df[['lot', 'brand']].groupby('brand').count().plot(kind='bar')
         plt.title('Cars vendus by brand Histogram', color = 'black', fontsize = 16)
         plt.xlabel('Years',color = 'black')
         plt.ylabel('Number of Cars vendus', color = 'blue')
         plt.xticks(color = 'blue')
         plt.yticks(color = 'black')
         plt.show()
```



Les cars le plus vendus sont ford=1189, dodge=424 et chevrolet=262

2. Visualisez la variation de vente annuelle de chacune des 6 marques (sur un même graphique)

```
In [17]: FreqData2_table = pd.crosstab(columns=df2010['brand'], index=df2010['year'])
         FreqData2_table.reset_index()

         Stacked = FreqData2_table.plot(kind="bar",figsize=(8,8),stacked=True,title='Stacked Bar Chart',fontsize=12)
         Stacked.set_ylabel("Count of Vents",fontsize=12)
         Stacked.set_xlabel("Year",fontsize=12)

Out[17]: Text(0.5, 0, 'Year')
```



Étape 4 : Analyse des relations

Pensez-vous qu'il y a une relation entre le prix price et la distance parcourue mileage ? Expliquez la démarche et

les résultats obtenus.

In [18]:

```
df2010.head(5)
```

Out[18]:

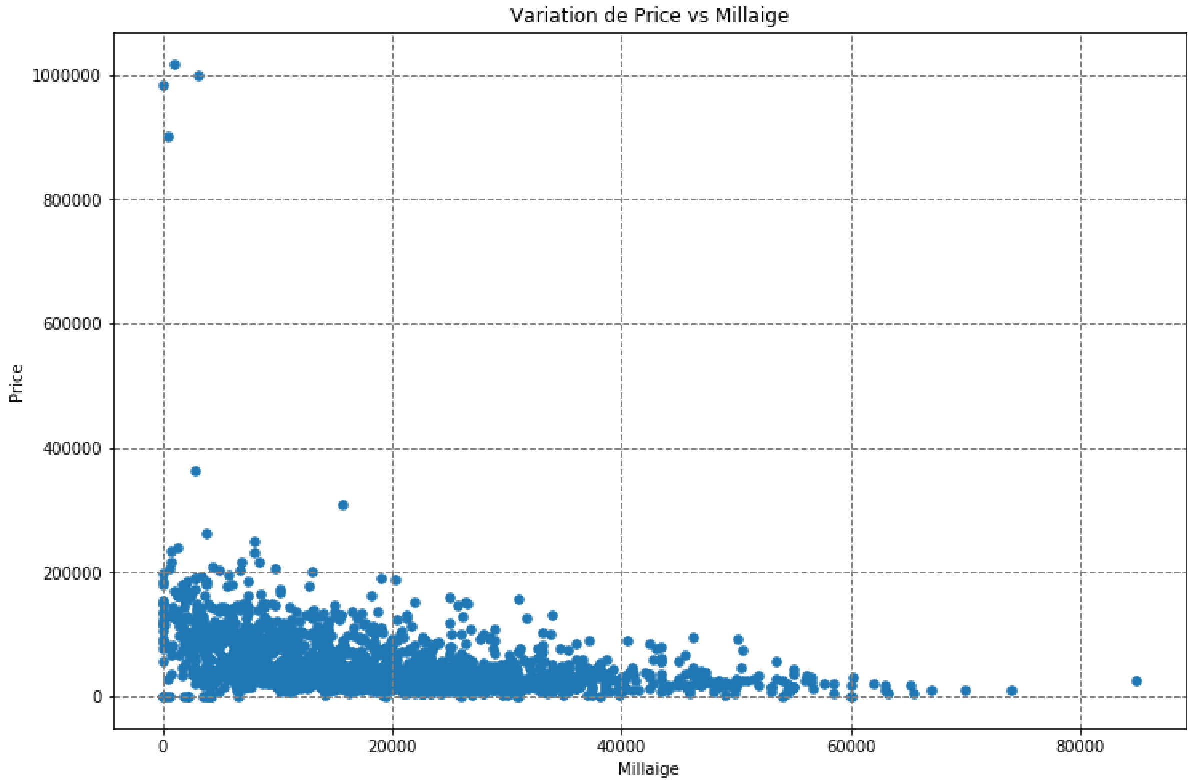
	Unnamed: 0	price	brand	model	year	title_status	mileage	color	vin	lot	state	country	condition
1	1	2899	ford	se	2011	clean vehicle	190552.0	silver	2fmdk3gc4bbb02217	166951262	tennessee	usa	6 days left
2	2	5350	dodge	mpv	2018	clean vehicle	39590.0	silver	3c4pdcgg5jt346413	167655728	georgia	usa	2 days left
3	3	25000	ford	door	2014	clean vehicle	64146.0	blue	1ftfw1et4efc23745	167753855	virginia	usa	22 hours left
4	4	27700	chevrolet	1500	2018	clean vehicle	6654.0	red	3gcpcrec2jg473991	167763266	florida	usa	22 hours left
5	5	5700	dodge	mpv	2018	clean vehicle	45561.0	white	2c4rdgeg9jr237989	167655771	texas	usa	2 days left

In [19]:

```
fig = plt.figure(figsize=(12,8))
plt.scatter( df2010['price'], df2010["mileage"], s=30)
plt.grid(color='gray', linestyle='--', linewidth=1)
plt.ylabel("Price")
plt.xlabel("Millaige")
plt.title('Variation de Price vs Millaige')
```

Out[19]:

```
Text(0.5, 1.0, 'Variation de Price vs Millaige')
```



Graphiquement on a pas une relation lineaire entre les deux variables.

In [20]:

```
# Regression Lineaire

from sklearn import linear_model
from sklearn.linear_model import LinearRegression

x1 = np.array(df2010['price']).reshape((-1,1))
y1 = df2010['mileage']

mymodel = LinearRegression()
results = mymodel.fit(x1,y1)

print("Coeficient determination: \n", results.score(x1, y1))
print("Intercept: \n", results.intercept_)
print("Slope: \n", results.coef_)
```

```
Coeficient determination:
 0.12446583982016178
Intercept:
 80379.37301767626
Slope:
 [-1.62942198]
```

Le coeficient de determination cest tres bas, donc pas de relation lineaire entre variables.

In [21]:
```python
from statsmodels.formula.api import ols

model2 = ols('price ~ mileage', data=df2010).fit() #x vs y
print(model2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.124
Model:                            OLS   Adj. R-squared:                  0.124
Method:                 Least Squares   F-statistic:                     336.9
Date:                Tue, 02 Mar 2021   Prob (F-statistic):           1.82e-70
Time:                        21:36:42   Log-Likelihood:                -25456.
No. Observations:                2372   AIC:                         5.092e+04
Df Residuals:                    2370   BIC:                         5.093e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   2.294e+04    305.908     74.992      0.000    2.23e+04    2.35e+04
mileage       -0.0764      0.004    -18.355      0.000      -0.085      -0.068
==============================================================================
Omnibus:                      520.702   Durbin-Watson:                   1.689
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1181.266
Skew:                           1.228   Prob(JB):                    3.10e-257
Kurtosis:                       5.434   Cond. No.                     9.88e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.88e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

R-squared confirm que on a pas relation linear entre variables price and mileage

3. Les variables prix price et title_status contiennent plusieurs valeurs aberrantes. En fixant un seuil à 2 (threshold = 2), supprimer

les valeurs aberrantes en utilisant la distance interquartile.

In [22]:

```python
def is_outlier(value, p25, p75):
    # Check if value is an outlier
    lower = p25 - 1.5 * (p75 - p25)
    upper = p75 + 1.5 * (p75 - p25)
    return value <= lower or value >= upper


def get_indices_of_outliers(values):
    #Get outlier indices (if any)
    p25 = np.percentile(values, 25)
    p75 = np.percentile(values, 75)

    indices_of_outliers = []
    for ind, value in enumerate(values):
        if is_outlier(value, p25, p75):
            indices_of_outliers.append(ind)
    return indices_of_outliers


indices_of_outliers = get_indices_of_outliers(df2010['price'])
df2010['price'][indices_of_outliers] = mean; #np.percentile(df2010['price'], 75, interpolation = 'midpoint') - np.
percentile(df2010['price'], 25, interpolation = 'midpoint')
```

```
C:\Users\valm044\Anaconda3\lib\site-packages\ipykernel_launcher.py:21: SettingWithCopyWarning:
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/user_guide/indexing.html#return
ing-a-view-versus-a-copy

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\series.py in __setitem__(self, key, value)
   1013         try:
-> 1014             self._set_with_engine(key, value)
   1015         except com.SettingWithCopyError:

~\Anaconda3\lib\site-packages\pandas\core\series.py in _set_with_engine(self, key, value)
   1053         try:
-> 1054             self.index._engine.set_value(values, key, value)
   1055             return

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.set_value()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.set_value()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

TypeError: '[38, 43, 88, 120, 252, 313, 326, 335, 341, 347, 422, 555, 576, 771, 775, 1016, 1119, 1161, 1196, 1199,
1228, 1231, 1232, 1235, 1237, 1238, 1240, 1241, 1242, 1243, 1244, 1269, 1270, 1286, 1296, 1315, 1324, 1347, 1351,
 1354, 1359, 1373, 1461, 1536, 1538, 1597, 1600, 1670, 1749, 1752, 1786, 1789, 1791, 1867, 1935, 1941, 1945, 1947,
2083, 2085, 2087]' is an invalid key

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-22-fa86f6ac5400> in <module>
     19
     20 indices_of_outliers = get_indices_of_outliers(df2010['price'])
---> 21 df2010['price'][indices_of_outliers] = mean; #np.percentile(df2010['price'], 75, interpolation = 'midpoin
t') - np.percentile(df2010['price'], 25, interpolation = 'midpoint')

~\Anaconda3\lib\site-packages\pandas\core\series.py in __setitem__(self, key, value)
   1040                     pass
   1041
-> 1042             self._set_with(key, value)
   1043
   1044         if cacher_needs_updating:

~\Anaconda3\lib\site-packages\pandas\core\series.py in _set_with(self, key, value)
   1090             if key_type == "integer":
   1091                 if self.index.inferred_type == "integer":
-> 1092                     self._set_labels(key, value)
   1093                 else:
   1094                     return self._set_values(key, value)

~\Anaconda3\lib\site-packages\pandas\core\series.py in _set_labels(self, key, value)
   1103         mask = indexer == -1
   1104         if mask.any():
-> 1105             raise ValueError(f"{key[mask]} not contained in the index")
   1106         self._set_values(indexer, value)
   1107

ValueError: [313 347] not contained in the index
```

Excercise 4

In [25]:

```
from scipy import stats

results = stats.ttest_1samp(df2010['price'], 20000, 0)
print('stattiscit: ', results[0])
print('p_value: ', results[1])
```

```
stattiscit:  -3.3341306580162486
p_value:  0.0008688293624366382
```

In [26]:

```
# interpret p-value
alpha = 0.05
if results[1] <= alpha:
    print('(reject H0)')
else:
    print('(H0 holds true)')
```

```
(reject H0)
```

In [ ]:

```
# Nos rejectons que la moyenne cest 20000
```