Unsupervised Machine Learning
Final project

1. Objective :

Considering the financial and economic difficulties that have arisen since "the revolution," the Ministry of Transport and the Ministry of Commerce have started looking into potential new import routes. As many overseas markets have indicated interest in supporting with this endeavour, the ideal profile must conform to logistical and technological limits and closely mirror European models. In order to address this, a preliminary research has produced a dataset that lists the specifications of the many car models that are offered, together with information on the vehicles' respective origins—European, Japanese, or American. The goal of this study is to use clustering analysis to identify the origin that would be most appropriate for the Tunisian market and might replace the European profile.

a. Data: We have a text file with a variety of automobiles listed along with their distinctive attributes and places of origin..

```
#importation des données /chargement du fichier
import pandas
Voiture = pandas.read_table("Voitures_Origine.txt",sep="\t",header=0,index_col=0)
```

```
#dimensions : nombre de lignes, nombre de colonnes
print(Voiture.shape)
```

```
(392, 6)
```

```python
#Les noms des colonnes
print(Voiture.columns)
```

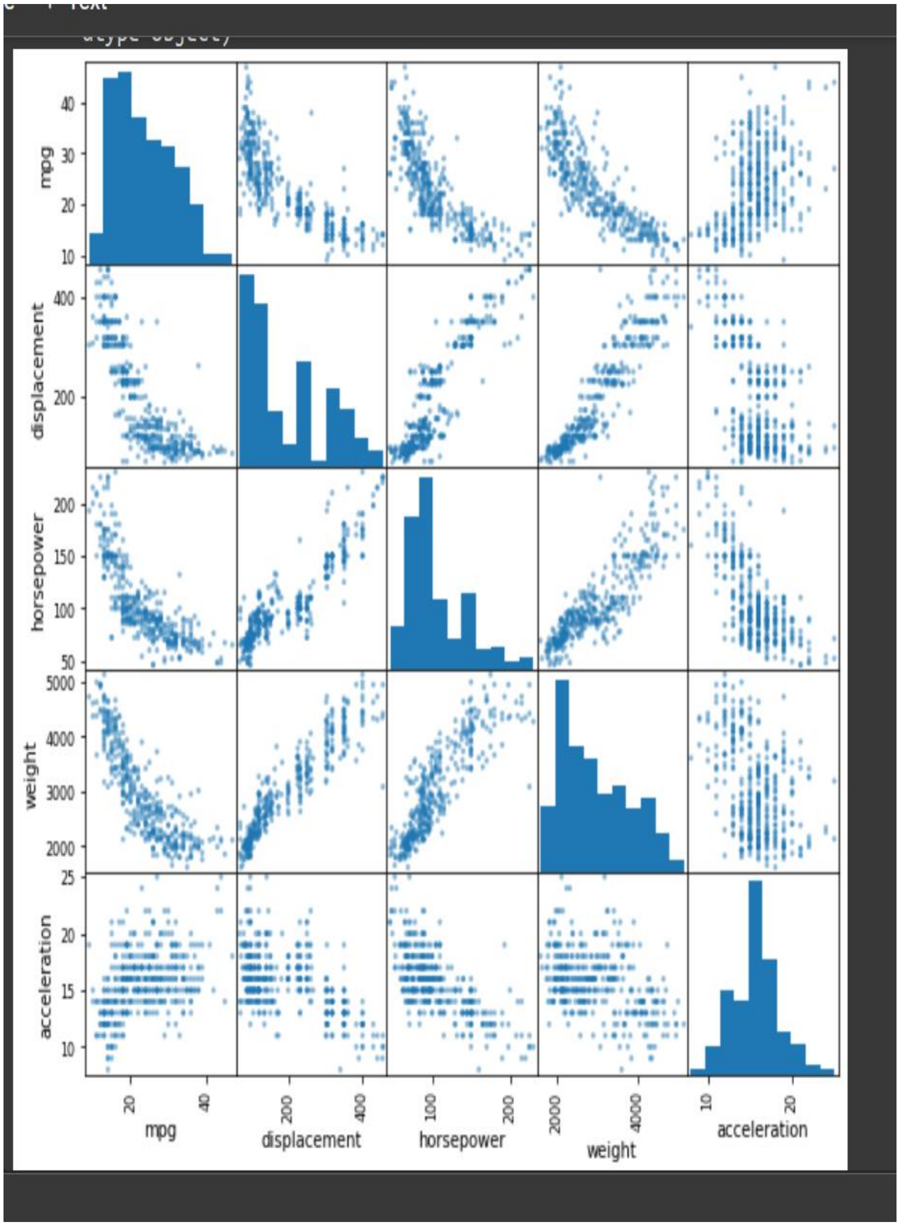
```
Index(['mpg', 'displacement', 'horsepower', 'weight', 'acceleration',
       'origin'],
      dtype='object')
```

```python
#Type de chaque colonne
print(Voiture.dtypes)
```

```
mpg              int64
displacement     int64
horsepower       int64
weight           int64
acceleration     int64
origin          object
dtype: object
```

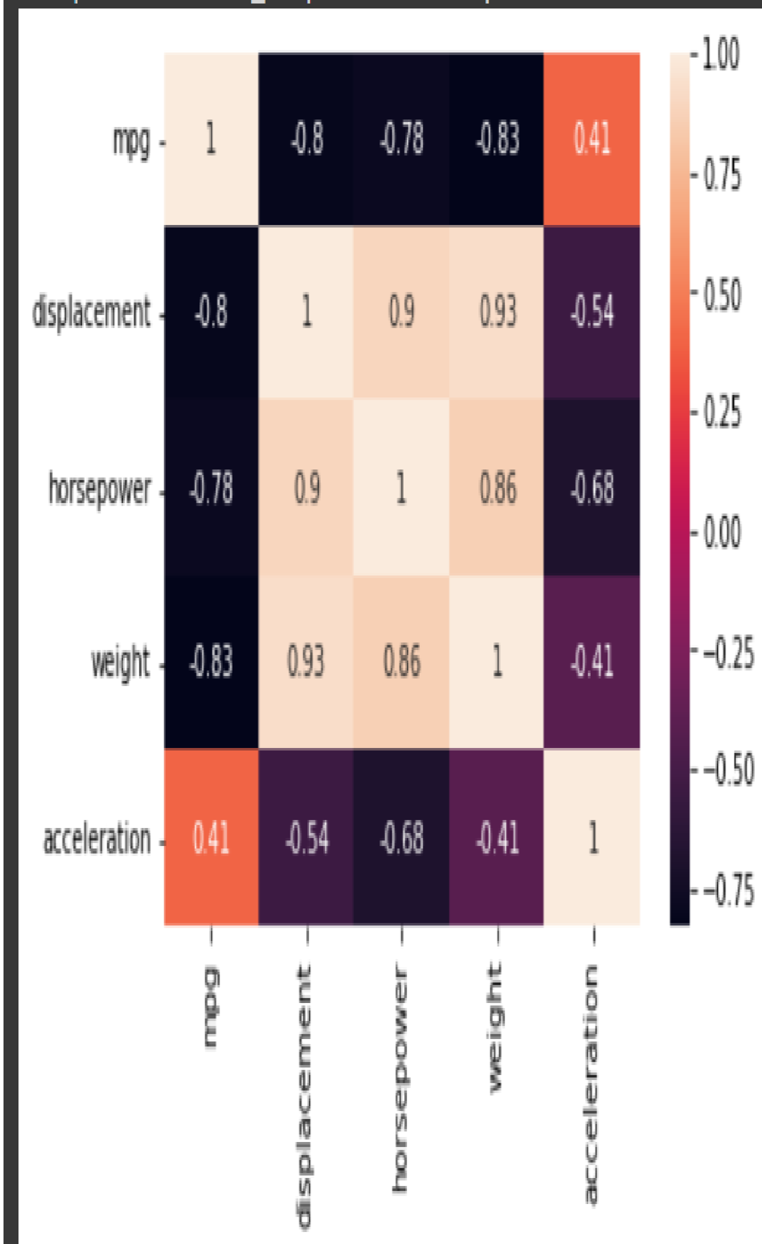
```python
#Description des données
print(Voiture.describe())
```

```
              mpg  displacement  horsepower       weight  acceleration
count  392.000000    392.000000  392.000000   392.000000    392.000000
mean    23.492347    194.410714  104.469388  2977.584184     15.681122
std      7.799924    104.645191   38.491160   849.402560      2.761232
min      9.000000     68.000000   46.000000  1613.000000      8.000000
25%     17.000000    105.000000   75.000000  2225.250000     14.000000
50%     23.000000    151.000000   93.500000  2803.500000     16.000000
75%     29.000000    275.750000  126.000000  3614.750000     17.000000
max     47.000000    455.000000  230.000000  5140.000000     25.000000
```


mpg
displacement
horsepower
weight
acceleration
mpg
displacement
horsepower
weight
acceleration

```python
# Obtain the correlation matrix
sns.heatmap(Voiture.corr(), annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f55f583ded0>
```



1. Models : clustering

i. K-means

```python
# valeurs de toutes les colonnes
#colonnes => 0:5 (0 à 5 [non inclus])
#lignes = : (toutes les colonnes)
V_SansLabels=Voiture.iloc[:,0:5];
#Labels
V_Labels=Voiture.iloc[:,5];
```

```python
#k-means
import numpy as np
from sklearn import cluster
kmeans = cluster.KMeans(n_clusters=2);
kmeans.fit(V_SansLabels);
#index triés des groupes
idk = np.argsort(kmeans.labels_);
#affichage des observations et leurs groupes
print(pandas.DataFrame(V_SansLabels.index[idk],kmeans.labels_[idk]));
kmeans.labels_
#distances aux centres de classes des observations
print(kmeans.transform(V_SansLabels));
#correspondance avec les groupes réels
pandas.crosstab(V_Labels,kmeans.labels_)
```

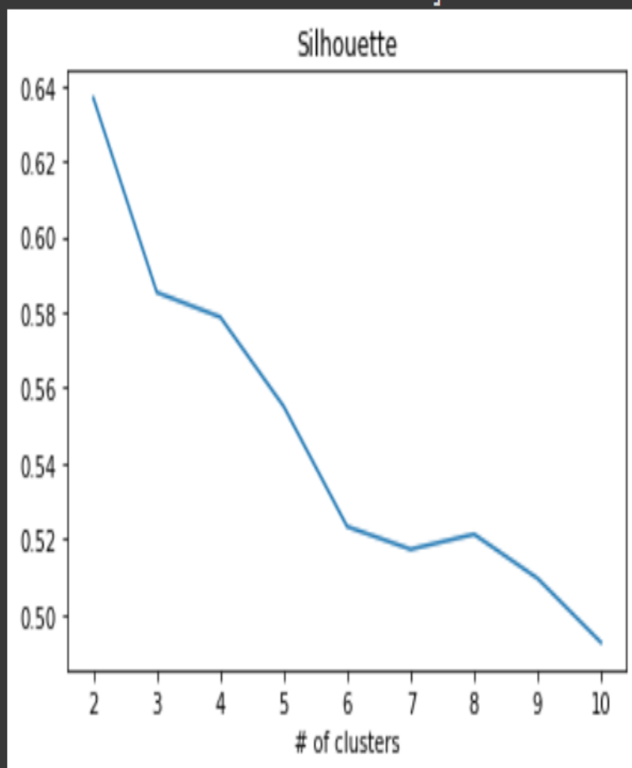
col_0	0	1
origin		
american	99	146
european	58	10
japanese	79	0

K-Means does not come with built-in methods for determining the ideal number of clusters, in contrast to Agglomerative Hierarchical Clustering (CAH). To solve this, we must either use specialised packages that provide pertinent methods or write our own algorithms in Python. The overall procedure continues to be the same: we iteratively change the number of clusters and track the development of a solution quality indicator. This indicator measures how much more similar individuals are to those in their own cluster than they are to those in other clusters.

In the study that follows, we calculate the "silhouette" measure for a variety of cluster sizes created by the K-Means approach.

In this case, the Elbow approach was used to determine the ideal number of clusters.

```
[0.63670095 0.58526049 0.57873747 0.55506504 0.52325224 0.51725055
 0.52122797 0.50956765 0.49260061]
```



```python
#librairie pour évaluation des partitions
from sklearn import metrics

#utilisation de la métrique "silhouette"
#faire varier le nombre de clusters de 2 à 10
res = np.arange(9,dtype="double")
for k in np.arange(9):
  km = cluster.KMeans(n_clusters=k+2)
  km.fit(V_SansLabels)
  res[k] = metrics.silhouette_score(V_SansLabels,km.labels_)
print(res)
#graphique
import matplotlib.pyplot as plt
plt.title("Silhouette")
plt.xlabel("# of clusters")
plt.plot(np.arange(2,11,1),res)
plt.show()
```

a. Data balancing: retraining data using a K-means model and oversampling BSMOTE for three clusters

```python
from collections import Counter
from imblearn.over_sampling import BorderlineSMOTE
```

```python
def BSMOTE(X,y):
# summarize class distribution
    counter = Counter(y)
    print(counter)
# transform the dataset
    X, y = BorderlineSMOTE().fit_resample(X, y)
# summarize the new class distribution
    counter = Counter(y)
    print(counter)
    return X,y
```

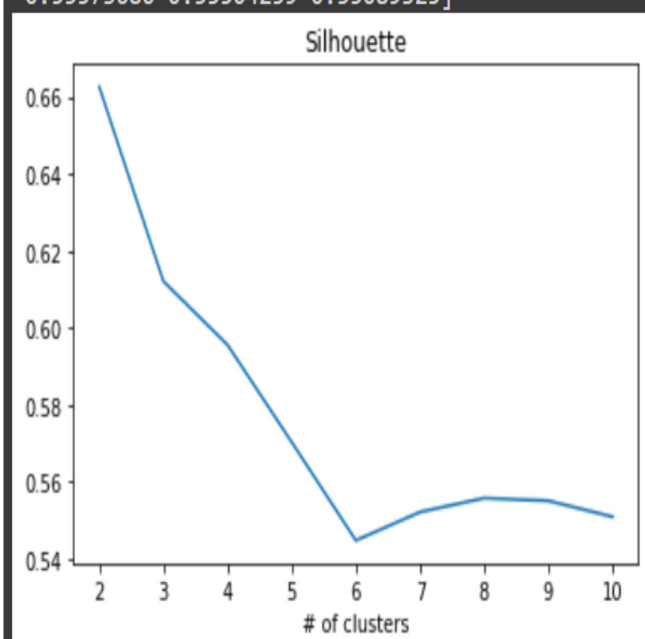
```python
V_SansLabels,V_Labels=  BSMOTE(V_SansLabels,V_Labels)
```

```
Counter({'american': 245, 'asian': 79, 'european': 68})
Counter({'asian': 245, 'american': 245, 'european': 245})
```

```
#librairie pour évaluation des partitions
from sklearn import metrics

#utilisation de la métrique "silhouette"
#faire varier le nombre de clusters de 2 à 10
res = np.arange(9,dtype="double")
for k in np.arange(9):
  km = cluster.KMeans(n_clusters=k+2)
  km.fit(V_SansLabels)
  res[k] = metrics.silhouette_score(V_SansLabels,km.labels_)
print(res)
#graphique
import matplotlib.pyplot as plt
plt.title("Silhouette")
plt.xlabel("# of clusters")
plt.plot(np.arange(2,11,1),res)
plt.show()
```

```
[0.66259946 0.61208252 0.59554235 0.57027846 0.54473164 0.55210926
 0.55573686 0.55504259 0.55089325]
```



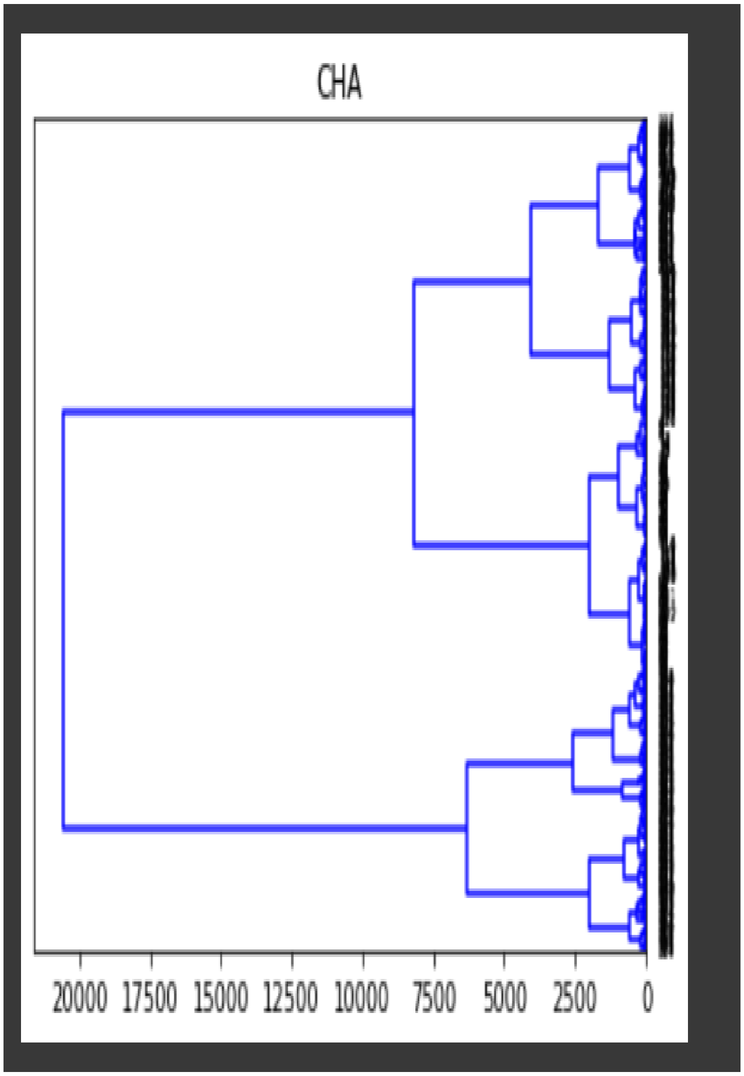
b.

CAH : here we used the CAH method for 2 clusters

```python
#librairies pour la CAH
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage,fcluster
```

```python
#générer la matrice des distances
Z = linkage(V_SansLabels,method='ward',metric='euclidean')
```

```python
#affichage du dendrogramme
plt.title("CHA")
dendrogram(Z,labels=V_SansLabels.index,orientation='left',color_threshold=0)
plt.show()
```

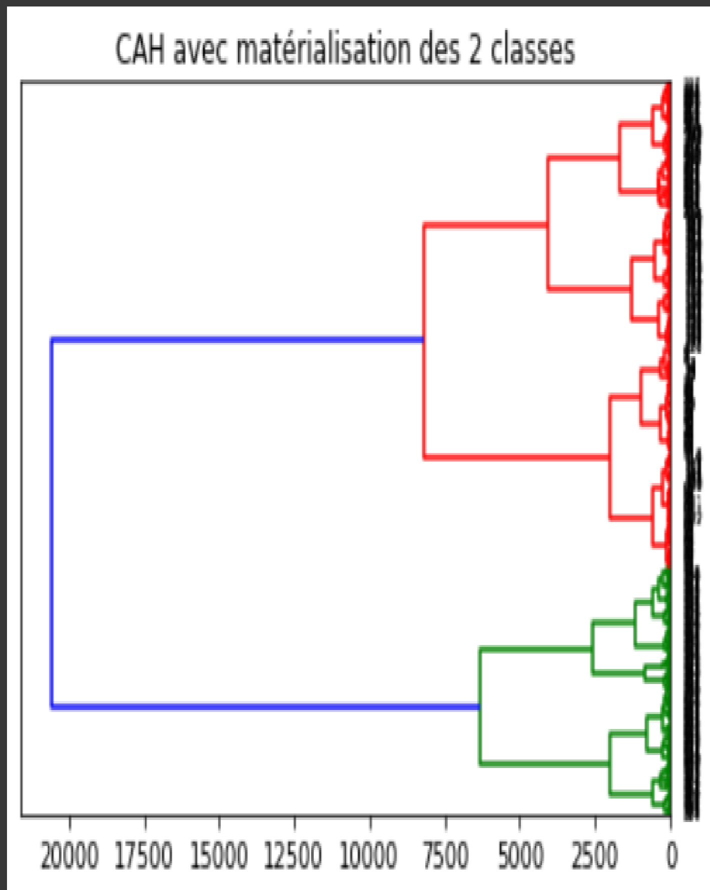

CHA
20000
17500
15000
12500
10000
7500
5000
2500
0

```python
# matérialisation des deux classes (hauteur t = 17500)
plt.title('CAH avec matérialisation des 2 classes')
dendrogram(Z,labels=V_SansLabels.index,orientation='left',color_threshold=17500)
plt.show()

#découpage à la hauteur t = 17500==> identifiants de 2 groupes obtenus
groupes_cah = fcluster(Z,t=17500,criterion='distance')
print(groupes_cah)

#index triés des groupes
import numpy as np
idg = np.argsort(groupes_cah)

#affichage des observations et leurs groupes
print(pandas.DataFrame(V_SansLabels.index[idg],groupes_cah[idg]))
```




```
#correspondance les vrais labels avec les groupes de la CAH
pandas.crosstab(V_Labels,groupes_cah)
```

col_0	1	2
origin		
american	129	116
european	3	65
japanese	0	79

2. Key Findings and Insights: It was shown that Japanese automobiles are the best suitable substitutes for European cars in the Tunisian market after applying clustering methods to the dataset. It's interesting to note that the CAH and K-means models assigned the same origin category to both European and Japanese automobiles.

3. Upcoming Steps: The dataset will be used to train further clustering models in order to increase accuracy even more. In addition, given the few characteristics in our data, we may investigate the possibility of dimensionality reduction as a preprocessing step. However, this choice is dependent on the unique features of our dataset.