Rapport du projet TPT : partie Machine Learning

1- Introduction :

Nous allons présenter un petit rapport sur la partie Machine Learning de notre projet intitulé : Analyse de la clientèle d'un concessionnaire automobile dans l'objectif de pouvoir recommander des modèles de véhicules.

2- Bibliothèques et Framework utilisées :

Pour les bibliothèques de Machine Learning et de python que nous avons utilisé sont :

- Scikitlearn : utilisé pour faire la partie pré-traitement des données et la partie modélisation.
- Pandas : utilisé pour l'importation des données et création des dataframes.
- Numpy : utilisé pour travailler avec des tableaux et des matrices.
- Seaborn : utilisé pour faire la visualisation des données.
- Matplotlib : utilisé pour faire la visualisation des données.

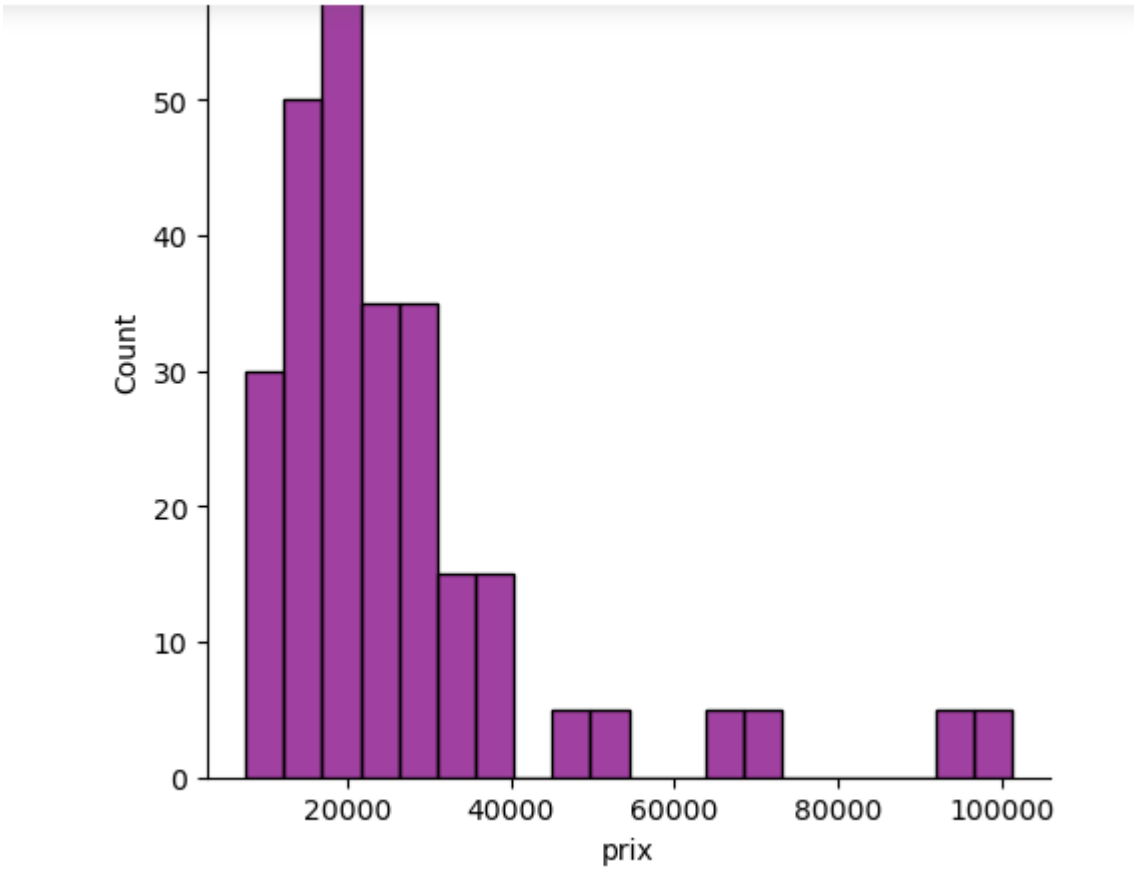
3- Analyse exploratoire et préparation des données catalogues :

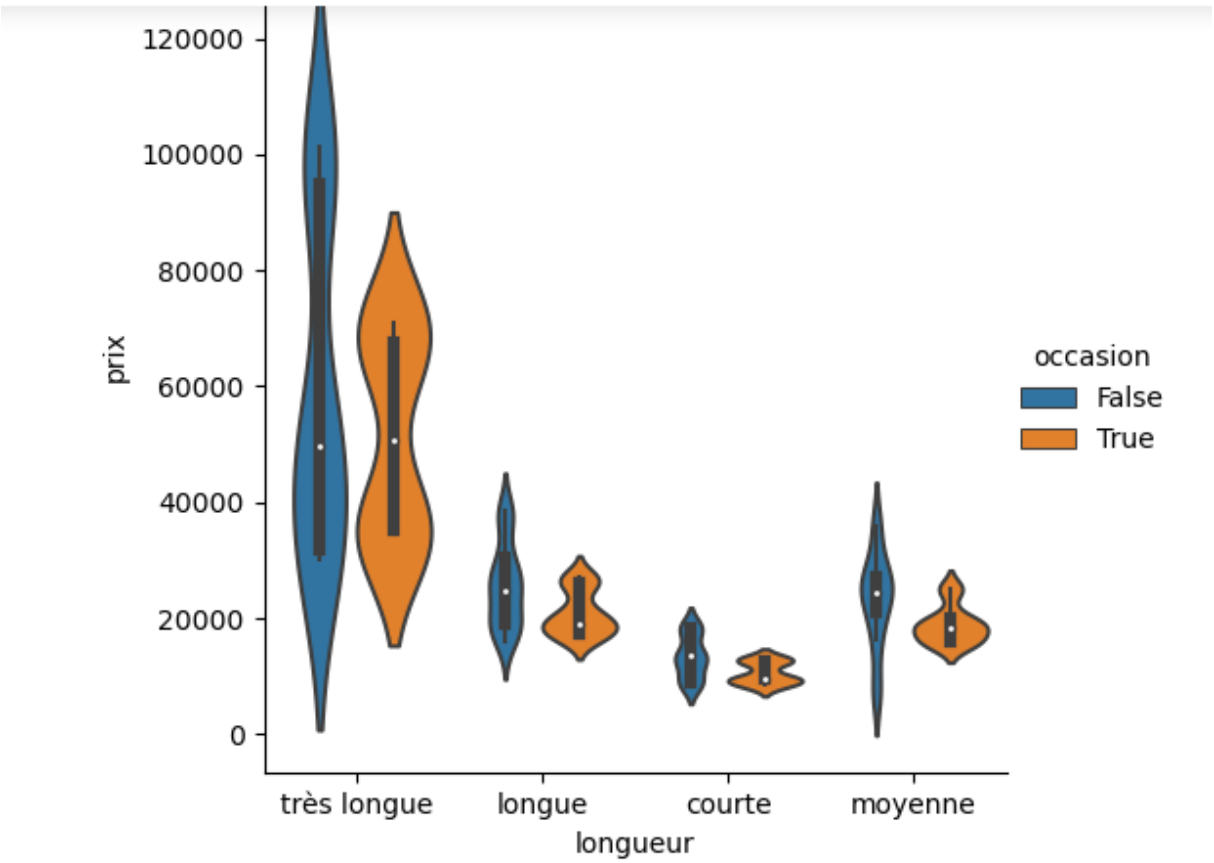
Dans un premier temps nous allons travailler avec le dataset Catalogue.csv pour faire l'analyse et l'exploration des données en commençant par faire une petite description statistiques de notre dataset en utilisant des fonctions comme info() et describe(), et le premier résultats de cette description est qu'on pourrait soupçonner que la moyenne des prix des véhicules est autour de 26668 pour des véhicules dont la puissance vaut en moyenne 157.59157.59 avec un nombre de porte et un nombre de places qui varie en moyennes autour de 55 et 44. Par ailleurs 3030 des observations de cet échantillons révèle que pour les véhicules ayant le même nombre de porte et de place, ont une puissance de 147.

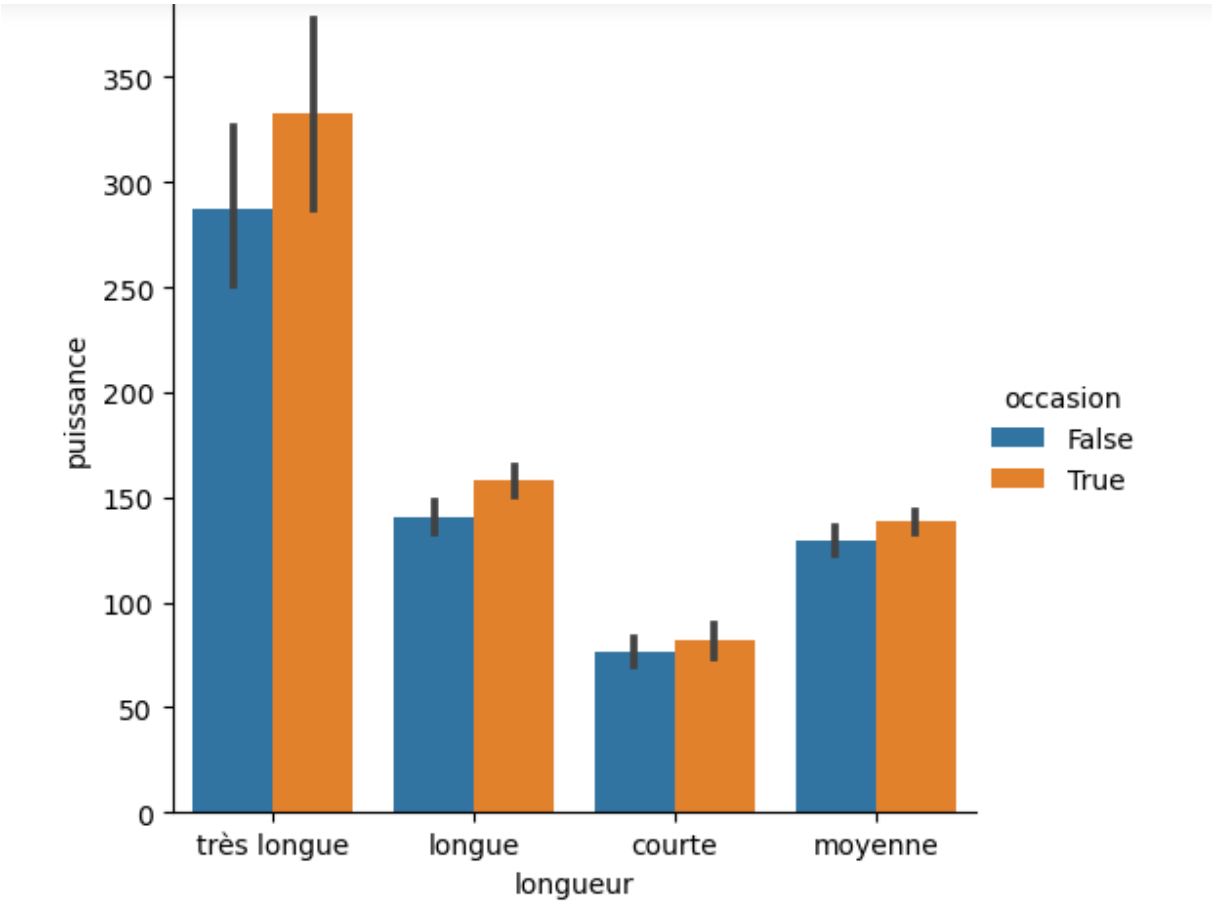
```
catalogue.describe()
```

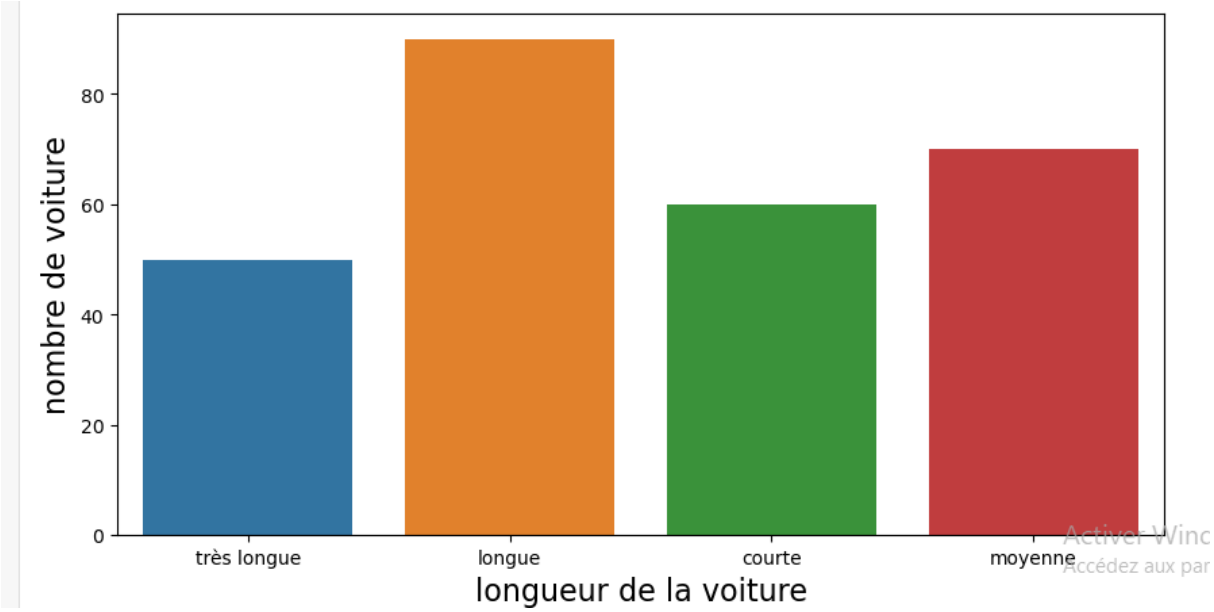
	puissance	nbPlaces	nbPortes	prix
count	270.000000	270.000000	270.000000	270.000000
mean	157.592593	5.222222	4.814815	26668.055556
std	90.551289	0.629707	0.580798	19050.121112
min	55.000000	5.000000	3.000000	7500.000000
25%	109.000000	5.000000	5.000000	16029.000000
50%	147.000000	5.000000	5.000000	20597.500000
75%	170.000000	5.000000	5.000000	30000.000000
max	507.000000	7.000000	5.000000	101300.000000

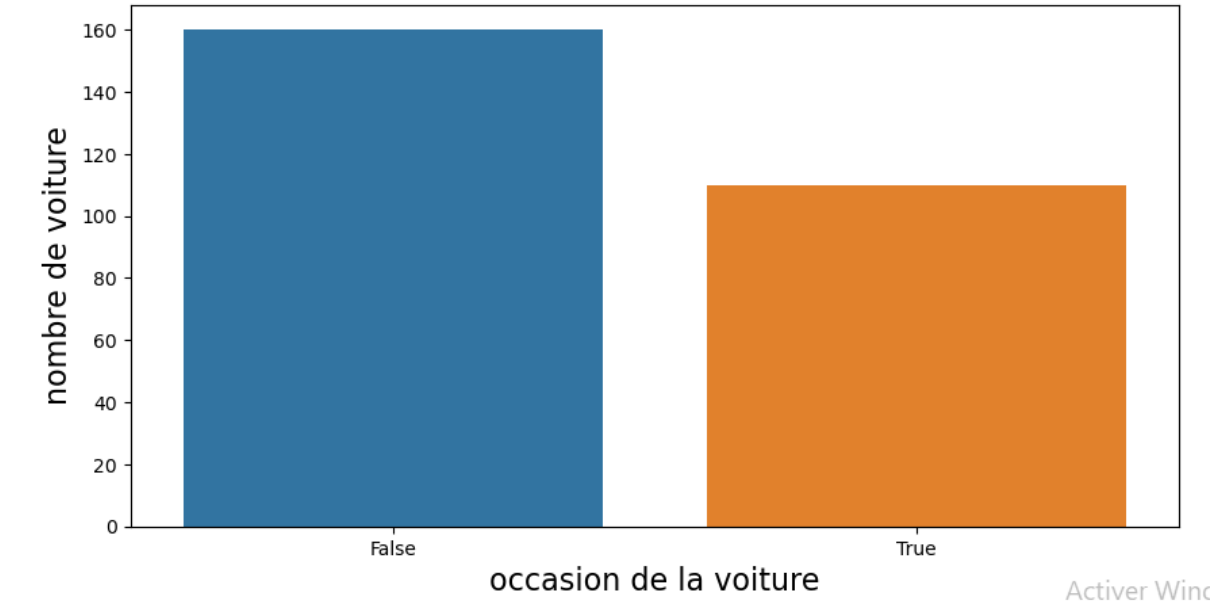
Et aussi nous allons voir si on a des valeurs manquantes, après cela nous avons commencé à faire des visualisations de nos différentes variables en utilisant des fonctions de Seaborn et aussi Matplotlib.




120000
100000
80000
60000
40000
20000
0
prix
très longue
longue
courte
moyenne
longueur
occasion
False
True


350
300
250
200
150
100
50
0
puissance
très longue
longue
courte
moyenne
longueur
occasion
False
True





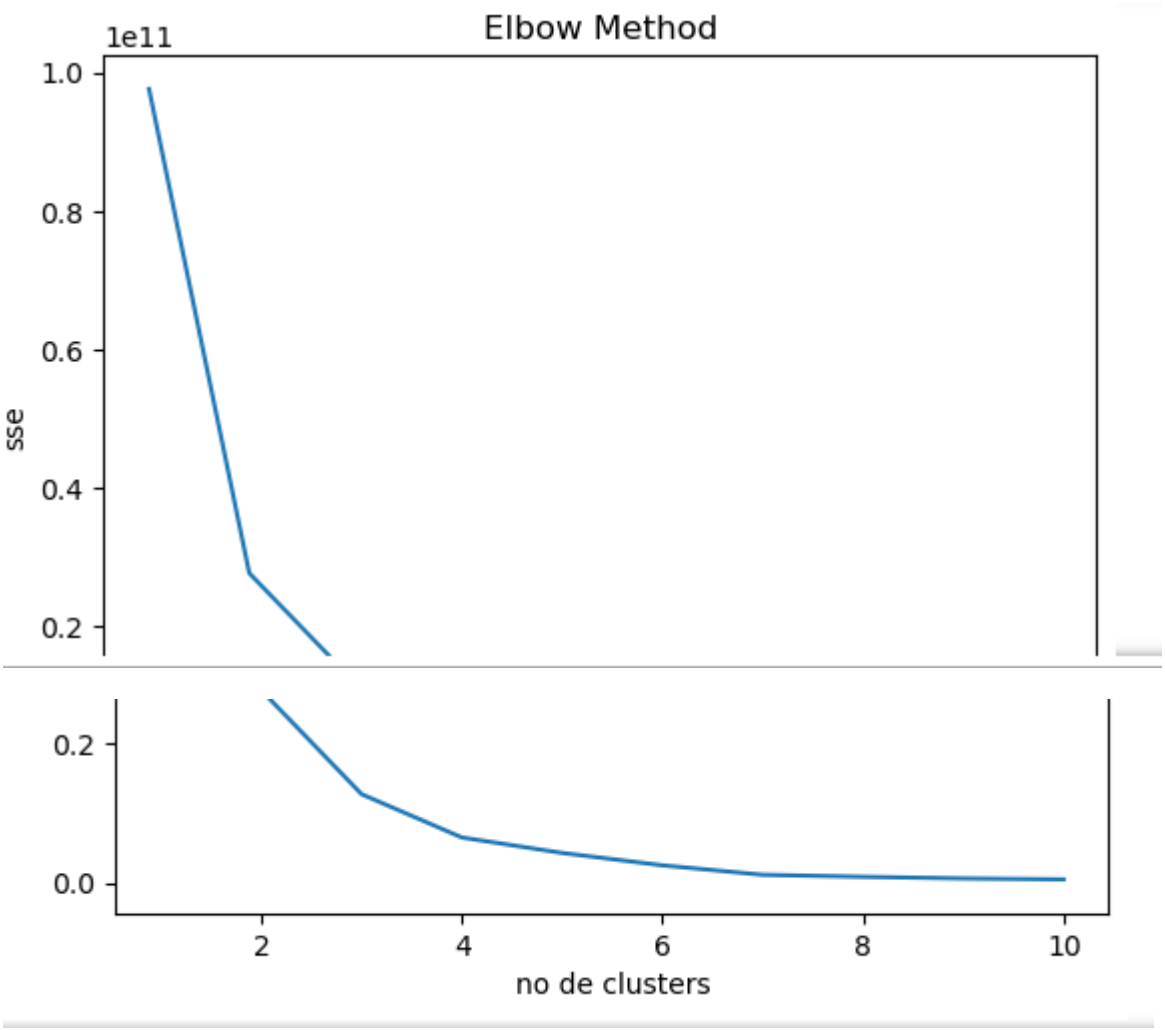
Puis nous allons passer à la partie pré-traitement des données en faisant des opérations d'encodage pour transformer nos variables catégorielles en variables numériques et aussi on peut faire la normalisation de nos différentes variables.

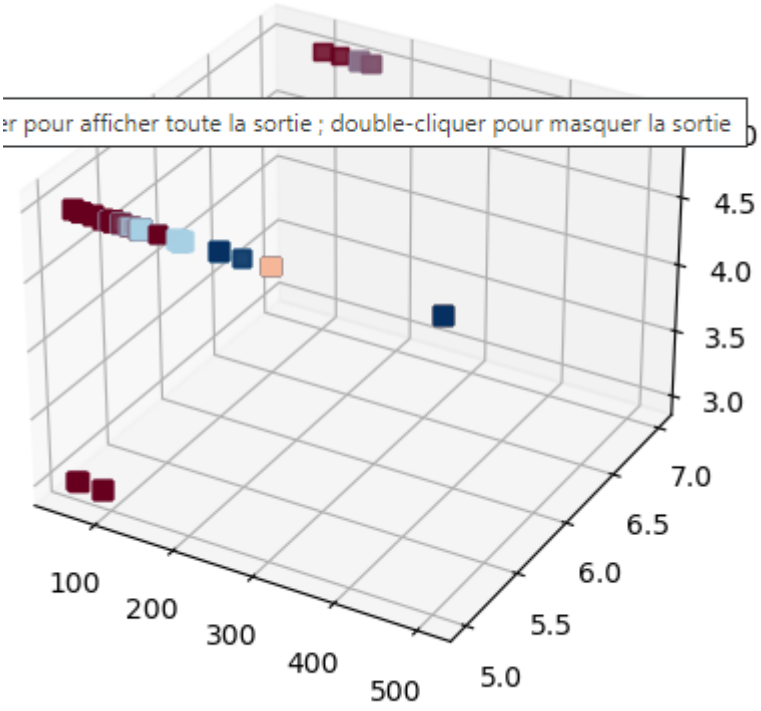
4- Tests des différentes approches de clustering :

Nous allons nous intéresser à des approches de clustering comme Kmeans.

Nous allons créer un modèle Algorithme KMeans pour décider du nombre de cluster optimal, KMeans++ en utilisant Elbow méthod pour comprendre K pour KMeans et pour le calcul KMEANS++.

Pour l'apprentissage non supervisé, nous utilisons « fit_predict() » dans lequel pour l'apprentissage supervisé, nous utilisons « fit_tranform() » y_kmeans est le modèle final. Maintenant, nous allons visualiser et tester une autre approche de clusterisation.

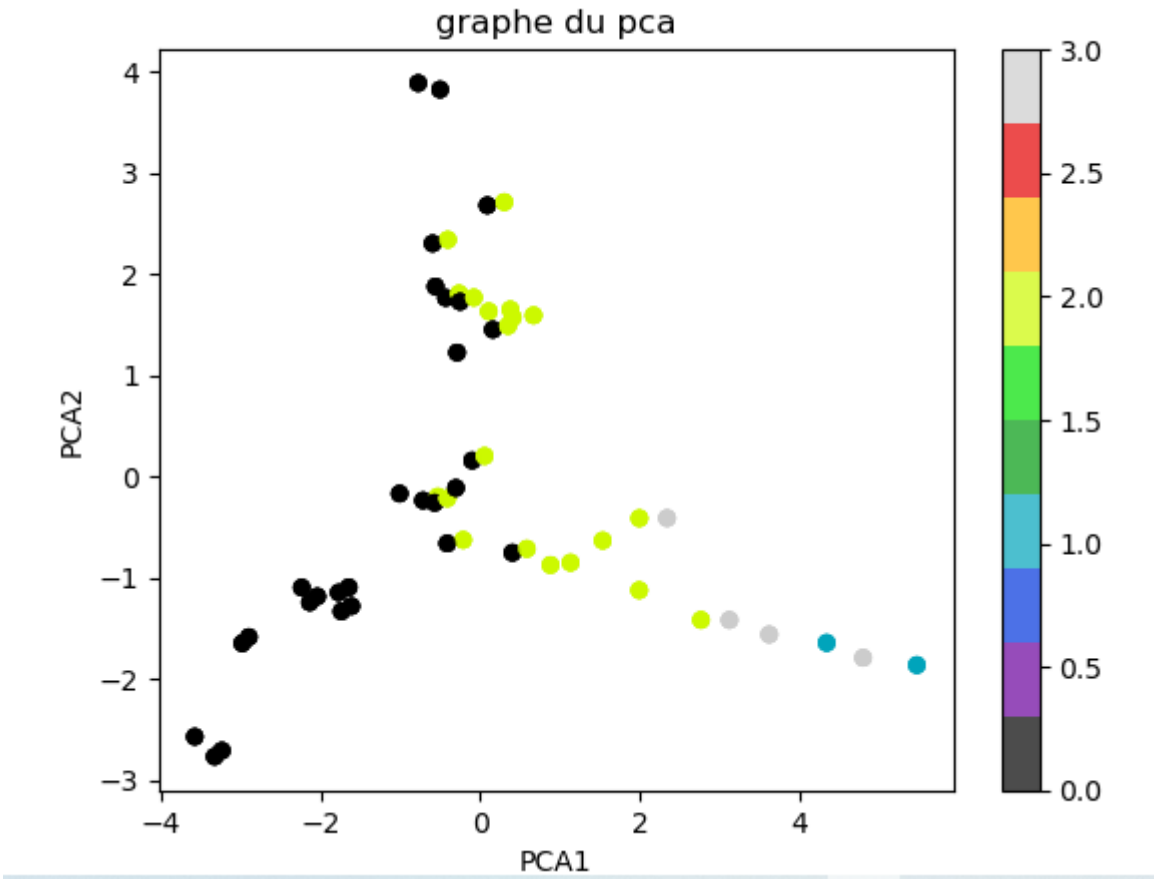




5-Analyse des principal composantes PCA :

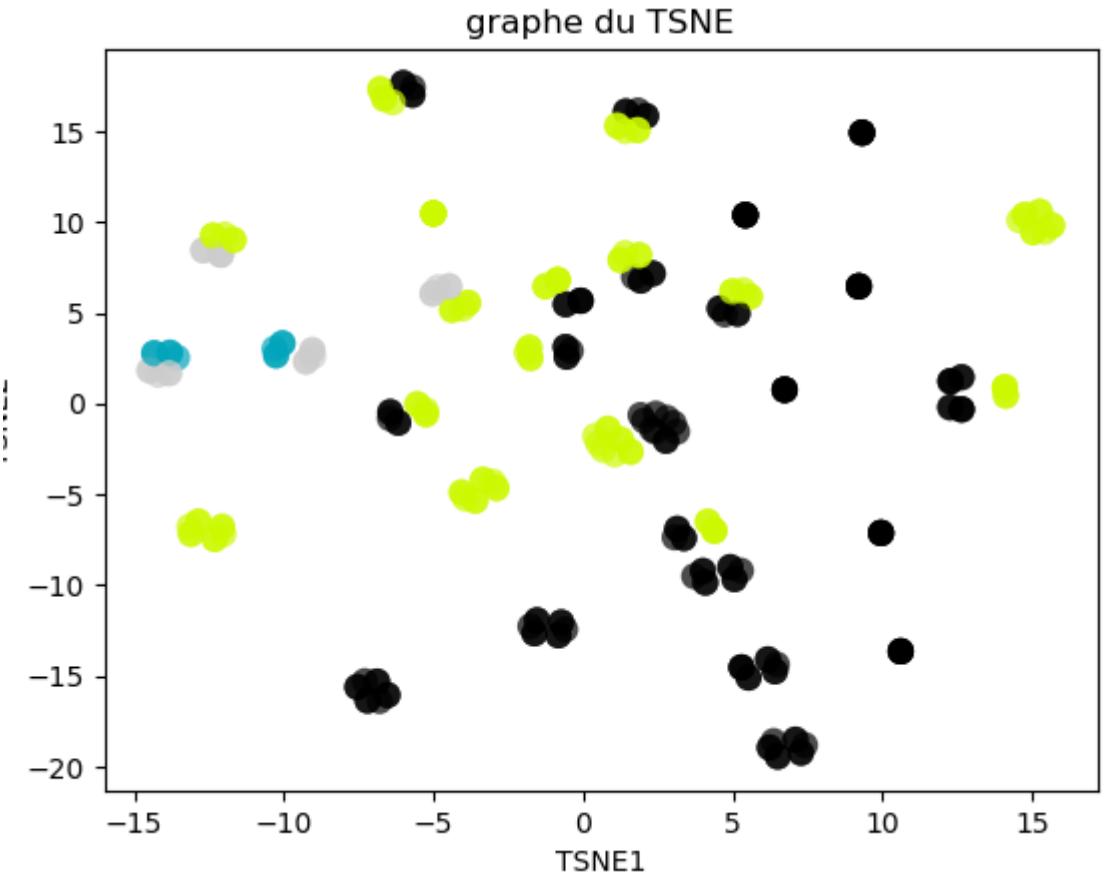
Puisqu'il est difficile d'évaluer la qualité de cet algorithme non supervisé en raison de l'absence d'une métrique de qualité explicite telle qu'utilisée dans l'apprentissage supervisé, nous allons effectuer une réduction de dimension enfin de mieux visualiser nos données et de nous aider à gérer plus facilement la multi-colinéarité qui pourras nous aider lors de l'apprentissage superviser.

L'analyse en composantes principales est l'une des méthodes les plus simples, les plus intuitives et les plus fréquemment utilisées pour la réduction de la dimensionnalité, projetant des données sur son sous-espace orthogonal d'entités.



6- T-distributed Stochastic Neighbor Embedding.TSNE:

Après la visualisation la méthone des TSNE nous montre que nous pouvons travailler avec plus de %4% clusters, et de plus elle a un temps d'exécution très supérieure au PCA.



7-Conclusion du partie 1:

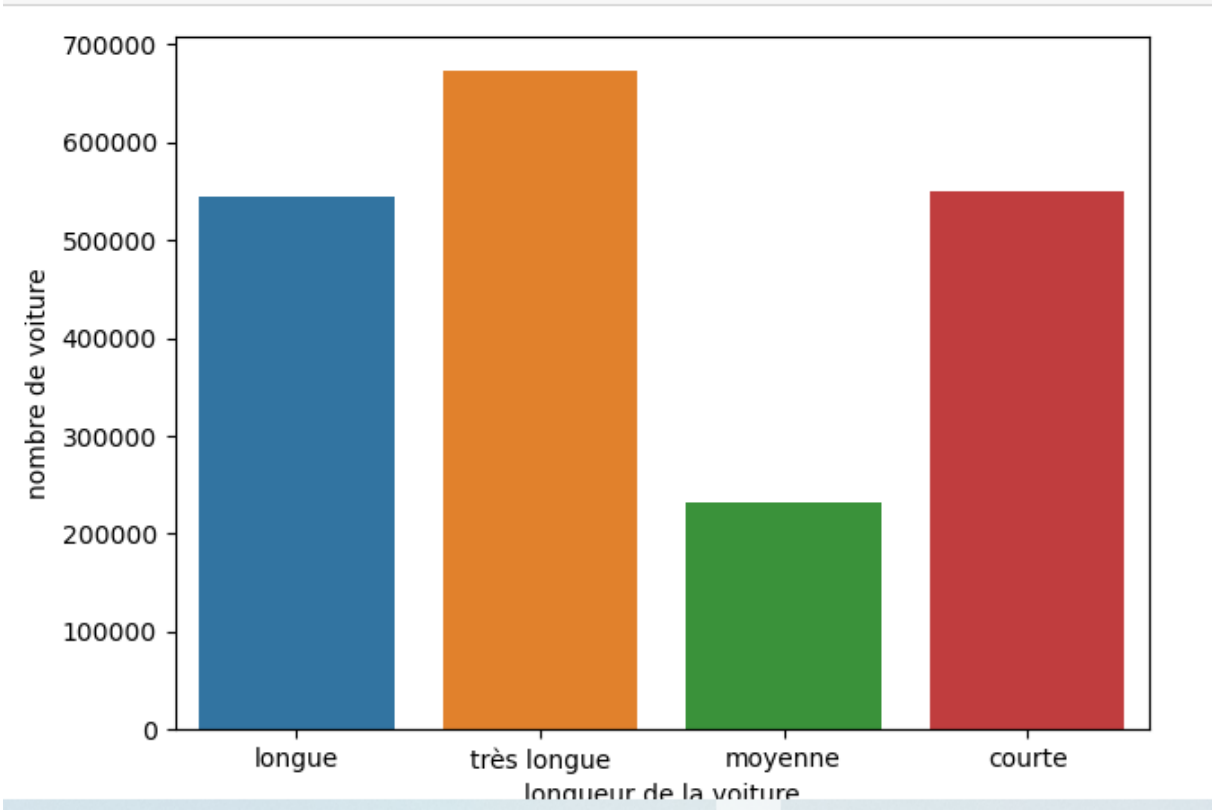
Au regard des données catalogues, ou nous avons pu faire une exploration généraliser des données, nous avons pu comprendre que seule la variable catégorielle marque, longueur, occasion contribut a la compréhension des futurs clients, néanmoins nous avons pu par les méthodes de clustérisations identifier les quatres différents catégories qu'un client pourrais facilement s'y insérer. Cependant parmit ces méthodes de clustérisation, nous pouvons retenir la méthode, par analyse principal des composantes *PCA_, puisqu'elle nous donne un meilleur ratio de la variation* 1212 *% comparer à sa deuxième composante à peine* 11 *%. Nous ne pouvons pas en parler du _PCA* sans en parle du graphique du *PCA* qui nous a bien démontre un nombre exact de clusters. Dans cette optique, la suite de notre travail consistera à crée une catégorie à quatre choix *citadelle* , *ClasRoutier_, _routier* et *sport*. Ensuite nous allons appliquer différents méthodes d'apprentissage superviser enfin de prédire la catégorie d'un client lambda.

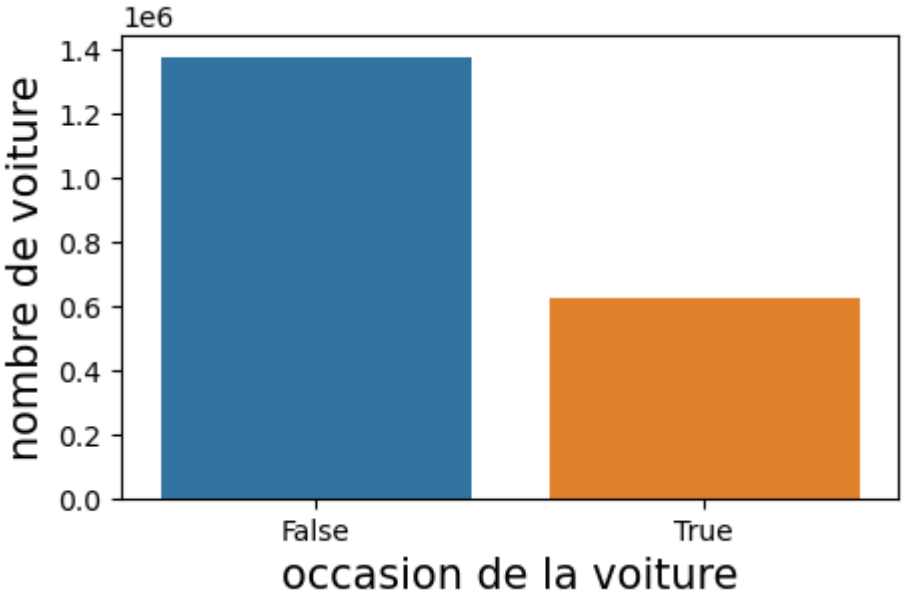
Partie 2 :

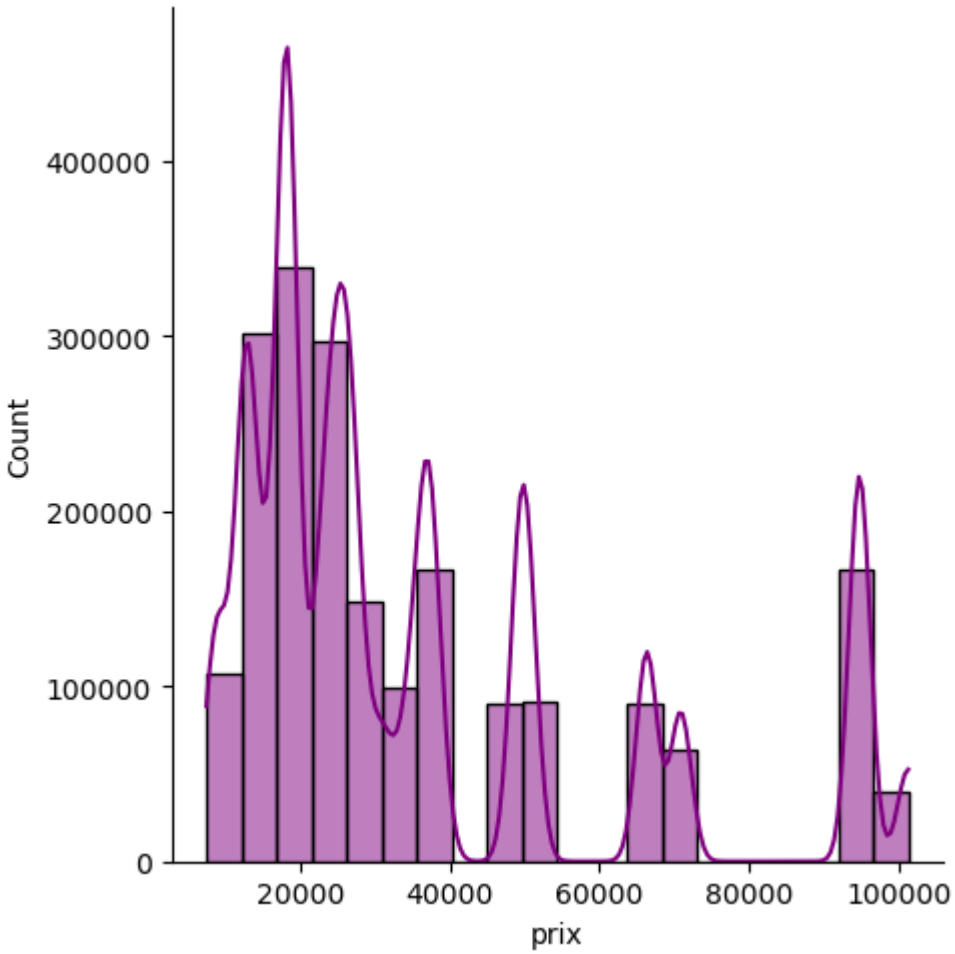
1-Test du PCA sur le dataframe Immatriculation :

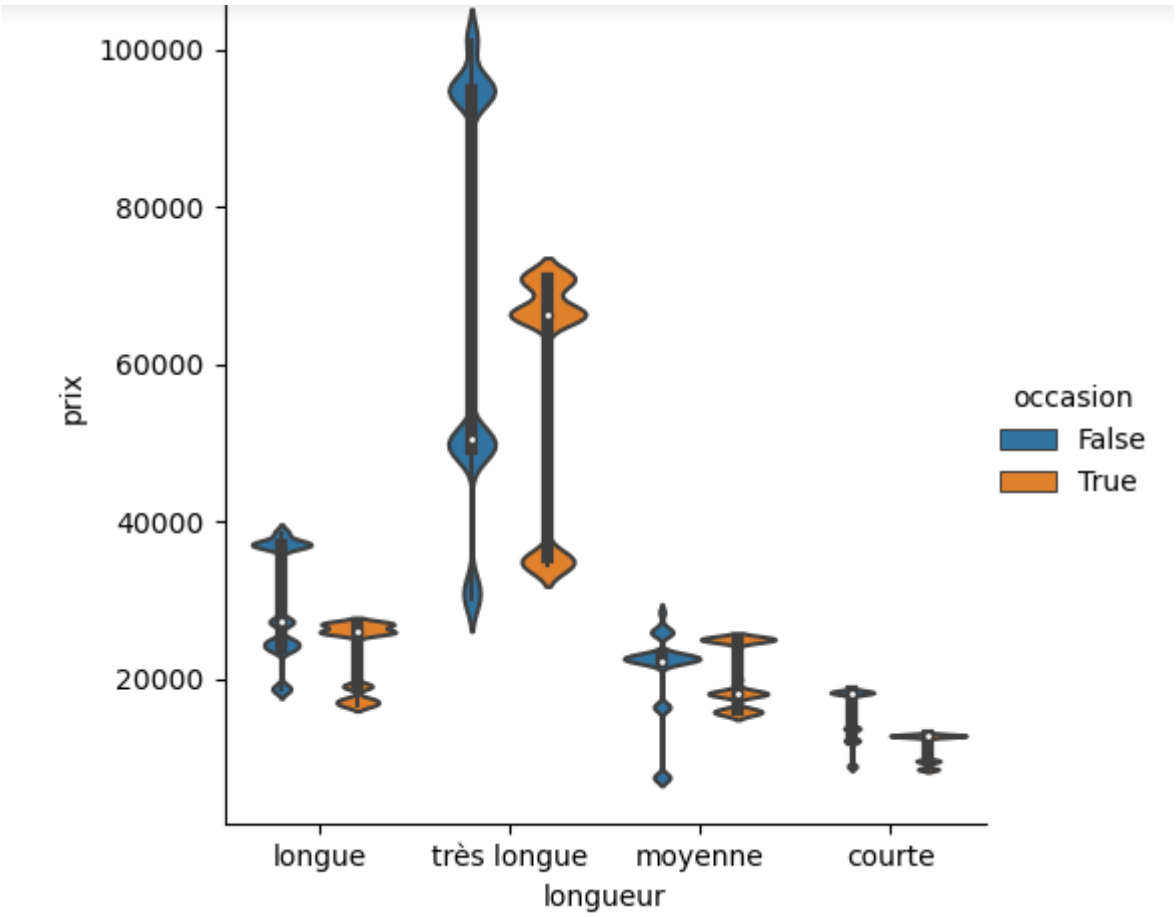
Nous allons travailler maintenant sur la nouvelle Dataset Immatriculations.csv dans laquelle on a ajouté la variable Immatriculation.

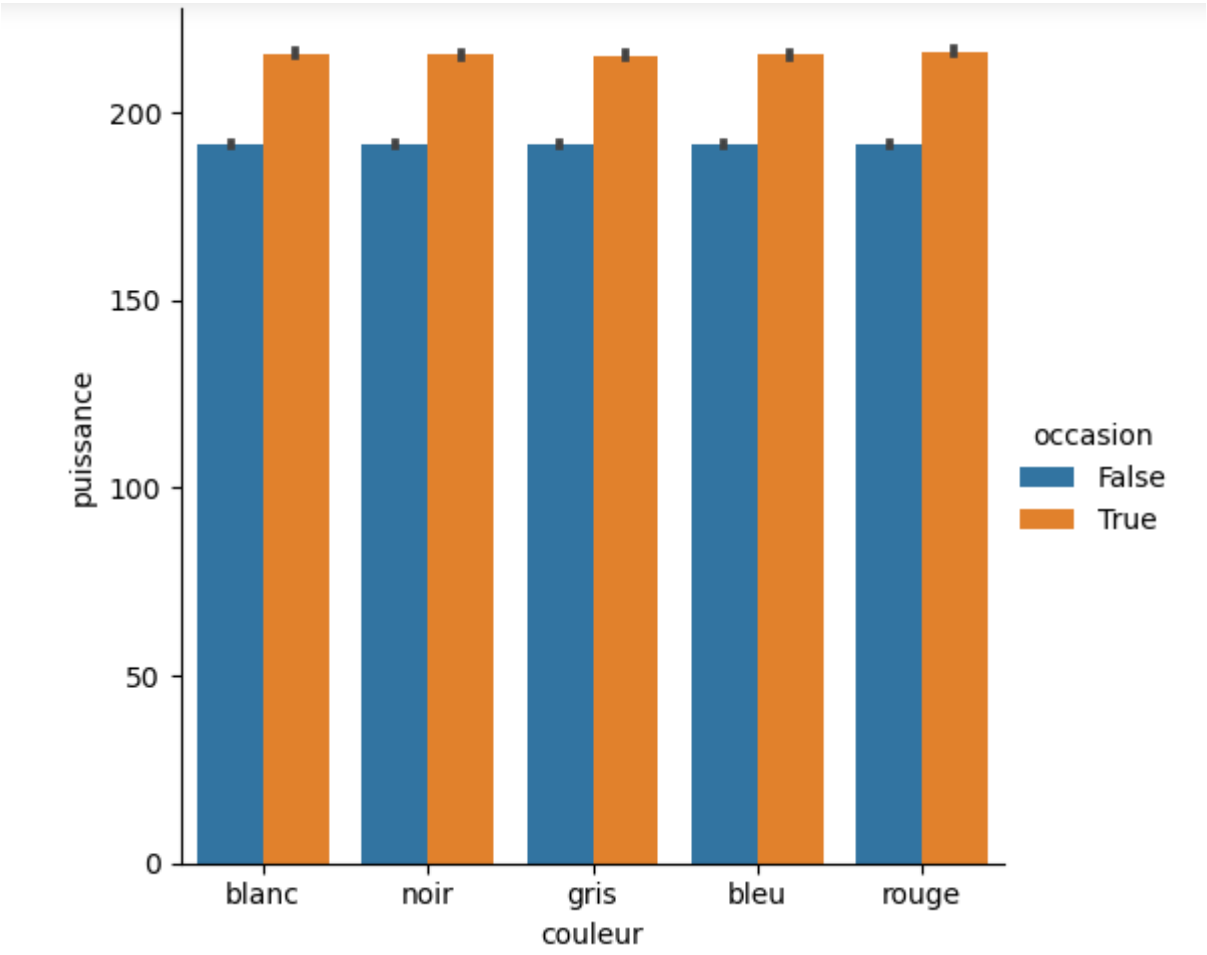
Nous avons fait quelques visualisations en utilisant des fonctions de Seaborn et Matplotlib ainsi que nous avons fait du PCA sur cette base de données.

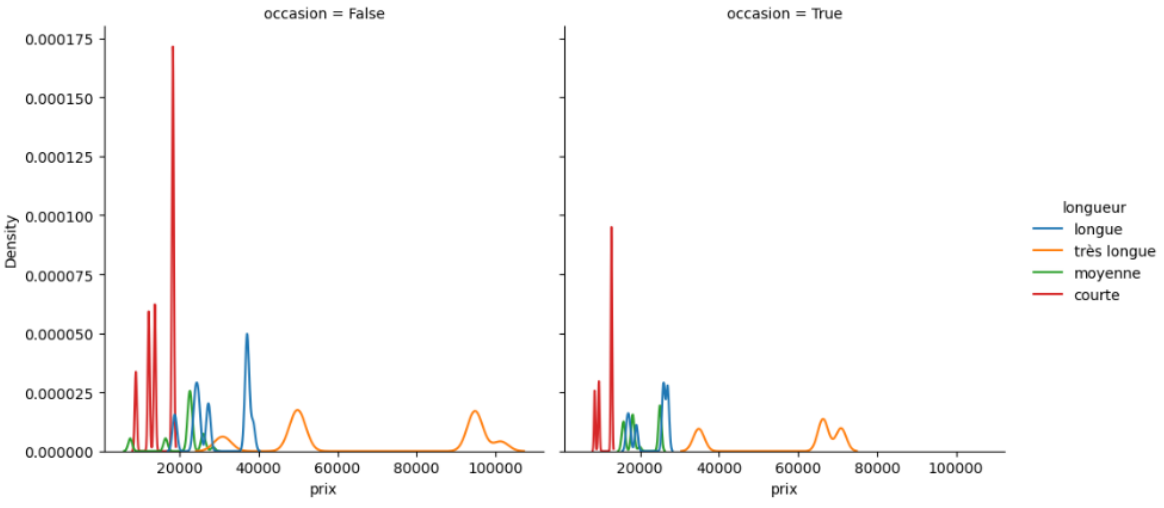


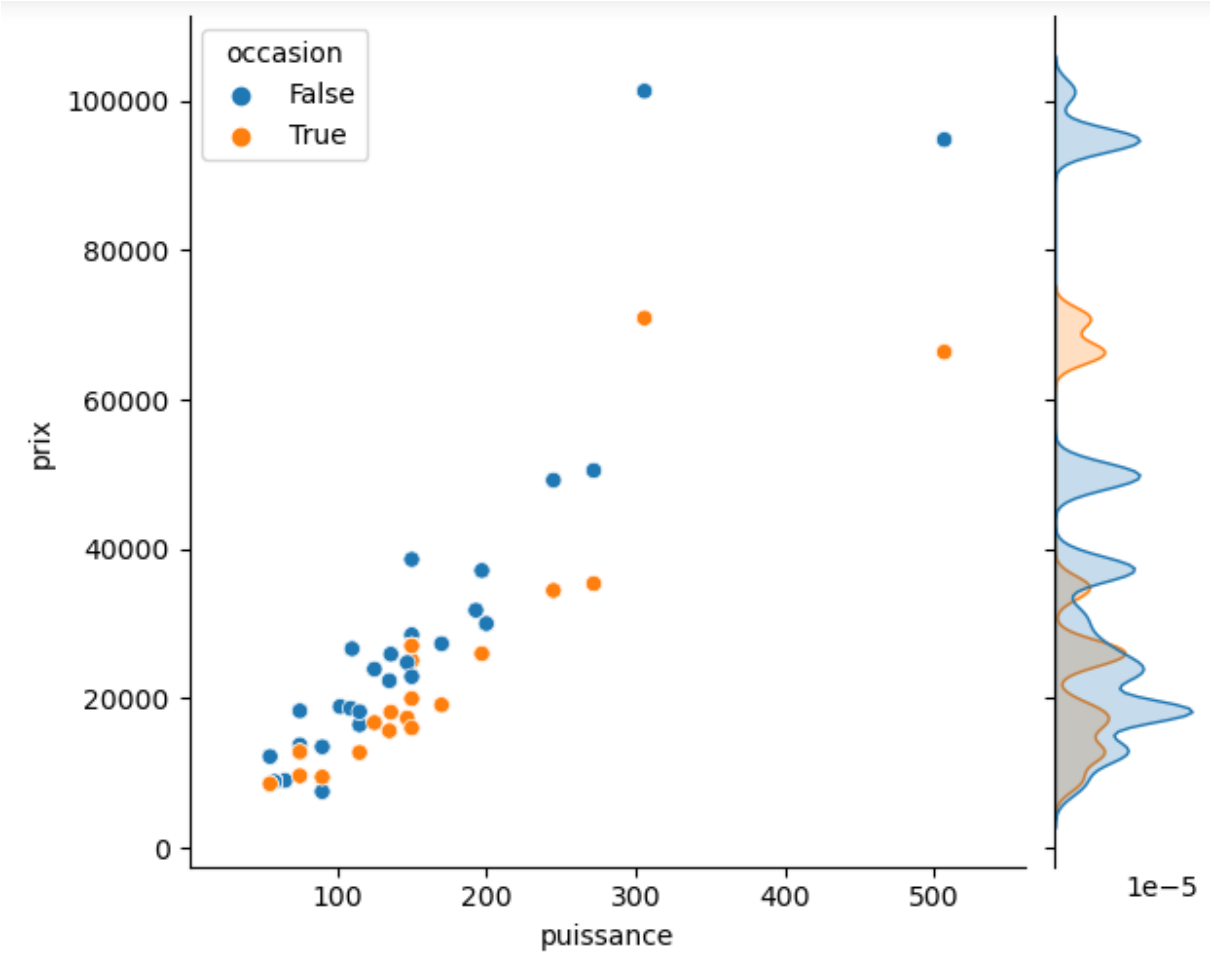

1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
nombre de voiture
False
True
occasion de la voiture

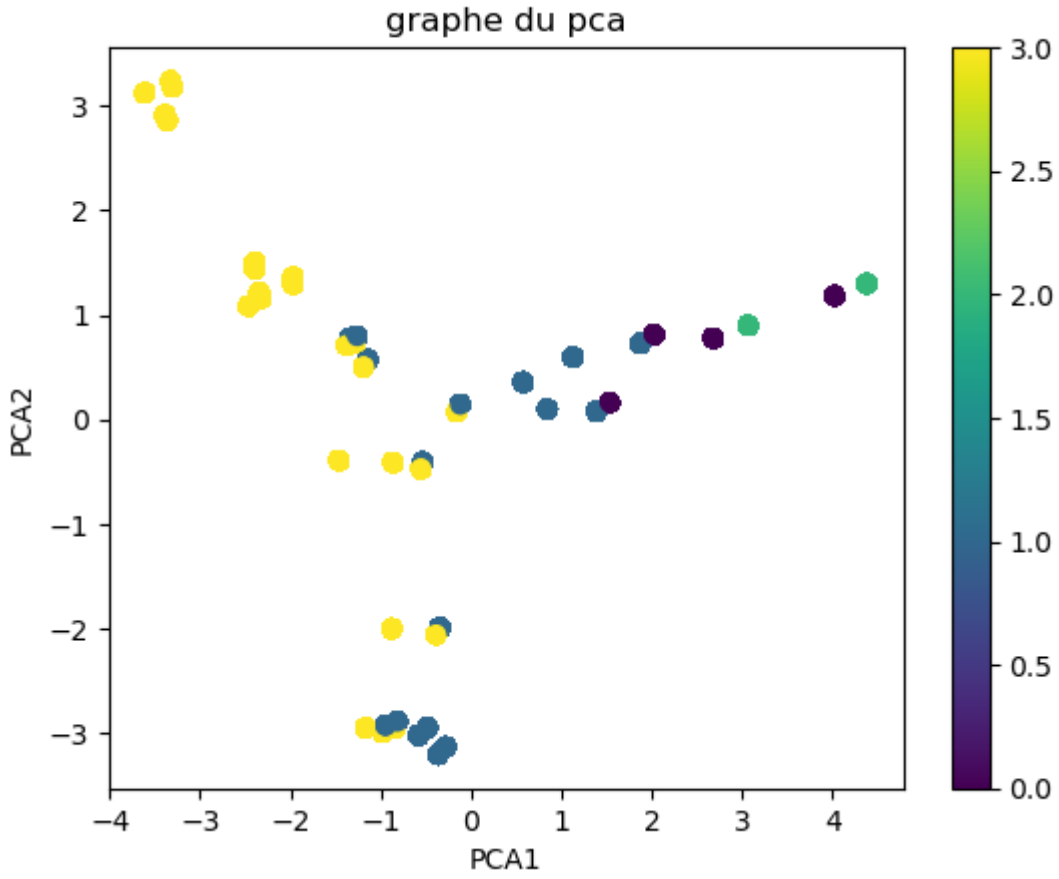

400000
300000
200000
100000
0
Count
20000
40000
60000
80000
100000
prix


100000
80000
60000
40000
20000
prix
occasion
False
True
longue
très longue
moyenne
courte
longueur


puissance
200
150
100
50
0
blanc
noir
gris
bleu
rouge
couleur
occasion
False
True


occasion = False
occasion = True
Density
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
20000
40000
60000
80000
100000
prix
longueur
longue
très longue
moyenne
courte


occasion
False
True
100000
80000
60000
40000
20000
0
prix
100
200
300
400
500
puissance
1e−5



Partie 3 :

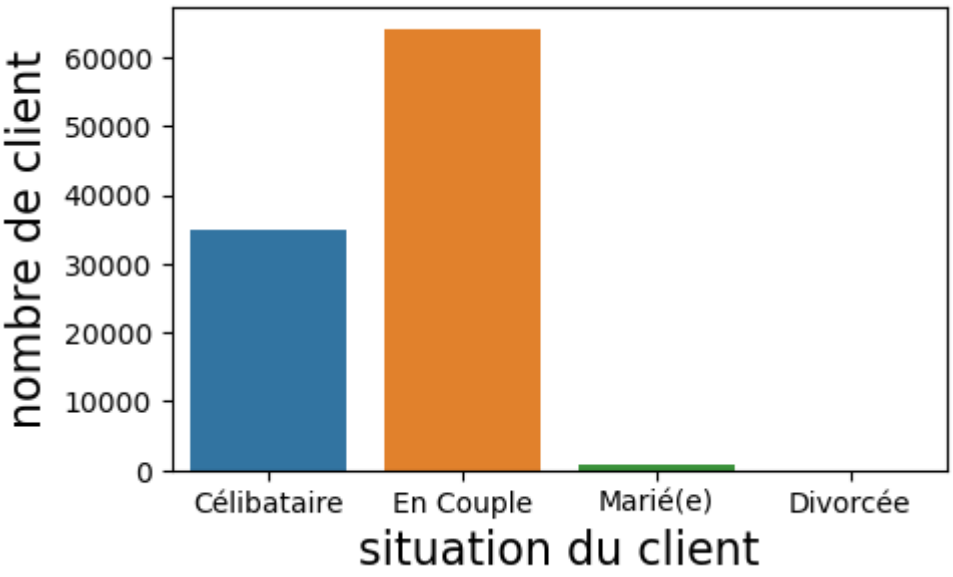
Dans cette partie nous allons intéresser au Dataframe Client.

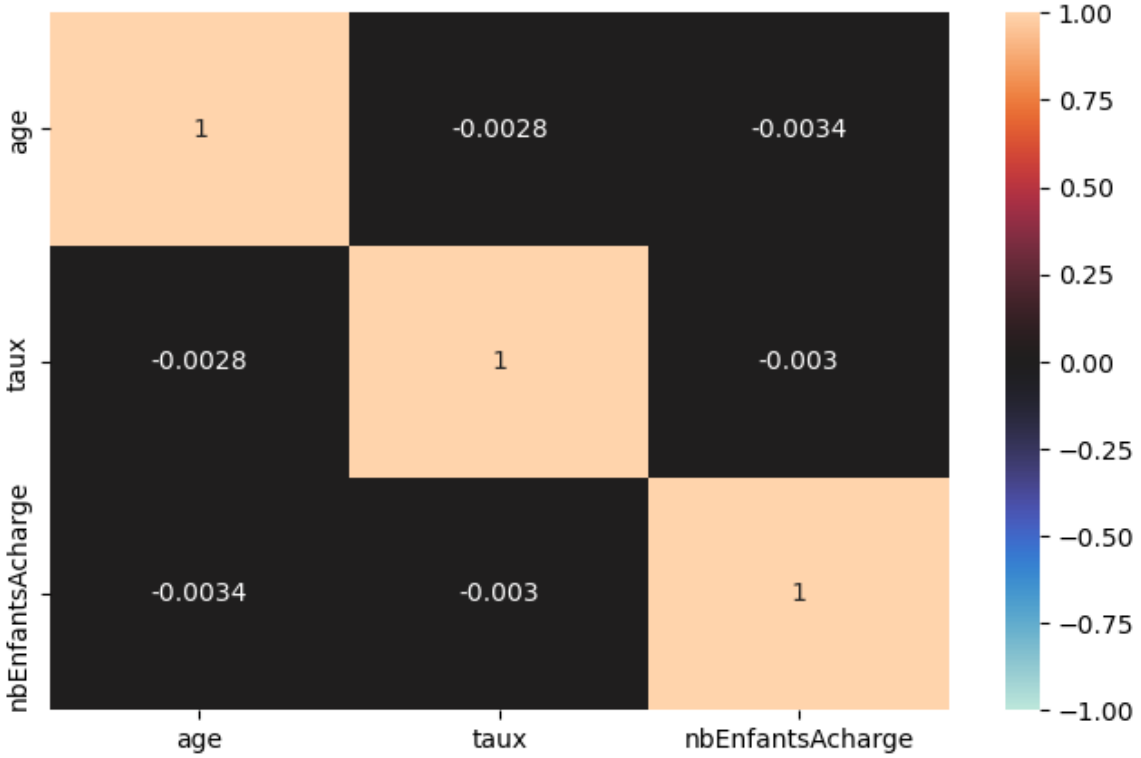
Nous avons fait une analyse et exploration des données sur cette Dataframe puis nous avons effectué un pré-traitement par remplacer nos valeurs manquantes avec des valeurs statistiques comme à chaque variable quantitative remplaçons ces variables manquantes par la médiane et à chaque variable qualitative remplaçons ces variables manquantes par le mode.

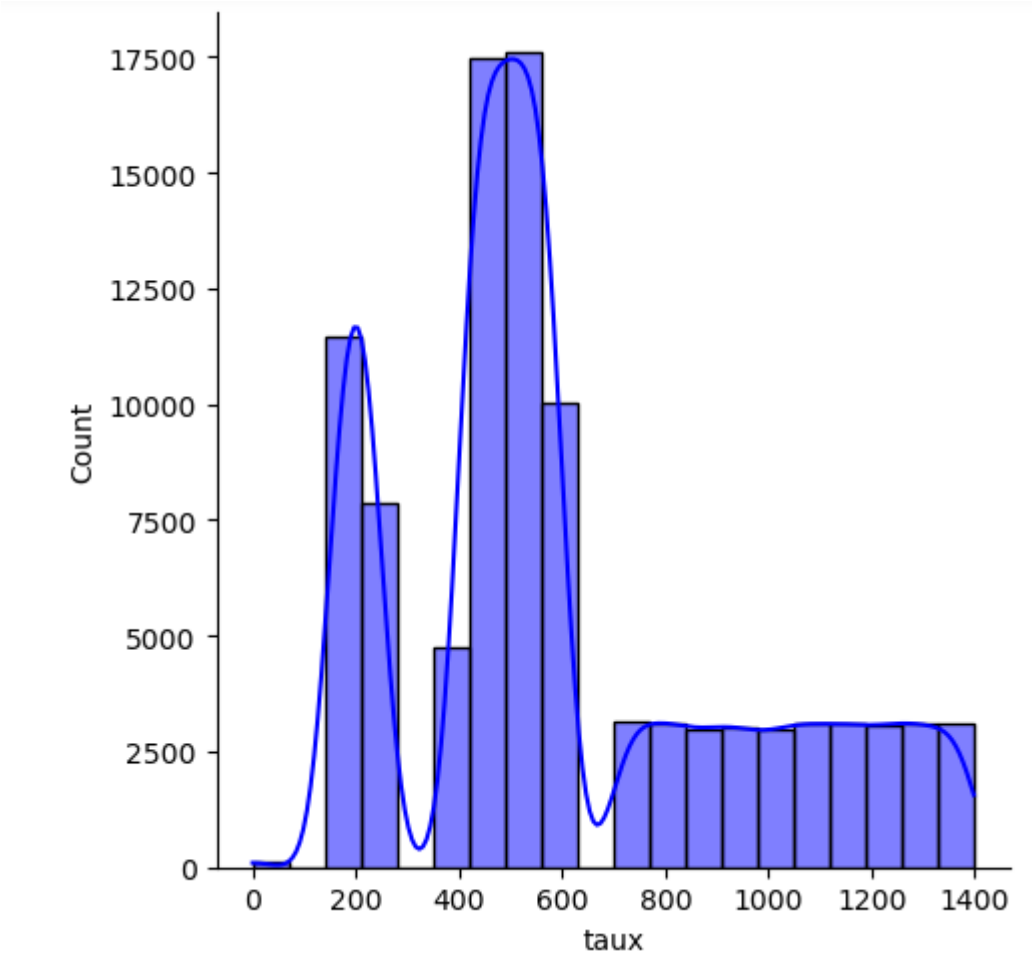
```
df_client.describe()
```

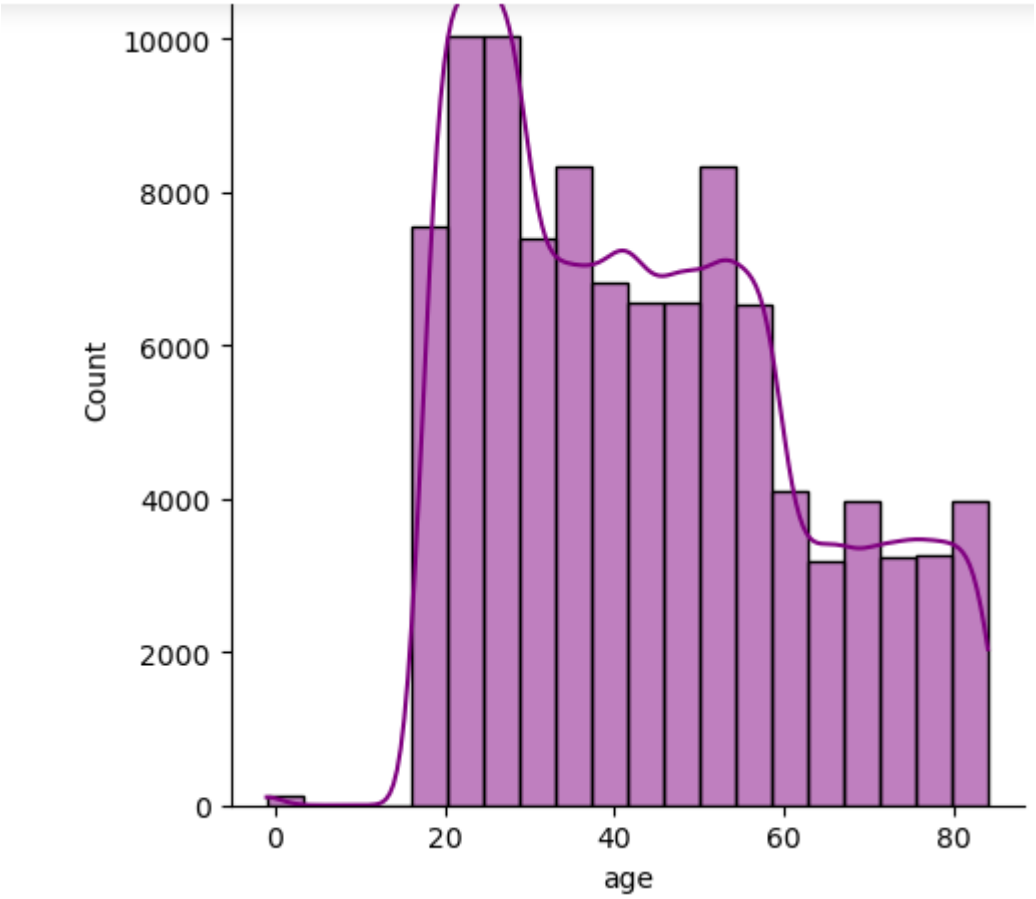
	age	taux	nbEnfantsAcharge
count	99808.000000	99786.000000	99802.000000
mean	43.631412	609.848556	1.243632
std	18.378725	336.212495	1.388275
min	-1.000000	-1.000000	-1.000000
25%	27.000000	422.000000	0.000000
50%	41.000000	522.000000	1.000000
75%	57.000000	827.000000	2.000000
max	84.000000	1399.000000	4.000000

Ensuite nous avons effectué des visualisations de nos différentes variables de notre Dataframe avec Seaborn et Matplotlib.


nombre de client
60000
50000
40000
30000
20000
10000
0
Célibataire
En Couple
Marié(e)
Divorcée
situation du client


age
taux
nbEnfantsAcharge
1
-0.0028
-0.0034
-0.0028
1
-0.003
-0.0034
-0.003
1
age
taux
nbEnfantsAcharge
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


17500
15000
12500
10000
7500
5000
2500
0
Count
0
200
400
600
800
1000
1200
1400
taux



1- Fusion des données clients avec celle d'immatriculation :

Maintenant nous allons construire une grande Dataframe en fusion les données du Client avec celle d'immatriculation.

Aussi sur cette Dataframe nous avons fait une analyse et exploration des données puis un pré-traitement sur cette Dataframe comme l'encodage et une normalisation de quelques variables de notre Dataframe.

2- Modélisation :

Maintenant après avoir travaillé sur notre Dataframe et effectuer des opérations nécessaires notre est prêt et nous allons choisir un bon modèle et l'entrainer sur cette Dataframe.

Nous avons entrainé un modèle d'Arbre de décision et nous avons une précision de 48%.

Après avoir effectué une optimisation des hyperparamètres de notre modèle avec GridSearchCV de Sklearn nous avons obtenu une amélioration de notre précision de 75%.

Nous avons essayé un autre modèle et entrainé le modèle de Random Forests et nous avons obtenu une précision de 72%, et après avoir effectué une optimisation des hyperparamètres de notre modèle nous avons une précision de 85%.

3- Utilisation des pipelines :

Nous pouvons aussi utiliser des pipelines en Machine Learning qui nous permet de construire une chaine de transformation.

En combinant un transformer, de normalisation ou encodage par exemple LabelEncoder de Sklearn, avec un estimateur ou modèle de Machine Learning nous obtenons une Pipeline c-a-d une chaine de transformation. Cet estimateur dispose des méthodes fit(), predict(), et score().

Pour créer une Pipeline en Machine Learning on doit utiliser la classe Pipeline de Sklearn puis la fonction make_pipeline pour créer un Pipeline.

Ça reste une méthode simple et rapide pour développer des modèles de ML efficace en combinant notre transfomer et notre modèle en une seule chaine de transformation en seule ligne de code.