



Rapport du TPT

**Analyse de la clientèle d'un
concessionnaire automobile dans
l'objectif de pouvoir recommander
des modèles de véhicules**

2022-2023

Enseignant

Mme. Evgeniya Ishkina

Etudiants

Dje Bi Mointi Patrice Jean-Marc

Yapi Mpkesso Carole

Table des matières

I- Identification des catégories des véhicules à partir des informations du Catalogue ..	3
1- Description du Dataset.....	3
2- Analyse exploratoire	3
II- Attribution à chaque véhicule vendu d'une catégorie	11
1- Méthodologie.....	11
2- Modèles de clustering et entraînements	11
3-Attribution des catégories.....	14
III- Entraînement d'un modèle de prédiction de catégorie de véhicule.....	16
1- Description du Dataset des clients	16
2- Chargement, vérification et correction des données	16
3. Fusion des données	19
4. Analyse exploratoire	19
5- Modèles de prédiction des catégories et entraînements	23
IV- Test et déploiement du modèle de prédiction	25
1- Test sur les données de Marketing.....	25
2- Déploiement du modèle.....	25
Conclusion.....	26

I- Identification des catégories des véhicules à partir des informations du Catalogue

1- Description du Dataset

Le Dataset à notre disposition possède 270 observations et 12 variables. Le tableau ci-dessous résume les caractéristiques des variables :

Variables	Type	Commentaire
marque	Chaîne de caractères	Marque d'un véhicule
nom	Chaîne de caractères	Modèle du véhicule
puissance	Entier	Puissance du véhicule
longueur	Entier	Longueur du véhicules
nbPlaces	Entier (5 ou 7)	Nombre de places
nbPortes	Entier (3 ou 5)	Nombre de portes
couleur	Chaîne de caractères	Couleur du véhicules
occasion	Entier (0 ou 1)	0 si le véhicule est neuf et 1 si le véhicule est d'occasion
prix	Entier	Prix du véhicule
bonus_malus	Entier	Bonus-malus écologique du véhicule
rejets_co2_gkm	Entier	Mesure de l'émission CO2 du véhicule
cout_energie	Entier	Coût énergétique du véhicule

Tableau 1 : Description des caractéristiques des variables du catalogue

Les variables catégorielles sont les suivantes : marque, nom, nbPlaces, nbPortes, couleur, occasion, longueur. Les variables numériques sont les suivantes : puissance, prix, bonus_malus, rejets_co2_gkm, cout_energie.

2- Analyse exploratoire

2.1 Chargement des données

Pour commencer l'analyse exploratoire nous avons chargé les données, effectué quelques observations et vérifié s'il y a des valeurs manquantes.

- Chargement des données

```
catalogue = pd.read_csv("data/Catalogue_cleaned_local.csv", encoding='utf8')
```

```
catalogue.shape
```

```
(270, 12)
```

Nous avons utilisé une version du catalogue de véhicules dans laquelle nous avons ajouté les informations d'émissions CO2 des véhicules.

- **Describe du DataFrame catalogue**

```
catalogue.describe()
```

	puissance	nbPlaces	nbPortes	occasion	prix	bonus_malus	rejets_co2_gkm	cout_energie
count	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000
mean	157.592593	5.222222	4.814815	0.407407	26668.055556	-2648.185185	58.685185	289.907407
std	90.551289	0.629707	0.580798	0.492264	19050.121112	4871.531247	59.340184	214.495281
min	55.000000	5.000000	3.000000	0.000000	7500.000000	-6000.000000	0.000000	72.000000
25%	109.000000	5.000000	5.000000	0.000000	16029.000000	-6000.000000	15.000000	96.000000
50%	147.000000	5.000000	5.000000	0.000000	20597.500000	-6000.000000	26.000000	206.000000
75%	170.000000	5.000000	5.000000	1.000000	30000.000000	1387.000000	108.000000	457.000000
max	507.000000	7.000000	5.000000	1.000000	101300.000000	8237.000000	187.000000	749.000000

- **Affichage des 5 premières valeurs**

```
catalogue.head()
```

	marque	nom	puissance	longueur	nbPlaces	nbPortes	couleur	occasion	prix	bonus_malus	rejets_co2_gkm	cout_energie
0	Volvo	S80 T6	272	très longue	5	5	blanc	0	50500	-6000.0	42.0	72.0
1	Volvo	S80 T6	272	très longue	5	5	noir	0	50500	-6000.0	42.0	72.0
2	Volvo	S80 T6	272	très longue	5	5	rouge	0	50500	-6000.0	42.0	72.0
3	Volvo	S80 T6	272	très longue	5	5	gris	1	35350	-6000.0	42.0	72.0
4	Volvo	S80 T6	272	très longue	5	5	bleu	1	35350	-6000.0	42.0	72.0

- **Vérification des valeurs manquantes**

```
: catalogue.isnull().sum()
```

```
: marque      0
   nom        0
   puissance  0
   longueur   0
   nbPlaces   0
   nbPortes   0
   couleur    0
   occasion   0
   prix       0
   bonus_malus 0
   rejets_co2_gkm 0
   cout_energie 0
   dtype: int64
```

On remarque qu'il n'y a pas de valeurs manquantes. A ce stage, nous avons chargé et prévisualisé les données sous forme de DataFrame. Dans la suite, nous effectuons des visualisations graphiques de l'ensemble des données.

2.2 Visualisation des données

• Variables continues

Nous avons commencé par faire un **pairplot** sur les variables numériques pour étudier leurs relations :

```
sns.pairplot(catalogue[["puissance", "prix", "bonus_malus", "rejets_co2_gkm", "cout_energie"]])
```

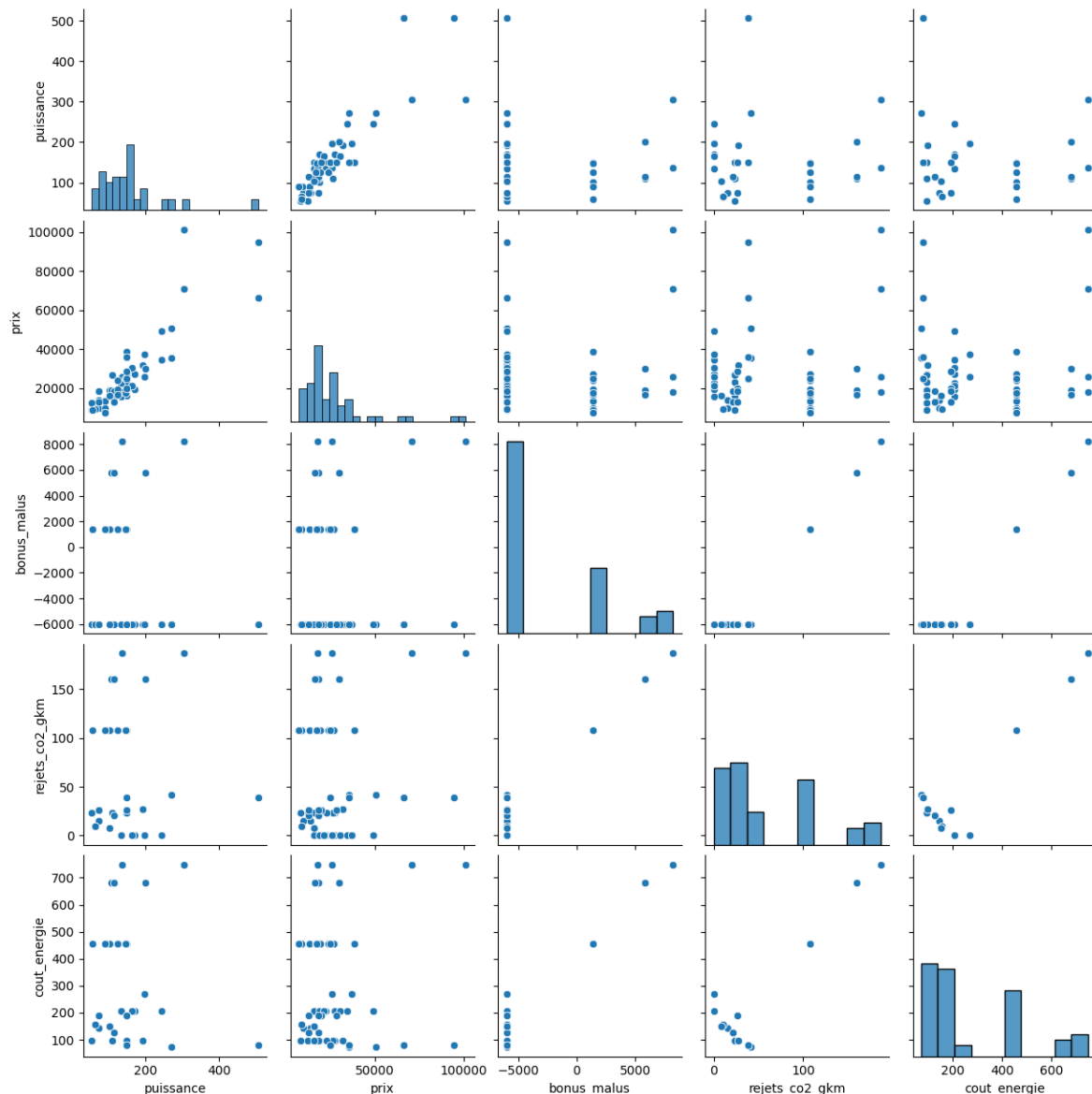


Figure 1 : Relation entre les variables numériques du catalogue

Nous avons remarqué que :

- La puissance et le prix sont liés et qu'ils possèdent tous deux des valeurs aberrantes
- Le bonus_malus, le cout_energie et le rejets_co2_gkm semblent être corrélés entre eux
- La puissance et le prix ne sont pas corrélés avec les variables d'émission carbone

- Variables catégorielles

Nous observons à la fois les différentes valeurs possibles des variables catégorielles et leur distribution :

```
sns.pairplot(catalogue[["puissance", "prix", "bonus_malus", "rejets_co2_gkm", "cout_energie"]])
```

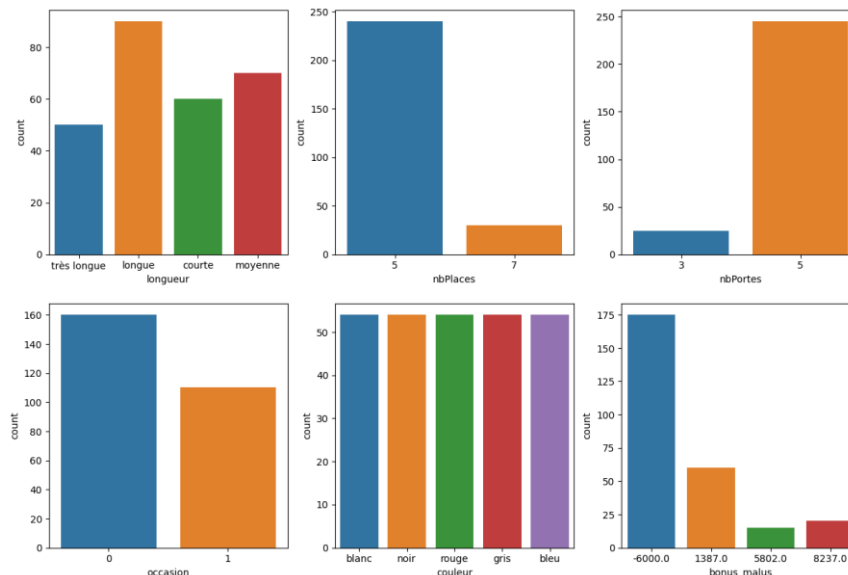


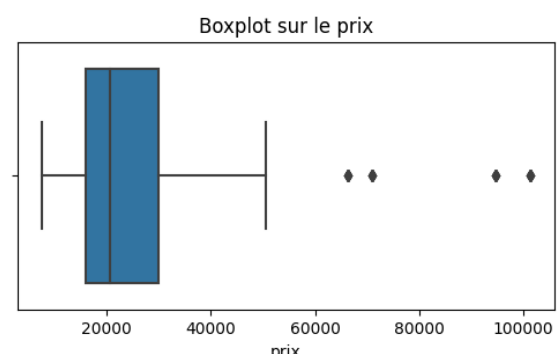
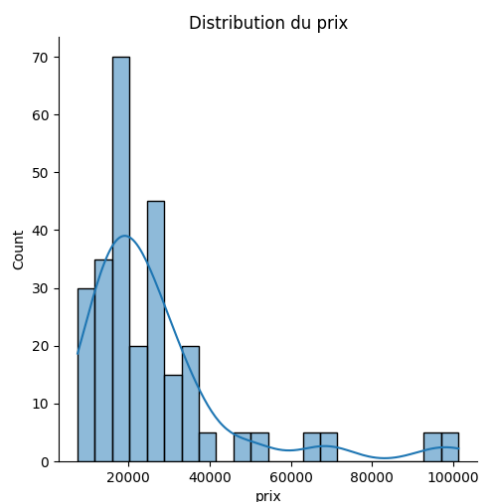
Figure 2 : Distribution des variables catégorielles

Les tableaux ci-dessous résume l'ensemble des observations :

	Domaine de valeurs	Commentaires
longueur	très longue, longue, courte, moyenne	Les voitures longues sont les plus nombreuses tandis que les très longues sont les moins nombreuses
occasion	0, 1	Il y a plus de véhicules neufs que de véhicules d'occasions
nbPlaces	5, 7	Il y a plus de véhicules ayant 5 places que de véhicules ayant 7 places
nbPortes	3, 5	Il y a plus de véhicules ayant 5 portes que de véhicules ayant 3 portes
couleur	blanc, noir, rouge, gris, bleu	La répartition des couleurs est uniforme

Tableau 2 : Tableau des observations faites sur la figure 2

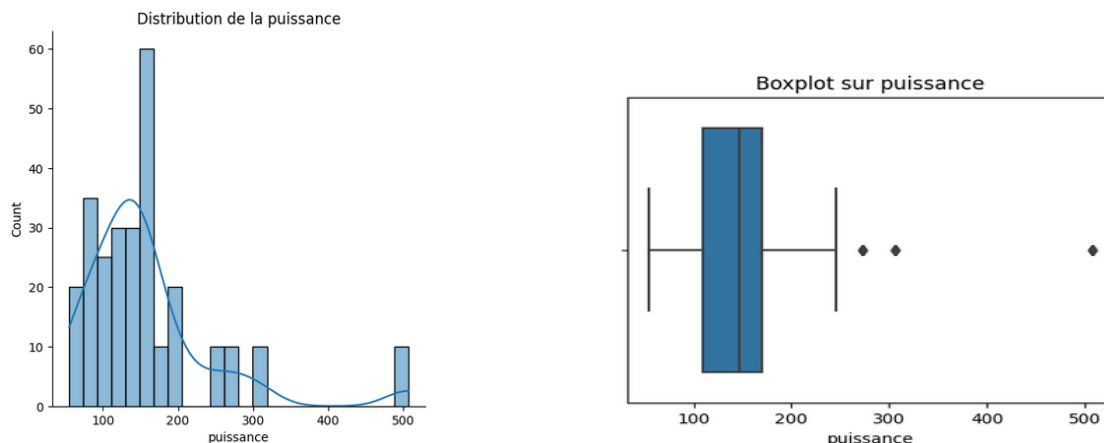
- Distribution du prix



Ci-dessus, nous avons visualisé la distribution du prix à l'aide d'un **displot** et d'un **boxplot**. Nous remarquons que les valeurs aberrantes commencent à partir de 49200. D'où on pourrait considérer que les valeurs supérieures à 49200 sont celles des véhicules les plus chers.

- **Distribution de la puissance**

Après le prix, nous avons fait les mêmes visualisations sur la puissance :



Cette fois, nous avons remarqué que les valeurs aberrantes commencent à partir de 272. On pourrait donc considérer que les valeurs supérieures à 272 sont celles des véhicules les plus puissants.

- **Bilan des visualisations**

Il est possible de séparer les variables en différent groupe :

Type de variables	Variables
Apparence	couleur
Identification du véhicule	nom, marque
Taille et spatialité	longueur, nbPlaces, nbPortes
Coût et performance	prix, puissance, occasion
Emission CO2	bonus_malus, rejets_co2_gkm, cout_energie

Tableau 3 : Tableau des types de variables dans catalogue

En particulier, plus le prix d'un véhicule est élevé, plus sa puissance l'est aussi. De plus, la puissance et le prix possèdent des valeurs aberrantes.

2.3 Relation entre les variables

Après visualisation des données nous nous sommes intéressés aux relations entre les variables comme suit :

- Relation entre prix, puissance et longueur
- Relation entre puissance et occasion
- Relation entre prix, cout_energie et rejets_co2_gkm
- Corrélation entre toutes les variables

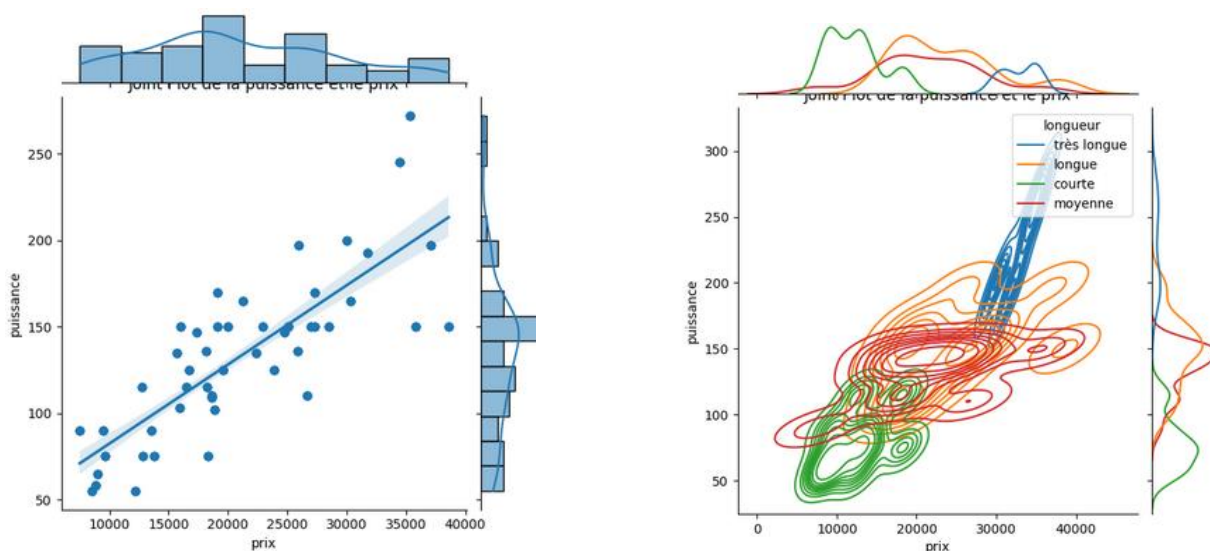
Pour classifier les prix et les puissances nous nous sommes basés sur le référentiel ci-dessous :

Classe	Puissance	Prix
Très faible	moins de 50	moins de 10 000
Faible	de 50 à 99	de 10 000 à 20 000
Moyen	de 100 à 149	de 20 000 à 30 000
Élevé	de 150 à 199	de 30 000 à 50 000
Très élevé	plus de 200	plus de 50 000

Tableau 4 : Classification des prix et puissances des véhicules en Europe selon Chat GPT

NB : Nous avons retiré les valeurs les plus aberrantes pour faciliter l'étude des relations.

- **Relation entre prix, puissance et longueur**



Les figures ci-dessus (lineplot à gauche et joinplot à droite) représentent d'une part la relation entre la puissance et le prix, mais aussi la relation entre le prix, la puissance et la longueur. La figure de droite nous montre que la règle selon laquelle plus le véhicule est cher plus sa puissance est élevée n'est pas toujours vraie car certains véhicules sont chers mais possèdent une puissance plus modérée. La figure de droite nous donne les informations ci-dessous :

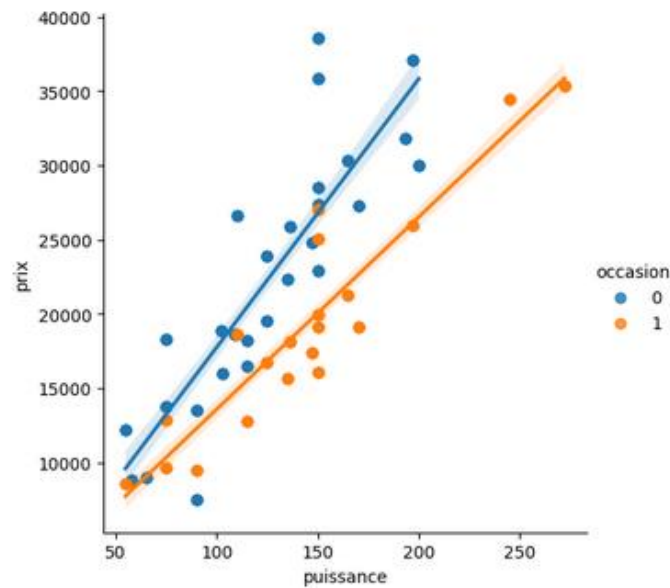
Longueur	Puissance(approximation)	Prix (approximation)
Court	[55,149]]8000,22000]
Moyen	[50,160]]7500,42000]
Long	[80,220]]11000,45000]
Très long	[150,330[]2000,38000[

Tableau 5 : Tableau des valeurs approximatives de la puissance du prix d'un véhicule en fonction de sa longueur

On remarque qu'il existe des inclusions subtiles entre la longueur, le prix et la puissance et il est difficile de caractériser les véhicules sur la base de la longueur. Cependant il est possible d'identifier les caractéristiques qu'on ne peut pas avoir :

- Il n'y a pas de voitures courtes, chères et très puissantes
- Il n'y a pas de voitures longues, bon marché, faible ou très puissantes
- Il n'y a pas de voitures moyennes, très chère et très puissante
- Il n'y a pas de voitures très longues ayant un prix et une puissance inférieur à la moyenne

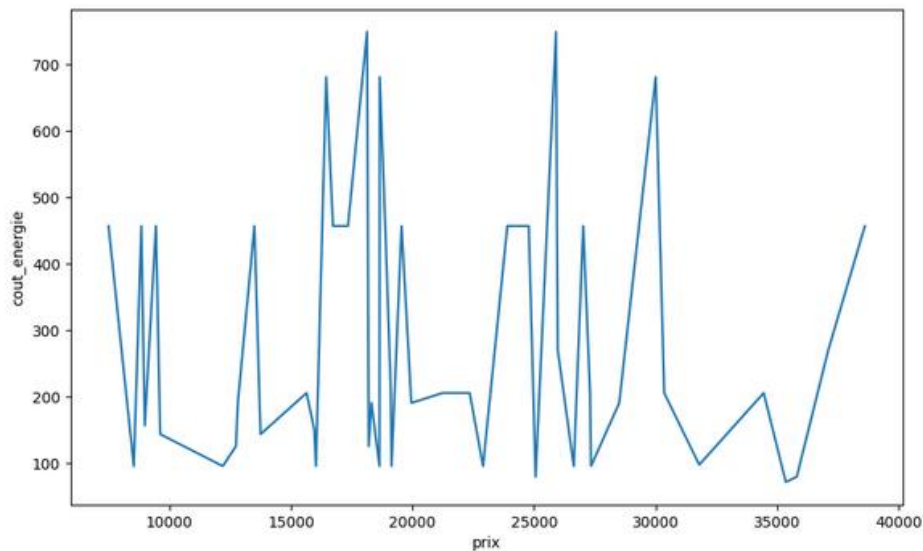
- **Relation entre puissance, prix et occasion**



La figure ci-dessous représente la relation entre la puissance et l'occasion. On remarque que les véhicules les plus chers sont le plus souvent neufs tandis que les plus puissants sont d'occasion en majorité.

- **Relation entre prix, cout_energie et rejets_co2_gkm**

```
plt.figure(figsize=(10,6))
sns.lineplot(catalogue2,x="prix",y="cout_energie")
<Axes: xlabel='prix', ylabel='cout_energie'>
```



D'après la figure ci-dessous, prix et cout_energie n'évoluent pas en même temps. Etant donné que le cout_energie est lié aux rejets_co2_gkm on aura la même représentation avec le prix. En sommes, les variables ne sont pas corrélées et il serait difficile de catégoriser les véhicules en utilisant les données d'émission CO2.

- **Matrice de corrélation**

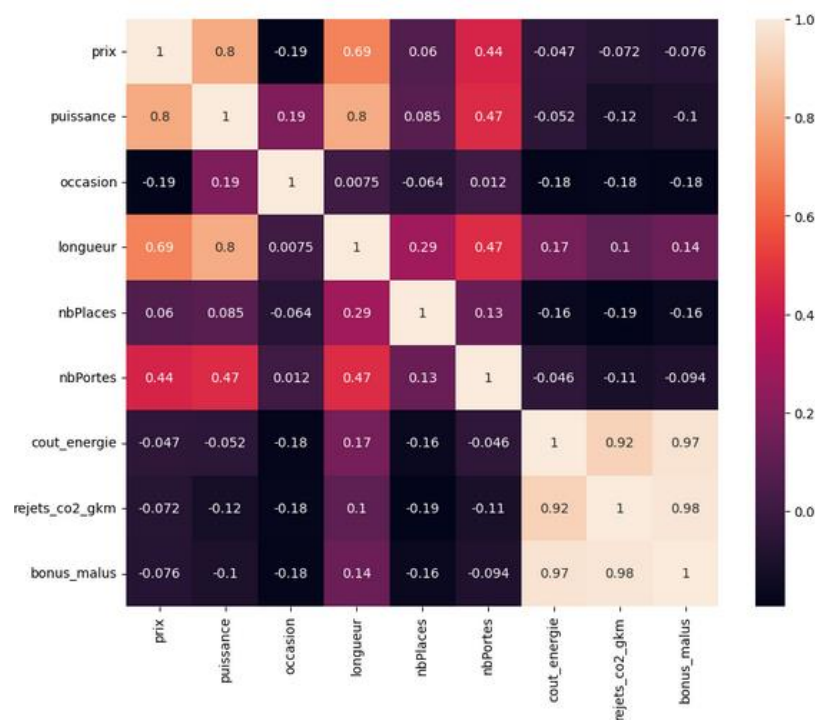


Tableau 6 : Matrice de corrélation données du catalogue

D'après la matrice de corrélation, les variables d'émission CO2 sont corrélées uniquement entre elles. Les autres variables possèdent des liens souvent forts souvent faibles.

2.4 Bilan de l'analyse exploratoire

Pour catégoriser les véhicules nous avons décidé de nous focaliser sur les variables qui décrivent les caractéristiques des véhicules : la puissance, le prix et la longueur. Pour inclure de la nuance, nous utilisons les variables occasion, nbPortes et nbPlaces. Nous avons effectué une catégorisation provisoire des véhicules sur la base de l'analyse exploratoire :

Catégorie	Description
Véhicules économiques	Véhicule court, faible puissance et coût inférieur à 20 000
Véhicules abordables	Véhicule court, faible puissance et coût entre 20 000 et 30 000
Véhicules familiales	Véhicule moyen ou long, puissance moyenne et coût entre 20 000 et 35 000
Véhicules haut de gamme	Véhicule long, puissance élevée et coût entre 35 000 et 50 000
Véhicules de luxe	Véhicule très long, très puissant et coût supérieur à 50 000

Tableau 7 : Tableau des catégories provisoire des véhicules

La catégorisation ci-dessus n'est pas la catégorisation définitive. Elle servira de base pour orienter le choix de la catégorisation final en fonction des résultats du clustering. On continuera avec l'idée que nous avons 4 ou 5 catégories véhicules.

II- Attribution à chaque véhicule vendu d'une catégorie

1- Méthodologie

Pour catégoriser les véhicules vendus nous avons entraîné des modèles de clustering sur le catalogue. L'attribution des catégories a été faite à partir du meilleur modèle entraîné, en prédisant les catégories des véhicules vendus. Nous avons donc travaillé avec l'hypothèse selon laquelle, 4 ou 5 clusters seraient optimal pour bien catégoriser les véhicules.

2- Modèles de clustering et entraînements

2.1 Préparation des données

Nous avons commencé par retirer quelque valeur aberrante de notre jeu de données. En effet, certaines valeurs aberrantes représentent une catégorie et toutes les supprimer reviendrait à supprimer cette catégorie. Dans les faits, les véhicules concernés sont ceux dont le prix est inférieur au calcul du quantile à 0.995 sur le prix. Pour trouver cette valeur nous avons étudié les valeurs du quantile à partir desquelles les valeurs étaient les plus rares par rapport à la réalité. Ainsi, le retrait des valeurs aberrantes ciblées c'est fait comme suit :

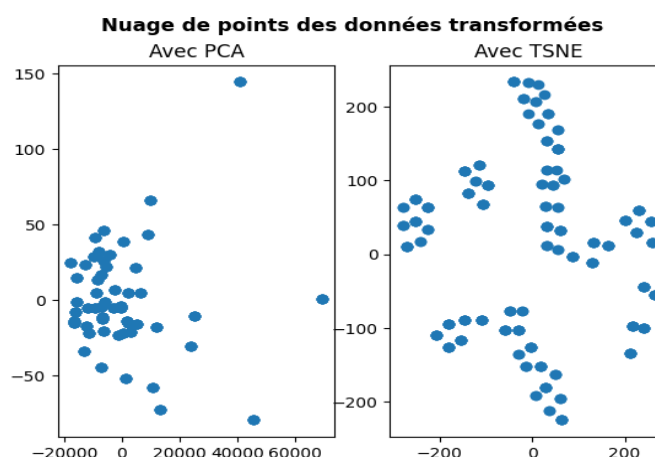
```
X = catalogue.copy()
X = catalogue[catalogue.prix > catalogue.prix.quantile(0.995)]
```

Les variables que nous avons sélectionnées pour le clustering sont les suivantes :

- Variables numériques : prix, puissance
- Variables catégorielles : longueur, nbPortes, nbPlaces, occasion

Pour leur prétraitement nous avons utilisé un pipeline qui contient les transformations ci-dessous :

Transformation	Variables
OneHotEncoder	["longueur", "nbPortes", "nbPlaces", "occasion"]
passthrough	["prix", "puissance"]



2.2 Définition des modèles

Nous avons utilisé GridSearch pour l'optimisation des hyperparamètres et la silhouette comme métrique de scoring :

Modèle	Hyperparamètres	Valeurs testées
KMeans	n_clusters	4, 5
	random_state	0,1,2,3,4
	init	random,k-means++
	n_init	auto,1,2,3,4
	max_iter	100,300,500
	tol	0.0001,0.001,0.01
KPrototypes	n_clusters	4,5
	gamma	0.1, 0.5,1.0, 2.0,5.0
	n_init	5,10,15,20
	max_iter	50,100,200
AgglomerativeClustering	n_clusters	4,5
	linkage	complete,average,single
	metric	euclidean,manhattan,cosine
GaussianMixture	n_components	4,5
	covariance_type	full, tied,diag,spherical
	init	random,k-means
	max_iter	50,100,200

2.3 Entraînements des modèles

Pour entraîner nos modèles nous avons utilisé la méthodologie suivante :

- 1- Transformation des données avec le pipeline du modèle
- 2- Entraînement du modèle avec GridSearch
- 3- Réentraînement du modèle avec les hyperparamètres optimisés
- 4- Stockage des résultats

Ci-dessous un extrait du code de l'entraînement du modèle AgglomerativeClustering.

```

#### Definition de la pipeline
agglo_pipeline = Pipeline(steps=[('transformations', transformers)])

#### fit de la pipeline
agglo_pipeline.fit(X)

#### Transformation des données
X_train = agglo_pipeline.transform(X)

#### Grille paramètre
param_grid = {
    'n_clusters': [4,5],
    'linkage': ['complete', 'average', 'single'],
    'metric': ['euclidean', 'manhattan', 'cosine']
}

#### Entraînement
agglo = AgglomerativeClustering()
grid = GridSearchCV(agglo,param_grid=param_grid,cv=4,scoring=agglo_silhouette_score)
grid.fit(X_train)

```

2.4 Evaluation des résultats et sélection du meilleur modèle

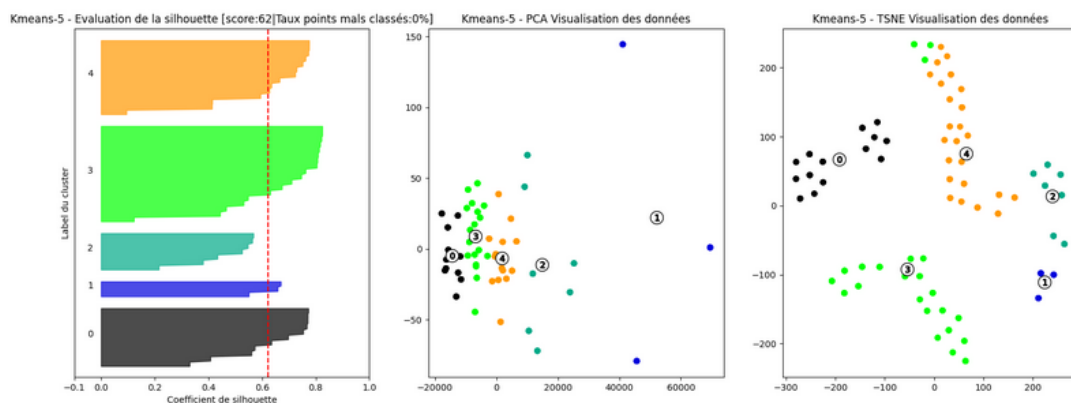
Modèle	Train score	Test score	Taux de points mal classés
KMeans	0.69	0.62	0%
KPrototypes	0.72	0.60	7%
AgglomerativeClustering	0.64	0.51	5%
GaussianMixture	0.34	0.34	1%

Le taux de points mal classés est une métrique qui donne le taux de points dont la silhouette est négative. Ce qui signifie que les points concernés sont dissimilaires des points de leur cluster et similaires aux points d'autres clusters.

D'après les résultats ci-dessous, L'AgglomerativeClustering a le meilleur score d'entraînement et KMeans le meilleur score de test. De plus, KMeans possède 0% comme taux de points mal classés et son score d'entraînement est proche de celui de AgglomerativeClustering. Nous avons sélectionné KMeans comme meilleur modèle de clustering. Ci-dessous les meilleurs hyperparamètres du Kmeans :

Modèle	Hyperparamètres	Valeurs testées
KMeans	n_clusters	5
	random_state	3
	init	random
	n_init	2
	max_iter	100
	tol	0.001

On remarque que le nombre de clusters est 5. Ce qui implique qu'on aura bel et bien 5 catégories. Ci-dessous le résultat visuel du clustering avec KMeans :



On remarque grâce à la représentation avec TSNE que quelques véhicules du cluster 3 semblent proches du cluster 4. Cela signifie que quelques véhicules du cluster 3 pourraient être assimilés au cluster 4.

2.4 Déduction de la catégorisation

L'analyse de la composition des clusters ci-dessous nous a permis de déduire une catégorisation des véhicules en fonction des clusters :

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
marques	['Volkswagen' 'Peugeot' 'Mini' 'Lancia' 'Kia' ...]	['Mercedes' 'BMW']	['Volvo' 'Saab' 'Renault' 'Jaguar' 'BMW']	['Volkswagen' 'Seat' 'Renault' 'Nissan' 'Mini'...]	['Volkswagen' 'Skoda' 'Saab' 'Renault' 'Nissan'...]
noms	['Polo 1.2 6V' '1007 1.4' 'Copper 1.6 16V' 'Yp...']	['S500' 'M5']	['S80 T6' '9.3 1.8T' 'Vel Satis 3.5 V6' 'X-Typ...']	['Touran 2.0 FSI' 'New Beetle 1.8' 'Golf 2.0 F...']	['Touran 2.0 FSI' 'New Beetle 1.8' 'Golf 2.0 F...']
puissances	55-115	306-507	150-272	75-170	110-200
prix	7500-13750	66360-101300	34440-50500	15644-22350	22900-31790
longueurs	['courte' 'moyenne']	['très longue']	['très longue' 'longue' 'moyenne']	['longue' 'moyenne' 'courte']	['longue' 'moyenne' 'très longue']
nbPlaces	5	5	5	5	5
nbPortes	5	5	5	5	5
nbVehicules	55	20	35	90	70

Cluster	Description	Catégorie
0	Contient les véhicules court et moyens à prix et puissance relativement faible	Citadine
1	Contient les véhicules chères, très longs et très puissant	Haut de gamme
2	Contient les véhicules moyens, longs et très longs à prix et puissances relativement moyenne et élevée	Familiale
3	Contient les véhicules court, moyen et longs à prix et puissance relativement faible et moyen	Economique
4	Contient les véhicules moyens, longs et très longs à prix et puissance relativement moyenne et élevé	Standard (Moyen)

3-Attribution des catégories

3.1 Chargement des données et vérifications

Nous avons commencé par charger les données pour ensuite les contrôler :

```
immatriculations = pd.read_csv("data/Immatriculations.csv", encoding='latin-1')

print("Doublons d'immatriculations :", immatriculations.immatriculation.duplicated().sum())
print("Lignes en doublon :", immatriculations.duplicated().sum())
print("Nombre de lignes : ", immatriculations.shape[0])

Doublons d'immatriculations : 3368
Lignes en doublon : 24
Nombre de lignes : 2000000
```

Nous avons remarqué la présence de 24 immatricules en double et 3368 doublons sur les 2000000 véhicules vendus. Nous avons supprimé ces doublons comme suit :

```
immatriculations2 = immatriculations.drop(index=immatriculations[immatriculations.immatriculation.duplicated()].index)
immatriculations2.drop_duplicates(inplace=True)

print("Doublons d'immatriculations :", immatriculations2.immatriculation.duplicated().sum())
print("Lignes en doublon :", immatriculations2.duplicated().sum())
print("Nombre de lignes : ", immatriculations2.shape[0])

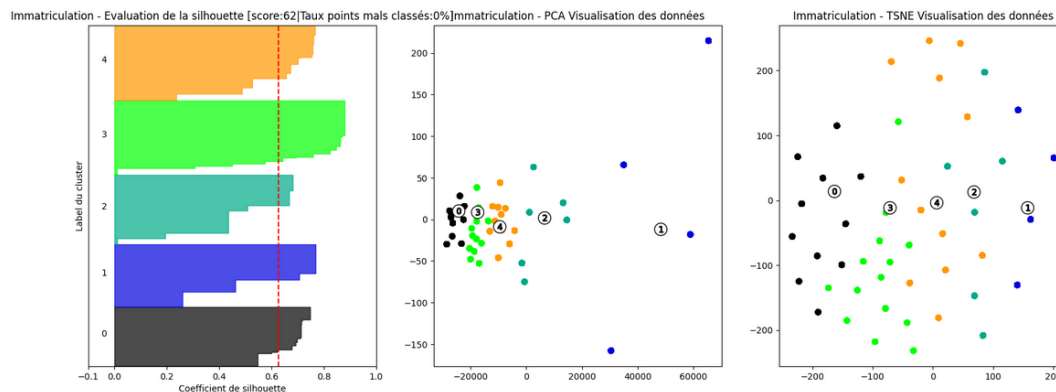
Doublons d'immatriculations : 0
Lignes en doublon : 0
Nombre de lignes : 1996632
```

3.2 Prédiction des catégories

Pour attribuer les catégories nous avons appliqué notre modèle de clustering sur les véhicules. Pour ce faire nous avons exécuté le code ci-dessous :

```
X_im = pipeline.transform(immatriculations2)
labels = categorieClusteringModel.predict(X_im)
immatriculations2["categorie_label"] = labels
```

Les figures ci-dessous présentent l'ensemble des résultats des prédictions :



La visualisation du clustering montre une similitude entre les prédictions sur les véhicules vendus et celles sur le catalogue. Mais avec TSNE on remarque que les points sont dispersés.

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
marques	['Peugeot' 'Volkswagen' 'Kia' 'Daihatsu' 'Audi...']	['BMW' 'Mercedes']	['Volvo' 'Jaguar' 'Renault' 'Saab']	['Volkswagen' 'Audi' 'Renault' 'Mercedes' 'For...']	['Renault' 'Skoda' 'BMW' 'Saab' 'Jaguar' 'Ford...']
noms	['1007 1.4' 'Polo 1.2 6V' 'Picanto 1.1' 'Cuore...']	['M5' 'S500']	['S80 T6' 'X-Type 2.5 V6' 'Vel Satis 3.5 V6' '...']	['Golf 2.0 FSI' 'A2 1.4' 'Megane 2.0 16V' 'A20...']	['Laguna 2.0T' 'Superb 2.8 V6' '120i' '9.3 1.8...']
puissances	55-115	306-507	150-272	75-170	110-200
prix	7500-13750	66360-101300	34440-50500	15644-22350	22900-31790
longueurs	['courte' 'moyenne']	['très longue']	['très longue' 'longue']	['moyenne' 'courte' 'longue']	['longue' 'très longue' 'moyenne']
nbPlaces	5	5	5	5	5
nbPortes	5	5	5	5	5
nbVehicules	345888	359719	412607	439363	439055

Le DataFrame ci-dessus montre que la composition des clusters concorde avec celle obtenue lors de l'entraînement du modèle.

	immatriculation	marque	nom	puissance	longueur	nbPlaces	nbPortes	couleur	occasion	prix	categorie_label	categorie
0	3176 TS 67	Renault	Laguna 2.0T	170	longue	5	5	blanc	False	27300	4	Moyen
1	3721 QS 49	Volvo	S80 T6	272	très longue	5	5	noir	False	50500	2	Familliale
2	9099 UV 26	Volkswagen	Golf 2.0 FSI	150	moyenne	5	5	gris	True	16029	3	Economique
3	3563 LA 55	Peugeot	1007 1.4	75	courte	5	5	blanc	True	9625	0	Citadine
4	6963 AX 34	Audi	A2 1.4	75	courte	5	5	gris	False	18310	3	Economique
5	5592 HQ 89	Skoda	Superb 2.8 V6	193	très longue	5	5	bleu	False	31790	4	Moyen

Le DataFrame des 6 premiers véhicules catégorisés montrent que la catégorisation est cohérente vis-à-vis des caractéristiques des véhicules.

III- Entraînement d'un modèle de prédiction de catégorie de véhicule

1- Description du Dataset des clients

L'objectif de cette partie est de mettre en place un modèle qui permet de classer les clients par catégorie de véhicule. Cette approche permet de mettre en place un modèle de recommandation de catégories de véhicules. Une fois la catégorie identifiée, il devient plus facile de proposer des véhicules adaptés au client. Pour ce faire, nous avons à notre disposition 2 Dataset (Clients_51.csv et Clients_52.csv) de 10 000 observations chacun et 7 variables explicatives.

Variables	Type	Commentaire
age	Entier	Age du client
sexe	Chaîne de caractères	Sexe du client (F ou M)
taux	Entier	Capacité d'endettement du client (30% du salaire)
situationFamiliale	Chaîne de caractères	Marié, Célibataire, Divorcé, Seule, Seul, En couple
nbEnfantsAcharge	Entier	Nombre de places
2eme voiture	Booléen	Le client possède-t-il déjà un véhicule
immatriculation	Chaîne de caractères	Immatriculation du véhicule vendu

Les variables catégorielles sont les suivantes : sexe, age, situationFamiliale, 2eme voiture, nbEnfantsAcharge. Les variables numériques sont les suivantes : taux. Les données en l'état ne possèdent pas d'informations sur la catégorie de véhicule vendu. Ainsi, nous avons utilisé la variable immatriculation pour fusionner les données des véhicules vendus et les données des clients pour faire ressortir les catégories des véhicules achetés par les clients.

2- Chargement, vérification et correction des données

2.1 Chargement des données

Les données des clients étant dans deux fichiers distincts nous avons chargés les fichiers puis concaténé les données pour obtenir un seul DataFrame.

```
# Données des clients
clients_51 = pd.read_csv("data/Clients_51.csv", encoding='latin-1')
clients_52 = pd.read_csv("data/Clients_52.csv", encoding='latin-1')
clients_5 = pd.concat([clients_51, clients_52])
clients_5.head(5)
```

	age	sexe	taux	situationFamiliale	nbEnfantsAcharge	2eme voiture	immatriculation
0	49	F	914	En Couple	1	false	2170 DJ 60
1	18	M	563	En Couple	4	false	8132 RT 49
2	82	M	417	Célibataire	0	false	4764 CE 84
3	72	M	442	En Couple	4	false	6239 YO 57
4	41	M	592	Célibataire	0	false	9318 FD 10

2.1 Vérification et correction des données

- Vérification des valeur manquantes

```
clients_data.isnull().sum()
```

```
age                0
sexe               0
taux              0
situationFamiliare 0
nbEnfantsAcharge  0
2emevoiture       0
immatriculation    0
dtype: int64
```

La figure ci-dessus montre qu'il n'y a pas de valeurs manquantes.

- Vérification des types

```
clients_data.dtypes
```

```
age                object
sexe              object
taux              object
situationFamiliare object
nbEnfantsAcharge  object
2emevoiture       object
immatriculation   object
dtype: object
```

La figure ci-dessus montre que tous les types sont de type objet ce qui n'est pas correcte. Nous avons donc corrigé les types incorrects comme suit :

```
clients_data.age = pd.to_numeric(clients_data.age, errors="coerce")
clients_data.taux = pd.to_numeric(clients_data.taux, errors="coerce")
clients_data.nbEnfantsAcharge = pd.to_numeric(clients_data.nbEnfantsAcharge, errors="coerce")
clients_data["2emevoiture"] = clients_data["2emevoiture"].map({
    "true":True,
    "false":False
})
```

- Vérification des valeurs

Nous savions que les données possédaient des valeurs incohérentes à savoir : " ", "?", "N/D", "-1". Nous avons remplacé ces valeurs par NaN.

```
# Remplacement des valeurs inattendue par nan
clients_data.replace([" ", "?", "N/D", "-1", -1], np.nan, inplace=True)
```

Nous avons aussi remarqué que la variable sexe présentait des valeurs inattendues (Masculin, Féminin, Homme, Femme) au lieu de F ou M. Nous avons corrigé cela comme suit :

```
clients_data["sexe"] = sexe_vals = clients_data.sexe.map({
    "Masculin": "M",
    "Homme": "M",
    "Femme": "F",
    "Féminin": "F",
    "M": "M",
    "F": "F"
})
clients_data.sexe.value_counts()
```

```
M    139400
F     60014
```

Nous avons remarqué que la variable situationFamiliale présentait valeurs qui désignait le même ensemble de valeurs (Seule, Seul, Divorcé = Célibataire ; Marié(e) = En Couple). Nous avons donc regroupé l'ensemble des valeurs en deux types : Célibataire et En Couple.

```
clients_data.situationFamiliale = clients_data.situationFamiliale.map({
    "En Couple": "En Couple",
    "Célibataire": "Célibataire",
    "Seule" : "Célibataire",
    "Marié(e)" : "En Couple",
    "Seul" : "Célibataire",
    "Divorcée": "Célibataire"
})
clients_data.situationFamiliale.value_counts()
```

```
En Couple      129438
Célibataire    69930
```

- Correction des valeurs manquantes

La suppression des valeurs inattendues a engendré l'apparition de valeurs manquantes dans le Dataset. Nous avons étudié ces valeurs et avons remarqué qu'il y avait peu de valeurs manquantes par rapport à l'ensemble des données :

```
(clients_data.isnull().sum() / clients_data.shape[0]) * 100
```

```
age          0.2845
sexe         0.2930
taux         0.3205
situationFamiliale 0.3160
nbEnfantsAcharge 0.3020
2emevoiture  0.1930
immatriculation 0.0000
dtype: float64
```

```
# Nombre de lignes avec valeurs manquantes
nbr_miss_line = clients_data[clients_data.isna().any(axis=1)].shape[0]
nbr_miss_line
```

```
3396
```

```
x = (nbr_miss_line/clients_data.shape[0]) * 100
print("Le taux de lignes manquantes est :",x)
```

```
Le taux de lignes manquantes est : 1.698
```

Il y a 1,698% de lignes avec des valeurs manquantes, soit 3394 lignes. Ce nombre étant assez faible nous avons opté pour la suppression des valeurs manquantes :

```
clients_data2 = clients_data.copy()
clients_data2.dropna(inplace=True)
clients_data2.isnull().sum()
```

```
age          0
sexe         0
taux         0
situationFamiliale 0
nbEnfantsAcharge 0
2emevoiture  0
immatriculation 0
dtype: int64
```

- Ajustement des types

Nous avons constaté que les types étaient correctes (Objet ou float) mais qu'ils n'étaient pas exacts. C'est-à-dire qu'un type int pouvait être un float. Nous avons donc défini les types exacts des variables.

```
clients_data2["age"] = clients_data2["age"].astype(np.int64)
clients_data2["2emevoiture"] = clients_data2["2emevoiture"].astype(bool)
clients_data2["nbEnfantsAcharge"] = clients_data2["nbEnfantsAcharge"].astype(np.int64)
clients_data2["taux"] = clients_data2["taux"].astype(np.int64)
```

3. Fusion des données

La fusion des données s'est faite à partir de la variable immatriculation qui est présente dans les DataFrame des clients et des véhicules vendus :

```
cols = ["age", "sexe", "taux", "nbEnfantsAcharge", "situationFamiliare", "2emevoiture", "categorie_label", "categorie"]
```

```
clients_immatriculations = clients_data2.merge(immatriculations, on="immatriculation")
```

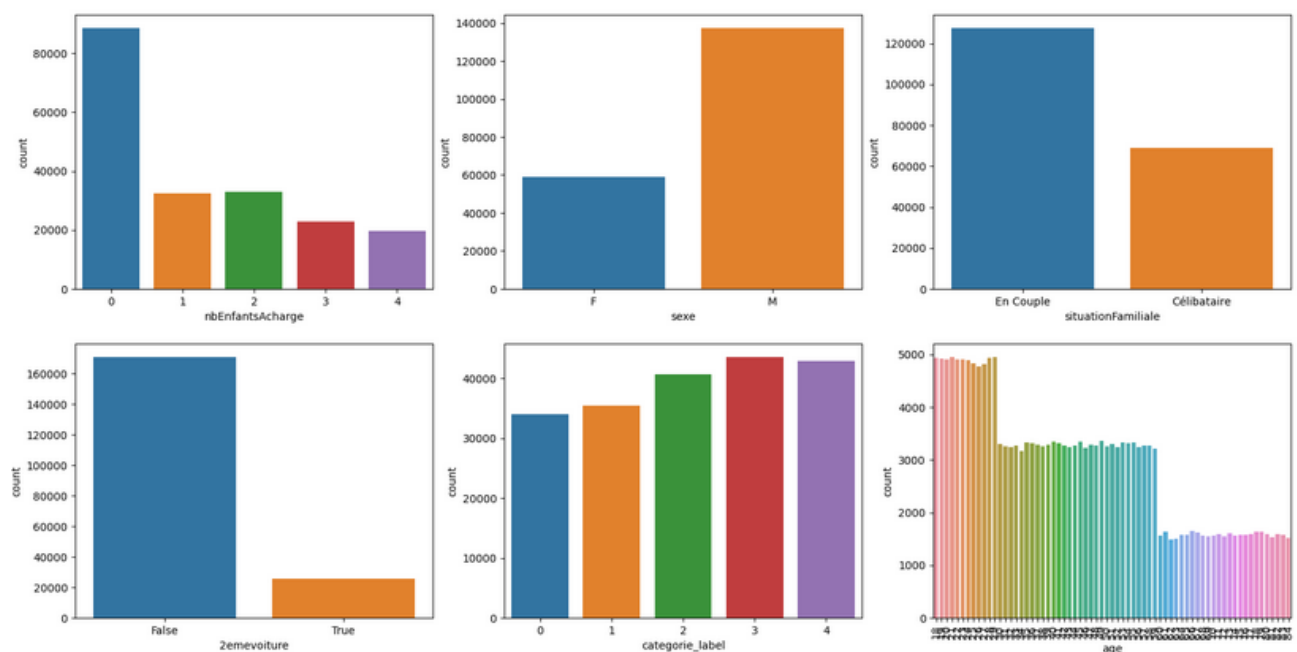
```
clients_immatriculations = clients_immatriculations[cols]
```

```
clients_immatriculations.head(10)
```

	age	sexe	taux	nbEnfantsAcharge	situationFamiliare	2emevoiture	categorie_label	categorie
0	49	F	914	1	En Couple	False	2	Haut de gamme
1	18	M	563	4	En Couple	False	1	Luxe
2	82	M	417	0	Célibataire	False	3	Economique
3	72	M	442	4	En Couple	False	1	Luxe
4	41	M	592	0	Célibataire	False	4	Familiale

4. Analyse exploratoire

4.1 Visualisation des données



Les tableaux ci-dessous résume l'ensemble des observations sur les variables catégorielles :

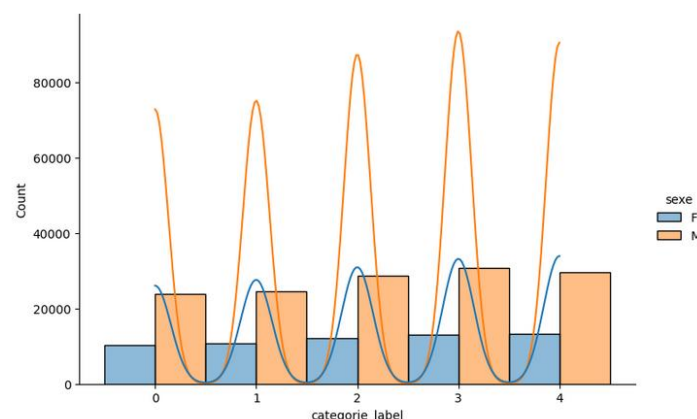
Variable	Domaine de valeurs	Commentaires
nbEnfantsAcharge	0,1,2,3,4,5	Les clients célibataires sans enfants sont plus nombreux. Ceux avec 1 ou 2 enfants sont largement moins nombreux et ceux avec 3 et 4 sont les moins nombreux. Les proportions sont donc inégalement réparties globalement
sexe	F, M	Il y a plus de clients hommes que femmes
situationFamiliale	En couple, Célibataire	Il y a plus de clients en couple que de clients célibataires
2emevoiture	True, False	La majorité des clients sont sans véhicules
categorie_label	0 (Citadine), 1 (Haut de gamme), 2 (Familiale), 3 (Economique), 4 (Moyen)	Les véhicules les plus vendus sont les moyens et les économiques. Les moins vendus sont les citadines et les hauts de gamme. Les familiales viennent juste après les économiques
age	Entre 18 et 84	Il y a des clients de différents âges. Plus l'âge avance moins il y a de clients de cet âge. Il y a globalement 3 tendances au niveau des âges : 18-30, 31-60, 61-84

4.2 Relation entre les variables

Après visualisation des données nous nous sommes intéressés aux relations entre les variables. Nous présentons quelques-unes :

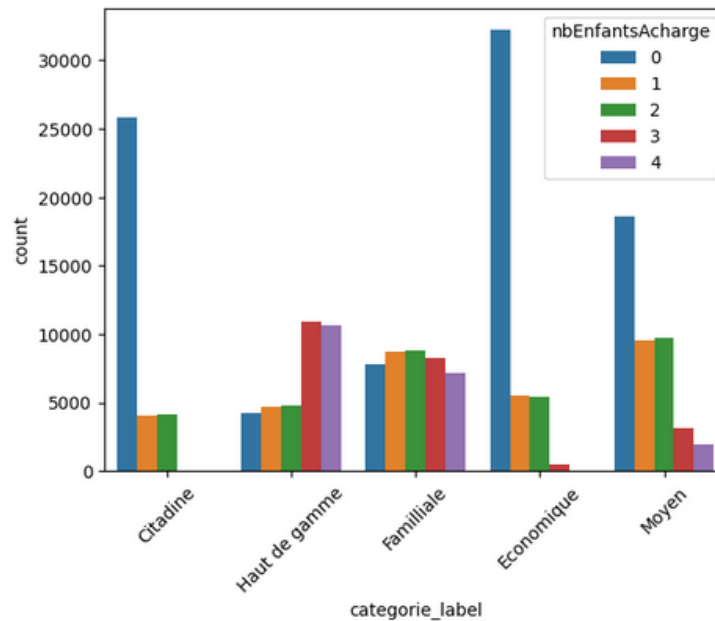
- Relation entre la catégorie de véhicules et le sexe
- Relation entre la catégorie de véhicules et le nombre d'enfants à charge
- Relation entre la catégorie de véhicules et la deuxième voiture
- Relation entre la catégorie de véhicules, âge et taux
- Corrélation entre toutes les variables

• Relation entre la catégorie de véhicules et le sexe



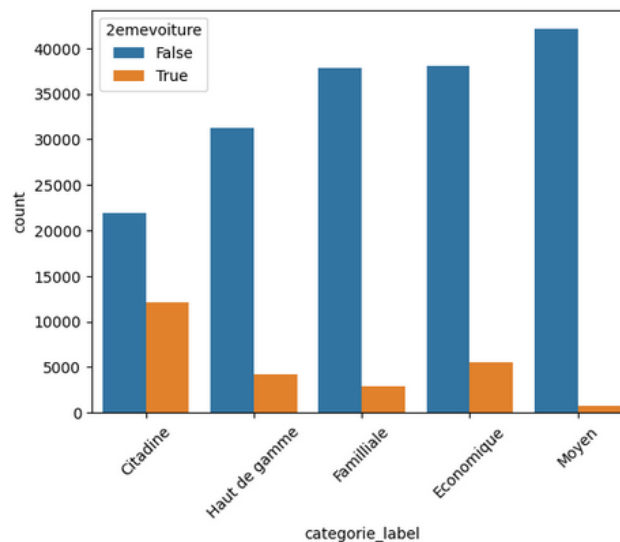
Les hommes sont les plus nombreux. Il est donc normal que leur nombre soient plus important sur la figure. On remarque en particulier que les distributions des hommes et des femmes ont la allure pour toutes les catégories. On déduit donc que le sexe n'a pas d'influence sur le choix du véhicule.

- **Relation entre la catégorie de véhicules et le nombre d'enfants à charge**



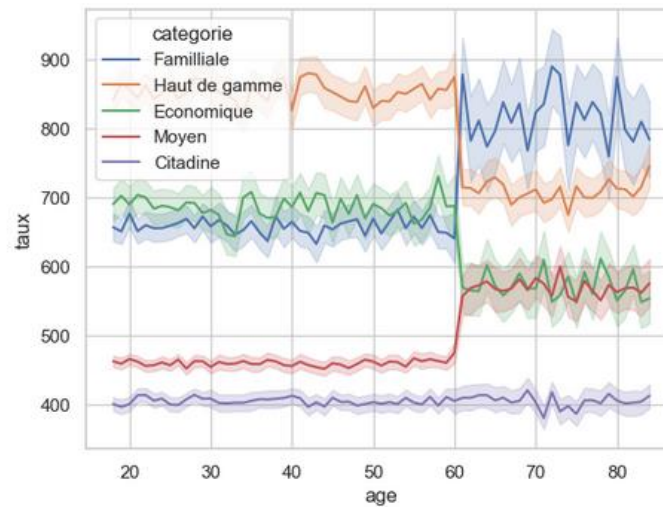
Les clients sans enfants optent plus pour les citadines et les économiques (les moins chères ou les plus équilibrés). Néanmoins, une importante portion opte aussi les familiales. Les clients avec 1 et 2 enfants optent plus pour les moyens et les familiales (Les plus équilibrés). Les clients avec 3 et 4 enfants optent plus pour les hauts de gamme et les familiales (les plus grandes).d

- **Relation entre la catégorie de véhicules et la deuxième voiture**



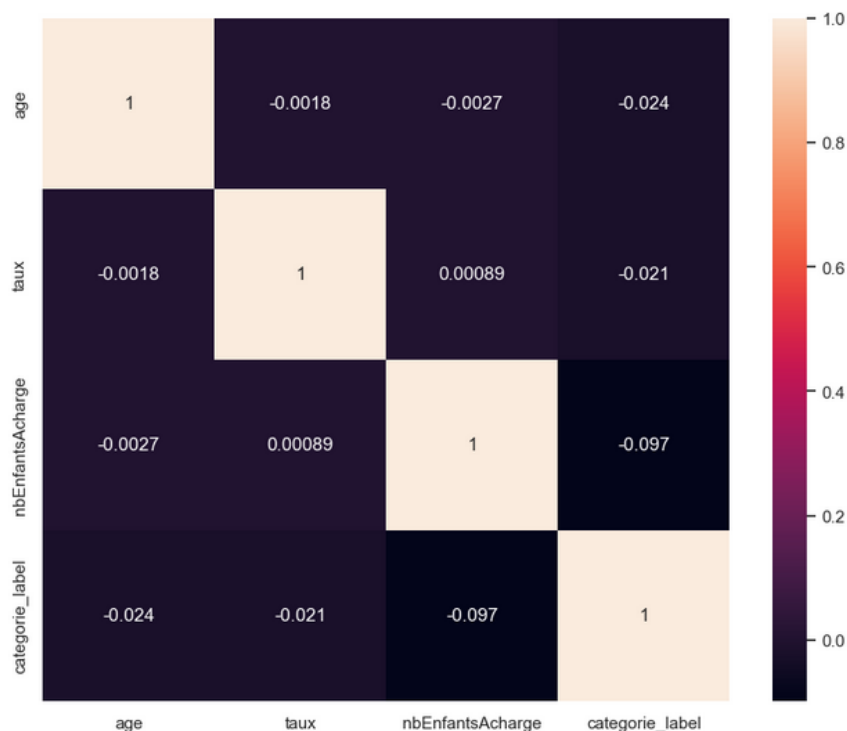
On remarque plus de la moitié des clients ayant acheté une citadine possèdent déjà voiture. Pour les autres catégories, peu de clients en possédaient une, en particulier pour les clients ayant acheté une familiale.

- **Relation entre la catégorie de véhicules, âge et taux**



D'après les figures ci-dessus la variation du taux change en fonction de la tranche d'âge. Il y a deux tranches d'âges dans la figure : 20-59 (jeunes adultes) et 60-80 (personne âgée). Si la tranche d'âge change alors le taux change aussi, or, au taux et à la tranche d'âge correspond une catégorie de véhicules par conséquent le couple taux-âge peut influencer sur la sélection de la catégorie du véhicule d'un client.

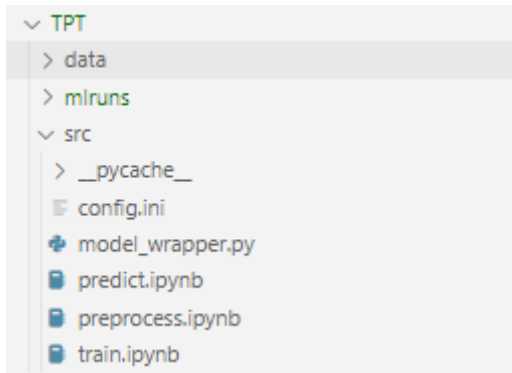
- **Corrélation entre toutes les variables**



5- Modèles de prédiction des catégories et entraînements

5.1 Méthodologie

L'entraînement et l'évaluation des modèles ont été fait en utilisant mlflow avec l'architecture ci-dessous :



- Le fichier config.ini contient les chemins des fichiers et dossiers de l'architecture
- data contient les fichiers des données et mlruns celui des entraînements avec mlflow
- Le fichier preprocess.ipynb contient les codes de préparation de données
- Le fichier train.ipynb contient les codes d'entraînement des modèles effectué avec mlflow : les modèles entraînés sont enregistrés dans le registre des modèles de mlflow
- Le fichier predict.ipynb est utilisé pour tester le meilleur modèle sur les données de test

5.2 Préparation des données

Nous avons divisé nos données en un jeu de test et un jeu d'entraînement avec un ratio de 20% pour les données de test, soit 157283 données pour l'entraînement et 39321 pour les données de test.

```
# Split des données
X = data.drop("categorie_label",axis=1)
y = data["categorie_label"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=RANDOM_STATE)
```

Les variables que nous avons sélectionnées pour le clustering sont les suivantes :

- Variables numériques : taux, nbEnfantsACharges, age
- Variables catégorielles : situationFamiliiale, 2emevoiture,

Le prétraitement consiste à appliquer un simple onehot sur les variables catégorielles dans un pipeline. Les variables numériques n'ont pas subi de transformations.

```
categorical_features = ["situationFamiliiale","2emevoiture"]
numeric_features = ["taux","nbEnfantsACharge","age"]

data_preprocessor = ColumnTransformer(
    transformers=[
        ("onehot", preprocessing.OneHotEncoder(), categorical_features),
        ("passthrough", "passthrough", numeric_features),
    ]
)
```

5.3 Définition des modèles

Nous avons utilisé GridSearch pour l'optimisation des hyperparamètres et l'accuracy comme métrique de scoring :

Modèle	Hyperparamètres	Valeurs testées
LogisticRegression	solver	liblinear, newton-cholesky
	C	0.2, 0.5
	random_state	0, 1
DecisionTreeClassifier	criterion	gini, entropy
	max_depth	4,8
	random_state	0, 1
RandomForestClassifier	n_estimators	4,5
	max_depth	4,8
	random_state	0, 1
Knn	n_neighbors	4,5
	algorithm	auto, ball_tree, kd_tree















5.4 Entraînements des modèles

Ci-dessous un extrait du code de la boucle d'entraînement des modèles avec mlflow :

```
for i, config in enumerate(runs):
    print("Train :",config["run_name"])
    with mlflow.start_run(run_name=config["run_name"],experiment_id=experiment_id) as run:
        model_run(config["model"],config["run_name"],config["param_grid"],i,X_train,X_test, y_train, y_test)
```

5.5 Evaluation des résultats et sélection du meilleur modèle

L'évaluation du modèle, c'est fait à partir de l'interface graphique de mlflow qui se présente comme suit :

Run Name	Created	Duration	Source	Models	best_score	test_accuracy_
 logr	27 days ago		 C:\Users\...	-	-	-
 knn	 27 days ago	1.6min	 C:\Users\...	 vehicule_c.../5	0.751	0.754
 random_forest	 27 days ago	40.7s	 C:\Users\...	 vehicule_c.../8	0.774	0.772
 decision_tree	 27 days ago	5.0s	 C:\Users\...	 vehicule_c.../9	0.774	0.774

On constate alors que le meilleur modèle est l'arbre de décision avec un test_accuracy de **0.774** :

Modèle	Hyperparamètres	Meilleurs valeurs
DecisionTreeClassifier	criterion	entropy
	Max_depth	4
	random_state	0

On sélectionne donc l'arbre de décision comme notre modèle de prédiction.

IV- Test et déploiement du modèle de prédiction

1- Test sur les données de Marketing

Notre modèle de prédiction étant prêt, nous l'avons testé sur les données de Marketing dans le fichier predict.ipynb. Après chargement des données, nous avons lancé une prédiction puis affiché les résultats. Ci-dessous le résultat des prédictions :

	age	sexe	taux	situationFamiliale	nbEnfantsAcharge	2emevoiture	vehicule_categorie_id	vehicule_categorie_label
0	21	F	1396	Célibataire	0	False	3	Economique
1	35	M	223	Célibataire	0	False	0	Citadine
2	48	M	401	Célibataire	0	False	0	Citadine
3	26	F	420	En Couple	3	True	2	Familliale
4	80	M	530	En Couple	3	False	1	Haut de gamme
5	27	F	153	En Couple	2	False	3	Economique
6	59	F	572	En Couple	2	False	4	Moyen
7	43	F	431	Célibataire	0	False	0	Citadine
8	64	M	559	Célibataire	0	False	3	Economique
9	22	M	154	En Couple	1	False	3	Economique
10	79	F	981	En Couple	2	False	2	Familliale
11	55	M	588	Célibataire	0	False	0	Citadine
12	19	F	212	Célibataire	0	False	0	Citadine
13	34	F	1112	En Couple	0	False	2	Familliale
14	60	M	524	En Couple	0	True	0	Citadine
15	22	M	411	En Couple	3	True	2	Familliale
16	58	M	1192	En Couple	0	False	2	Familliale
17	54	F	452	En Couple	3	True	2	Familliale
18	35	M	589	Célibataire	0	False	0	Citadine
19	59	M	748	En Couple	0	True	3	Economique

On remarque la majorité des couples ayant au moins 1 enfant ont soit une familiale, soit une citadine. Les célibataires ont une citadine pour ceux ayant un taux faible et une économique pour ceux ayant un taux élevé. On a aussi un couple avec enfant possédant une moyenne et un autre sans enfants avec un taux très élevés possédant une familiale. Globalement, on constate que les résultats ne présentent pas beaucoup d'incohérences et sont donc acceptables.

2- Déploiement du modèle

Nous avons effectué un déploiement du modèle en local sur notre machine à l'aide d'un conteneur Docker. Le modèle a été déployé sur une api qui est manipulable depuis une interface web, et qui permet à un utilisateur d'entrer les informations d'un client depuis un formulaire et de recevoir en réponse le véhicule qui correspond le mieux au client. Ci-dessous les captures du l'interface web et du Dockerfile :

- **Interface web**

age	sexe	taux	nbEnfantsAcharge	situationFamiliale	Zemevoiture	vehicule_categorie_id	vehicule_categorie_label
23	M	0	0	Célibataire	False	0	Citadine

- **Dockerfile**

```

1 FROM python:3.8-slim
2
3 WORKDIR /app
4 ADD . /app
5
6 # install dependencies
7 RUN pip install -r requirements.txt
8
9 # expose port
10 EXPOSE 5000
11
12 # run application
13 CMD ["gunicorn", "--bind", "0.0.0.0:5000", "app:app"]

```

Conclusion

En conclusion, nous pouvons dire que l'objectif qui était de mettre en place un outil permettant à un concessionnaire automobile de mieux cibler les véhicules susceptibles d'intéresser ses clients a été atteint. Nous avons réussi à mettre en place un outil prêt à l'emploi qui peut être facilement déployé avec Docker. Nous avons d'abord commencé par identifier les catégories de véhicules du catalogue du concessionnaire à l'aide d'un modèle de clustering, Kmeans avec une silhouette de **62%**, puis utilisé ce modèle pour attribuer **5 catégories** aux véhicules vendus, et enfin, nous avons utilisé les données véhicules vendus catégorisé pour entraîner un modèle de recommandation de véhicules basé sur les arbres de décision, utilisable depuis une interface web. Notre modèle n'est pas parfait car sa précision est de **77%**. Comme piste d'amélioration nous pouvons revoir le modèle de clustering, en augmentant ou en diminuant le nombres de clusters, ou encore en essayant de re-entraîner le modèle avec d'autres hyperparamètres pour améliorer sa silhouette.