

PROJET DATA EXPLORATION

Youssef Saidi et Lucas Terra

Dans ce rapport on va détaillé tout ce qu'on a pu trouver et découvrir sur un jeu de données qui regroupe des informations sur des voitures en effectuant toute l'étude vue redécampent en classe.

Préparation des données

Je commence par importer toutes les librairies dont je vais avoir besoin le long de l'étude. C'est des librairies basiques qui sont tout le temps utilisé lors d'une étude de data.

```
In [ ]: %matplotlib widget
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

dataset = pd.read_csv(
    filepath_or_buffer="Jeux/Cars.csv",
    decimal=".",
    sep=";"
)
```

```
In [ ]: print("Sur ce projet on a", dataset.shape[0], "voiture et accompagné", dataset.shape[1], "caractéristique(variables)")

Sur ce projet on a 193 voiture et accompagné 25 caractéristique(variables)
```

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

```
Out[ ]:
```

	continent	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	engine-type	num-of-cylinders	...	peak-rpm	city-mpg	highway-mpg	price	wheel-base	length	\
0	europe	alfa-romero	gas	std	two	convertible	rwd	front	dohc	four	...	5000	21	27	13495	88.6	168.8	
1	europe	alfa-romero	gas	std	two	convertible	rwd	front	dohc	four	...	5000	21	27	16500	88.6	168.8	
2	europe	alfa-romero	gas	std	two	hatchback	rwd	front	ohcv	six	...	5000	19	26	16500	94.5	171.2	
3	europe	audi	gas	std	four	sedan	fwd	front	ohc	four	...	5500	24	30	13950	99.8	176.6	
4	europe	audi	gas	std	four	sedan	4wd	front	ohc	five	...	5500	18	22	17450	99.4	176.6	

5 rows × 25 columns

```
In [ ]: dataset.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193 entries, 0 to 192
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   continent         193 non-null    object  
 1   make              193 non-null    object  
 2   fuel-type          193 non-null    object  
 3   aspiration         193 non-null    object  
 4   num-of-doors       193 non-null    object  
 5   body-style          193 non-null    object  
 6   drive-wheels        193 non-null    object  
 7   engine-location     193 non-null    object  
 8   engine-type         193 non-null    object  
 9   num-of-cylinders    193 non-null    object  
 10  fuel-system         193 non-null    object  
 11  bore               193 non-null    float64 
 12  stroke              193 non-null    float64 
 13  compression-ratio   193 non-null    float64 
 14  horsepower          193 non-null    int64   
 15  peak-rpm             193 non-null    int64   
 16  city-mpg             193 non-null    int64   
 17  highway-mpg          193 non-null    int64   
 18  price               193 non-null    int64   
 19  wheel-base           193 non-null    float64 
 20  length                193 non-null    float64 
 21  width                 193 non-null    float64 
 22  height                193 non-null    float64 
 23  curb-weight           193 non-null    int64   
 24  engine-size           193 non-null    int64   
dtypes: float64(7), int64(7), object(11)
memory usage: 37.8+ KB

```

```
In [ ]: quanti = dataset.select_dtypes(include=["float64", "int64"]).columns
quali = dataset.select_dtypes(include=["object"]).columns
```

Analyse univariée

Maintenant que nos données sont prêtes on va commencer par effectuer une analyse univariée. Pour faire une analyse poussée dans cette partie on a utilisé un outil très puissant de visualisation de données qui est PowerBI vous allez trouver ci-joint des captures de graphique et ce qu'on en a déduit.

Je vais commencer par fournir les informations générales sur chaque variable.

```
In [ ]: dataset[quanti].describe()
```

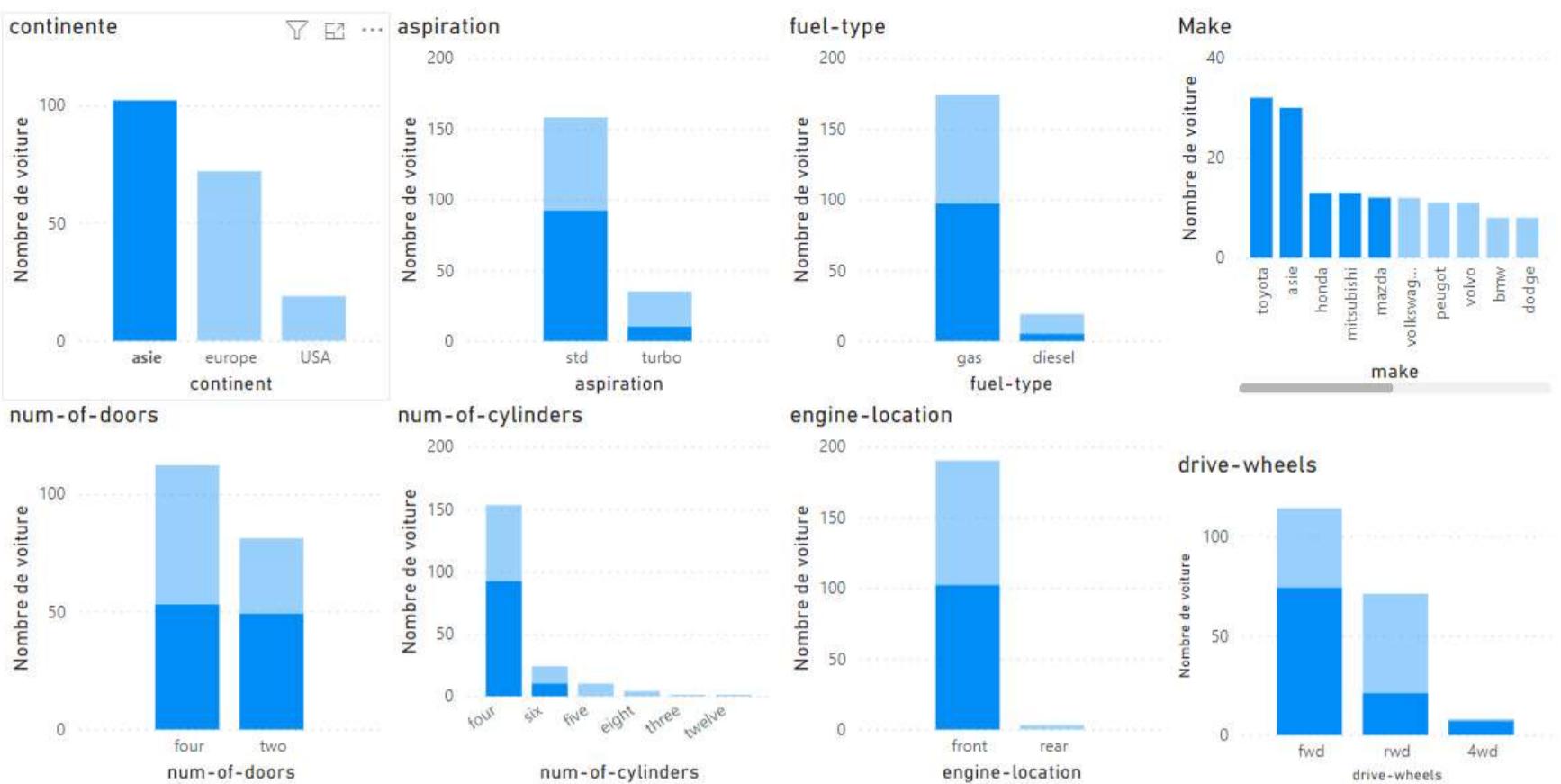
	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	wheel-base	length	width
count	193.000000	193.000000	193.000000	193.000000	193.000000	193.000000	193.000000	193.000000	193.000000	193.000000	193.000000
mean	3.330622	3.248860	10.143627	103.481865	5099.740933	25.326425	30.787565	13285.025907	98.923834	174.326425	65.893782
std	0.272385	0.315421	3.977491	37.960107	468.694369	6.387828	6.816910	8089.082886	6.152409	12.478593	2.137795
min	2.540000	2.070000	7.000000	48.000000	4150.000000	13.000000	16.000000	5118.000000	86.600000	141.100000	60.300000
25%	3.150000	3.110000	8.500000	70.000000	4800.000000	19.000000	25.000000	7738.000000	94.500000	166.300000	64.100000
50%	3.310000	3.290000	9.000000	95.000000	5100.000000	25.000000	30.000000	10245.000000	97.000000	173.200000	65.400000
75%	3.590000	3.410000	9.400000	116.000000	5500.000000	30.000000	34.000000	16515.000000	102.400000	184.600000	66.900000
max	3.940000	4.170000	23.000000	262.000000	6600.000000	49.000000	54.000000	45400.000000	120.900000	208.100000	72.000000

Pour les variables quantitatives, le tableau d'avant nous donne une vue d'ensemble sur les véhicules qu'on a. Par exemple on sait que 75% des voitures coutent moins que 16515 dollars et que donc on a surtout affaire à des voitures de gamme moyenne. On jetant un petit coup d'œil aux horsepowers on peut confirmer ça, étant donné, que 75% des voitures ont moins que 116 chevaux et ne sont donc pas très puissantes.

```
In [ ]: dataset[quali].describe()
```

	continent	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	engine-type	num-of-cylinders	fuel-system
count	193	193	193	193	193	193	193	193	193	193	193
unique	3	20	2	2	2	5	3	2	5	6	7
top	asie	toyota	gas	std	four	sedan	fwd	front	ohc	four	mpfi
freq	102	32	174	158	112	92	114	190	141	153	88

Ensuite essayons de voir quelque graphique de distribution intéressant.



On remarque à partir de ces graphes qu'on a beaucoup de voitures en Asie, un peu moins en Europe et beaucoup moins aux USA (que 19) ce qui n'est pas représentatif et peut donc fausser nos résultats. On remarque aussi qu'une grande partie des voitures sont sous un système std d'aspiration et qu'ils sont à gaz. La marque de voiture la plus vendue est Toyota. On a une grande partie des voitures qui est à 4 portes et à 4 cylindres.

Analyses bivariées

Quantitatif x Quantitatif

Dans cette partie on va essayer d'analyser les relations qui sont entre les variables. Pour ça on va afficher des nuages de points de plusieurs paires de variables quantitatives qui sont intéressantes ainsi que leur droite de régression linéaire et ensuite on va étudier le coefficient de corrélation qui se trouve entre eux.

In []: `dataset[quanti]`

	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	wheel-base	length	width	height	curb-weight	engine-size
0	3.47	2.68	9.0	111	5000	21	27	13495	88.6	168.8	64.1	48.8	2548	130
1	3.47	2.68	9.0	111	5000	21	27	16500	88.6	168.8	64.1	48.8	2548	130
2	2.68	3.47	9.0	154	5000	19	26	16500	94.5	171.2	65.5	52.4	2823	152
3	3.19	3.40	10.0	102	5500	24	30	13950	99.8	176.6	66.2	54.3	2337	109
4	3.19	3.40	8.0	115	5500	18	22	17450	99.4	176.6	66.4	54.3	2824	136
...
188	3.78	3.15	9.5	114	5400	23	28	16845	109.1	188.8	68.9	55.5	2952	141
189	3.78	3.15	8.7	160	5300	19	25	19045	109.1	188.8	68.8	55.5	3049	141
190	3.58	2.87	8.8	134	5500	18	23	21485	109.1	188.8	68.9	55.5	3012	173
191	3.01	3.40	23.0	106	4800	26	27	22470	109.1	188.8	68.9	55.5	3217	145
192	3.78	3.15	9.5	114	5400	19	25	22625	109.1	188.8	68.9	55.5	3062	141

193 rows × 14 columns

Je commence par afficher les différentes nuages de point avec la variable Price, je n'ai traité ici que la variable price parce que sinon j'aurais beaucoup de nuage de points. Et j'ai aussi supprimé plusieurs variables quantitatives qui n'étaient pas importantes.

In []: `data=['price']`

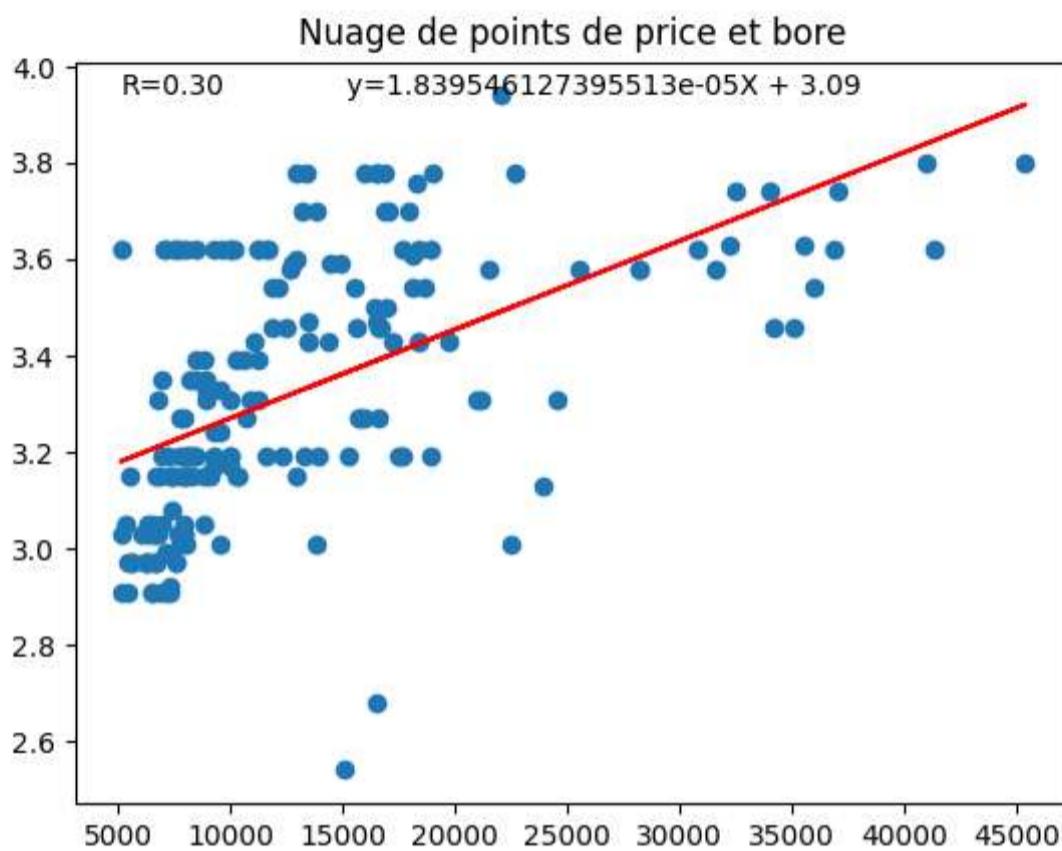
```
for x in data:
    for y in quanti.drop(['stroke', 'compression-ratio', 'peak-rpm', 'height']):
        if x!=y:
            lr = LinearRegression()
            X = dataset[x].values.reshape(-1,1)
            Y = dataset[y].values.reshape(-1,1)
            lr.fit(X, Y)
            yPred= lr.predict(X)
            fig = plt.figure()
            plt.scatter(X,Y)
```

```

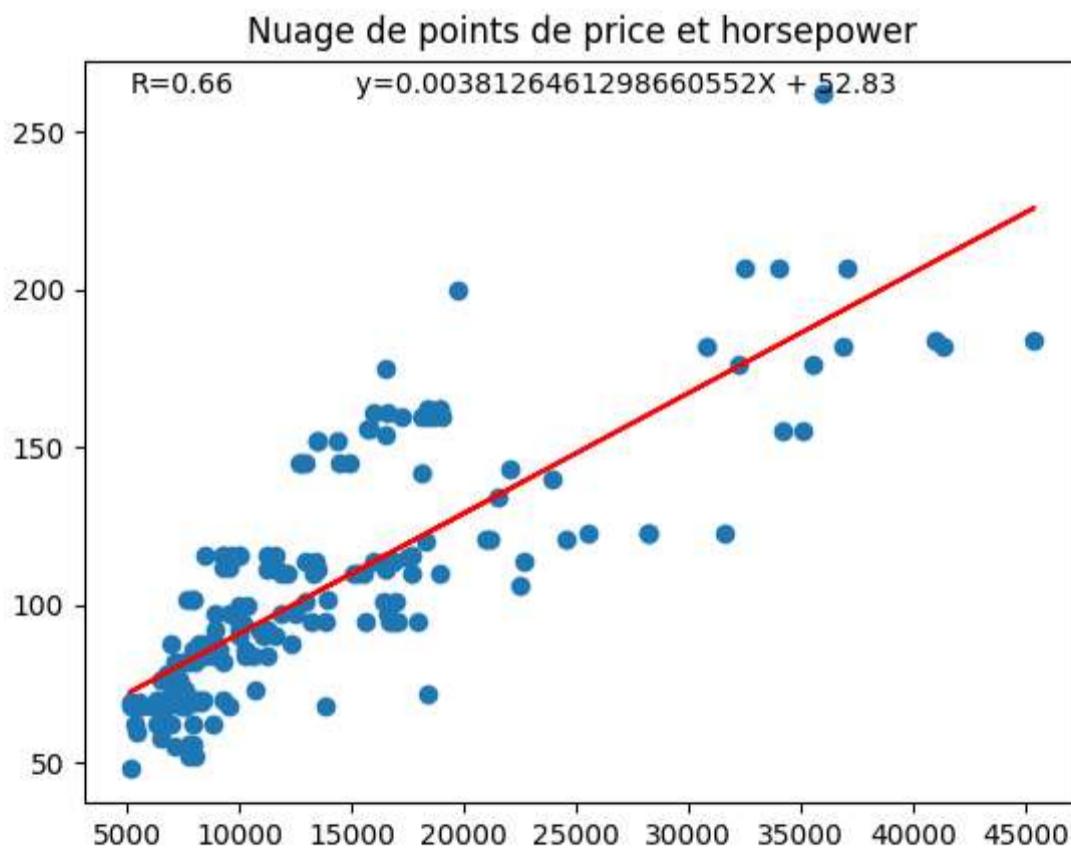
plt.title("Nuage de points de "+x+" et "+y)
plt.plot(X, yPred, color='red')
plt.text(
    x=X.min(),
    y=Y.max(),
    s="R="+format(lr.score(X,Y), ".2f"),
    fontsize=10
)
plt.text(
    x=X.max()/3,
    y=Y.max(),
    s="y=" + str(lr.coef_[0][0]) + "X + " +format(lr.intercept_[0], ".2f"),
    fontsize=10
)
plt.show()

```

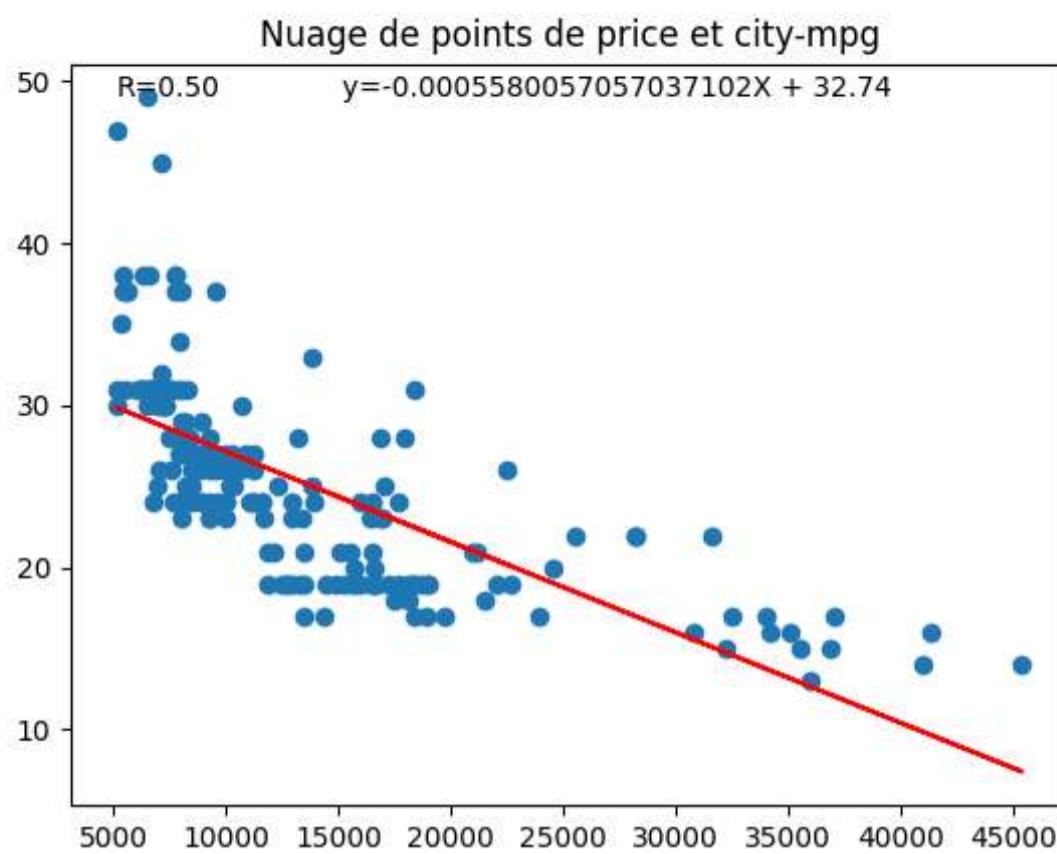
Figure



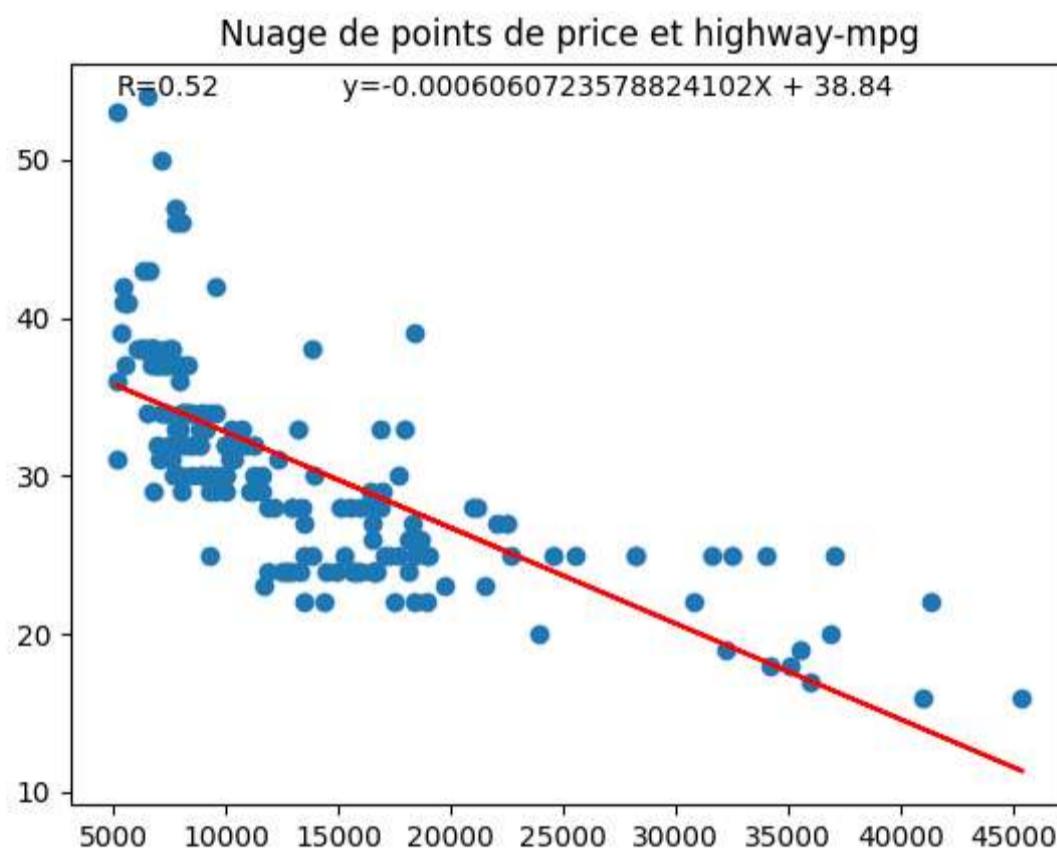
Figure



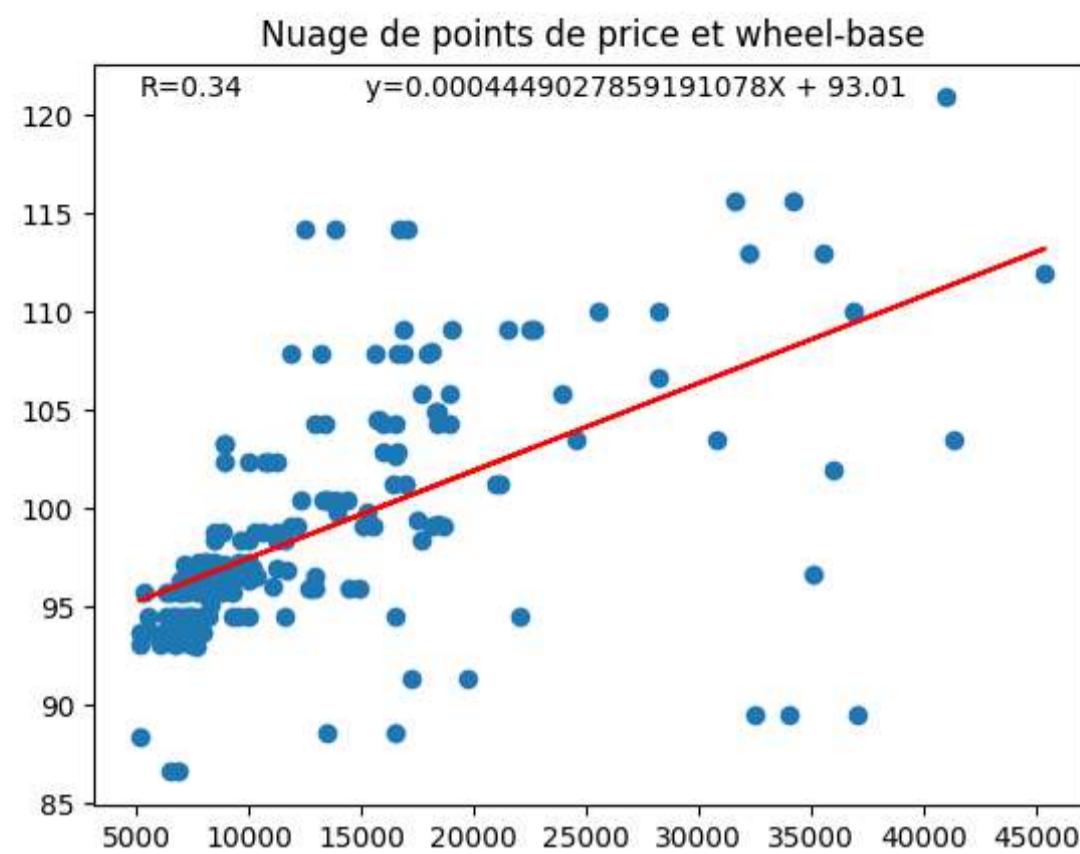
Figure



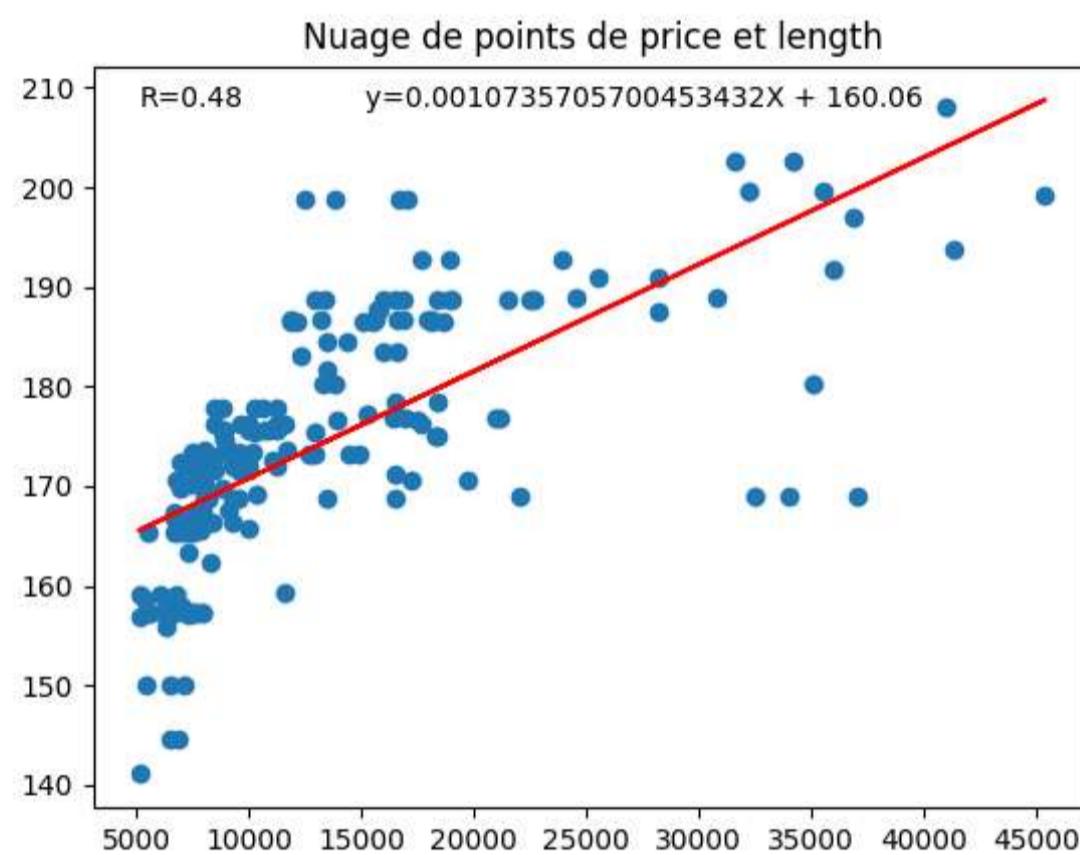
Figure



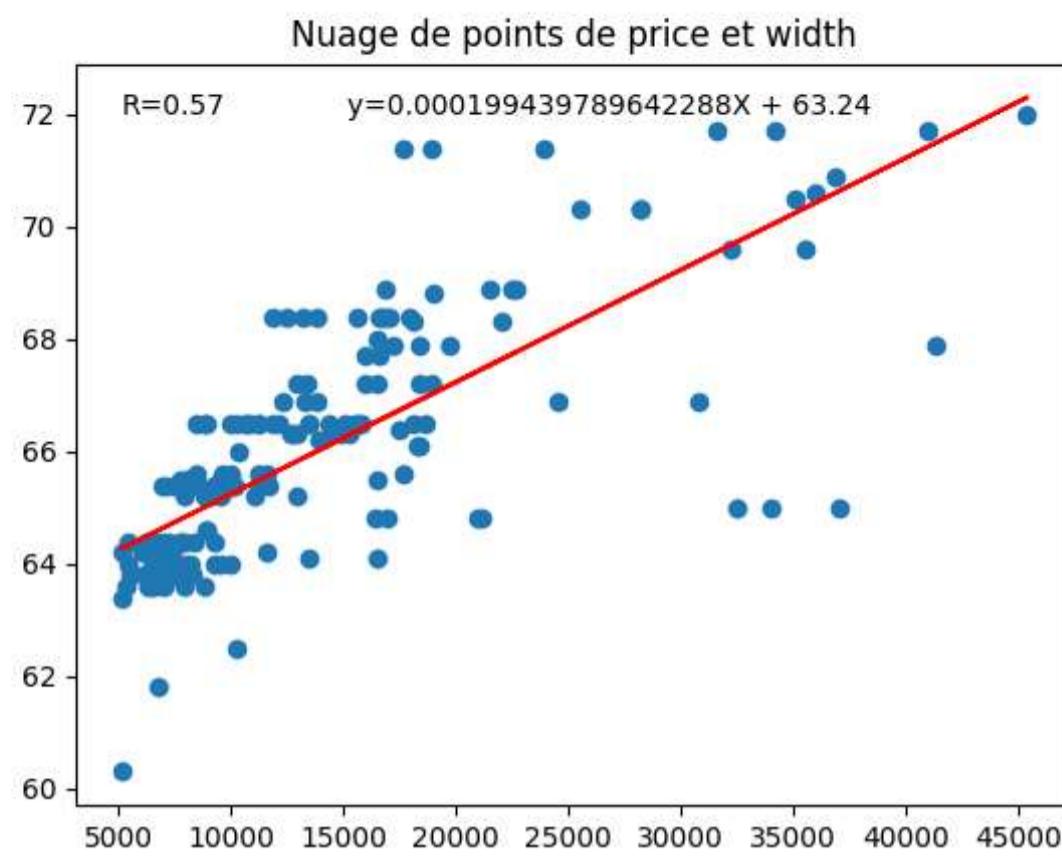
Figure



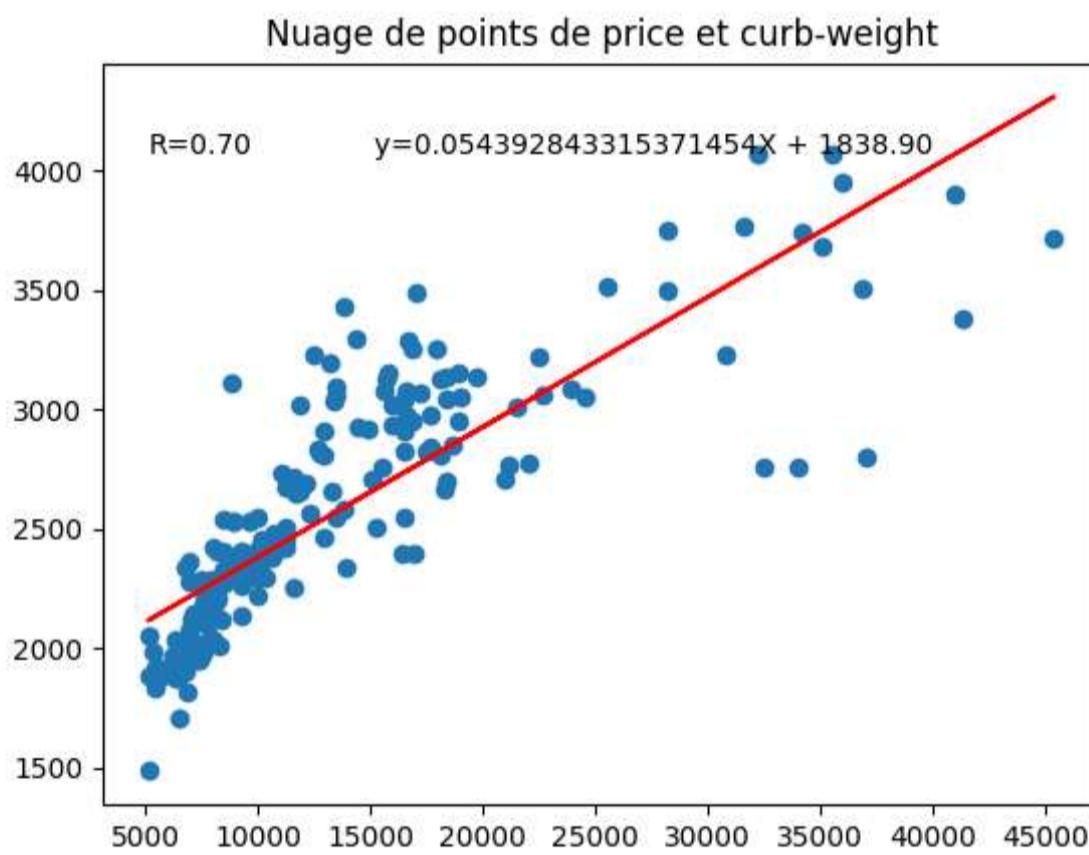
Figure



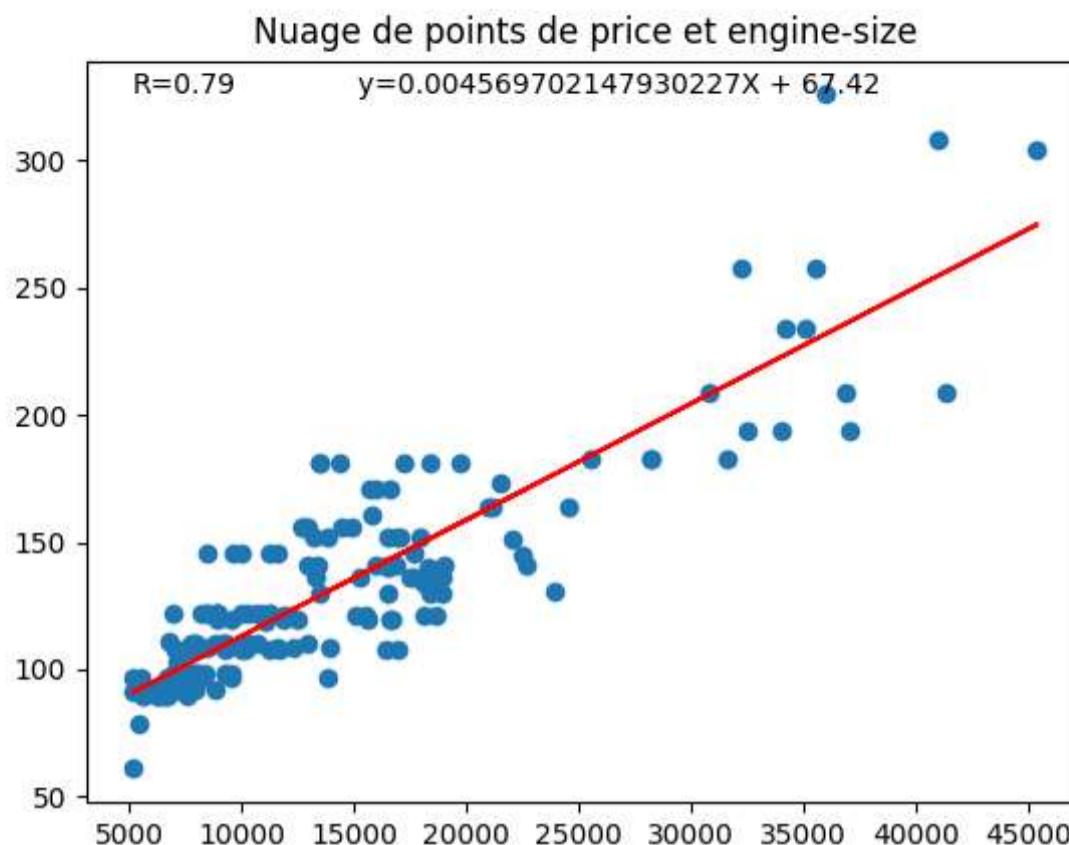
Figure



Figure



Figure



Ensute je dessine un tableau de correlation

```
In [ ]: array = dataset[quanti].values[:,]
cor = dataset.corr(numeric_only=True)

cor.style.background_gradient(cmap='coolwarm')
```

	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	wheel-base	length	width	height
bore	1.000000	-0.065038	-0.004172	0.572972	-0.273766	-0.601369	-0.608804	0.546295	0.495957	0.606373	0.541633	0.182445
stroke	-0.065038	1.000000	0.199600	0.102913	-0.068420	-0.031248	-0.040274	0.096007	0.174225	0.121888	0.188733	-0.054338
compression-ratio	-0.004172	0.199600	1.000000	-0.203818	-0.439741	0.314648	0.249669	0.074483	0.252234	0.156061	0.188631	0.253934
horsepower	0.572972	0.102913	-0.203818	1.000000	0.101383	-0.833615	-0.812078	0.812453	0.377040	0.589650	0.621532	-0.081730
peak-rpm	-0.273766	-0.068420	-0.439741	0.101383	1.000000	-0.061032	-0.008412	-0.103835	-0.350823	-0.276144	-0.247612	-0.257334
city-mpg	-0.601369	-0.031248	0.314648	-0.833615	-0.061032	1.000000	0.971975	-0.706618	-0.504499	-0.702143	-0.657153	-0.111166
highway-mpg	-0.608804	-0.040274	0.249669	-0.812078	-0.008412	0.971975	1.000000	-0.719178	-0.571771	-0.731264	-0.702009	-0.159850
price	0.546295	0.096007	0.074483	0.812453	-0.103835	-0.706618	-0.719178	1.000000	0.584951	0.695928	0.754649	0.136234
wheel-base	0.495957	0.174225	0.252234	0.377040	-0.350823	-0.504499	-0.571771	0.584951	1.000000	0.879307	0.818465	0.591239
length	0.606373	0.121888	0.156061	0.589650	-0.276144	-0.702143	-0.731264	0.695928	0.879307	1.000000	0.857368	0.491050
width	0.541633	0.188733	0.188631	0.621532	-0.247612	-0.657153	-0.702009	0.754649	0.818465	0.857368	1.000000	0.310640
height	0.182445	-0.054338	0.253934	-0.081730	-0.257334	-0.111166	-0.159850	0.136234	0.591239	0.491050	0.310640	1.000000
curb-weight	0.645070	0.175349	0.161030	0.762154	-0.278528	-0.777763	-0.818104	0.835368	0.782173	0.882694	0.867640	0.305837
engine-size	0.581854	0.214518	0.025257	0.845325	-0.217769	-0.716378	-0.737531	0.888778	0.568375	0.686998	0.739903	0.026906

Comme pour les nuages de points, je supprime les variables qui ont que très puissante corrélation avec les autres variables pour avoir quelque chose de plus propre.

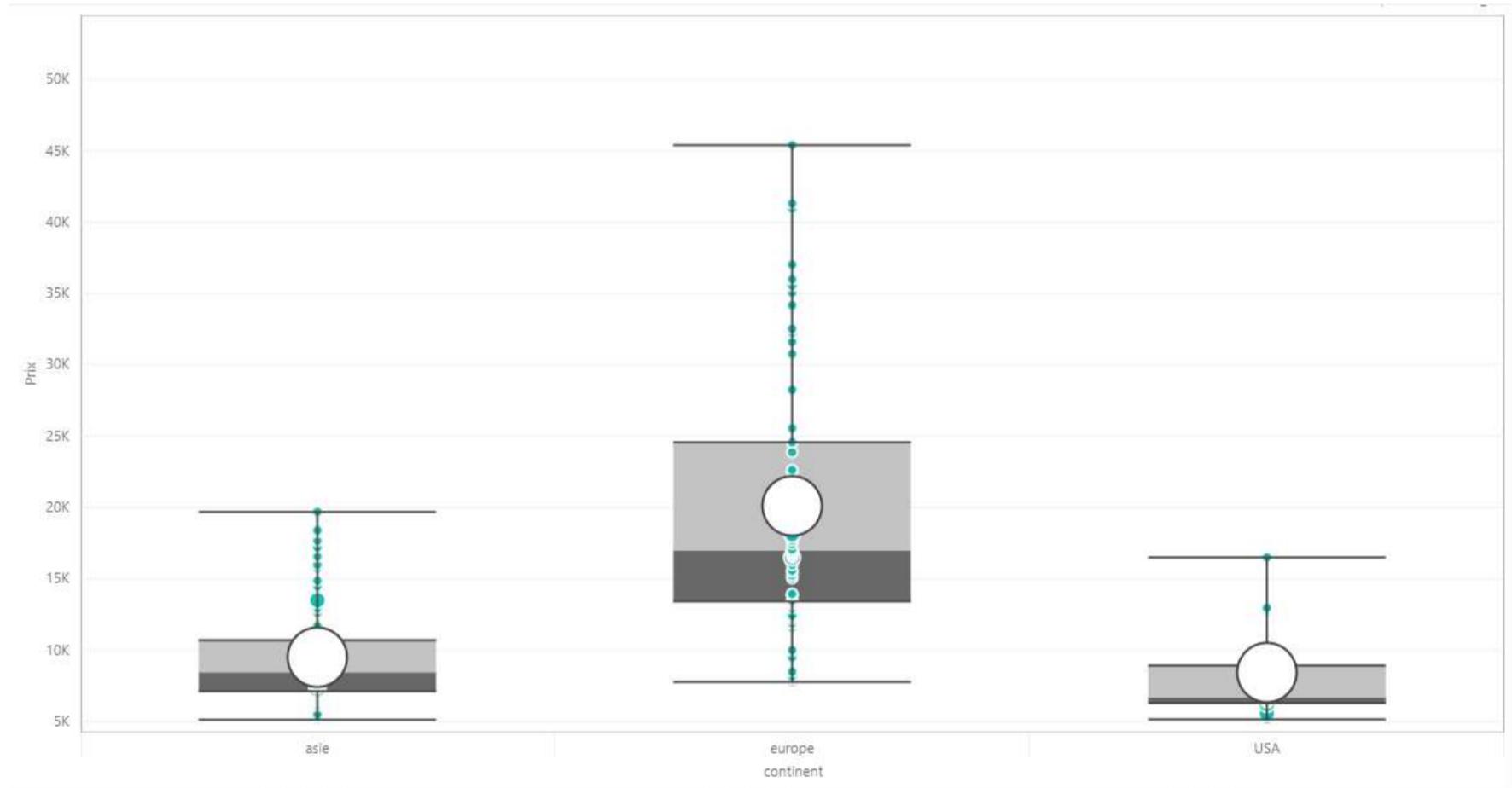
```
In [ ]: dataset2 = dataset.drop(columns=["stroke", "compression-ratio", "peak-rpm"])
cor2 = dataset2.corr(numeric_only=True)
cor2.style.background_gradient(cmap='coolwarm')
```

Out[]:

	bore	horsepower	city-mpg	highway-mpg	price	wheel-base	length	width	height	curb-weight	engine-size
bore	1.000000	0.572972	-0.601369	-0.608804	0.546295	0.495957	0.606373	0.541633	0.182445	0.645070	0.581854
horsepower	0.572972	1.000000	-0.833615	-0.812078	0.812453	0.377040	0.589650	0.621532	-0.081730	0.762154	0.845325
city-mpg	-0.601369	-0.833615	1.000000	0.971975	-0.706618	-0.504499	-0.702143	-0.657153	-0.111166	-0.777763	-0.716378
highway-mpg	-0.608804	-0.812078	0.971975	1.000000	-0.719178	-0.571771	-0.731264	-0.702009	-0.159850	-0.818104	-0.737531
price	0.546295	0.812453	-0.706618	-0.719178	1.000000	0.584951	0.695928	0.754649	0.136234	0.835368	0.888778
wheel-base	0.495957	0.377040	-0.504499	-0.571771	0.584951	1.000000	0.879307	0.818465	0.591239	0.782173	0.568375
length	0.606373	0.589650	-0.702143	-0.731264	0.695928	0.879307	1.000000	0.857368	0.491050	0.882694	0.686998
width	0.541633	0.621532	-0.657153	-0.702009	0.754649	0.818465	0.857368	1.000000	0.310640	0.867640	0.739903
height	0.182445	-0.081730	-0.111166	-0.159850	0.136234	0.591239	0.491050	0.310640	1.000000	0.305837	0.026906
curb-weight	0.645070	0.762154	-0.777763	-0.818104	0.835368	0.782173	0.882694	0.867640	0.305837	1.000000	0.857188
engine-size	0.581854	0.845325	-0.716378	-0.737531	0.888778	0.568375	0.686998	0.739903	0.026906	0.857188	1.000000

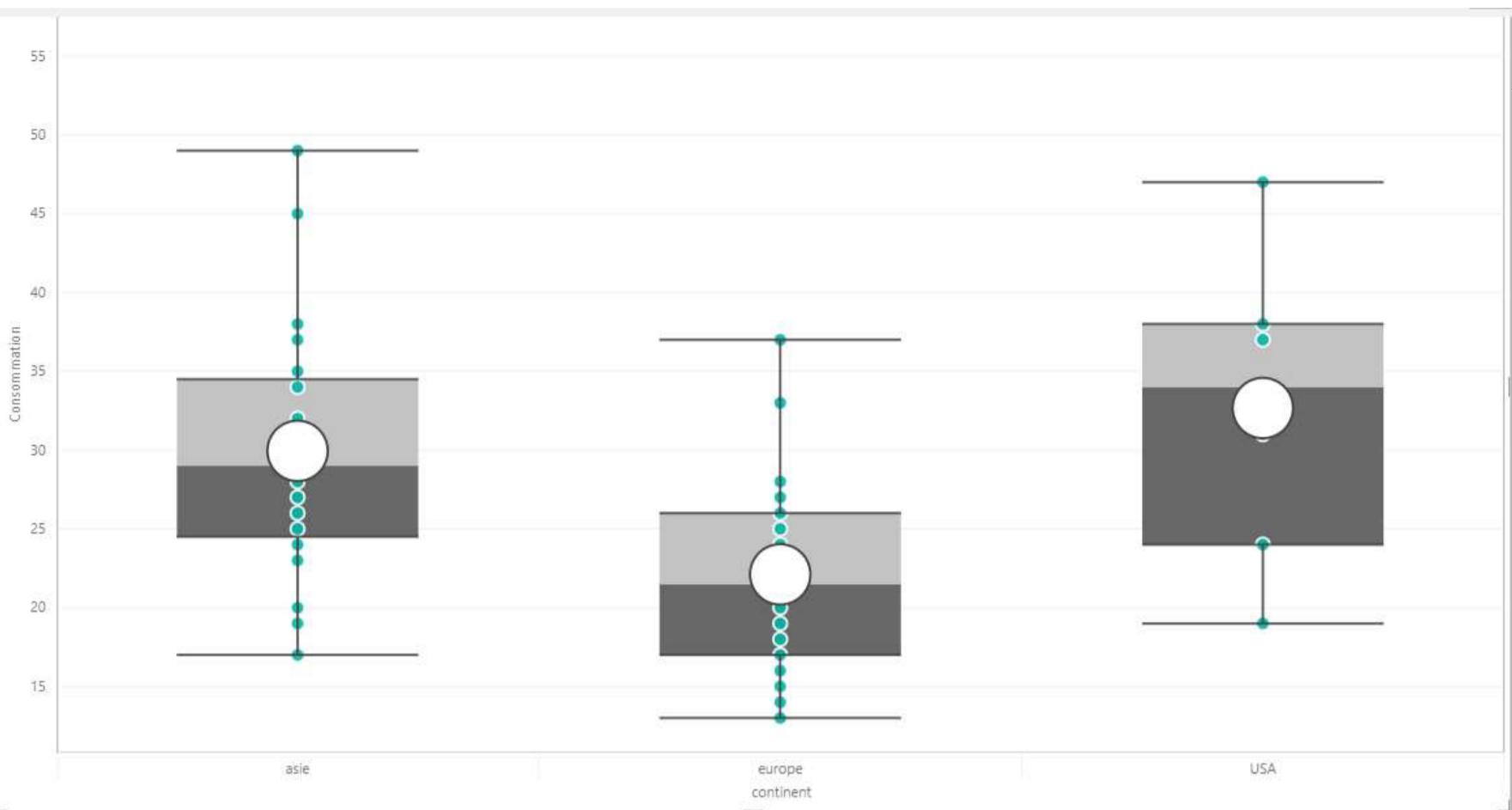
À partir des deux résultats qu'on a, on comprend que l'horsePower depend énormément de la taille de l'engin et donc plus on a de la puissance plus le moteur est grand et plus il est lourd ce qui donc influence le poids de la voiture et donc indirectement sa consommation. Et on a le prix depend énormément de l'horsePower et donc indirectement du poids, de la taille de l'engin et de la consommation et il de pond aussi de la taille de la voiture. On peut donc supposer que les deux paramètres qui influencent le plus le prix de la voiture sont son moteur et sa taille.

Quali x Quanti



Boîte à moustache du Price

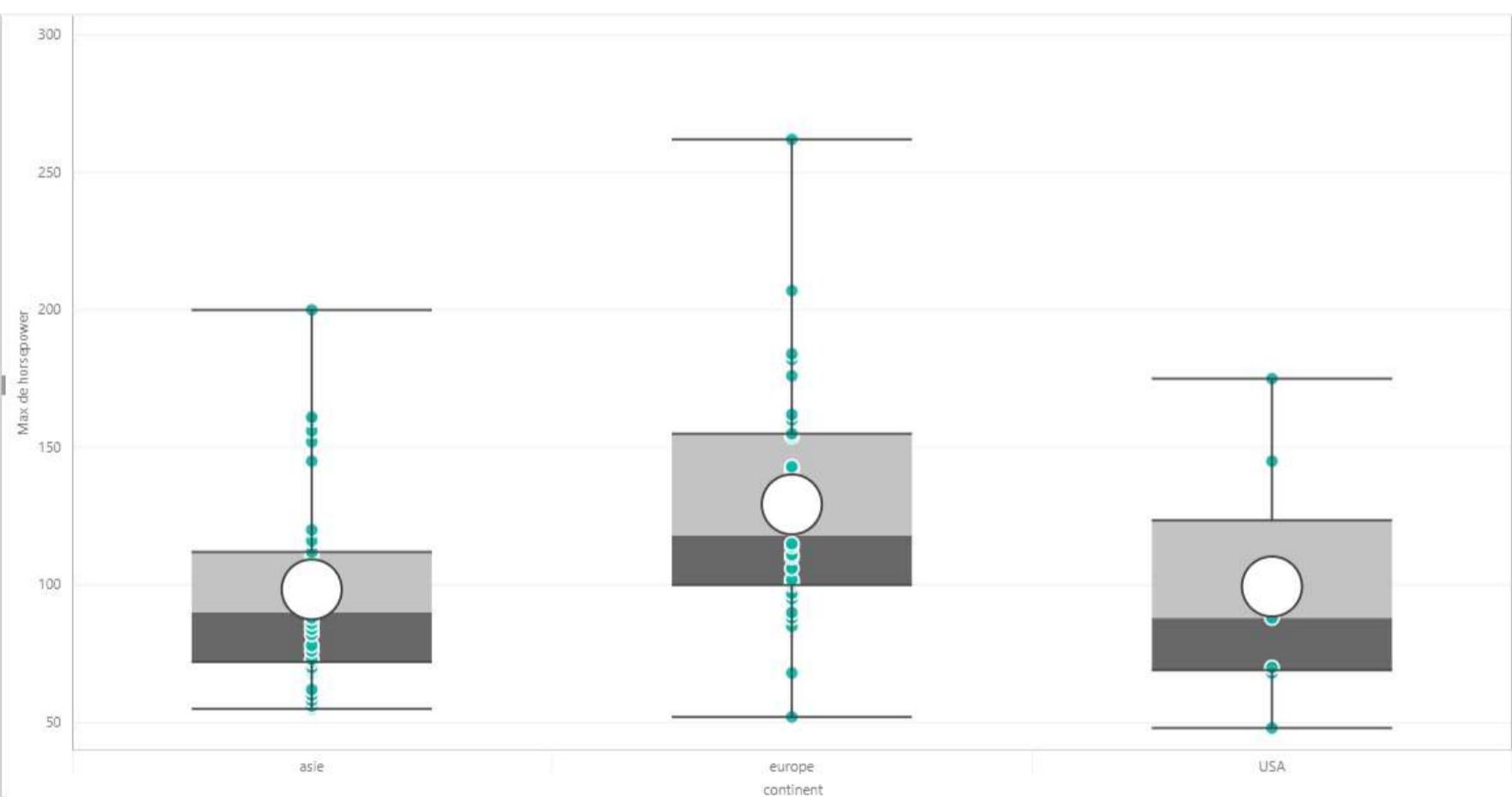
On remarque que la boîte à moustache de l'Europe a une étendue supérieure au double de l'étendue de la boîte à moustache de l'Asie ou des USA. La moyenne des prix en Europe est supérieure au maximum des prix en Asie et aux USA. Et que le prix minimum européen est lui aussi plus élevé qu'en Asie ou aux USA. Ainsi on peut dire que le premier quartile des voitures européennes, a un prix moyen supérieur à celui du troisième quartile des voitures vendues en Asie et aux USA. On suppose à partir des données qu'on a, que les voitures en Europe coutent plus chères. Maintenant on va essayer de comprendre la raison qui pousse à avoir une aussi grande différence de prix. Pour ça on va étudier les performances des véhicules dans les trois pays.



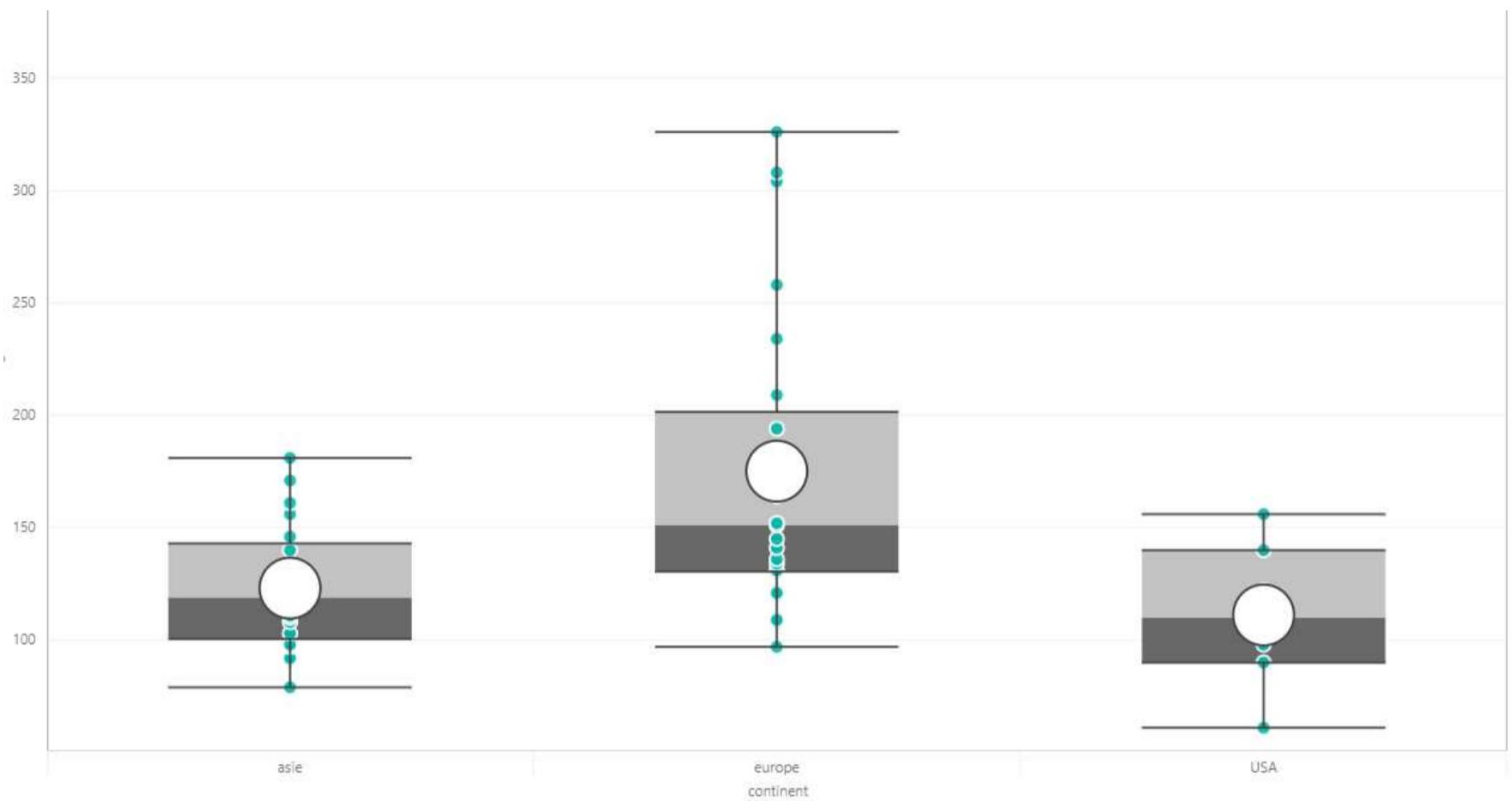
Boîte à moustache de la Consommation

On remarque que le nombre de miles parcourus par gallon est bien plus faible en Europe que dans les autres continents. Une voiture européenne consomme en moyen 1 gallon sur 22 miles. Tandis que les voitures asiatiques et américaines consomment un gallon tous les 29 et 34 miles. On peut donc dire que les voitures en Europe consomment beaucoup plus que ceux aux USA et en Asie. Ainsi on remarque que les voitures en Europe coutent plus chère et consomme plus.

Regardons maintenant grâce aux boîtes à moustaches des variables horsepower et taille d'engin, si les voitures achetées en Europe sont plus chères grâce à leur qualité ou si elles sont justes plus chères sans aucune raison.



Boîte à moustache de horsePower

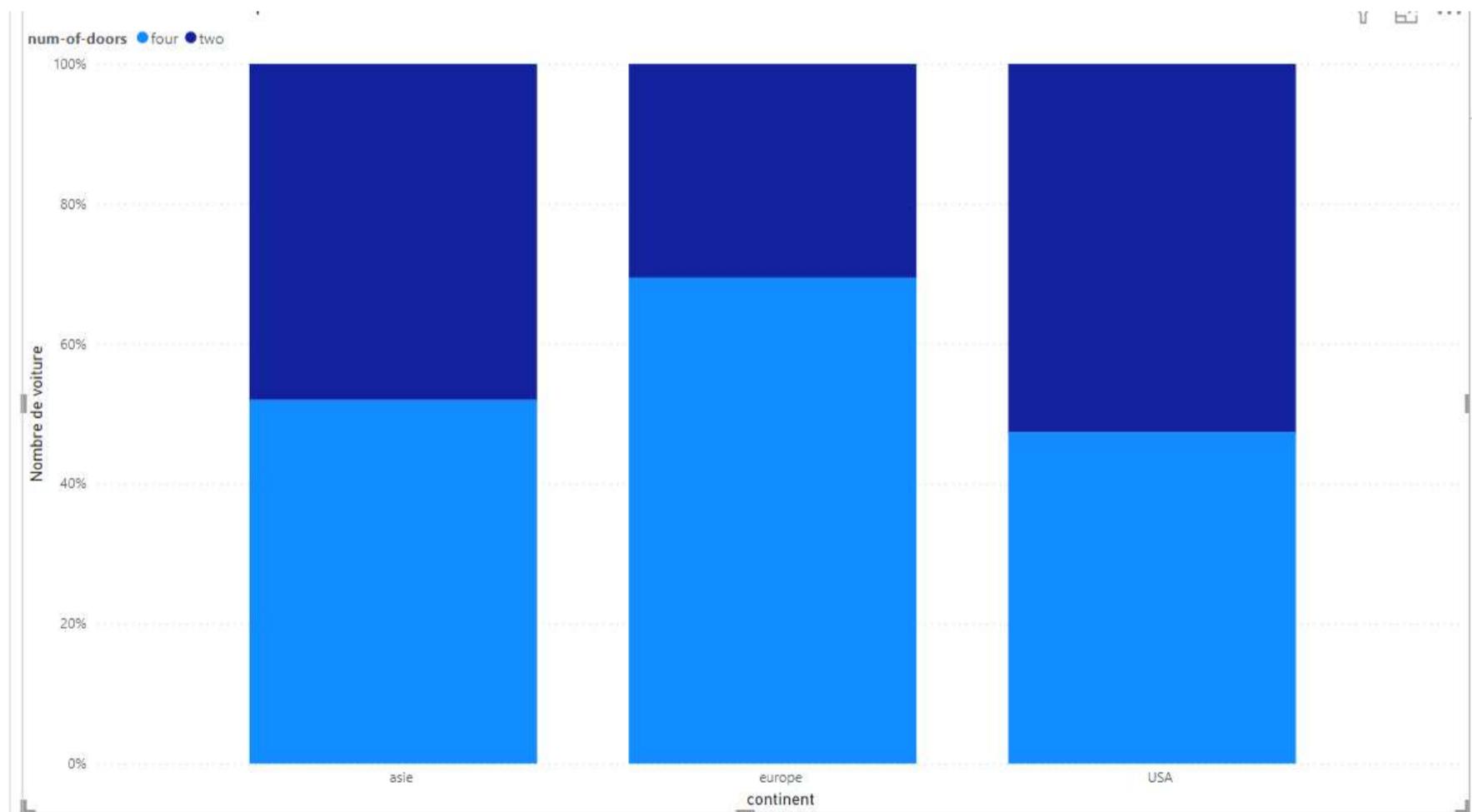


Boîte à moustache de Engine Size

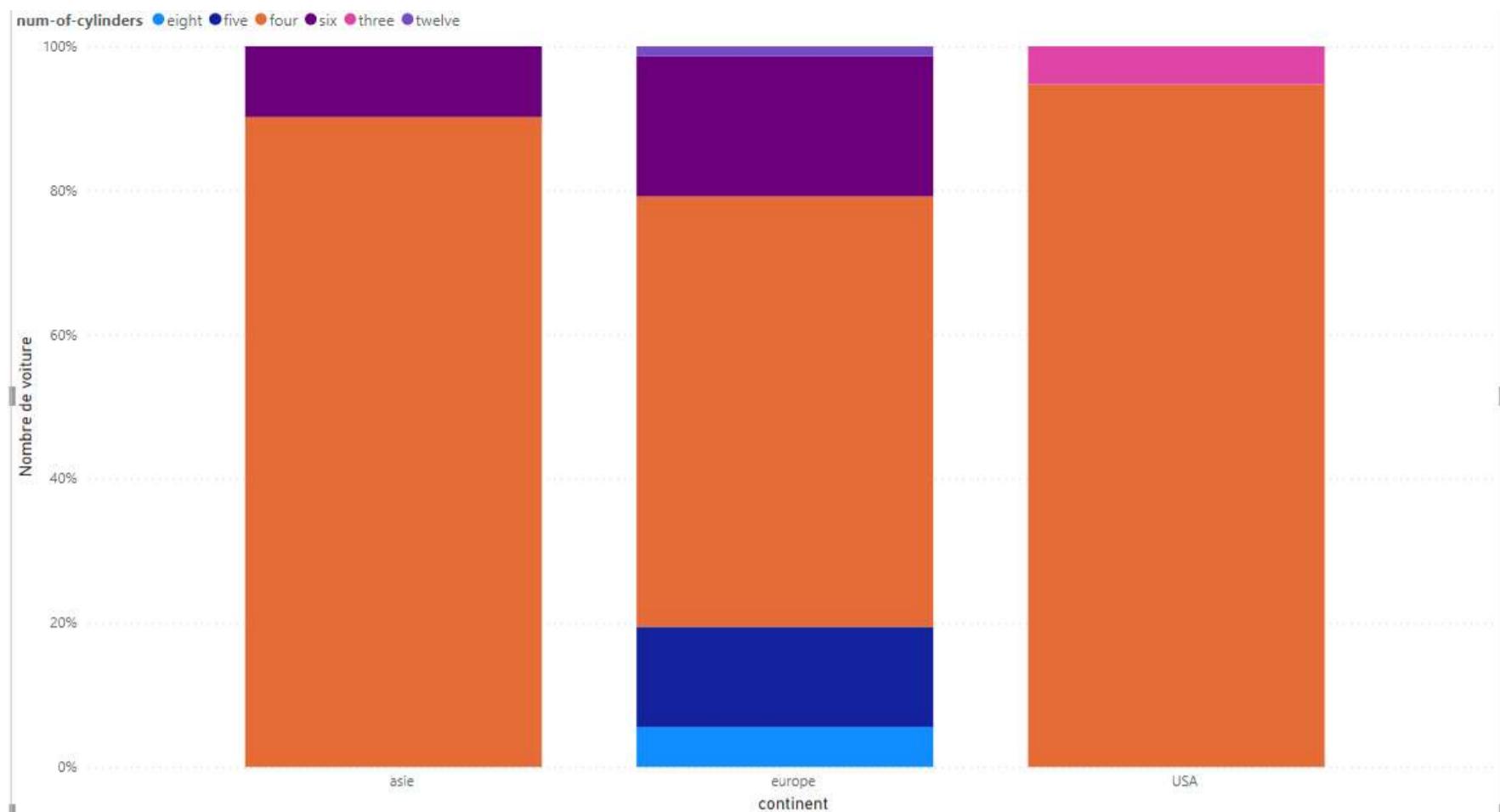
On remarque directement que l'étendue est significativement plus grande en Europe pour l'engine sise avec 229 contre 102 en Asie et 145 aux USA. Le troisième quartile européen est supérieur au maximum asiatique et américain. Maintenant si on considère la taille des engins des voitures en Europe, on remarque que le maximum américain et asiatique est inférieur aux Q3 de l'Europe. On a donc de plus grand moteur en Europe que dans les autres pays. Les mêmes remarques peuvent être faites pour l'horsepower. On a donc les voitures européennes qui sont bien plus performante que dans les 2 autres continents. On peut aussi expliquer la consommation plus élevée avec cette boîte à moustache étant donné qu'un engin plus grand implique un engin plus lourd et donc plus de poids et plus de consommation.

Les voitures vendues en Europe sont donc bien plus puissante et plus performante mais cela ce fait au détriment du prix et de la distance parcourue par gallon.

Quali x Quali



On remarque qu'en Asie il y a autant de voiture à deux portes qu'à quatre c'est aussi le cas aux USA mais on ne peut pas généraliser étant donné le très peu d'enregistrement qu'on a. Alors qu'en Europe les gens ont plus tendance à acheter des voitures à quatre portes qu'à deux portes. On peut donc supposer qu'en Europe les voitures à deux portes ne sont pas très attractives.



On remarque que les voitures à 4 cylindres sont les plus courantes mais on a quand même une petite diversité au niveau de l'Europe étant donné qu'on a un pourcentage non négligeable pour les autres cylindrées aussi.

Maintenant essayer de trouver les dépendances entre variables qualitatives qui existent. Pour ça on va calculer le khi2.

On a partout

```
In [ ]: from scipy.stats import chi2_contingency
print("On trouve que les couples de variable qui sont dependant sont:")
for x in quali:
    for y in quali[quali.get_loc(x)]:
        if x!=y:
            data = pd.crosstab(dataset[x],dataset[y])
            stat, p, dof, expected = chi2_contingency(data)
            alpha = 0.02

            if p <= alpha:
                print("." +x+ " "+y+ " ")
```

On trouve que les couples de variable qui sont dependant sont:

- .continent make
- .continent fuel-type
- .continent aspiration
- .continent body-style
- .continent drive-wheels
- .continent engine-type
- .continent num-of-cylinders
- .continent fuel-system
- .make fuel-type
- .make aspiration
- .make num-of-doors
- .make body-style
- .make drive-wheels
- .make engine-location
- .make engine-type
- .make num-of-cylinders
- .make fuel-system
- .fuel-type aspiration
- .fuel-type engine-type
- .fuel-type fuel-system
- .aspiration fuel-system
- .num-of-doors body-style
- .num-of-doors fuel-system
- .body-style drive-wheels
- .body-style engine-location
- .body-style fuel-system
- .drive-wheels engine-type
- .drive-wheels num-of-cylinders
- .drive-wheels fuel-system
- .engine-location engine-type
- .engine-location num-of-cylinders
- .engine-type num-of-cylinders
- .engine-type fuel-system
- .num-of-cylinders fuel-system

Analyse multivariée

Pour effectuer une analyse multivariée on doit effectuer un ACP. Pour cela on commence par standardisé toutes les valeurs.

```
In [ ]: values = StandardScaler().fit_transform(array)
values

Out[ ]: array([[ 0.51302731, -1.80818563, -0.28827253, ... , -2.12259761,
   -0.02571274,  0.04521533],
   [ 0.51302731, -1.80818563, -0.28827253, ... , -2.12259761,
   -0.02571274,  0.04521533],
   [-2.39482676,  0.70291796, -0.28827253, ... , -0.61541231,
   0.497764 ,  0.57555862],
   ... ,
   [ 0.91791839, -1.20424932, -0.33868626, ... ,  0.6824417 ,
   0.85753529,  1.0817954 ],
   [-1.18015354,  0.48041511,  3.24068891, ... ,  0.6824417 ,
   1.24776341,  0.40681302],
   [ 1.65408397, -0.31423793, -0.16223819, ... ,  0.6824417 ,
   0.95271288,  0.31038697]])
```

Ensuite a crée un modèle ACP et on l'entraîne

```
In [ ]: model_acp=PCA(0.8)

acp = model_acp.fit_transform(values)
acp.shape

Out[ ]: (193, 4)
```

Une fois l'entraînement fini on remarque qu'il nous faut 4 composantes pour ne pas avoir de perte de données(80%)

```
In [ ]: model_acp.n_components_

Out[ ]: 4

In [ ]: model_acp.explained_variance_ratio_

Out[ ]: array([0.53961228, 0.16018715, 0.08704088, 0.06524487])
```

On trouve que pour notre modèle on a 85% d'inertie

```
In [ ]: model_acp.explained_variance_ratio_.cumsum()

Out[ ]: array([0.53961228, 0.69979944, 0.78684031, 0.85208518])

In [ ]: model_acp.explained_variance_

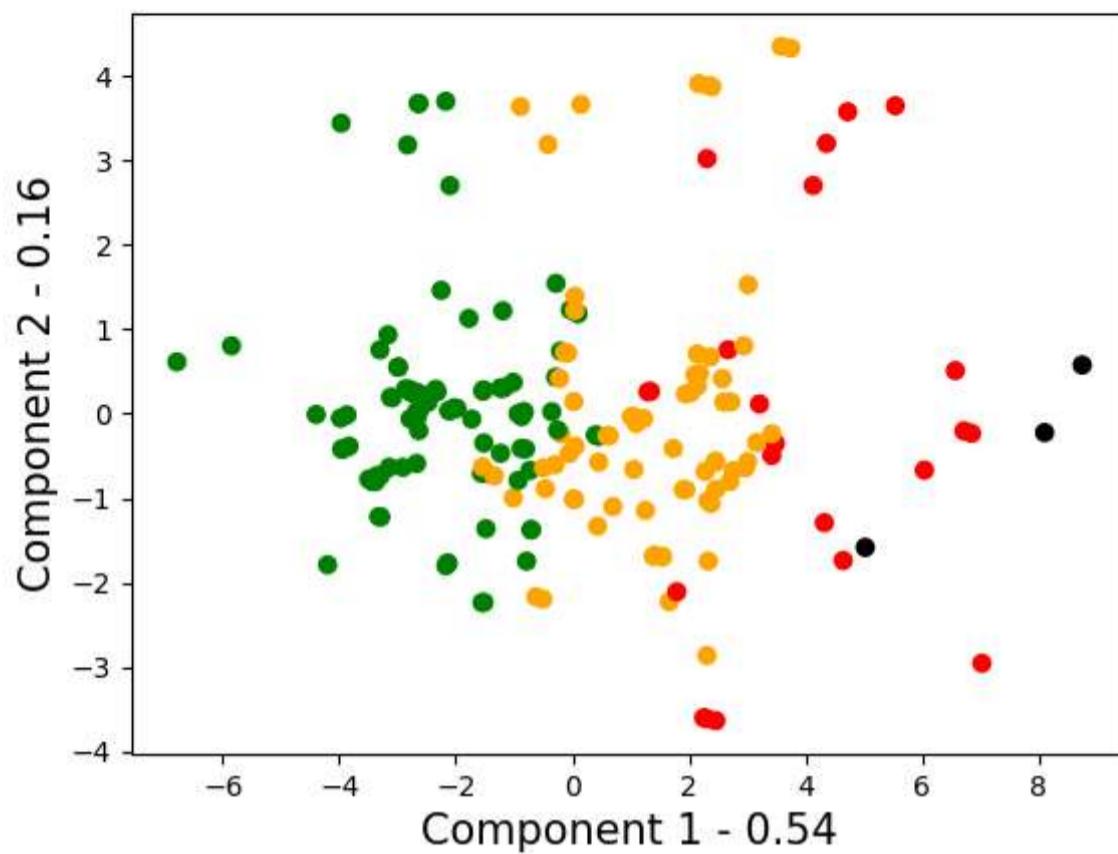
Out[ ]: array([7.5939187 , 2.25430048, 1.22491899, 0.91818558])
```

On dessine ensuite le résultat obtenu sur un plan 2D avec 2 composantes puis sur un plan 3D avec 3 composantes

```
In [ ]: comp1 = acp[:,0]
comp2 = acp[:,1]
comp3 = acp[:,2]

def mapc(p):
    mapping = [(10000, "green"), (20000, "orange"), (40000, "red"), (50000, "black")] # Add all your values and returns here
    c = []
    for i in p:
        for check, value in mapping:
            if int(i) <= check:
                c.append(value)
                break
    return c
fig = plt.figure()
plt.xlabel('Component 1 - '+format(model_acp.explained_variance_ratio_[0], '.2f'), fontsize=15)
plt.ylabel('Component 2 - '+format(model_acp.explained_variance_ratio_[1], '.2f'), fontsize=15)
plt.scatter(comp1,comp2,c=mapc(dataset["price"]))
plt.show()
```

Figure



On remarque que sur le plan 2d on peut délimiter 3 grandes régions tout à gauche les voitures pas chères en vert, aux milieux les voitures moyennes gamme en orange et à droite les voitures chères en rouge et en noir. On va voir qu'on est 3D cette delimitation est encore plus visible et on a l'émergence de 3 vraies régions avec très peu de chevauchement.

```
In [ ]: fig = plt.figure(figsize=(14,9))
ax = fig.add_subplot(111,
                     projection='3d')

ax.scatter(comp1,
           comp2,
           comp3,
           c=mapc(dataset["price"]),
           s=60)

ax.set_xlabel("PC1",
              fontsize=12)
ax.set_ylabel("PC2",
              fontsize=12)
ax.set_zlabel("PC3",
              fontsize=12)

ax.view_init(50, 200)
plt.title("3D PCA plot")
plt.show()
```

Figure

