Projet Car_prices

1. Les valeurs manquantes :

- Remplacer tous les « ? » par NaN
  - Remplacer par la moyenne des colonnes suivantes :
    - "normalized-losses": 41 données manquantes,
    - "stroke" : 4 données manquantes,
    - "bore" : 4 données manquantes,
    - "horsepower" : 2 données manquantes,
    - "peak-rpm" : 2 données manquantes,

    Il est possible de regrouper les données afin de calculer des moyennes plus « justes »

Remarque concernant la colonne « normalized-losses »

Bien entendu, il est possible de calculer les moyennes en regroupant les lignes par ressemblances ou par caractéristiques communes mais ce procédé n'est possible que dans le cas où le dataset est de taille raisonnable.

Exemple les « alfa-romero » ne possèdent pas de valeur « normalized-losses » mais elles ont pour les deux premières un « symboling » = 3, un engine size de 130 est sont toutes les deux de type « convertible »

La troisième a respectivement les valeurs 1, 152 et « hatchback »

Il est donc possible de calculer la moyenne des « normalized-losses » de véhicules ayant les mêmes caractéristiques.

On trouve 128 pour les premières et 123 pour la troisième ce qui finalement n'est pas très éloigné de la moyenne de la colonne : 122

- Remplacer par le mode (valeur la plus fréquente) utiliser *idxmax:*
  - "num-of-door": 2 données manquantes, remplacez-les par "four" (84% des berlines sont quatre portes)

- Supprimer les lignes pour lesquelles la valeur "price" est manquante.

2. Transtypage :

Corriger ensuite tous les types de données erronés. **.dtype()** puis **.astype()** pour modifier le type

- Bore, stroke, peak-rpm et price en float
- Normalized-losses en int

3. Un peu de conversion

Il est parfois utile pour la compréhension des données de convertir les unités utilisées. C'est le cas, ici, pour la consommation qui est mesurée en mpg (miles per gallon).

Convertir les colonnes concernées en L/100 km et les renommer. (L/100 km = 235/mpg)

4. Normaliser ou standardiser

Les colonnes numériques donnant les dimensions des voitures sont dans des échelles et intervalles différents length. Il convient de les réduire à une échelle commune pour pouvoir comparer leur influence sur la colonne price

Visualiser les distributions des colonnes length, width et height (utiliser la fonction *distplot()* de seaborn)

Choisir ensuite la bonne méthode : standartize (distribution normale et $x_{new} = \frac{x - x.mean}{x.std}$) ou normalize pour une distribution quelconque (dans ce cas $x_{new} = \frac{x}{x.max}$ ou $x_{new} = \frac{x - x.min}{x.max - x.min}$

Un peu de lecture sur le sujet : https://medium.com/@rrfd/standardize-or-normalize-examples-in-python-e3f174b65dfc (sujet de veille intéressant)

5. Découpage et regroupement :

Dans le dataset, "horsepower" est une variable réelle évaluée entre 48 et 288, elle a 57 valeurs uniques.

Il peut être intéressant de définir trois classes de voitures : de puissance forte, de puissance moyenne et de faible puissance ?

La réorganisation des données en trois "compartiments" peut simplifier l'analyse.

Utiliser la méthode *cut()* pour segmenter la colonne « horsepower » en 3 catégories.

https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.cut.html

6. Dummies (variable factice binaire)

Les colonnes Aspiration et fuel-type sont cocernées par cette modification.

La colonne "fuel-type" a deux valeurs uniques, "gas" ou "diesel". La régression ne comprend pas les chaines de caractères et se base uniquement sur les valeurs numériques. Dans ce cas les valeurs sont converties en valeurs factices (0 ou 1)
Utiliser la méthode *get_dummies()* de pandas