1 Introduction

Détection des anomalies et nouveautés avec la librairie Scikit-Learn (0.22.1) sous Python.

Ce tutoriel vient en complément du support de cours consacré à la détection des anomalies (''Détection des anomalies'', décembre 2019). Nous nous intéressons à deux aspects du domaine : la détection des observations atypiques ou déviantes dans une base, on parle de ''outlier'' ; par rapport à un jeu de données de référence non-pollué (qui joue le rôle d'ensemble d'apprentissage), l'identification des nouveautés parmi des individus supplémentaires, au sens où leurs caractéristiques s'en écartent significativement, on parle de ''novelty''. Nous utiliserons la librairie ''Scikit-Learn'' pour mener notre étude, avec en particulier les classes de calcul EllipticEnvelope et LocalOutlierFactor.

2 Données

Nous disposons d'un double dataset de véhicules décrits par leur prix, cylindrée, puissance, poids et consommation. Le premier – l'ensemble de référence – est composé de 27 observations (Tableau 1). Notre objectif est d'y déceler les individus atypiques c.-à-d. les véhicules dont les caractéristiques diffèrent sensiblement des autres.

Modele	Prix	Cylindree	Puissance	Poids	Conso
Daihatsu Cuore	11600	846	32	650	5.7
Suzuki Swift 1.0 GLS	12490	993	39	790	5.8
Fiat Panda Mambo L	10450	899	29	730	6.1
VW Polo 1.4 60	17140	1390	44	955	6.5
Opel Corsa 1.2i Eco	14825	1195	33	895	6.8
Subaru Vivio 4WD	13730	658	32	740	6.8
Toyota Corolla	19490	1331	55	1010	7.1
Ferrari 456 GT	285000	5474	325	1690	21.3
Mercedes S 600	183900	5987	300	2250	18.7
Opel Astra 1.6i 16V	25000	1597	74	1080	7.4
Peugeot 306 XS 108	22350	1761	74	1100	9
Seat Ibiza 2.0 GTI	22500	1983	85	1075	9.5
VW Golt 2.0 GTI	31580	1984	85	1155	9.5
Fiat Tempra 1.6 Liberty	22600	1580	65	1080	9.3
Fort Escort 1.4i PT	20300	1390	54	1110	8.6
Honda Civic Joker 1.4	19900	1396	66	1140	7.7
Volvo 850 2.5	39800	2435	106	1370	10.8
Ford Fiesta 1.2 Zetec	19740	1242	55	940	6.6
Hyundai Sonata 3000	38990	2972	107	1400	11.7
Lancia K 3.0 LS	50800	2958	150	1550	11.9
Mazda Hachtback V	36200	2497	122	1330	10.8
Mitsubishi Galant	31990	1998	66	1300	7.6
Opel Omega 2.5i V6	47700	2496	125	1670	11.3
Peugeot 806 2.0	36950	1998	89	1560	10.8
Nissan Primera 2.0	26950	1997	92	1240	9.2
Seat Alhambra 2.0	36400	1984	85	1635	11.6
Volvo 960 Kombi aut	49300	2473	125	1570	12.7

Tableau 1 - Ensemble de données de référence

Les 4 individus supplémentaires sont regroupés dans une seconde base (Tableau 2). Nous souhaitons y détecter des véhicules significativement différents de l'ensemble de référence.

Modele	Prix	Cylindree	Puissance	Poids	Conso
Citroen ZX Volcane	28750	1998	89	1140	8.8
Renault Safrane 2.2. V	36600	2165	101	1500	11.7
Toyota Previa salon	50900	2438	97	1800	12.8
Maserati Ghibli GT	92500	2789	209	1485	14.5

Tableau 2 - Observations supplémentaires

2.1 Vérification de la version de Scikit-Learn

Nous utilisons la version **0.22.1** de scikit-learn dans ce tutoriel.

```
#sklearn version
import sklearn
print(sklearn.__version__)
```

```
0.22.1
```

2.2 Importation et expertise du premier dataset

Nous importons le premier jeu de données composé de 27 observations et nous en affichons les premières lignes.

```
#modifier le dossier de travail
import os
os.chdir("... votre dossier ...")

#importer les données
import pandas
cars = pandas.read_excel("cars_outliers_novelties.xlsx",sheet_name="dataset",index_col=0)

#premières lignes
print(cars.head())
```

```
                      Prix  Cylindree  Puissance  Poids  Conso
Modele
Daihatsu Cuore       11600        846         32    650    5.7
Suzuki Swift 1.0 GLS 12490        993         39    790    5.8
Fiat Panda Mambo L   10450        899         29    730    6.1
VW Polo 1.4 60       17140       1390         44    955    6.5
Opel Corsa 1.2i Eco  14825       1195         33    895    6.8
```

Voici les caractéristiques complètes du dataset.

```
#info sur les données
print(cars.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 27 entries, Daihatsu Cuore to Volvo 960 Kombi aut
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
```

```
 0   Prix        27 non-null     int64
 1   Cylindree   27 non-null     int64
 2   Puissance   27 non-null     int64
 3   Poids       27 non-null     int64
 4   Conso       27 non-null     float64
dtypes: float64(1), int64(4)
```

Puisque nous avons peu de variables, nous affichons les graphiques nuages de points par paires de variables pour disposer d'une vision globale de la conformation des données. Rien de tel pour identifier en un coup d'œil les éventuelles corrélations et autres points singuliers. Nous utilisons la librairie ''seaborn''.

```
#graphique par paire
import seaborn as sns
sns.pairplot(cars)
```

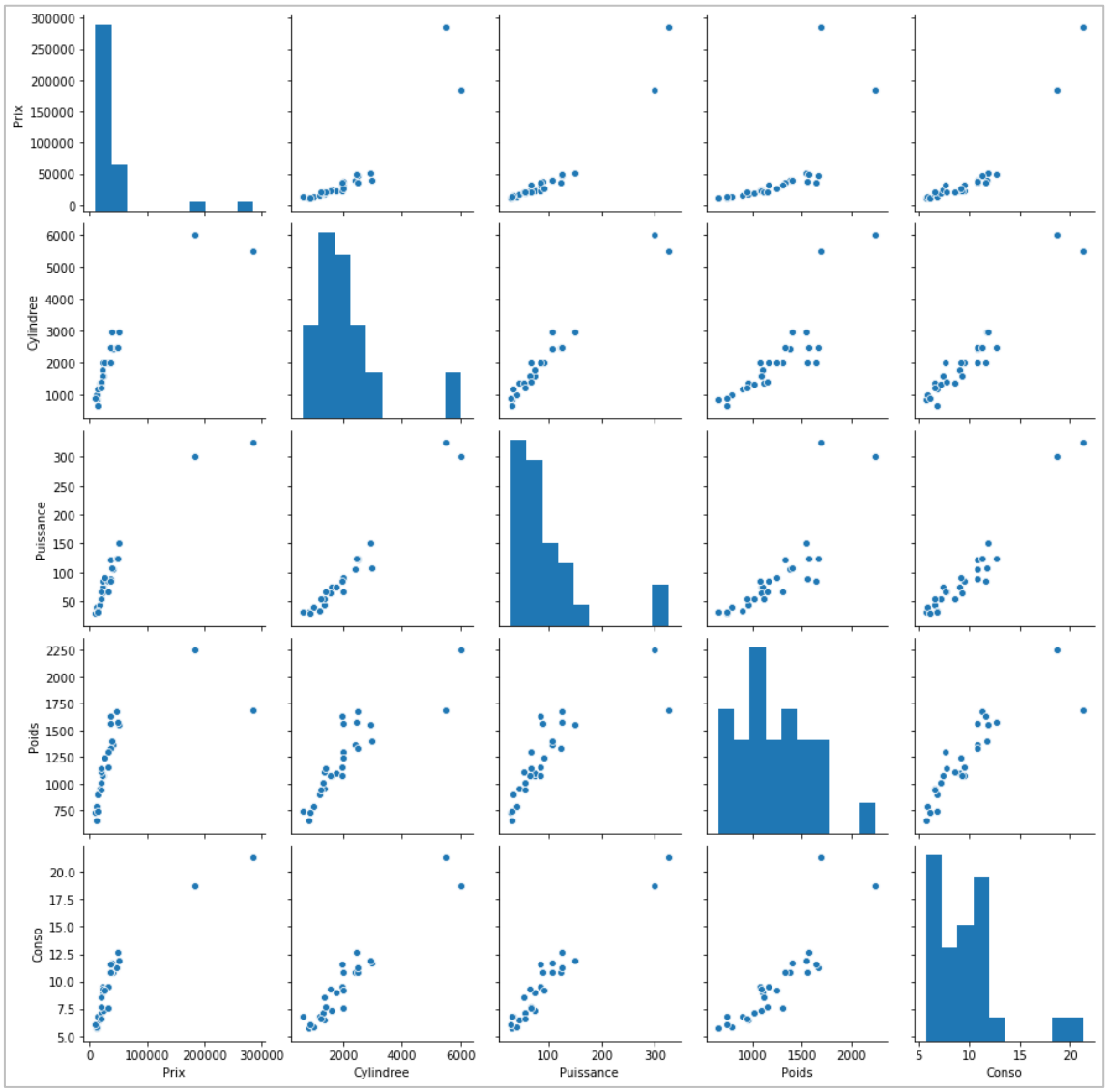

Figure 1 - Graphiques nuages de points par paires de variables

Visiblement (c'est le cas de le dire) : (1) Les variables sont toutes fortement corrélées. (2) Deux véhicules se distinguent par des caractéristiques hors normes sur l'ensemble des descripteurs (valeurs élevées quelles que soit les variables) (Figure 1), la Ferrari et la Mercedes (Tableau 1).

Le premier enjeu consiste à identifier ces deux observations. Ils n'appartiennent manifestement pas à la même population que les autres, voitures plébéiennes. Cela ne devrait pas être trop difficile. Il faut l'espérer en tous les cas. Mais est-ce qu'il y a d'autres observations atypiques ? Des véhicules dont les propriétés diffèrent des autres, mais qui n'apparaissent pas de manière évidente dans les graphiques où l'information est écrasée par la présence des deux véhicules aristocratiques ? Ce second enjeu fort de notre analyse sera autrement moins trivial. Nous verrons comment se comporteront les algorithmes de machine learning dans cette configuration.

3 Identification avec la distance au barycentre

3.1 Barycentre du nuage de points

En faisant l'hypothèse que la distribution des données est unimodale, une manière simple de détecter les observations déviantes est de calculer leurs distances à la référence constituée par le barycentre empirique du nuage de points. Nous obtenons les coordonnées de ce dernier en calculant le vecteur composé des moyennes par variable.

```
#calculer et afficher les moyennes
centroid = cars.mean()
print(centroid)
```

```
Prix         42506.481481
Cylindree     2056.074074
Puissance       93.111111
Poids         1222.777778
Conso            9.659259
```

Le véhicule ''moyen'' présente un prix de 42506, une cylindrée de 2056, etc. Essayons de le situer dans nos graphiques.

```
#créer un jeu de données avec la moyenne
import numpy
tmp = cars.append(centroid,ignore_index=True)
tmp['id'] = numpy.zeros(tmp.shape[0])
tmp.iloc[-1,-1] = 1

#graphique avec matérialisation des barycentres empiriques
sns.pairplot(tmp,hue='id')
```

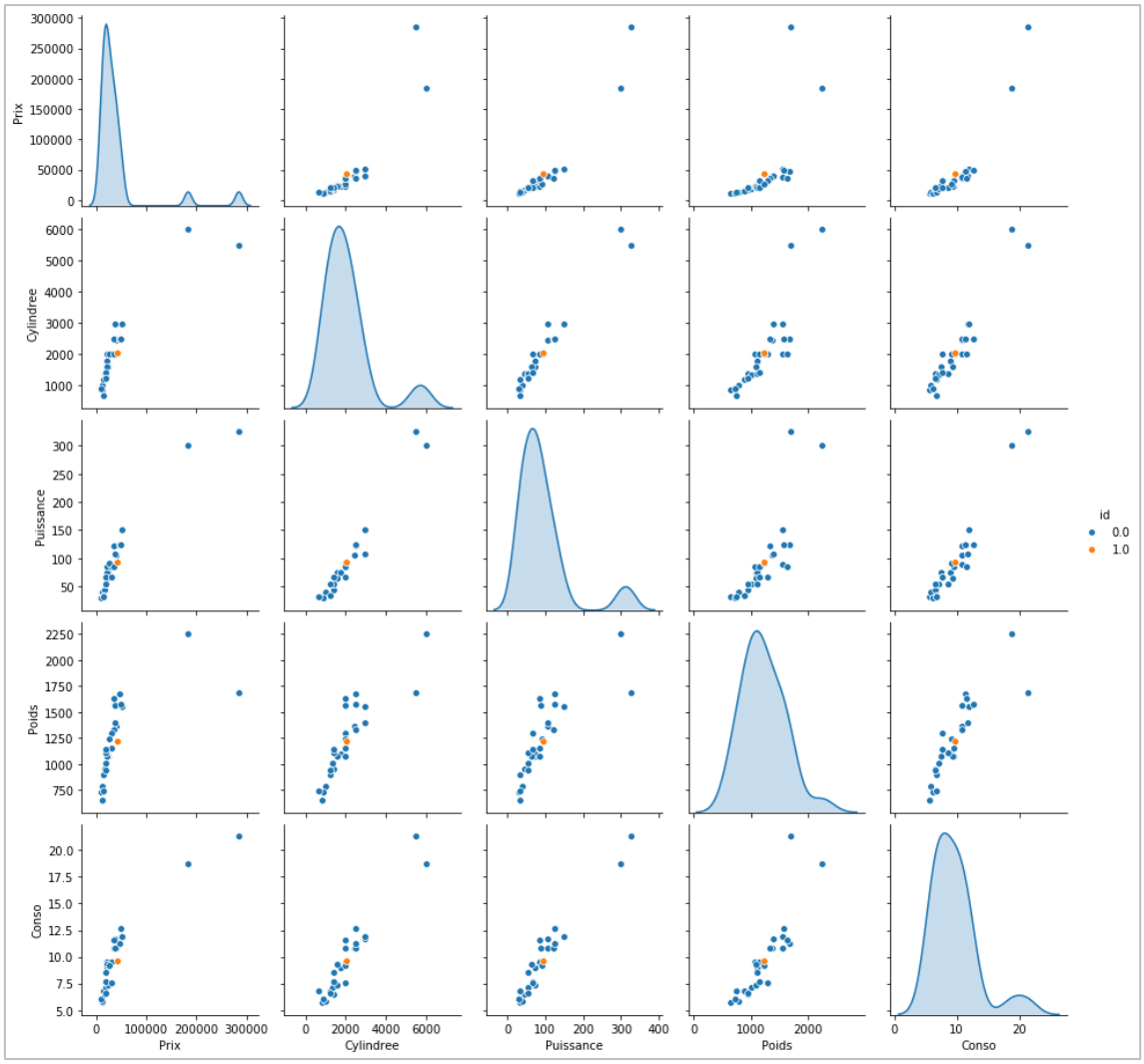

Figure 2 - Nuages de points avec le barycentre empirique (point orange)

La position du barycentre (Figure 2) souligne d'autant plus – si besoin était – le caractère extrême de la Ferrari et la Mercedes par rapport aux autres véhicules.

3.2 Distance de Mahalanobis empirique

Nous utilisons (le carré de) la distance de Mahalanobis pour calculer les écarts au barycentre. Par rapport à la distance euclidienne usuelle, elle possède l'avantage de tenir compte de la forme globale du nuage de points, essentielle dans notre cas compte tenu des fortes corrélations entre les variables. Nous nous appuyons sur la classe EmpiricalCovariance de ''scikit-learn''.

```
#classe pour le calcul de la covariance empirique
from sklearn.covariance import EmpiricalCovariance
empcov = EmpiricalCovariance()
#calcul de la covariance empirique
empcov.fit(cars)
```

```
#calcul de la distance de Mahalanobis (par rapport au barycentre) pour chaque observation
dm_emp = empcov.mahalanobis(cars)

#affichage - trié selon la distance
print(pandas.DataFrame({"Modele":cars.index,"dist":dm_emp}).sort_values(by='dist',ascending=False))
```

```
                     Modele       dist
7             Ferrari 456 GT  23.646937
8             Mercedes S 600  15.305658
18       Hyundai Sonata 3000  12.773687
25         Seat Alhambra 2.0   9.922763
21         Mitsubishi Galant   9.520826
19           Lancia K 3.0 LS   5.983910
22        Opel Omega 2.5i V6   5.388875
23           Peugeot 806 2.0   5.375939
26       Volvo 960 Kombi aut   4.867165
20         Mazda Hachtback V   4.515364
5           Subaru Vivio 4WD   3.816222
11        Seat Ibiza 2.0 GTI   3.795477
0             Daihatsu Cuore   3.261827
4        Opel Corsa 1.2i Eco   3.137841
3             VW Polo 1.4 60   2.976269
9        Opel Astra 1.6i 16V   2.687304
1       Suzuki Swift 1.0 GLS   2.536280
17     Ford Fiesta 1.2 Zetec   2.331469
15     Honda Civic Joker 1.4   2.306010
2         Fiat Panda Mambo L   2.242242
13  Fiat Tempra 1.6 Liberty   2.199231
14       Fort Escort 1.4i PT   1.508248
6             Toyota Corolla   1.126386
24        Nissan Primera 2.0   1.081980
16             Volvo 850 2.5   1.039813
10        Peugeot 306 XS 108   0.925045
12           VW Golt 2.0 GTI   0.727233
```

Comme prévu, les Ferrai et Mercedes sont mis en évidence. Plus surprenant, un trio de véhicules {Hyundai, Seat, Mitsubishi} semble se démarquer également. Que faut-il en penser ?

Réfléchissons un instant à la légitimité de notre démarche avant de se lancer dans des interprétations hasardeuses. Nous souhaitons identifier les observations atypiques à partir des écarts à la moyenne. Or le calcul de cette dernière est lui-même corrompu par leur présence. C'est un cercle vicieux. Pour que les distances soient exploitables dans notre contexte, il faut s'appuyer sur une caractéristique de tendance centrale qui soit une alternative robuste à la moyenne. C'est ce que propose justement la classe EllipticEnvelope.

3.3 Distance robuste – Identification des observations atypiques

Nous instancions la classe EllipticEnvelope en veillant à fixer (random_state = 0) pour que l'expérimentation soit reproductible à l'identique. Nous lançons les calculs en lui passant notre jeu

de données. Première information importante, nous disposons maintenant d'une estimation robuste du barycentre avec la propriété .location_ …

```
#EllipticEnveloppe
from sklearn.covariance import EllipticEnvelope
eev = EllipticEnvelope(random_state=0)
#modélisation
eev.fit(cars)
#affichage des moyennes robustes
print(pandas.DataFrame({'var':cars.columns,'location':eev.location_}))
```

```
         var      location
0       Prix  26989.473684
1  Cylindree   1756.157895
2  Puissance     78.000000
3      Poids   1169.210526
4      Conso      8.878947
```

… assez différente du vecteur des moyennes empiriques (page 4) qui étaient à l'évidence fortement biaisées par au moins les 2 observations manifestement singulières.

La moyenne robuste ou moyenne tronquée consiste à calculer une moyenne arithmétique en éliminant les observations extrêmes. L'objet EllicpticEnvelope fournit la liste des observations conservées pour les calculs avec la propriété .support_

```
#masque de calcul du centroïde robuste
print(eev.support_)
```

```
[ True  True  True False False False  True False False  True  True  True
  True  True  True  True  True  True False  True  True False  True  True
  True  True False]
```

Les véhicules exclus sont donc :

```
#véhicules exclus des calculs
print(cars.loc[eev.support_ == False,:])
```

```
                      Prix  Cylindree  Puissance  Poids  Conso
Modele
VW Polo 1.4 60       17140       1390         44    955    6.5
Opel Corsa 1.2i Eco  14825       1195         33    895    6.8
Subaru Vivio 4WD     13730        658         32    740    6.8
Ferrari 456 GT      285000       5474        325   1690   21.3
Mercedes S 600      183900       5987        300   2250   18.7
Hyundai Sonata 3000  38990       2972        107   1400   11.7
Mitsubishi Galant    31990       1998         66   1300    7.6
Volvo 960 Kombi aut  49300       2473        125   1570   12.7
```

Sur les observations conservées pour les estimations (support_ = True), nous retrouvons le barycentre robuste fourni par .location_ lorsque nous calculons explicitement les moyennes.

```
#barycentre pour les véhicules inclus dans le calcul
print(cars.loc[eev.support_ == True,:].mean())
```

```
Prix          26989.473684
Cylindree      1756.157895
Puissance        78.000000
Poids          1169.210526
Conso             8.878947
```

Affichons dans nos graphiques les barycentres empiriques (orange) et robustes (vert) (Figure 3). Le décalage, l'écart entre les moyennes calculées en atteste, ne semble pas manifeste dans les graphiques parce que ces derniers sont écrasés par les deux points extrêmes situés au nord-est.

```
#affichage des moyennes non-robustes et robustes
tmp = cars.append(centroid,ignore_index=True)
tmp = tmp.append(pandas.Series(index=cars.columns,data=eev.location_),ignore_index=True)
tmp['id'] = numpy.zeros(tmp.shape[0])
tmp.iloc[-2,-1] = 1
tmp.iloc[-1,-1] = 2
#graphique
sns.pairplot(tmp,hue='id',diag_kind='hist')
```

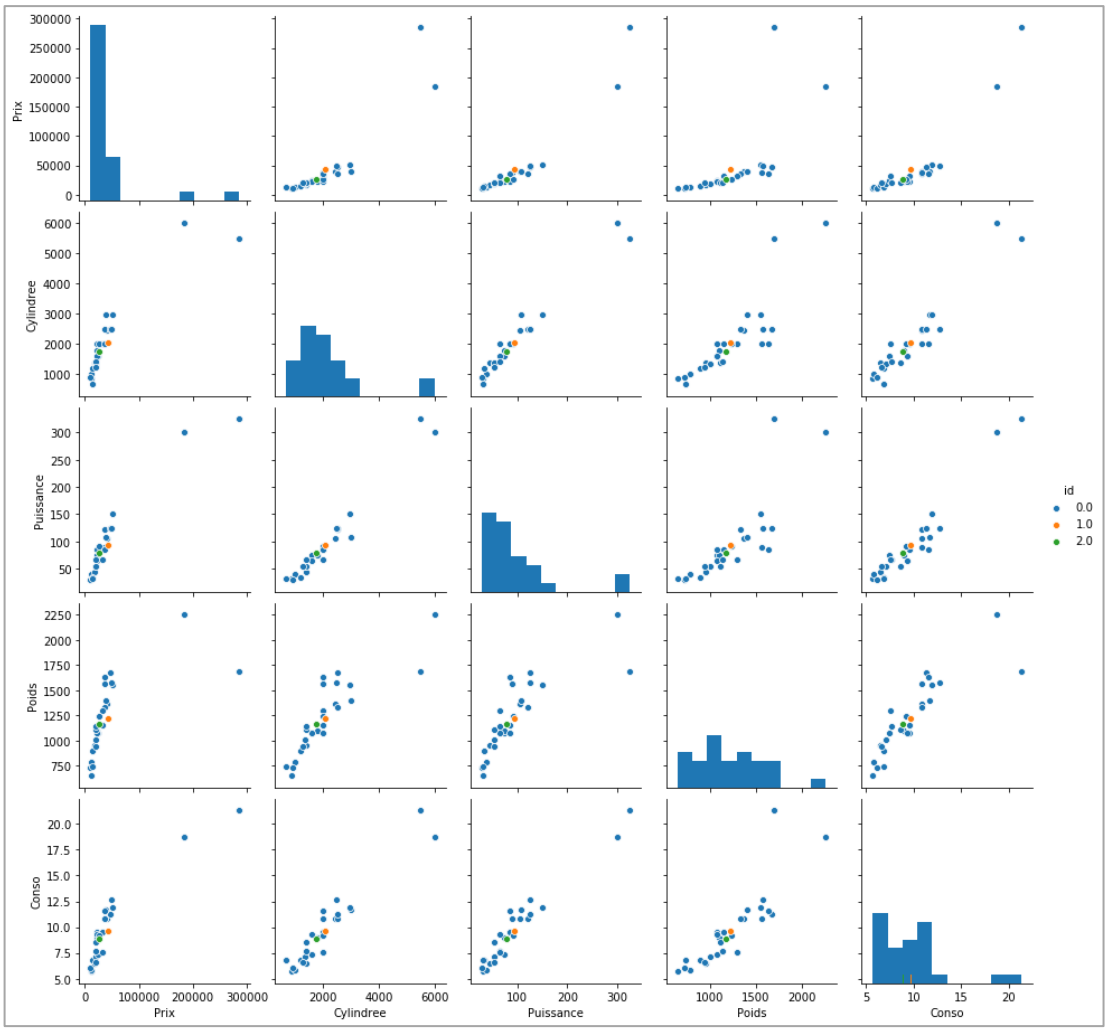

Figure 3 - Nuages de points avec les barycentres empiriques (orange) et robustes (vert)

La matrice de variance-covariance est également modifiée. Avec ces deux éléments (moyennes et covariances), nous disposons maintenant d'une distance de Mahalanobis au barycentre des observations que nous pouvons qualifier de ''robuste''.

```
#distance de Mahalanobis
dm = eev.mahalanobis(cars)

#afficher selon la distance
print(pandas.DataFrame({"Modele":cars.index,"dist":dm}).sort_values(by='dist',ascending=False))
```

```
                     Modele         dist
7            Ferrari 456 GT  8131.477760
8            Mercedes S 600  1465.908300
21        Mitsubishi Galant   166.039027
18      Hyundai Sonata 3000   135.785511
5          Subaru Vivio 4WD    65.484731
3            VW Polo 1.4 60    46.354647
26      Volvo 960 Kombi aut    43.699447
4       Opel Corsa 1.2i Eco    23.822921
16            Volvo 850 2.5     9.138922
19          Lancia K 3.0 LS     8.730551
25        Seat Alhambra 2.0     8.186580
0            Daihatsu Cuore     8.105323
11       Seat Ibiza 2.0 GTI     6.275879
22       Opel Omega 2.5i V6     6.153779
15    Honda Civic Joker 1.4     6.051504
13  Fiat Tempra 1.6 Liberty     5.662621
20        Mazda Hachtback V     5.628987
24       Nissan Primera 2.0     4.291482
23          Peugeot 806 2.0     4.277282
12          VW Golt 2.0 GTI     3.721891
9       Opel Astra 1.6i 16V     3.437647
2        Fiat Panda Mambo L     3.068669
1      Suzuki Swift 1.0 GLS     2.746950
6            Toyota Corolla     2.720209
17    Ford Fiesta 1.2 Zetec     2.557294
10       Peugeot 306 XS 108     2.204966
14      Fort Escort 1.4i PT     2.039465
```

L'outil calcule un seuil (.offset_) calibré de manière à ce que l'on identifie un certain nombre d'observations atypiques fixé par le paramètre ''contamination'' dont la valeur par défaut est 0.1, soit 0.1 x 27 ≈ à peu près 3 observations.

```
#valeur seuil
print(-eev.offset_)
```

```
147.88691769785044
```

Les observations qui présentent une distance supérieure à ce seuil sont considérées comme atypiques (outlier), les autres sont étiquetés ''normaux'' (inlier). La fonction decision_function() affiche la distance ajustée par rapport au seuil.

```
#decision function ==> valeur négative = outlier
df = eev.decision_function(cars)
```

```
#afficher selon la decision function triés de manière croissante
print(pandas.DataFrame({"Modele":cars.index,"decision":df}).sort_values(by='decision'))
```

```
                     Modele     decision
7            Ferrari 456 GT -7983.590842
8            Mercedes S 600 -1318.021383
21        Mitsubishi Galant   -18.152109
18      Hyundai Sonata 3000    12.101406
5          Subaru Vivio 4WD    82.402187
3            VW Polo 1.4 60   101.532270
26      Volvo 960 Kombi aut   104.187471
4       Opel Corsa 1.2i Eco   124.063996
16            Volvo 850 2.5   138.747995
19          Lancia K 3.0 LS   139.156367
25        Seat Alhambra 2.0   139.700338
0            Daihatsu Cuore   139.781595
11       Seat Ibiza 2.0 GTI   141.611039
22       Opel Omega 2.5i V6   141.733139
15    Honda Civic Joker 1.4   141.835414
13  Fiat Tempra 1.6 Liberty   142.224297
20        Mazda Hachtback V   142.257930
24       Nissan Primera 2.0   143.595436
23          Peugeot 806 2.0   143.609635
12          VW Golt 2.0 GTI   144.165026
9       Opel Astra 1.6i 16V   144.449271
2        Fiat Panda Mambo L   144.818248
1      Suzuki Swift 1.0 GLS   145.139968
6            Toyota Corolla   145.166709
17    Ford Fiesta 1.2 Zetec   145.329623
10       Peugeot 306 XS 108   145.681952
14      Fort Escort 1.4i PT   145.847453
```

Nous pouvons également solliciter la fonction predict() pour identifier les observations atypiques.

```
#prediction
statut = eev.predict(cars)
print(statut)
```

```
[ 1  1  1  1  1  1  1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1 -1  1  1
  1  1  1]
```

```
#afficher les véhicules concernés
print(cars.loc[statut == -1])
```

```
                     Prix  Cylindree  Puissance  Poids  Conso
Modele
Ferrari 456 GT     285000       5474        325   1690   21.3
Mercedes S 600     183900       5987        300   2250   18.7
Mitsubishi Galant   31990       1998         66   1300    7.6
```

Voyons la position de ces points dans nos graphiques par paires de variables (Figure 4).

```
#visualisation
cars_with_statut = cars.copy()
cars_with_statut['statut'] = -statut #pour que les points atypiques en rouge soient au premier plan
sns.pairplot(cars_with_statut,hue='statut',palette={-1:'silver',1:'red'},diag_kind='hist')
```

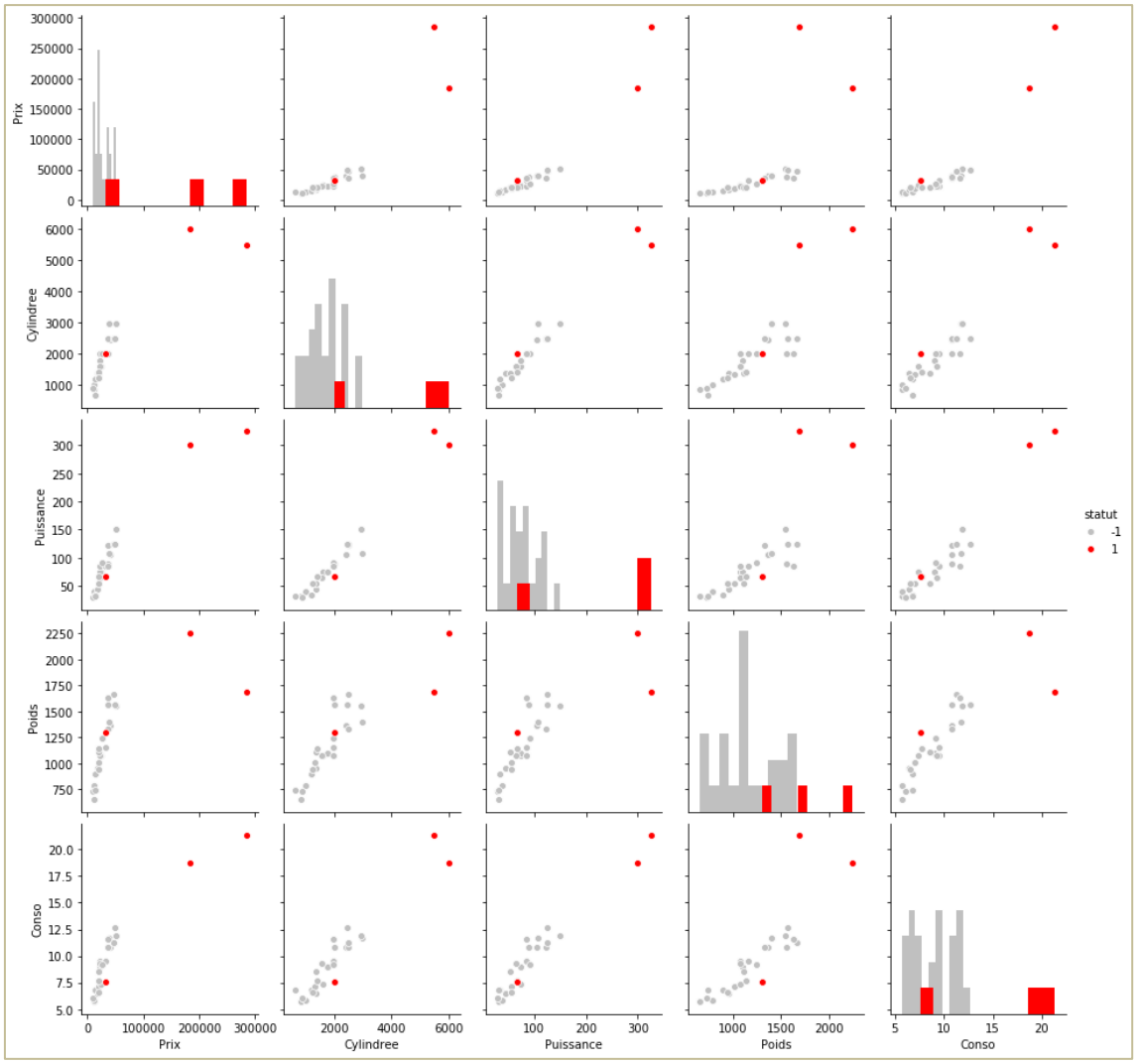

Figure 4 - Position des points atypiques (en rouge) dans les graphiques par paires

Il n'y avait pas trop de suspens concernant la Ferrari et la Mercedes (les deux points au nord-est).

La désignation de la Mitsubishi Galant comme atypique est un peu plus étonnante. Elle semble être située au cœur même des données (Figure 4). Nous remarquons néanmoins qu'elle s'écarte un peu de ses voisins directs : (1) (poids, puissance), à puissance égale, elle semble plus lourde ; (2) (poids, conso), à consommation égale, elle est encore une fois plus lourde.

Pour vérifier ces assertions, j'ai repris le tableau de données trié selon le poids croissant (Tableau 3). Nous constatons effectivement que les véhicules qui présentent une puissance similaire à la Mitsubishi (Fiat Tempra, Honda Civic) sont plus légères, celles qui présentent une consommation similaire (Toyota Corolla, Opeal Astra, Honda Civic) le sont également. Une distance de Mahalanobis plus élevée caractérise ces déphasages.

Modele	Prix	Cylindree	Puissance	Poids	Conso
Daihatsu Cuore	11600	846	32	650	5.7
Fiat Panda Mambo L	10450	899	29	730	6.1
Subaru Vivio 4WD	13730	658	32	740	6.8
Suzuki Swift 1.0 GLS	12490	993	39	790	5.8
Opel Corsa 1.2i Eco	14825	1195	33	895	6.8
Ford Fiesta 1.2 Zetec	19740	1242	55	940	6.6
VW Polo 1.4 60	17140	1390	44	955	6.5
Toyota Corolla	19490	1331	55	1010	7.1
Seat Ibiza 2.0 GTI	22500	1983	85	1075	9.5
Opel Astra 1.6i 16V	25000	1597	74	1080	7.4
Fiat Tempra 1.6 Liberty	22600	1580	65	1080	9.3
Peugeot 306 XS 108	22350	1761	74	1100	9
Fort Escort 1.4i PT	20300	1390	54	1110	8.6
Honda Civic Joker 1.4	19900	1396	66	1140	7.7
VW Golt 2.0 GTI	31580	1984	85	1155	9.5
Nissan Primera 2.0	26950	1997	92	1240	9.2
Mitsubishi Galant	31990	1998	66	1300	7.6
Mazda Hachtback V	36200	2497	122	1330	10.8
Volvo 850 2.5	39800	2435	106	1370	10.8
Hyundai Sonata 3000	38990	2972	107	1400	11.7
Lancia K 3.0 LS	50800	2958	150	1550	11.9
Peugeot 806 2.0	36950	1998	89	1560	10.8
Volvo 960 Kombi aut	49300	2473	125	1570	12.7
Seat Alhambra 2.0	36400	1984	85	1635	11.6
Opel Omega 2.5i V6	47700	2496	125	1670	11.3
Ferrari 456 GT	285000	5474	325	1690	21.3
Mercedes S 600	183900	5987	300	2250	18.7

Tableau 3 - Tableau de données trié selon le poids croissant

3.4 Remarque sur le taux de contamination, paramètre de l'algorithme

L'idée de s'appuyer sur des moyennes et covariances robustes pour calculer les distances et identifier les observations atypiques est séduisante. Mais faire reposer la décision atypique / non-atypique sur un paramètre d'entrée de l'algorithme, la proportion des données que l'on doit considérer comme ''outliers'' avec le paramètre (contamination), n'est pas très satisfaisant. On souhaite s'appuyer sur la méthode justement pour les identifier. Annoncer a priori leur nombre sur la foi de je ne sais quelle information est plus que contestable.

Une piste possible serait d'afficher un graphique de décroissance des distances et de déceler les ''décrochements'' pour identifier les observations à analyser en priorité. Cette fameuse ''méthode du coude'', pour empirique qu'elle soit, a montré son efficacité en statistique exploratoire à maintes reprises. Dans notre cas, Les Ferrari et Mercedes viennent définitivement d'une autre planète. C'est plus discutable, même si on peut l'envisager, en ce qui concerne la Mitsubishi et, dans une moindre mesure encore, la Hyundai.

```
#décroissance de la distance de Mahalanobis
tmp = pandas.DataFrame({"Modele":cars.index,"dist":dm}).sort_values(by='dist',ascending=False)
sns.barplot(x='dist',y='Modele',data=tmp)
```

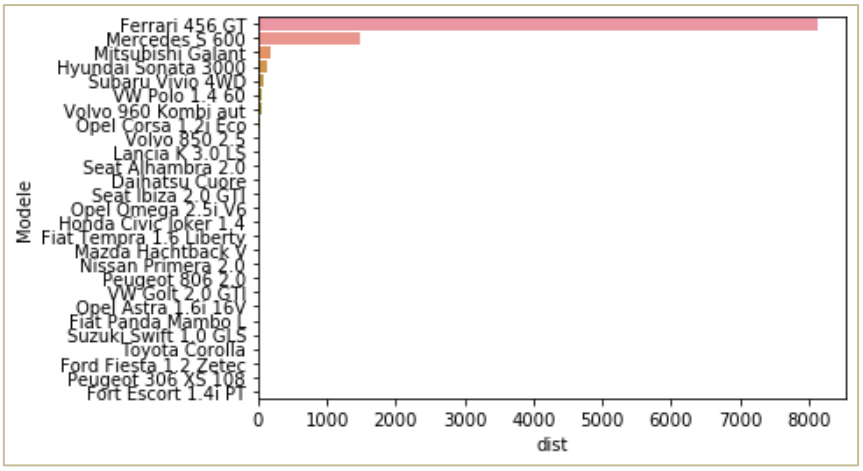

Figure 5 - Décroissance de la distance de Mahalanobis

Après, soyons honnête, cette approche n'est tenable que sur de petites bases. Pour les grands ensembles de données, inspecter unes à unes les observations n'est pas très réaliste.

3.5 Identification des nouveautés

Nous nous occupons des observations supplémentaires dans cette section. Dans quelle mesure les nouveaux véhicules (Tableau 2) diffèrent significativement du premier dataset qui sert maintenant de référence ? Attention : ce n'est pas tout le premier dataset qui sert de référence, mais uniquement ceux qui ont participé aux calculs du barycentre et de la matrice de variance covariance robustes. A savoir :

```
#véhicules de référence (issus du premier dataset -- Tableau 1)
print(cars.loc[eev.support_ == True,:])
```

```
                         Prix  Cylindree  Puissance  Poids  Conso
Modele
Daihatsu Cuore           11600        846         32    650    5.7
Suzuki Swift 1.0 GLS     12490        993         39    790    5.8
Fiat Panda Mambo L       10450        899         29    730    6.1
Toyota Corolla           19490       1331         55   1010    7.1
Opel Astra 1.6i 16V      25000       1597         74   1080    7.4
Peugeot 306 XS 108       22350       1761         74   1100    9.0
Seat Ibiza 2.0 GTI       22500       1983         85   1075    9.5
VW Golt 2.0 GTI          31580       1984         85   1155    9.5
Fiat Tempra 1.6 Liberty  22600       1580         65   1080    9.3
Fort Escort 1.4i PT      20300       1390         54   1110    8.6
Honda Civic Joker 1.4    19900       1396         66   1140    7.7
Volvo 850 2.5            39800       2435        106   1370   10.8
Ford Fiesta 1.2 Zetec    19740       1242         55    940    6.6
Lancia K 3.0 LS          50800       2958        150   1550   11.9
Mazda Hachtback V        36200       2497        122   1330   10.8
Opel Omega 2.5i V6       47700       2496        125   1670   11.3
Peugeot 806 2.0          36950       1998         89   1560   10.8
Nissan Primera 2.0       26950       1997         92   1240    9.2
Seat Alhambra 2.0        36400       1984         85   1635   11.6
```

Nous chargeons le second ensemble de données.

```
#véhicules supplémentaires
cars_supp = pandas.read_excel("cars_outliers_novelties.xlsx",sheet_name="novelty",index_col=0)
print(cars_supp)
```

```
                        Prix  Cylindree  Puissance  Poids  Conso
Modele
Citroen ZX Volcane     28750       1998         89   1140    8.8
Renault Safrane 2.2. V 36600       2165        101   1500   11.7
Toyota Previa salon    50900       2438         97   1800   12.8
Maserati Ghibli GT     92500       2789        209   1485   14.5
```

Bon, si on connait un petit peu les voitures, on perçoit tout de suite celle, racée, qui dénote des autres, loutres sur roues. Voyons si nous retrouvons la même chose par le calcul. Nous faisons appel à predict() sur ce nouvel ensemble de données pour identifier les ''nouveautés''.

```
#identifier leur positionnement
statut_supp = eev.predict(cars_supp)
print(statut_supp)
```

```
[ 1  1  1 -1]
```

A l'évidence, la Maserati, en dernière position dans le data frame, est celle qui dénote des véhicules de l'ensemble de référence. De manière très marquée si nous nous référons à decision_function() dont le calcul repose sur la distance de Mahalanobis.

```
#decision function - en négatif les 'novelties'
df_supp = eev.decision_function(cars_supp)
print(pandas.DataFrame({"Modele":cars_supp.index,"decision":df_supp}).sort_values(by='decision'))
```

```
                  Modele     decision
3     Maserati Ghibli GT -1060.781404
2    Toyota Previa salon   112.431935
1  Renault Safrane 2.2. V   137.498026
0     Citroen ZX Volcane   144.171539
```

Positionnons ces individus parmi les véhicules de référence dans l'espace des paires de variables pour appréhender différemment les résultats.

```
#afficher les véhicules parmi les non atypiques
tmp = cars.loc[statut == 1].append(cars_supp)
tmp['id'] = numpy.zeros(tmp.shape[0])
tmp.iloc[-4:-1,-1] = 1
tmp.iloc[-1,-1] = 2

#graphique
sns.pairplot(tmp,hue='id',palette={0:'silver',1:'green',2:'red'},diag_kind='hist')
```

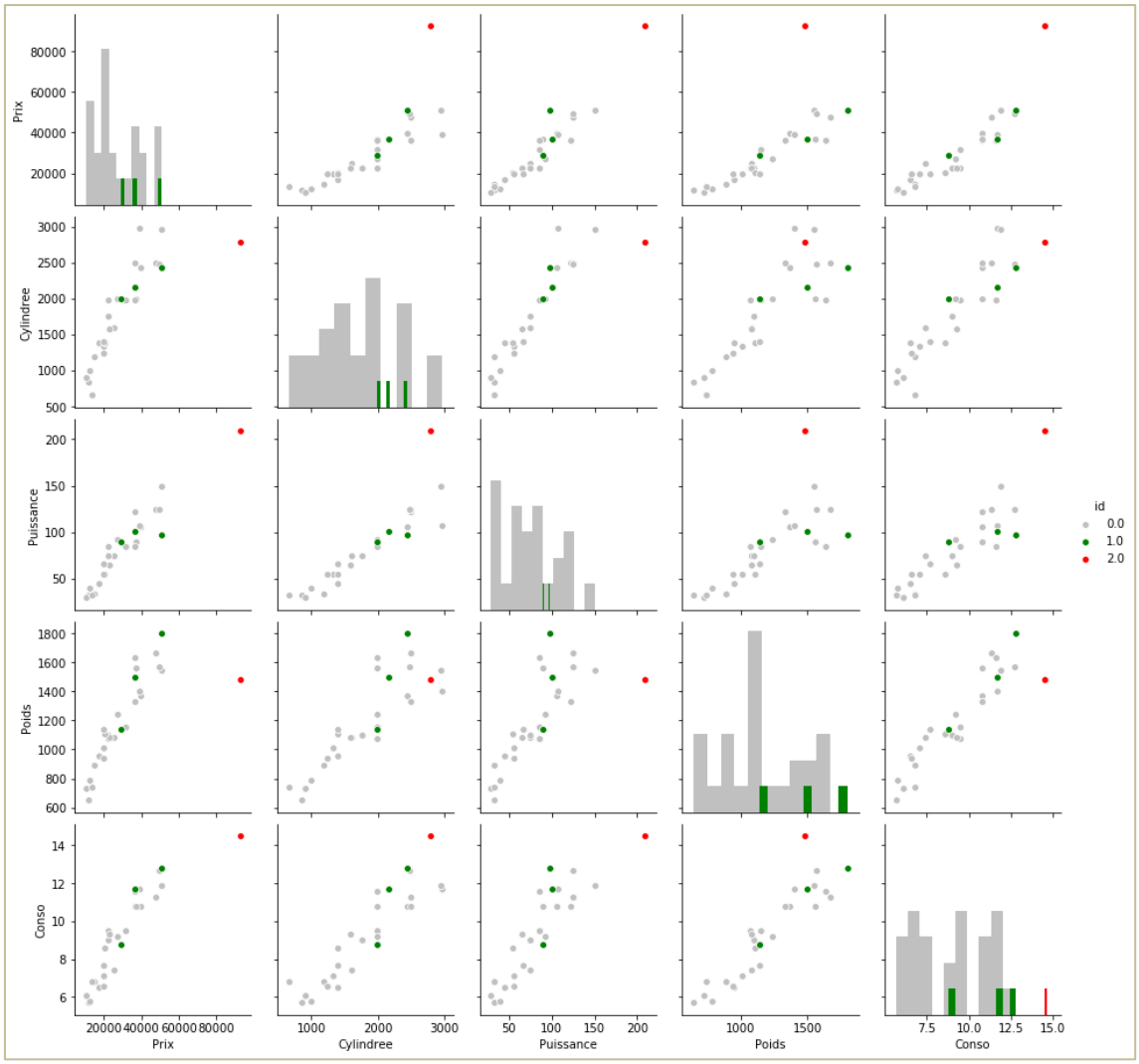

Figure 6 - Position des individus supp. par rapport à la référence (gris), 3 inliers (vert), 1 outlier (rouge)

(Toyota Previa, Renault Safrane, Citroën ZX), en vert, s'inscrivent parfaitement parmi les points de l'ensemble de référence (en gris) (Figure 6). La Maserati (en rouge) est à l'écart, elle occupe une place qui la rapprocherait plutôt de la Ferrari et la Mercedes, qui ont été écartées du ''modèle'' de référence rappelons-le.

4 Identification locale des observations atypiques

Dans cette section, notre démarche sera guidée par le support de cours rédigé récemment sur la méthode ''Local outlier factor'' ("Détection des anomalies", décembre 2019). On considère comme atypique un point dont la densité locale est plus faible que celle de ses voisins immédiats. Le nombre de voisins (n_neighbors) à prendre en compte est un facteur de la classe de calcul LocalOutlierFactor. Par rapport à l'approche précédente (basée sur la distance de Mahalanobis),

elle ne repose pas sur une estimation d'un barycentre global, elle reste donc valable même si les données sont organisées en clusters de densités différentes.

4.1 Préparation des données

Il va être question de calculs de distances euclidiennes entre paires d'observations. Les variables étant exprimées dans des unités différentes, ils peuvent être faussés. Nous décidons de réduire les variables c.-à-d. de les diviser par les écarts-type, mais sans les centrer. Nous utilisons la classe StandardScaler. Nous vérifions que les résultats (les écarts-type par variable) sont conformes à ce qui est attendu.

```
#réduction des variables
from sklearn.preprocessing import StandardScaler
stds = StandardScaler(with_mean=False,with_std=True)

#variables transformées + vérification de l'écart-type après transformation
Zcars = stds.fit_transform(cars)
print(numpy.std(Zcars,axis=0))
```

```
[1. 1. 1. 1. 1.]
```

4.2 Identification des observations atypiques

Nous pouvons instancier la classe de calcul. Nous décidons de fixer le nombre de voisins à prendre en compte à (n_neighbors = 3) (Attention : ce paramètre n'a rien à voir avec le nombre de points atypiques que l'on cherche à identifier ! qui ne fait pas partie des paramètres de l'algorithme). Nous l'appliquons sur la version transformée du premier ensemble de données. Nous affichons le diagnostic pour chaque observation.

```
#local outlier factor
from sklearn.neighbors import LocalOutlierFactor
loc = LocalOutlierFactor(n_neighbors=3)

#calcul
loc.fit(Zcars)

#degré d'anormalité des observations (NOF : negative outlier factor)
print(pandas.DataFrame({"Modele":cars.index,"NOF":loc.negative_outlier_factor_}).sort_values(by='NOF'))
```

```
                     Modele       NOF
7            Ferrari 456 GT -5.356444
8            Mercedes S 600 -5.027012
21        Mitsubishi Galant -1.518031
20        Mazda Hachtback V -1.206907
2        Fiat Panda Mambo L -1.121528
24       Nissan Primera 2.0 -1.098782
3            VW Polo 1.4 60 -1.085071
```

```
9       Opel Astra 1.6i 16V -1.077135
15    Honda Civic Joker 1.4 -1.075720
12          VW Golt 2.0 GTI -1.055123
14      Fort Escort 1.4i PT -1.031674
16            Volvo 850 2.5 -1.030281
22       Opel Omega 2.5i V6 -1.022065
17    Ford Fiesta 1.2 Zetec -1.015101
25        Seat Alhambra 2.0 -1.014424
10       Peugeot 306 XS 108 -1.008057
13  Fiat Tempra 1.6 Liberty -1.001974
19          Lancia K 3.0 LS -0.995477
26      Volvo 960 Kombi aut -0.995477
18      Hyundai Sonata 3000 -0.984054
23          Peugeot 806 2.0 -0.972561
0            Daihatsu Cuore -0.965632
5          Subaru Vivio 4WD -0.963006
1      Suzuki Swift 1.0 GLS -0.963006
4       Opel Corsa 1.2i Eco -0.934116
11       Seat Ibiza 2.0 GTI -0.915378
6            Toyota Corolla -0.910255
```

L'outil fournit le degré d'anormalité de chaque observation, que nous pouvons comparer à une valeur seuil (negative_outlier_factor_ < offset_ ==> outlier).

```
#valeur seuil
print(loc.offset_)
```

```
-1.5
```

La Ferrari et la Mercedes sont définitivement hors sujet. La Mitsibushi Galant, que l'on commence à bien connaître maintenant, est désignée ''outlier'', de très peu certes. Nous savons pourquoi.

4.3 Identification des nouveautés

Pour l'étiquetage des observations supplémentaires, la décision de conserver les points atypiques dans l'ensemble de référence se pose. En effet, un véhicule proche de la Ferrari ou de la Mercedes peut-il être vraiment considéré comme une ''nouveauté'' ? Ses congénères sont bien présents dans la base initiale. Il n'est pas vraiment ''surprenant'' à ce titre, même si ces véhicules sont hors normes. Le problème ne se posait pas pour la précédente méthode où les observations extrêmes étaient d'office exclus des calculs. Il est crucial ici car c'est la notion même de ''nouveauté'' qui est questionnable dans notre contexte. Pour ma part, sur cette exemple-ci, j'ai décidé de conserver la totalité de la base de référence.

Nous réduisons les observations supplémentaires avec les paramètres (écarts-type) calculés sur la base de référence.

```
#transformation des observations supplémentaires
Zcars_supp = stds.transform(cars_supp)
print(pandas.DataFrame(index=cars_supp.index,data=Zcars_supp,columns=cars_supp.columns))
```

```
                          Prix  Cylindree  Puissance     Poids     Conso
Modele
Citroen ZX Volcane      0.502963   1.658543   1.276686  3.187727  2.455964
Renault Safrane 2.2. V  0.640294   1.797170   1.448823  4.194378  3.265316
Toyota Previa salon     0.890464   2.023788   1.391444  5.033253  3.572311
Maserati Ghibli GT      1.618230   2.315153   2.998060  4.152434  4.046759
```

Et nous faisons appel à la fonction predict() pour identifier les observations singulières.

```
#identification - patatras
loc.predict(Zcars_supp)
```

```
AttributeError: predict is not available when novelty=False, use fit_predict if you
want to predict on training data. Use novelty=True if you want to use LOF for
novelty detection and predict on new unseen data.
```

Patatras, l'opération n'est pas possible. L'outil n'est utilisable que dans un seul contexte, soit pour la détection des ''outliers'' dans une base de données, soit pour la détection des ''novelties'' par rapport à une base de référence.

Nouvelle version de l'objet de calcul. Nous instancions donc un nouvel objet avec l'option (novelty = True) pour se placer dans le second contexte. Nous lui passons la base de référence en faisant appel à la fonction fit(). Puis nous identifions les nouveautés en faisant appel – de manière licite cette fois-ci – à la fonction predict().

```
#instaciation avec le bon paramètre
loc_bis = LocalOutlierFactor(n_neighbors=3, novelty=True)

#apprentissage et identification
loc_bis.fit(Zcars)

#prédiction
loc_atyp = loc_bis.predict(Zcars_supp)
print(pandas.DataFrame({"Modele":cars_supp.index,"Atypique":loc_atyp}))
```

```
                  Modele  Atypique
0      Citroen ZX Volcane         1
1  Renault Safrane 2.2. V         1
2     Toyota Previa salon         1
3      Maserati Ghibli GT        -1
```

La Maserati Ghibli est vraiment singulière, y compris par rapport à la Ferrari et la Mercedes. Tiens donc ! Un petit graphique pour bien situer tout cela (Figure 7).

```
#afficher les véhicules parmi les références
tmp = pandas.DataFrame(data=Zcars,columns=cars.columns)
tmp = tmp.append(pandas.DataFrame(Zcars_supp,columns=cars.columns))
tmp['id'] = numpy.zeros(tmp.shape[0])
tmp.iloc[-4:-1,-1] = 1
tmp.iloc[-1,-1] = 2
```

```
#graphique
sns.pairplot(tmp,hue='id',palette={0:'silver',1:'green',2:'red'},diag_kind='hist')
```

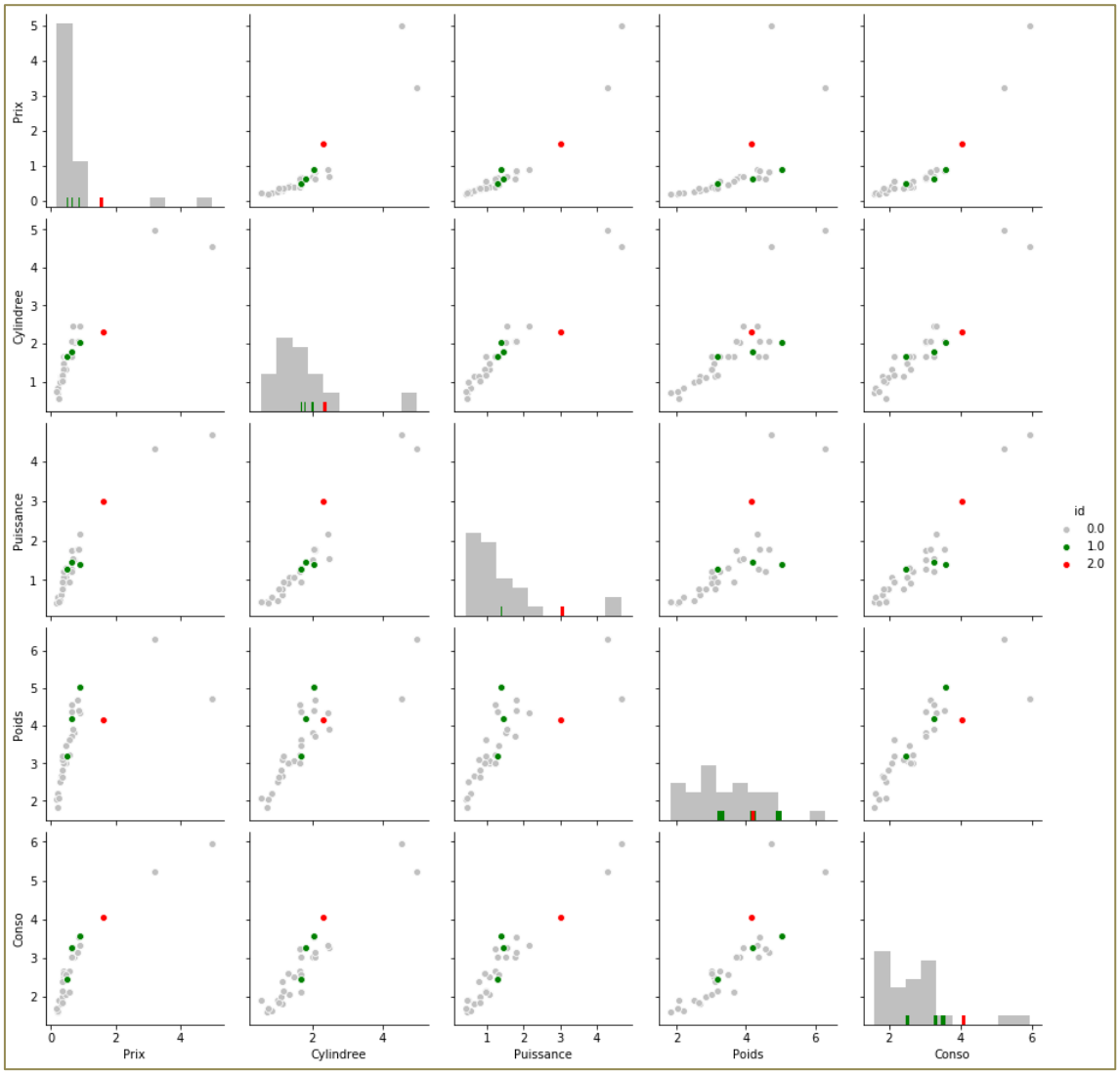

Figure 7 - Local outlier factor - Position des observations supplémentaires

La Maserati (en rouge) est effectivement isolée (Figure 7), à la fois des roturières et des aristocrates (un peu à mi-chemin entre les deux catégories en fait, surtout en termes de prix, de puissance et de consommation). Les Maserati ont toujours été un peu spéciales. La technique l'a bien identifiée.

5 Conclusion

La détection des anomalies prend de l'ampleur dans le contexte big data parce que la volumétrie empêche l'analyse au cas par cas des données. Il faut des techniques de machine learning pour

nous alerter sur la singularité de certaines observations, nous permettant ainsi de porter notre attention sur les situations réellement particulières, relevant par exemple de comportements frauduleux ou encore de défectuosités d'un système.

Nous nous sommes intéressés à deux approches implémentées dans la librairie Scikit-Learn pour Python dans ce tutoriel. Nous avons travaillé sur une petite base pour comprendre la teneur des algorithmes, mais la démarche est généralisable au traitement des très grandes bases. Nous avons constaté que ces techniques sont simples d'utilisation et proposent des résultats que nous pouvons assez facilement appréhender.

6 Références

Scikit-Learn, ''Novelty and Outlier Detection'', https://scikit-learn.org/stable/modules/outlier_detection.html

Tutoriel Tanagra, ''Détection des anomalies'', décembre 2019.