# Projet
# Py_ecommerce

*Formation Bootcamp Data analyst*

*Janvier - Mars 2021*

Réalisé par Ralph Coriolan & Sandy Semail

## Table des matières

# 1. Présentation du Dataset

Ce projet est basé sur un dataset disponible sur kaggle :
https://www.kaggle.com/retailrocket/ecommerce-dataset

Il est composé de 4 fichiers csv dont 2 fichiers représentent la même base séparée en deux pour des questions de poids.

- category_tree.csv
- events.csv
- item_properties_part1.csv
- item_properties_part2.csv

Ce dataset contient des données relatives à des actions faites sur un site de e-commerce pendant une durée de 4.5 mois.

Le fichier events est le fichier central : il indique les actions réalisées :

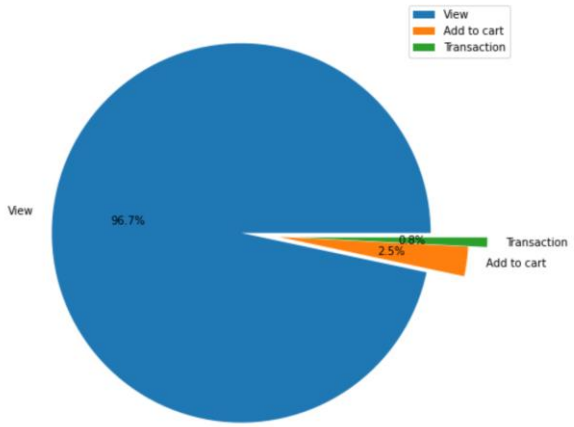
- Par un visiteur (visitor id)
- Pour un produit donnée (item id)
- Les 3 actions réalisables (event) : View, Addtocart et Transaction
- Lorsqu'un achat est réalisé, un transaction_id est intégré dans la base
- Tous ces éléments sont horodatés (timestamp)

| | timestamp | visitorid | event | itemid | transactionid |
|---|---|---|---|---|---|
| **0** | 1433221332117 | 257597 | view | 355908 | NaN |
| **1** | 1433224214164 | 992329 | view | 248676 | NaN |
| **2** | 1433221999827 | 111016 | view | 318965 | NaN |
| **3** | 1433221955914 | 483717 | view | 253185 | NaN |
| **4** | 1433221337106 | 951259 | view | 367447 | NaN |

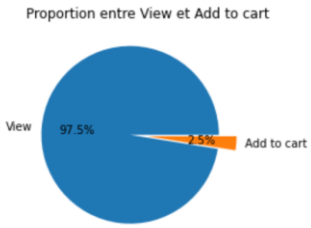
Les 2 autres fichiers apportent des informations complémentaires sur des catégories ou propriétés de produits. Nous allons principalement baser notre étude sur le fichier events qui contient beaucoup d'éléments à analyser.

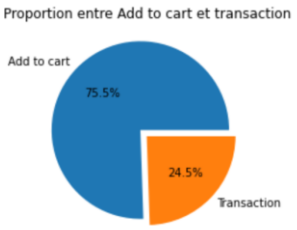
# 2. Analyse exploratoire des données

### 1. Analysons les proportions d'occurrence des différents "events"




Nous constatons que l'event "View" est prépondérant avec 96.7% des events de la base. Le tunnel de conversion d'un achat est View --> Add to cart --> Transactions. Voyons ce qui se passe entre ces 3 actions.




On constate que seul 2.5% des vues sont converties en ajout au panier, et la conversion de mise en panier en acte d'achat se produit dans 24.5% des cas. Par la suite, il pourrait être intéressant d'étudier les raisons du faible taux de conversion entre View et Add to cart (par exemple la non disponibilité des produits?

## 2. Comportement des visiteurs selon les events

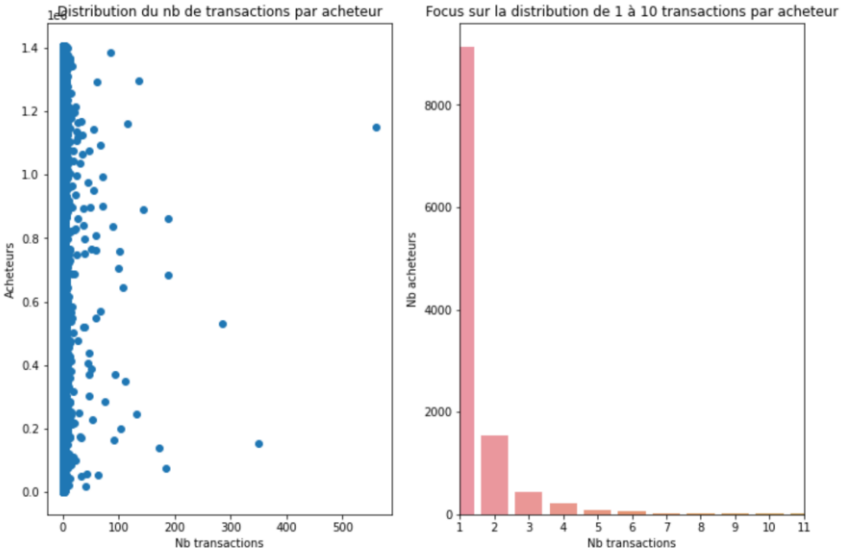
```
Nombre de visites:  2756101
Nombre de visteurs unique:  1407580
Nombre acheteurs unique: 11719
0.83 % des visteurs ont fait au moins une transaction.
```

La proportion très faible de visiteurs qui effectue un achat nous amènera à analyser si un nombre de visteurs anormaux n'affectent pas la base.

```
Les acheteurs font en moyenne 1.51 transactions
```

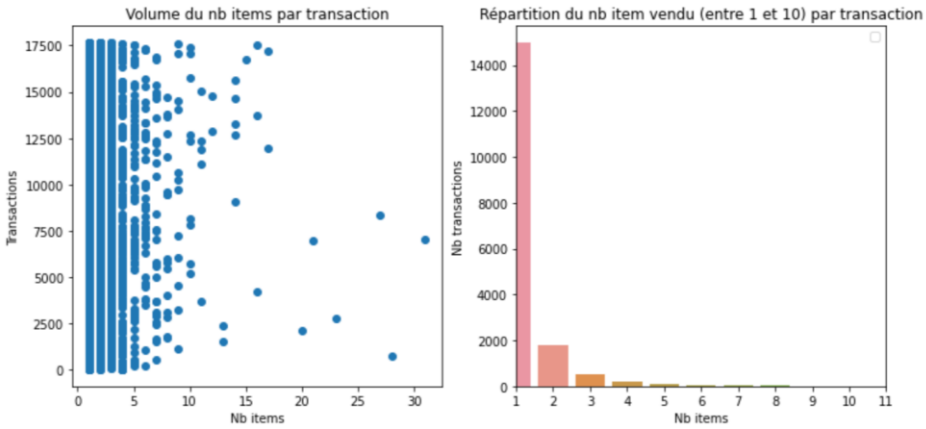
## 3. Analysons plus en détail les comportements d'achat des acheteurs

### ➔ selon le nombre de transactions qu'ils effectuent




On constate que le nombre de transactions par acheteur est faible. Le focus sur la distribution des visiteurs ayant fait entre 1 et 10 transactions nous confirme que le nombre de transaction est majoritairement de 1 et 2 transactions.

### ➔ selon le nombre de produits acheté par transaction

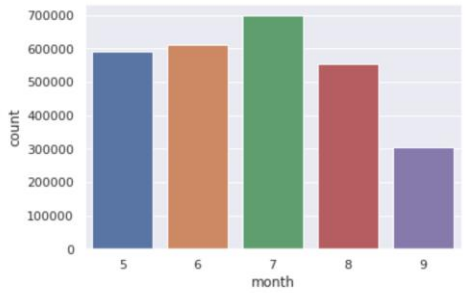



De nouveau, le nombre d'items le plus souvent commandés par transaction se concentre entre 1 et 2 items.

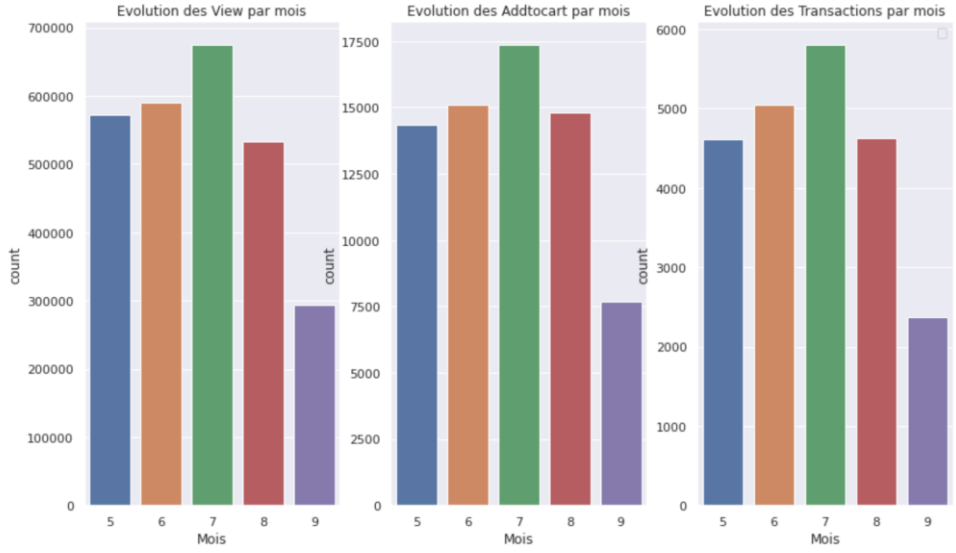
```
Nombre transactions qui intègre 1 à 2 items : 16756 , ce qui correspond à  91.0 % des transactions.
```

Cette première exploration des données concernant les transactions indique que les acheteurs ont majoritairement fait 1 à 2 transactions durant la période et que ces transactions intègrent le plus souvent 1 à 2 items.

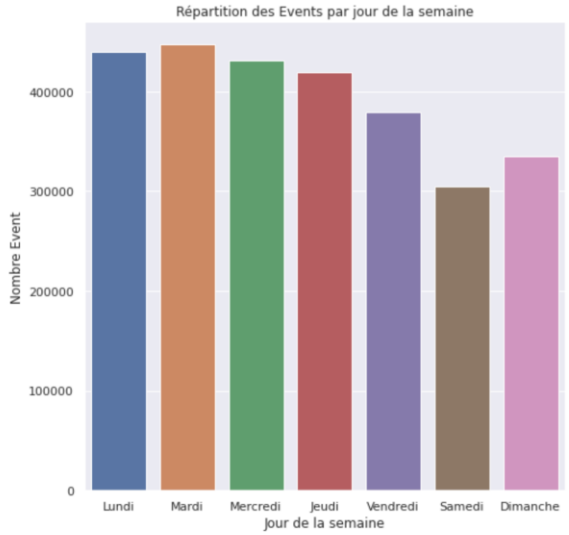
## 4. Analysons les comportements visiteurs selon les dates



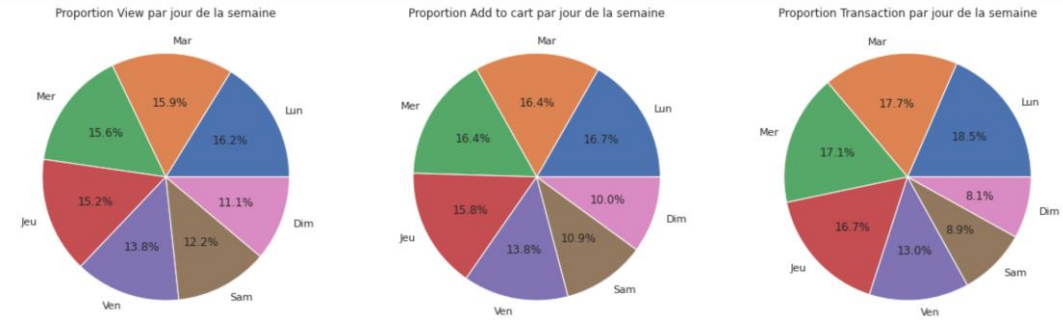

Comme prévu, les données sont sur 4.5 mois entre mai et mi-septembre. Le mois de juillet est celui qui a plus d'event. Sachant que le nombre de de View est prépondérant dans la base, vérifions si cette répartition change selon le type d'event.




Les 3 events suivent les mêmes tendances mensuellement. Analysons maintenant les tendances selon les jours de la semaine.




Le traffic est plus élevé les jours ouvrés que les week-end. Voyons si la tendance entre les 3 events est la même.



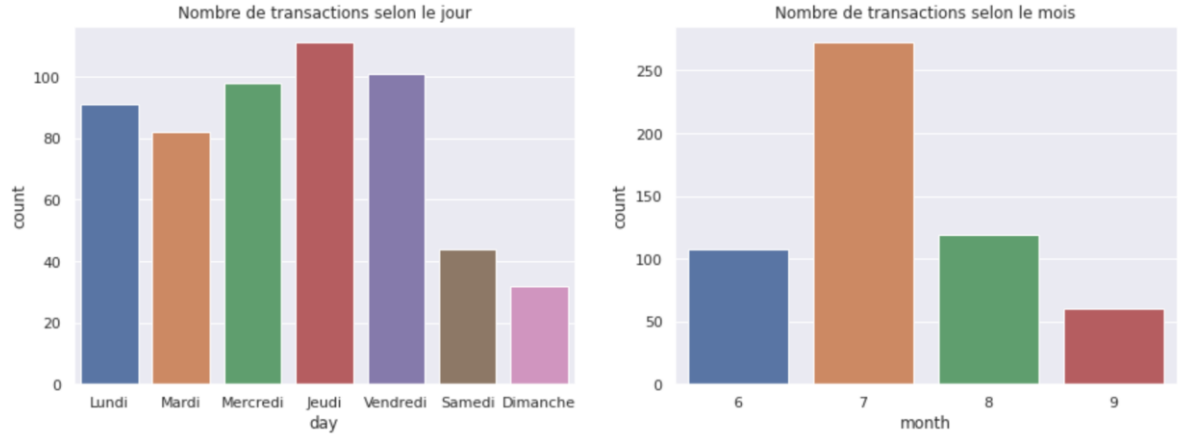

De nouveau, on s'aperçoit que les 3 types d'Events suivent la même tendance selon les jours de la semaine.

## 5. Déterminons les meilleurs clients du site

Durant les 4.5 mois, les 11 719 clients uniques ont acheté 1,2 voir plusieurs articles sur le site. Déterminons quels sont les 10 meilleurs clients.

| | transactionid |
|---|---|
| **visitorid** | |
| **1150086** | 559 |
| **152963** | 349 |
| **530559** | 286 |
| **684514** | 189 |
| **861299** | 188 |
| **...** | ... |
| **535433** | 1 |
| **535446** | 1 |
| **535618** | 1 |
| **535623** | 1 |
| **1407398** | 1 |

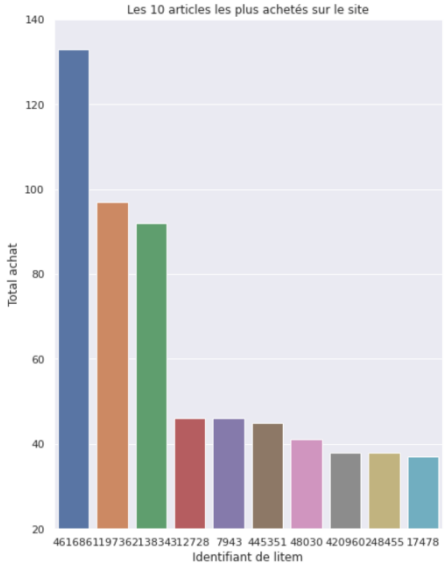
Plus de 90% des clients de ce site ont acheté entre 1 ou 2 articles. Toutefois, on peut remarquer qu'il y en a qui ont commandé plus d'un centaine. Le meilleur client, facilement reparable, est celui à l'indice 1150086 qui a commandé 559 fois.Ces transactions sont croissantes selon le jour de la semaine avec des pics le mercredi et le jeudi puis décroissantes en fin de semaine. Ce client a acheté plus de 250 articles en juillet soit plus du double comparé aux autres mois.




## 6. Les articles les plus commandés par les clients

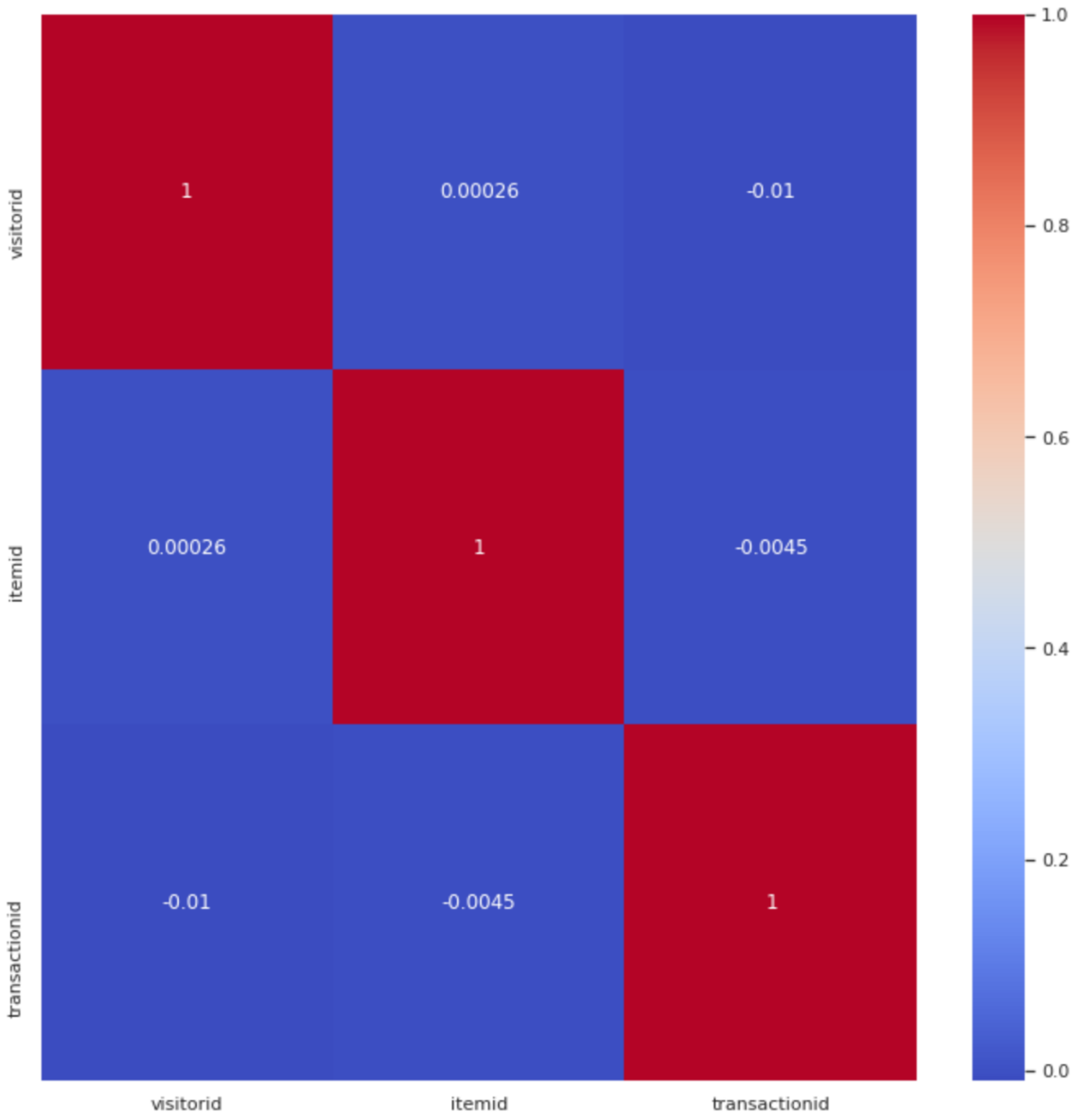
Intéressons-nous maintenant aux articles qui ont été commandés sur le site. On dénombre 12 025 articles uniques qui ont été achetés au cours de la période.

| | itemid | Total_purchased |
|---|---|---|
| **0** | 461686 | 133 |
| **1** | 119736 | 97 |
| **2** | 213834 | 92 |
| **3** | 312728 | 46 |
| **4** | 7943 | 46 |
| **5** | 445351 | 45 |
| **6** | 48030 | 41 |
| **7** | 420960 | 38 |
| **8** | 248455 | 38 |
| **9** | 17478 | 37 |



## 7. Analyse de corrélation

On constate que les 3 variables principales de ce dataset ne sont pas corrélées.



# 3. Clustering

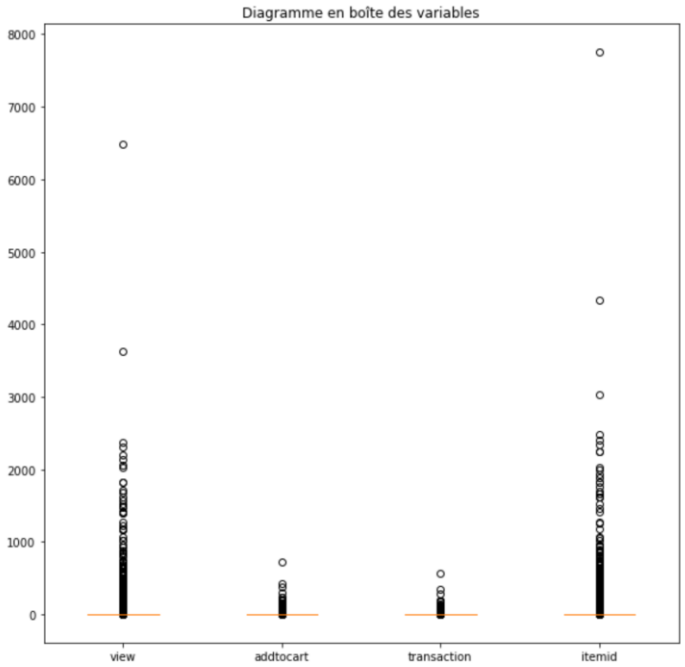
Notre première itération a pour objectif de déterminer des regroupements de comportements de clients pertinents pour adapter le discours marketing selon leur besoins/modes de consommation.

Nous allons réaliser une analyse de segmentation des visiteurs en se basant sur les événements réalisés et les articles achetés. Dans un premier, la variable 'event' sera dichotomisée afin de faire ressortir le poids de chaque type d'action.

Les variables d'interêt sont ensuite regroupées dans un nouveau dataset appelé profil_visitor

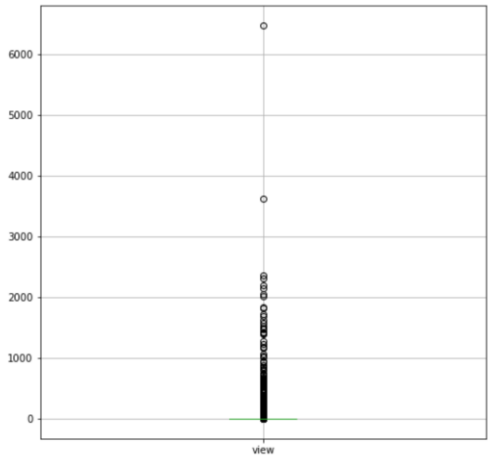
| | visitorid | view | addtocart | transaction | itemid |
|---|---|---|---|---|---|
| **1150086** | 1150086 | 6479.0 | 719.0 | 559.0 | 7757 |
| **152963** | 152963 | 2304.0 | 371.0 | 349.0 | 3024 |
| **530559** | 530559 | 3623.0 | 419.0 | 286.0 | 4328 |
| **684514** | 684514 | 1826.0 | 231.0 | 189.0 | 2246 |
| **861299** | 861299 | 1573.0 | 230.0 | 188.0 | 1991 |
| **76757** | 76757 | 1402.0 | 296.0 | 185.0 | 1883 |
| **138131** | 138131 | 1231.0 | 207.0 | 173.0 | 1611 |
| **890980** | 890980 | 662.0 | 210.0 | 145.0 | 1017 |

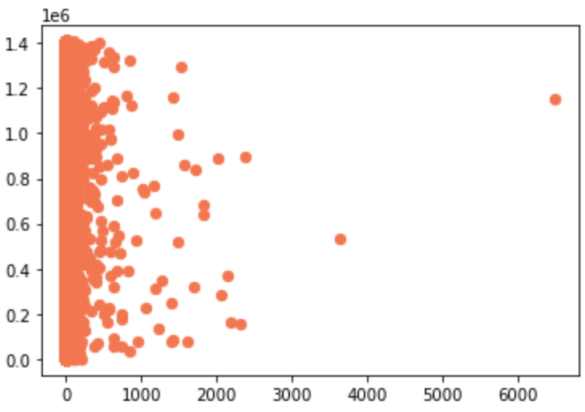
Ce premier aperçu de la base de données laisse présager de fortes variations dans la distribution des variables et donc la présence d'outliers. Ce qui se confirme grâce à la boite à moustaches suivant :




Étant donné l'échelle différente des variables, il est préférable de réaliser les boites à moustaches séparément afin de mieux faire ressortir ces valeurs extrêmes.

➔ Variable « View »

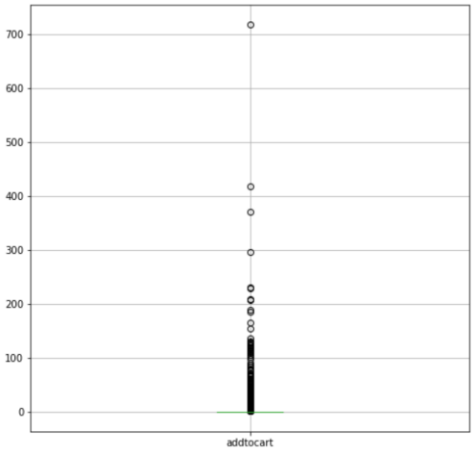


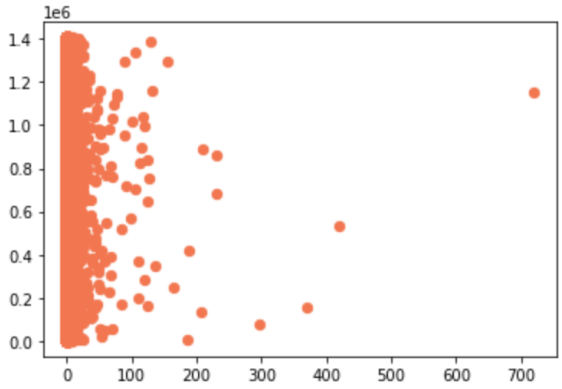



Les visiteurs ayant fait plus de 3000 « view » sont considérés comme outliers :

| | visitorid | view | addtocart | transaction | itemid |
|---|---|---|---|---|---|
| **530559** | 530559 | 3623.0 | 419.0 | 286.0 | 4328 |
| **1150086** | 1150086 | 6479.0 | 719.0 | 559.0 | 7757 |

➔ Variable « Addtocart »

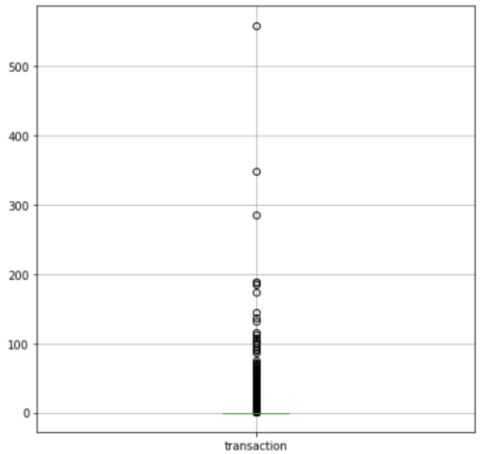


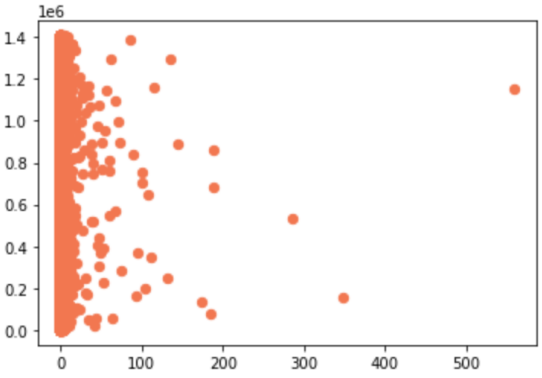




Les visiteurs ayant fait plus de 270 « addtocart » sont considérés comme outliers :

| | visitorid | view | addtocart | transaction | itemid |
|---|---|---|---|---|---|
| **76757** | 76757 | 1402.0 | 296.0 | 185.0 | 1883 |
| **152963** | 152963 | 2304.0 | 371.0 | 349.0 | 3024 |
| **530559** | 530559 | 3623.0 | 419.0 | 286.0 | 4328 |
| **1150086** | 1150086 | 6479.0 | 719.0 | 559.0 | 7757 |

➔ Variable « Transaction »

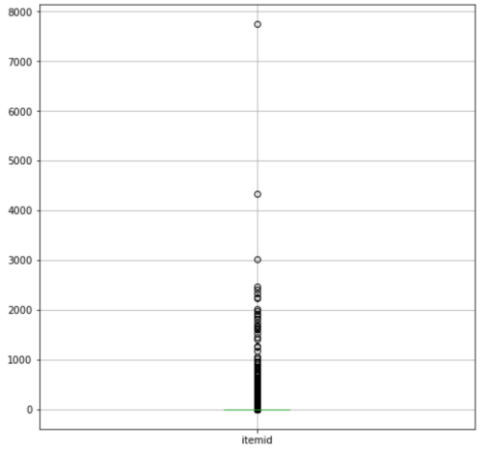


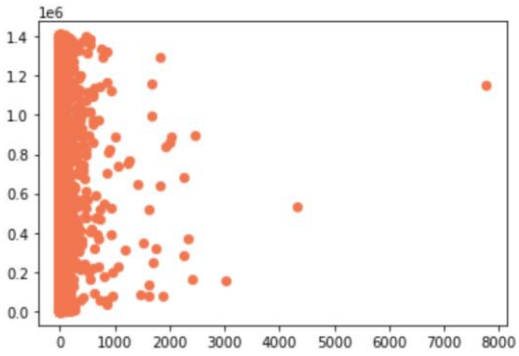




Les visiteurs ayant fait plus de 200 « transactions » sont considérés comme outliers :

| | visitorid | view | addtocart | transaction | itemid |
|---|---|---|---|---|---|
| **152963** | 152963 | 2304.0 | 371.0 | 349.0 | 3024 |
| **530559** | 530559 | 3623.0 | 419.0 | 286.0 | 4328 |
| **1150086** | 1150086 | 6479.0 | 719.0 | 559.0 | 7757 |

➔ Variable « Itemid »




Les visiteurs ayant acheté plus de 2800 « items » sont considérés comme outliers :

| | visitorid | view | addtocart | transaction | itemid |
|---|---|---|---|---|---|
| **152963** | 152963 | 2304.0 | 371.0 | 349.0 | 3024 |
| **530559** | 530559 | 3623.0 | 419.0 | 286.0 | 4328 |
| **1150086** | 1150086 | 6479.0 | 719.0 | 559.0 | 7757 |

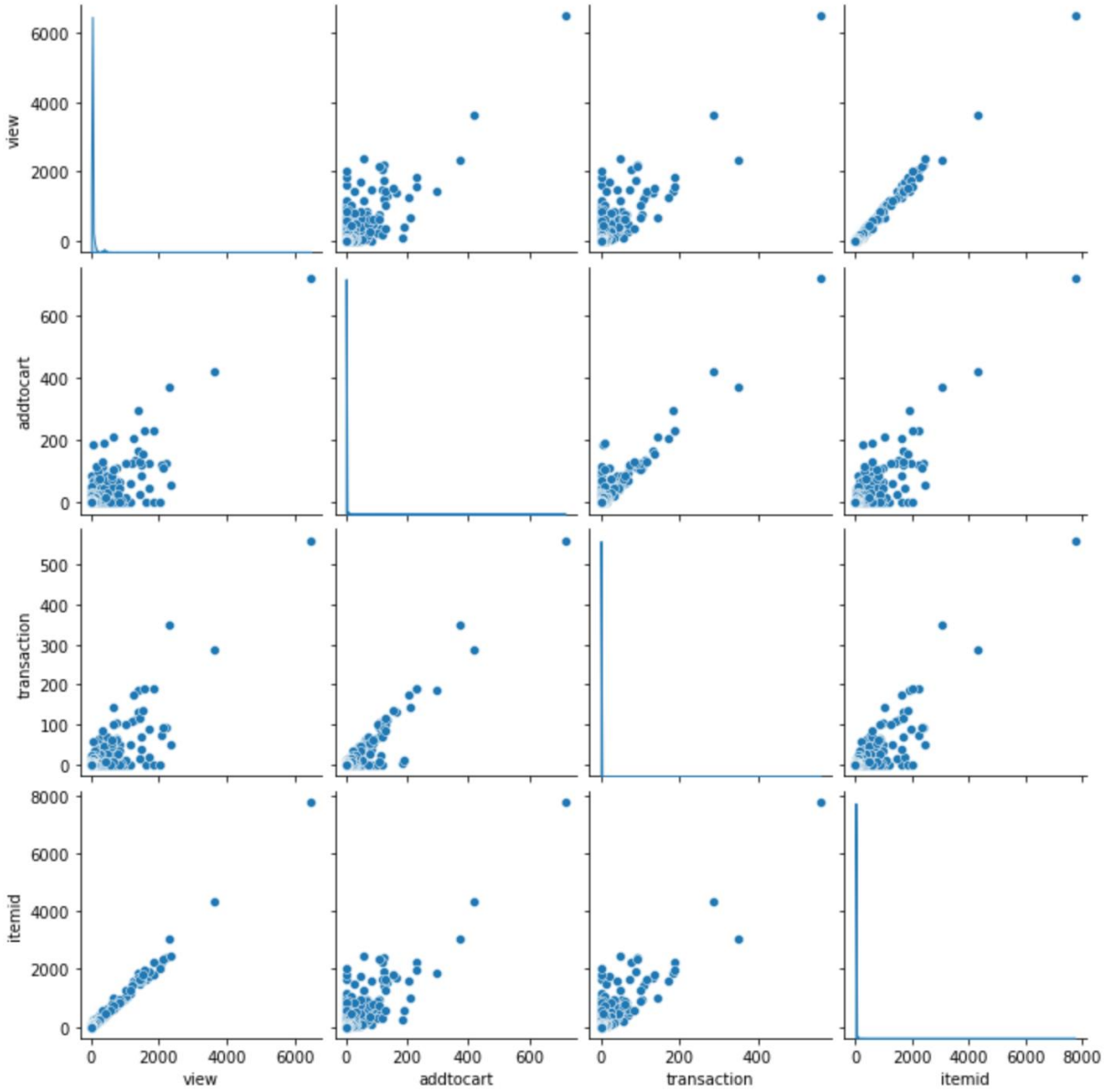
On remarque que les outliers concernent globalement les mêmes visiteurs. Affichons-les :

| | visitorid | view | addtocart | transaction | itemid |
|---|---|---|---|---|---|
| **76757** | 76757 | 1402.0 | 296.0 | 185.0 | 1883 |
| **152963** | 152963 | 2304.0 | 371.0 | 349.0 | 3024 |
| **530559** | 530559 | 3623.0 | 419.0 | 286.0 | 4328 |
| **1150086** | 1150086 | 6479.0 | 719.0 | 559.0 | 7757 |

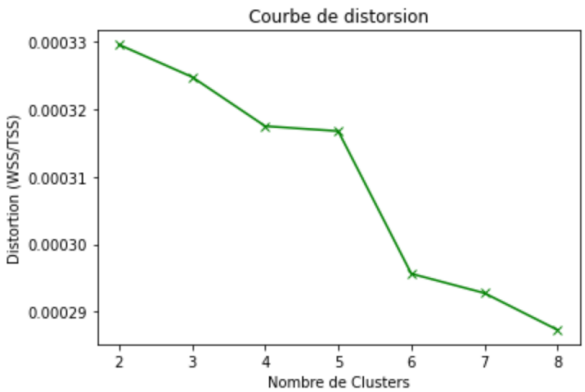
Les comportements des ces outliers différent grandement des autres visiteurs lorsqu'on analyse la distribution globale des variables d'intérêt. Ces visiteurs peuvent être tout aussi bien des gros clients (i.e grandes entreprises) qui passent de nombreuses commandes sur le site ou des robots. Étant donné le contexte métier et le modèle de clustering que l'on se propose de faire, les outliers seront gardés. Néanmoins, ils sont susceptibles d'être enlevés dans d'autres modélisations de machine learning qui sont en principe plus sensibles à ces valeurs extrêmes.

Avant de procéder au clustering des visiteurs, il est important de standardiser les données qui sont présentées suivant des échelles de grandeurs différentes. Deux choix sont souvent opposés : la normalisation ou la standardisation. Alors que la normalisation consiste à redimensionner les valeurs dans un intervalle fixe [0,1], la standardisation permet de redimensionner les données suivant une loi normale de moyenne nulle et variance unitaire.Ces deux techniques de feature scaling permettent de réduire la variance des variables et de minimiser du même coup l'effet des outliers. Grace à une analyse de la fonction de densité propre à chaque variable en diagonale, on peut voir que les features ne suivent une distribution normale. Le choix de la méthode de preprocessing se portera donc sur la normalisation Min-Max.

Pour compléter carte des corrélations, affichons un pairplot pour vérifier la linéarité entre les données :

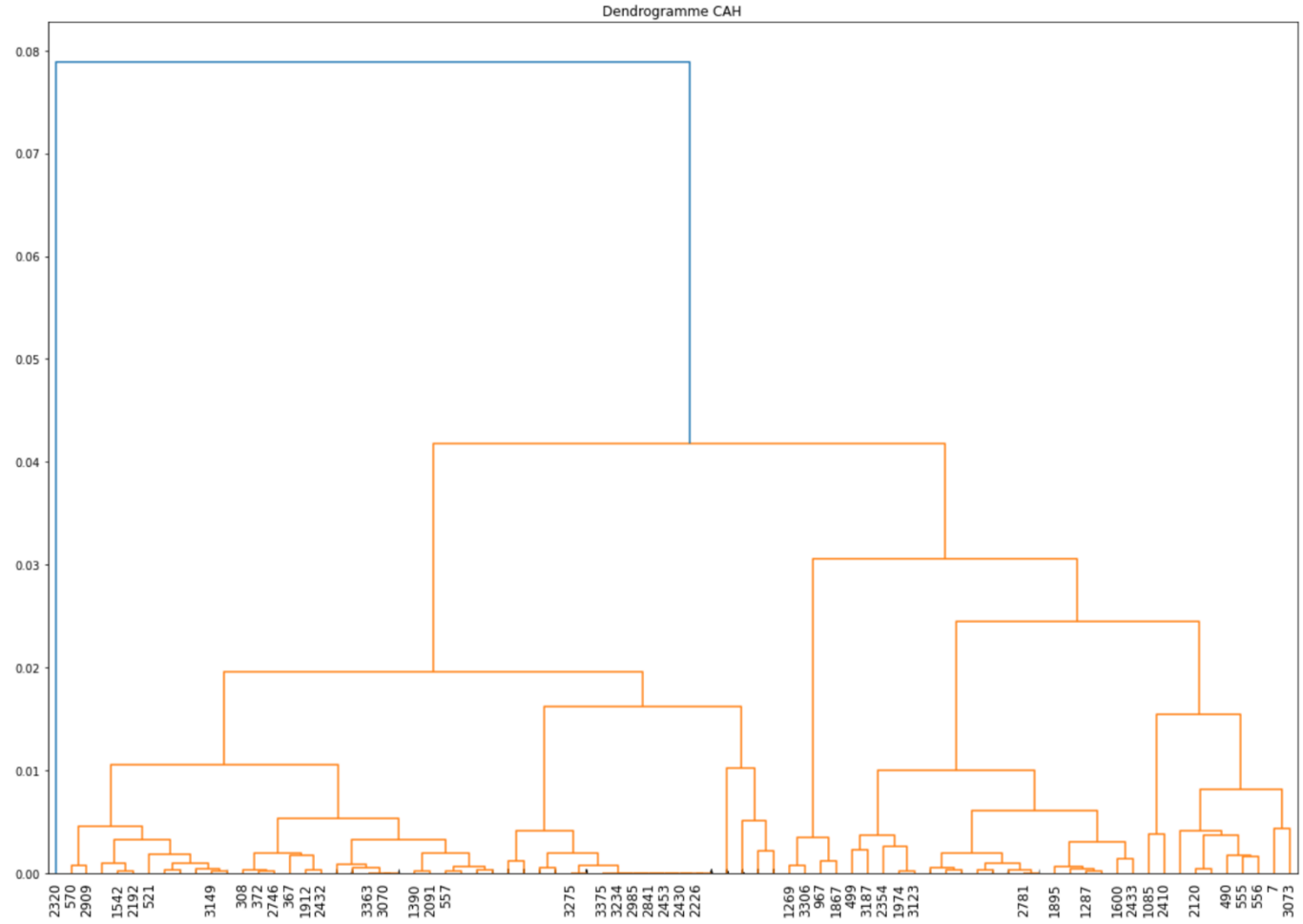



Passons maintenant à la segmentation des visiteurs. Au vue de la taille de l'échantillon de données et des paramètres de regroupement (4 variables), on peut supposer que le nombre de clusteurs est supérieur à 2. Utilisons la courbe de distorsion pour déterminer le nombre de clusters optimal.




La courbe de distorsion montre que le nombre de clusters optimal est 6 (décroissance régulière après une décroissance rapide). L'analyse des coefficients silhouette, qui mesurent à la fois l'homogénéité intra-cluster et la séparation inter-clusters,confirme également ce résultat.

A l'aide d'un dendogramme on peut visualiser de manière hérarchique les données. Vue que la première ligne du dendogramme représente l'ensemble des données (+ 1,4 millions de visiteurs), on ne peut que représenter un petit échantillon assez représentatif des données (0.25%).Pour finir, le dendogramme affiché sera un dendogramme tronqué qui montrera les 80 derniers regroupements de clusters grâce au paramètre 'truncate_mode'.




Selon les résultats précédents le nombre de clusters optimal serait de 6. Reprenons alors le regroupement avec 6 classes grâce à l'algorithme KMeans.

Une fois les clusters formés, étudions maintenant leurs caractéristiques, i-e les profils des visiteurs qui les composent. Ajoutons les identifiants des clusters dans le data set afin de mieux visualiser les visiteurs et leurs appartenances.

➔ K-means : 6 clusters

| | visitorid | view | addtocart | transaction | itemid | cluster_id |
|---|---|---|---|---|---|---|
| **0** | 0 | 3.0 | 0.0 | 0.0 | 3 | 0 |
| **1** | 1 | 1.0 | 0.0 | 0.0 | 1 | 0 |
| **2** | 2 | 8.0 | 0.0 | 0.0 | 8 | 0 |
| **3** | 3 | 1.0 | 0.0 | 0.0 | 1 | 0 |
| **4** | 4 | 1.0 | 0.0 | 0.0 | 1 | 0 |

| | visitorid | view | addtocart | transaction | itemid |
|---|---|---|---|---|---|
| **cluster_id** | | | | | |
| **0** | 1404763 | 2430778.0 | 46711.0 | 12650.0 | 2490139 |
| **1** | 21 | 32411.0 | 2909.0 | 2219.0 | 37539 |
| **2** | 1 | 6479.0 | 719.0 | 559.0 | 7757 |
| **3** | 96 | 51200.0 | 3911.0 | 2216.0 | 57327 |
| **4** | 2 | 5927.0 | 790.0 | 635.0 | 7352 |
| **5** | 2697 | 137517.0 | 14292.0 | 4178.0 | 155987 |

Les résultats de ce regroupement montre que les outliers ont fortement influencé la constitution des clusters. A titre d'exemple, un visiteur unique (visitor id 1150086) forme à lui seul un cluster. Afin de rendre plus compacte les clusters on choisira au final de laisser le nombre total à 4.

➔ K-means : 4 clusters

| | visitorid | view | addtocart | transaction | itemid | cluster_id | cluster_id_final |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 3.0 | 0.0 | 0.0 | 3 | 0 | 0 |
| **1** | 1 | 1.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| **2** | 2 | 8.0 | 0.0 | 0.0 | 8 | 0 | 0 |
| **3** | 3 | 1.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| **4** | 4 | 1.0 | 0.0 | 0.0 | 1 | 0 | 0 |

| | visitorid | view | addtocart | transaction | itemid |
|---|---|---|---|---|---|
| **cluster_id_final** | | | | | |
| **0** | 1407429 | 2558802.0 | 60649.0 | 16646.0 | 2636097 |
| **1** | 24 | 36069.0 | 3075.0 | 2370.0 | 41514 |
| **2** | 3 | 12406.0 | 1509.0 | 1194.0 | 15109 |
| **3** | 124 | 57035.0 | 4099.0 | 2247.0 | 63381 |

On remarque que le permier cluster (cluster id=0) réunit 99.9% des visiteurs du data set, tandis que les autres comptent moins de 200 visiteurs tous réunis. Il faut donc privilégier une approche relative afin de comprendre les comportements de ces derniers entre les groupes. L'analyse par les ratios ci-dessous montre clairement les disparités entre les clusters créés et permet de dégager effectivement 4 groupes avec des comportements distincts. Etant donné que l'acte d'achat est prioritaire on se propose de nommer les groupes ainsi :

Platinum : cluster 2 Gold : cluster 1 Silver : cluster 3 Bronze : cluster 0

## Pourcentage des visiteurs par cluster :

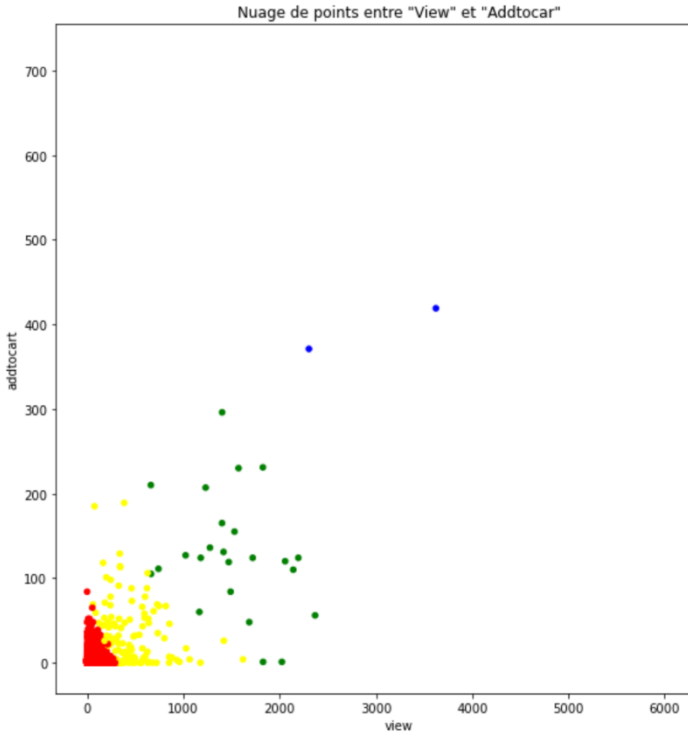
```
Visteurs cluster 0 en % : 99.989272368178
Visteurs cluster 1 en % : 0.0017050540644226262
Visteurs cluster 2 en % : 0.00021313175805282828
Visteurs cluster 3 en % : 0.008809445999516902
```

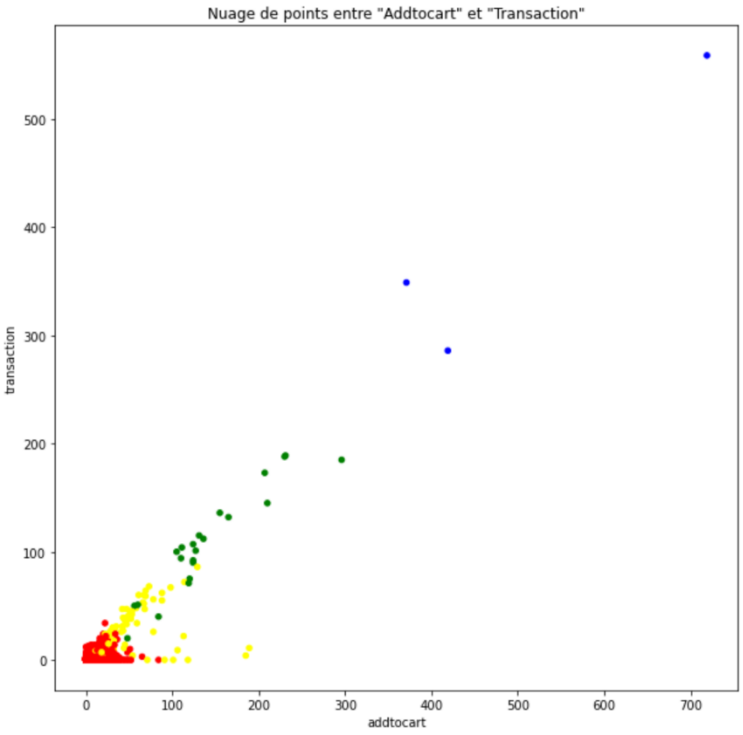
## Ratio des events par cluster :

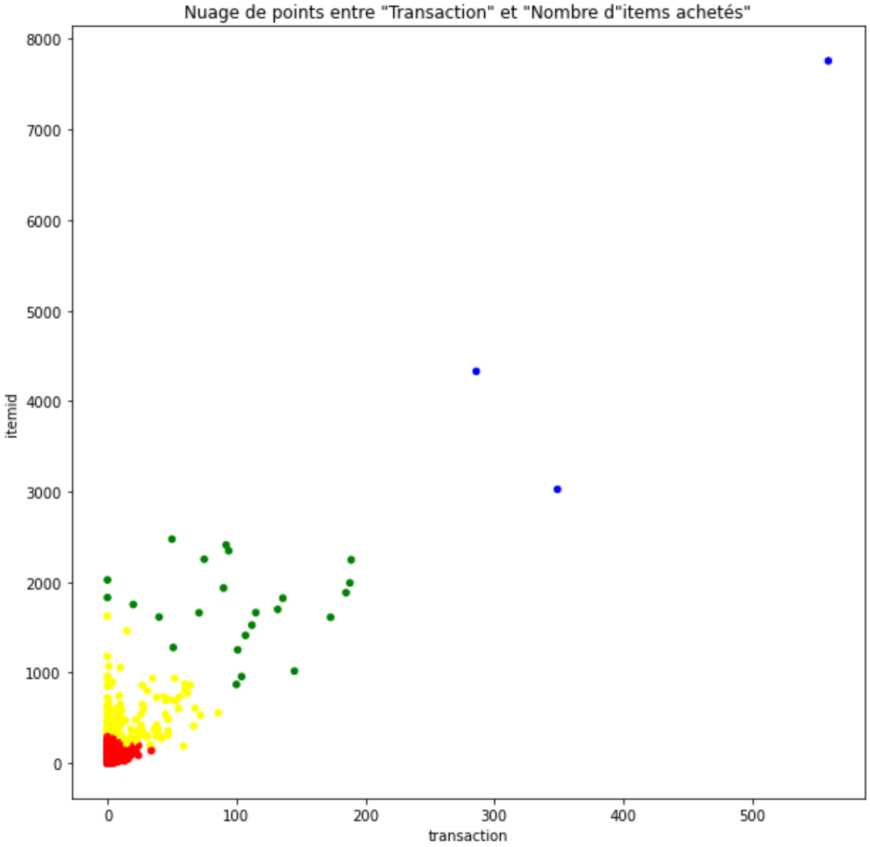
```
Ratio "views par visiteur" pour le cluster 0 : 1.818068264900041
Ratio "views par visiteur" pour le cluster 1 : 1502.875
Ratio "views par visiteur" pour le cluster 2 : 4135.333333333333
Ratio "views par visiteur" pour le cluster 3 : 459.9596774193548


Ratio "addtocart par visiteur" pour le cluster 0 : 0.04309204940355783
Ratio "addtocart par visiteur" pour le cluster 1 : 128.125
Ratio "addtocart par visiteur" pour le cluster 2 : 503.0
Ratio "addtocart par visiteur" pour le cluster 3 : 33.056451612903224


Ratio "transaction par visiteur" pour le cluster 0 : 0.011827239597876696
Ratio "transaction par visiteur" pour le cluster 1 : 98.75
Ratio "transaction par visiteur" pour le cluster 2 : 398.0
Ratio "transaction par visiteur" pour le cluster 3 : 18.120967741935484


Ratio "nb. articles par visiteur" pour le cluster 0 : 1.8729875539014758
Ratio "nb. articles par visiteur" pour le cluster 1 : 1729.75
Ratio "nb. articles par visiteur" pour le cluster 2 : 5036.333333333333
Ratio "nb. articles par visiteur" pour le cluster 3 : 511.13709677419354
```

La classe Platinum (cluster 2) est constitué de 3 outliers qui ont été précédemment identifiés lors de l'analyse descriptive des données.

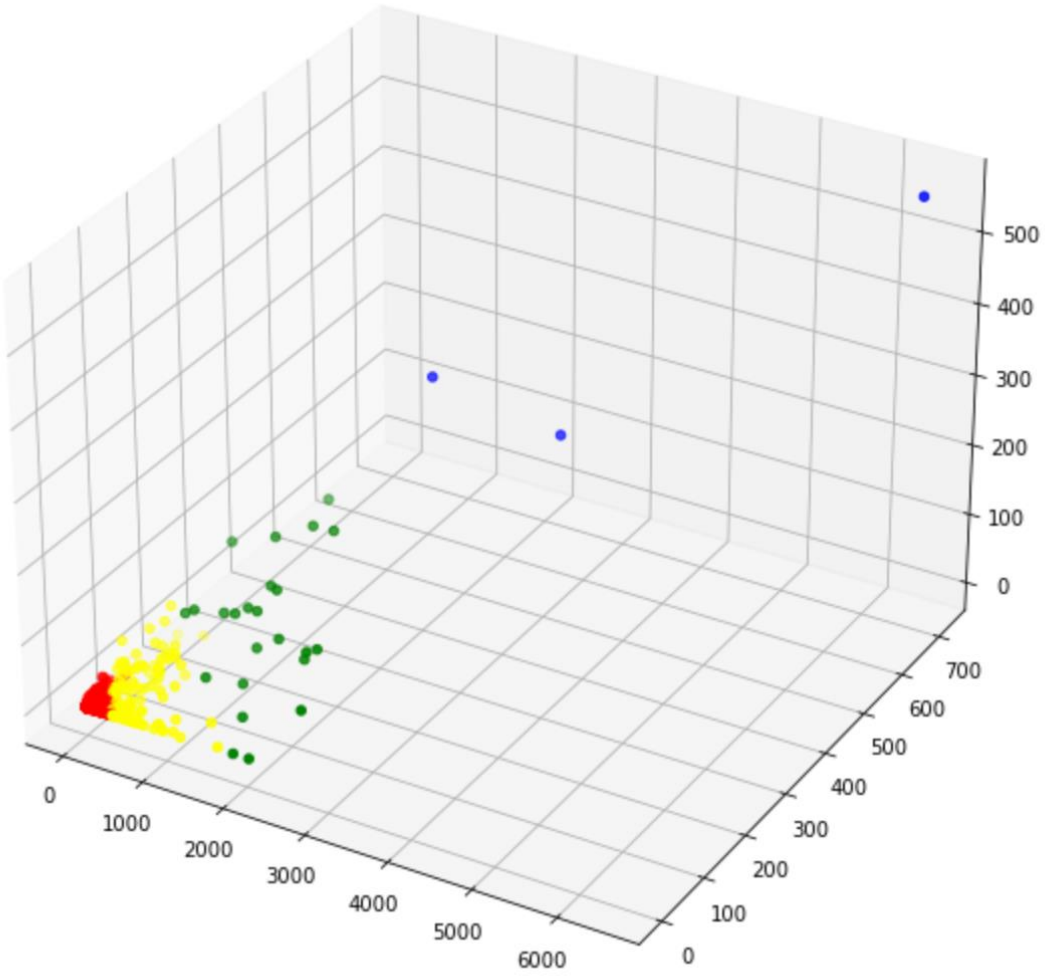
Passons à l'étape de visualisation des clusterings



Visualisation 3D des 3 profils visiteurs : view, addtocart et transaction :



# 4. Modélisation

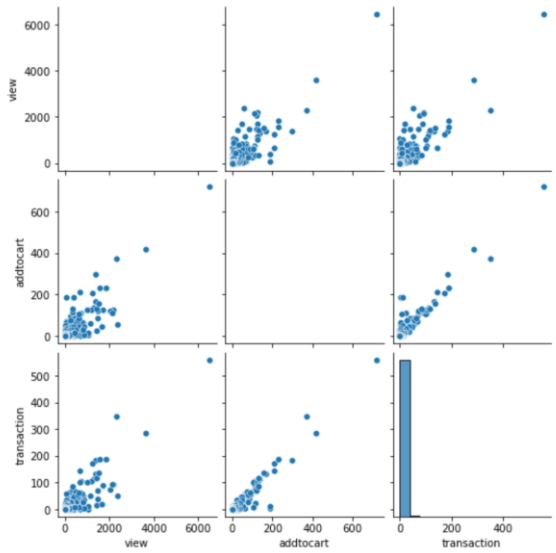
Après le regroupement des visiteurs en quatre clusters par un algorithme d'apprentissage non supervisé, on s'intéresse maintenant à prédire le nombre de transactions effectuées par un visiteur en fonction du nombre de vues et du nombre de mises au panier à l'aide d'un modèle de régression linéaire.

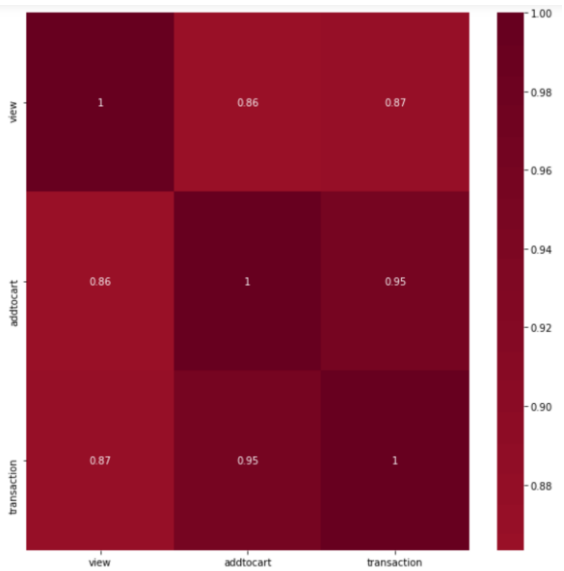
➔ Etapes préparatoires à la modélisation :

- La variable 'event' est dichotomisée afin de faire ressortir chaque type d'action puis est ajouté au dataframe transac_df
- Les variables d'interêt sont ensuite regroupées dans un nouveau dataset appelé transac_df_summary
- On ne garde que les visiteurs qui ont effectué au moins une transaction sur le site au cours de la période.

| visitorid | view | addtocart | transaction |
|---|---|---|---|
| **172** | 33.0 | 3.0 | 2.0 |
| **186** | 2.0 | 1.0 | 1.0 |
| **264** | 3.0 | 2.0 | 2.0 |
| **419** | 4.0 | 1.0 | 1.0 |
| **539** | 4.0 | 2.0 | 1.0 |

La prédiction par la régression linéaire suppose à priori une relation linéaire entre les variables et une certaine corrélation entre elles. Le graphique pairplot ci-dessous confirme l'hypothèse de linéarité entre la variable d'intérêt (nombre de transactions) et les autres variables explicatives (nombre de vues/mises au panier). Elles sont également très corrélées avec la variable indépendante (coefficient de corrélation très proche de l'unité). Par ailleurs, on détecte une forte multicolinéarité entre les features qui risque de conduire à un sur-apprentissage du modèle.




- On normalise les données avec la méthode Min-Max
- On stocke la variable 'transaction' de data dans target et le reste des variables dans features. On sépare ensuite les données en un ensemble d'apprentissage et un ensemble de test contenant 20% des données. La reproductibilité de l'aléatoire est fixé à 150.
- Ajustement du modèle de régression

Le modèle de régression linéaire estimé s'écrit : transaction = -0.001258 + 0.221049*view* + *0.722688*addtocart. Le nombre de vues et plus encore le nombre de mises au panier augmentent positivement la probabilité de faire une transaction sur le site.

| | Coefficient estimé |
|---|---|
| **intercept** | -0.001258 |
| **view** | 0.221049 |
| **addtocart** | 0.722688 |

➔ Faisons maintenant le diagnostic du modèle au travers de différents tests : ajustement du modèle (coefficient de détermination), analyse des résidus,...)

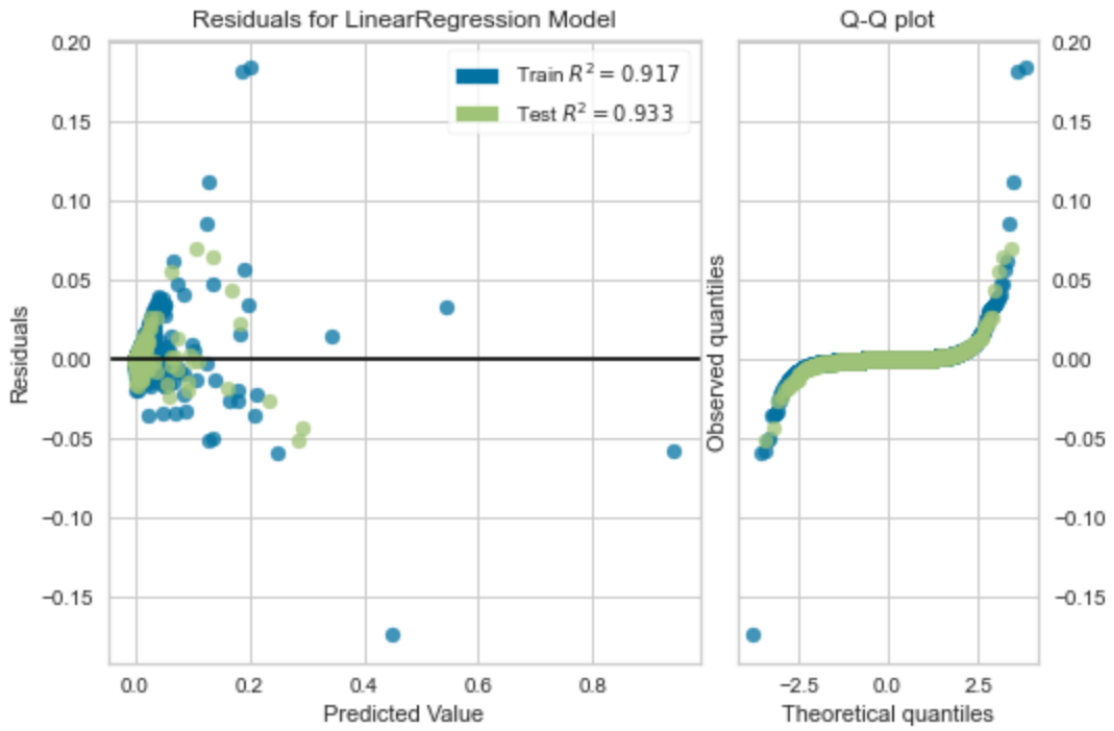
Les coefficients de détermination obtenus sur les deux échantillons sont très bons et très proches. Le modèle est bien ajusté aux données. Toutefois, le R2 obtenu par validation croisée est plus faible laissant présager d'un problème over-fitting :

```
R2 du modèle-échantillon d'entrainement : 0.9169061520023433
R2 obtenu par Cv : 0.7217781731544015
R2 du modèle-échantillon test : 0.9331201785673228
```

Analyse des résidus du modèle de régression :

```
MSE échantillon-entrainement : 2.1955758096655935e-05
MSE échantillon-test: 1.3426653616593717e-05
```

L'analyse des résidus révèle qu'ils ne sont pas parfaitement disséminés autour de la droite d'équation y=0 que ce soit pour l'échantillon d'entrainement ou de test.
Ils sont donc à priori hétérocédastiques.
De plus, le diagramme Quantile-Quantile (QQ-Plot) indique clairement que leur distribution ne suit pas une loi normale car ils ne sont pas alignés sur la première bissectrice.



➔ Une autre méthode pour la régression linéaire avec la bibliothèque statsmodels qui donne des résultats très proches de ceux obtenus avec sklearn

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            transaction   R-squared:                       0.920
Model:                            OLS   Adj. R-squared:                  0.920
Method:                 Least Squares   F-statistic:                 6.744e+04
Date:                Thu, 25 Feb 2021   Prob (F-statistic):               0.00
Time:                        16:09:14   Log-Likelihood:                 46739.
No. Observations:               11719   AIC:                         -9.347e+04
Df Residuals:                   11716   BIC:                         -9.345e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0013   4.23e-05    -30.133      0.000      -0.001      -0.001
view           0.1764      0.005     35.620      0.000       0.167       0.186
addtocart      0.7658      0.005    153.693      0.000       0.756       0.776
==============================================================================
Omnibus:                    21253.271   Durbin-Watson:                   2.029
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        332122593.704
Skew:                         -12.265   Prob(JB):                         0.00
Kurtosis:                     827.361   Cond. No.                         164.
==============================================================================
```

Conclusion : le modèle est certes bien ajusté aux données (R2 > 90 %), mais il ne possède pas de bonnes propriétés statistiques (résidus hétérocédastiques et non normaux).
De plus, il présente des signes de sur-apprentissage probablement dus à la multicolinéarité entre les variables explicatives.

➔ Grâce à la régression régularisée, on peut corriger cet effet de sur-apprentissage dans le modèle. On se propose de tester trois algorithmes (la régression ridge, la régression Lasso et la régression ElasticNet) sur des données qui auront été préalablement standardisées.

- On commence par normaliser les variables en utilisant la méthode StandardScaler de la classe sklearn.preprocessing
- On isole la variable cible (target2) et les variables explicatives (features2) on scinde les données en deux échantillons. L'échantillon test représente 20% de l'ensemble des observations; la reproductibilité de l'aléatoire est fixé à 280.

------ Régression Ridge -----

Nous allons utiliser la classe RidgeCV qui par validation croisée permettra de trouver le meilleur modèle prédictif. Différentes valeurs de alpha (coefficient de pénalité) seront testées pour trouver celle qui minimise le critère d'erreur.

```
RidgeCV(alphas=array([1.0e-03, 5.0e-02, 1.0e-02, 1.0e-01, 3.0e-01, 5.0e-01, 7.0e-01,
       1.0e+00, 5.0e+00, 1.0e+01, 1.5e+01, 3.0e+01, 5.0e+01]))
```

| | Coefficient estimé |
|---|---|
| **intercept** | -0.001303 |
| **view** | 0.186705 |
| **addtocart** | 0.768730 |

```
Valeur optimal de alpha : 0.001
Score échantillon-entrainement : 0.9343477476135365
Score obtenu par Cv : 0.8202125801817377
Score échantillon-test : 0.87692386259216
```

La valeur optimale de alpha qui minimise le critère d'erreur est de 0.001. Le modèle affiche de très bonnes performances à la fois sur l'échantillon d'entrainement que sur l'échantillon test. Le score sur l'échantillon d'entrainement obtenu avec la régression rigde s'est légèrement améliorée par rapport à la régression linéaire, mais sur l'échantillon test il a régressé (87.7% contre 93.3% auparavant).

Analysons l'erreur quadratique moyenne de prédiction sur les deux échantillons :

```
MSE échantillon-entrainement : 0.06078056910110858
MSE échantillon-test: 0.15956248989003244
```

Conclusion : L'erreur quadratique moyenne de prédiction sur l'échantillon d'entrainement est deux fois moindre que sur l'échantillon test. Avec son coefficient de détermination qui est aussi plus élevé, on peut déduire que le sur-aprentissage persiste sur l'échantillon d'entrainement.

------ Régression Lasso -----

Déterminons la valeur optimale de alpha par validation croisée à partir de la classe LassoCV

```
LassoCV(alphas=[0.001, 0.05, 0.01, 0.1, 0.3, 0.5, 0.7, 1, 5, 10, 15, 30, 50],
        cv=10)
```

| | Coefficient estimé |
|---|---|
| **intercept** | -0.001307 |
| **view** | 0.186613 |
| **addtocart** | 0.767782 |

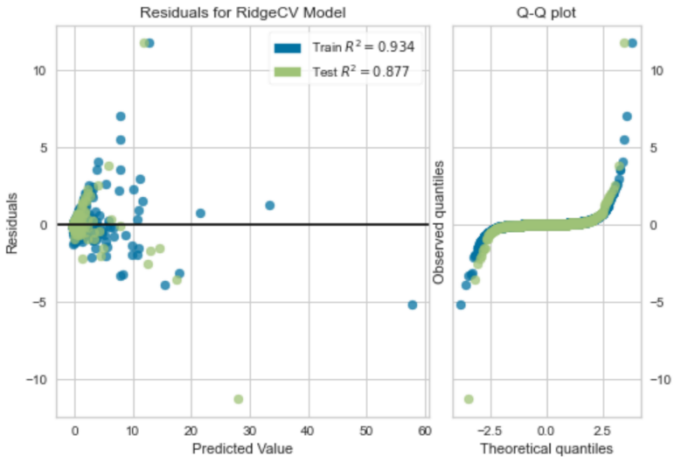
```
Valeur optimale de alpha : 0.001
Score échantillon-entrainement : 0.9343466286938873
Score obtenu par Cv : 0.820596761230075
Score échantillon-test : 0.8767269912766635
```

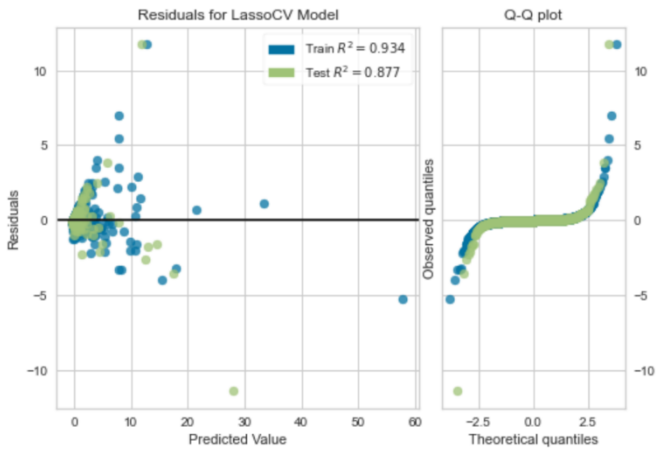
La valeur optimale de alpha qui minimise le critère d'erreur est de 0.001.
La régression Lasso conduit à un modèle avec les mêmes performances que celui obtenu avec la régression Ridge : un score légèrement plus élevé avec l'échantillon d'entrainement qu'avec l'échantillon test.

L'erreur quadratique moyenne de prédiction sur l'échantillon d'entrainement est aussi deux plus élevé que sur l'échantillon test.

```
MSE échantillon-entrainement : 0.060781604992042666
MSE échantillon-test: 0.15981772439730701
```

Conclusion : Les propriétés statistiques des résidus ne se sont pas améliorées avec les deux modèles.



------ Régression ElasticNet -----

Les méthodes de régression Ridge et Lasso n'ont pas permis pour l'instant de trouver le meilleur modèle. Nous nous proposons de tester la méthode ElasticNet qui, par construction, combine les coefficients de pénalités des régressions précédentes.

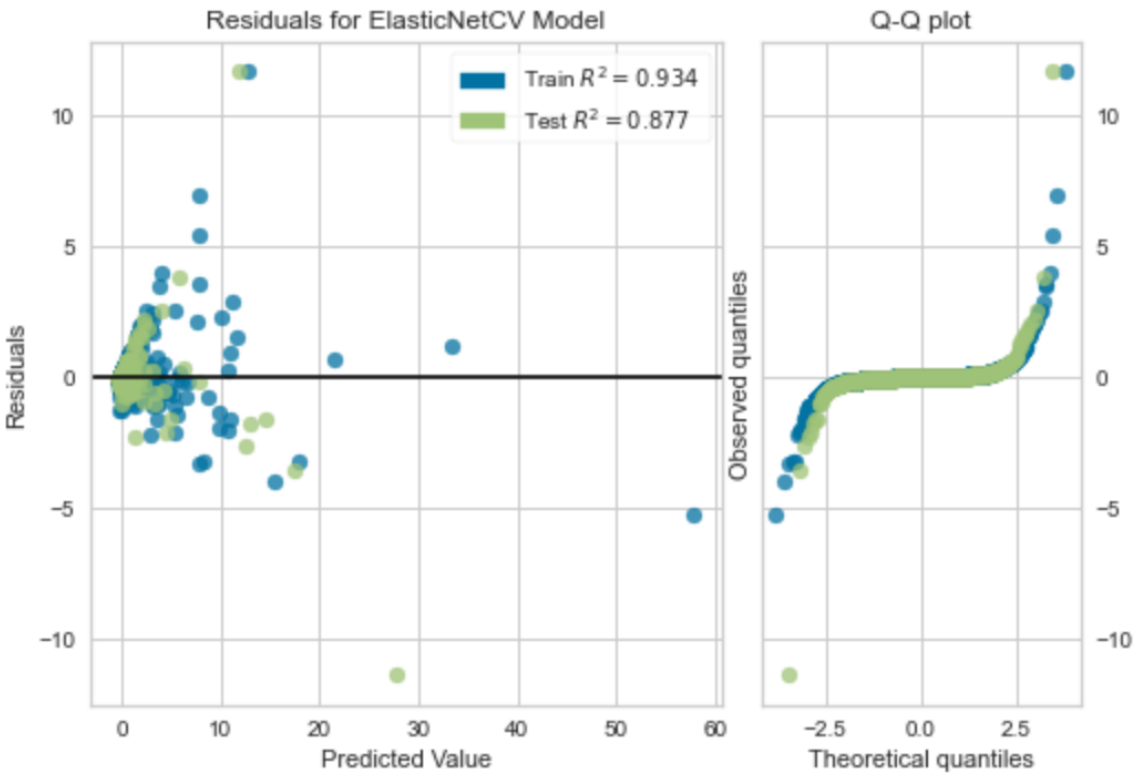
```
ElasticNetCV(alphas=(0.001, 0.05, 0.01, 0.1, 0.3, 0.5, 0.7, 1, 5, 10, 15, 30,
                     50),
             cv=12,
             l1_ratio=(0.001, 0.05, 0.01, 0.1, 0.3, 0.5, 0.7, 1, 5, 10, 15, 30,
                       50, 70))
```

| | Coefficient estimé |
|---|---|
| **intercept** | -0.001315 |
| **view** | 0.188946 |
| **addtocart** | 0.765867 |

```
Valeur optimale de alpha : 0.001
Score échantillon-entrainement : 0.934345832469382
Score obtenu par Cv : 0.8204749333252874
Score échantillon-test : 0.8767333929422961
```

Les performances du modèle prédictif sont également identiques.

```
MSE échantillon-entrainement : 0.06078234213321921
MSE échantillon-test: 0.15980942493545122
```




Conclusion : La régression Ridge a sensiblement amélioré le modèle par rapport aux modèles. Elle a dans un premier temps renforcé l'influence de la variable 'addtocart' comme facteur explicatif principal des transactions. Elle a augmenté sensiblement la performance du modèle sur l'échantillon test qui est passé de 87.2% à 87.7%. L'erreur de prédiction a chuté considérablement pour les deux échantillons quoique l'écart entre les deux persiste.

➔ Synthèse des résultats obtenus :

- Les coefficients des paramètres estimés :

| | Régression linéaire | Régression Ridge | Régression Lasso | Régression ElasticNet |
|---|---|---|---|---|
| **intercept** | -0.001258 | -0.001303 | -0.001307 | -0.001315 |
| **view** | 0.221049 | 0.186705 | 0.186613 | 0.188946 |
| **addtocart** | 0.722688 | 0.768730 | 0.767782 | 0.765867 |

- Les performances :

| | Régression linéaire | Régression Ridge | Régression Lasso | Régression ElasticNet |
|---|---|---|---|---|
| **Train** | 0.933838 | 0.934348 | 0.934347 | 0.934346 |
| **C-V** | 0.820445 | 0.820213 | 0.820597 | 0.820475 |
| **Test** | 0.872897 | 0.876924 | 0.876727 | 0.876733 |

- Les erreurs de prédiction :

| | Régression linéaire | Régression Ridge | Régression Lasso | Régression ElasticNet |
|---|---|---|---|---|
| **Train** | 0.926039 | 0.060781 | 0.060782 | 0.060782 |
| **Test** | 1.296775 | 0.159562 | 0.159818 | 0.159809 |

Nous avons également testé le modèle randomforestRegressor mais les résultats n'étaient pas plus performants.