Royaume du Maroc
UNIVERSITÉ MOHAMED V - RABAT

ECOLE NATIONALE SUPÉRIEURE D'INFORMATQUE
ET D'ANALISE DES SYSTÈMES

Rapport de projet Machine Learning
Filière Business Intelligence And Analytics (BI&A)

Analyse Comparative des algorithmes ML/DL dans une tâche de Classification.

Cas d'étude : Prédire si un visiteur d'une boutique en ligne va effectuer un achat, estimation du taux de conversion des visiteurs en clients.

Présenté par :
ABERQI Nezar
ARAB Noureddine
BOUSSAIRI Hamza
MALIKI Ayoub

Encadrant :
Mme. BENBRAHIM Houda

Année universitaire : 2022 - 2023

Remerciements

Louange à ALLAH seul, que ses bénédictions soient sur notre seigneur et maitre Mohamed et sur les siens.

Tout d'abord nous souhaitons témoigner notre immense gratitude et notre profonde reconnaissance à Madame BENBRAHIM Houda pour sa disponibilité et ses efforts considérables tout au long de ce semestre, afin de nous orienter pour mieux cerner notre projet.

Nos remerciements s'adressent aussi à M. DERROUZ Hatim qui nous a transmis son savoir-faire et les méthodes de raisonnement pendant les séances des travaux pratiques.

Nos gratifications sont aussi adressées à tout le corps professoral et à toute l'équipe pédagogique de l'École Nationale Supérieur d'Informatique et d'Analyse des Systèmes de nous avoir passé une meilleure formation qui nous a permis d'exécuter un tel projet de manière juste et efficace.

Résumé

Le présent rapport constitue une synthèse de notre projet de Machine Learning et Deep Learning ayant comme objectif la prédiction de la conversion visiteur-client pour une boutique en ligne.

En effet, lorque la conversion visiteur-client pour une boutique en ligne est généralement faible, cela signifie que la plupart des visiteurs consultant la boutique n'effectuent pas un achat et donc ne génèrent aucun revenu.

Notre projet est donc conçu pour implémenter et exploiter la puissance de l'apprentissage automatique pour élaborer une solution capable de révéler des tendances que nous pouvons ensuite utiliser pour atteindre notre objectif qui est l'augmentation des ventes

Abstract

This report is a summary of our Machine Learning and Deep Learning project aimed at predicting visitor to customer conversion for an online store.

In fact, when the visitor-to-customer conversion for an online store is usually low this means that most visitors who visit the site don't make any purchase hence not generating any revenue .

Our project is therefore designed to leverage the power of machine learning in order to build a solution capable of revealing trends that we can then use to achieve our goal of increasing sales.

Table des matières

Introduction Générale

Dans ce projet, nous nous intéressons à la problématique de la prédiction de la conversion d'un visiteur à un client pour les boutiques en ligne en utilisant des techniques de Machine Learning et Deep Learning. La conversion visiteur-client est un KPI très important pour les sites E-commerce, car elle mesure le taux de visiteurs qui effectuent un achat sur le site. La possibilité de prédire ce taux de conversion va donc nous être très utile pour pouvoir estimer les revenus et mieux gérer l'entreprise.

Nous utilisons donc des techniques de machine Learning et du Deep Learning telles que les arbres de décision, SVM et les réseaux de neurones artificiels pour prédire cette conversion sans oublier les différentes techniques de clustering comme K-Means, DBSCAN et le Clustering hiérarchique pour segmenter les visiteurs et révéler certaines tendances.

Notre but est donc de développer un modèle de prévision de l'achat des produits d'une boutique en ligne afin d'élaborer des campagnes publicitaires rentables et essayer d'optimiser l'expérience du visiteur tout au long de sa consultation du site (User-experience). Les algorithmes de Machine Learning sont testés pour comprendre lesquels sont les plus performants, ces modèles disposent bien évidemment d'hyperparamètres qu'on doit ajuster par des techniques spécifiques afin d'assurer une performance optimale.

Les résultats de nos expériences sont intéressants non seulement d'un point de vue **Machine Learning** mais aussi d'un point de vue **Business Intellingence**, notre finalité étant non seulement de développer des modèles de performances élevées mais aussi de savoir bien interpréter ces modèles pour résoudre cette problématique et savoir les exploiter afin d'aider les dirigeants dans leurs décisions pour assurer la prospération des revenus .

Chapitre 1

Travaux connexes et état de l'art

1.1 Problématique

La problématique de ce projet est de développer une solution efficace pour prédire la conversion visiteur-client pour une boutique en ligne, cette dernière attire un grand nombre de visiteurs chaque jour, mais seulement une petite proportion de ces visiteurs finissent par effectuer un achat. Le défi est donc de développer un modèle de prédiction qui peut prévoir avec précision les visiteurs qui sont les plus susceptibles de devenir des clients.

1.2 Méthodes utilisées pour résoudre le problème

Il existe différentes techniques qui ont été mises en œuvre pour résoudre notre Problème.

L'une des méthodes couramment utilisées est l'utilisation de modèles statistiques de prédiction tels que les régressions logistiques et les réseaux de neurones. Ces modèles utilisent des données historiques sur les visiteurs et les clients pour prédire les conversions futures. Les caractéristiques telles que les habitudes de navigation, les habitudes d'achat et les caractéristiques démographiques des visiteurs sont souvent utilisées pour entraîner les modèles.

Les algorithmes d'apprentissage automatique ont également été utilisés pour identifier les caractéristiques des visiteurs ayant le plus de chances de devenir des clients, afin de cibler ces visiteurs avec des campagnes marketing ciblées.

D'autres méthodes utilisent des techniques de scoring pour affecter une note à chaque visiteur en fonction de sa probabilité de convertir, ciblant ainsi les visiteurs ayant les notes les plus élevées. Il est également possible de combiner plusieurs de ces méthodes pour améliorer la précision des prévisions.

1.3 Approche proposée

On a décider d'aborder une autre méthode. En effet, notre approche consiste à considérer les contextes dans lesquels les achats ont été effectués, tels que la localisation des clients, le moment de l'achat et les pages visitées. En utilisant ces informations, il est possible de déterminer les facteurs qui ont conduit à l'achat et d'utiliser ces informations pour cibler les visiteurs similaires.

Il est également important de considérer les clients existants et les nouveaux visiteurs séparément, car ils peuvent avoir des comportements d'achat différents. En utilisant ces informations, il serait possible d'ajuster les campagnes marketing pour maximiser les conversions.

1.4 Différence entre la méthode abordée et les autres méthodes

La différence principale entre notre méthode et les autres méthodes pour résoudre le problème de prédiction de la conversion est le fait que notre méthode se concentre sur les clients existants et les visiteurs similaires plutôt que sur l'ensemble des visiteurs. En utilisant les informations sur les clients existants qui ont acheté des produits auparavant, il est possible de découvrir des caractéristiques similaires chez les visiteurs qui ont visité le site mais qui n'ont pas acheté de produits. Il est également intéressant de se concentrer sur les visiteurs qui se sont arrêtés juste avant de faire un achat, car ils sont les plus proches de devenir des clients.

En somme, notre méthode se concentre sur une segmentation plus fine des visiteurs de la boutique en ligne et sur les contextes dans lesquels les achats ont été effectués, et utilise ces informations pour cibler efficacement les visiteurs les plus susceptibles de devenir des clients.

Chapitre 2

Données d'expérimentation

2.1 Analyse globale

2.1.1 Vue d'ensemble

L'ensemble de données qu'on va exploiter dans notre projet est un dataset qu'un étudiant de l'Université de **Waterloo** a pu récupérer lors d'un hackaton, le lien de son repository est mentionné en annexe. On est satisfait de la qualité, la quantité de données fournies et la varitété des attributs. Grâce à ces données, on pourra entamer notre analyse et commencer à développer nos modèles.

Les consignes du défi étaient comme suit : ***"Le propriétaire d'une boutique en ligne souhaite augmenter ses revenus en effectuant des prévisions précises sur l'achat ou non des produits des visiteurs. IL vous a embauché, un BI analyst formé à Waterloo, pour résoudre cette tâche."***
Analytiquement parlant, c'est une tâche que nous jugons loin d'être facile, en effet, on peut se poser tant de questions qui qui se focalisent sur la probabilté de retour d'un client, les similarités entre les clients qui ne génèrent pas de revenu, la variation du taux de conversion entre les nouveaux visiteurs et les visiteurs réguliers etc..

2.1.2 Caractéristiques et défis des données :

Notre échantillion de données consiste en 8622 entrées, la variable à prédire 'Revenu' qu'on a décidé de renommer par '**HasBought**' vaut 1 pour les visiteurs effectuant un achat et 0 sinon.

— **Données déséquilibrées :** Ce jeu de données est très déséquilbré, seulement ***1335 (15.5%)*** des visiteurs se sont convertis en clients, chose qui peut altérer la qualité des algorithmes qui vont se concentrer sur la classe majoritaire. Nous devons donc penser à palier à ce problème et choisir la bonne métrique d'évaluation des modèles.

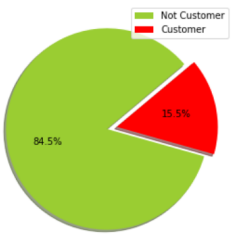


Figure 2.1 – Répartition des visiteurs de la boutique.

— **Les attributs de notre dataset**

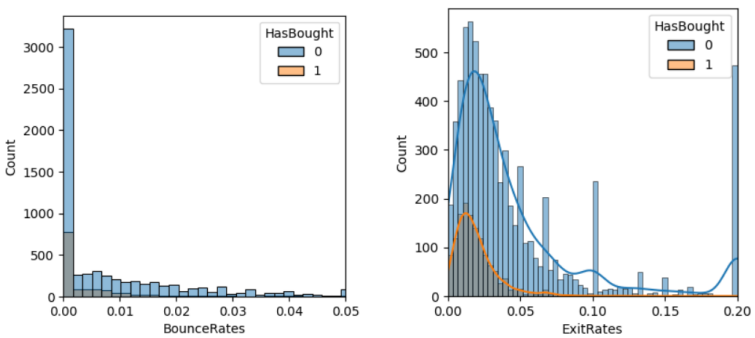
ID	identifier le visiteur
Administrative	nombre des pages administratives (page login, mon compte, gestion des transaction ...) consultées par les visiteurs.
Administrative_Duration	Durée totale de visite des pages administratives par le visiteur.
Informational	nombre des pages informationnelles (page d'accueil, blogs, articles..) consultés par le visiteur.
Informational Duration	Durée totale de visite des pages informationnelles par le visiteur.
Product Related	nombre de pages contenant les produits de la boutique consultés par le visiteur.
Product Related Duration	Durée totale de visite des pages Product Related par le visiteur.
Exit Rate	proportion des visiteurs ayant quitté la boutique après avoir consulté une page, ce nombre va donc représenter la moyenne des exit rate des pages vues au cours de la session d'un visiteur.
Bounce Rate	proportion des visiteurs ayant quitté la boutique après avoir consulté une seule page uniquement, ce nombre va donc refléter la moyenne des bounce rate des pages vues au cours de la session d'un visiteur.
Page Values	Moyenne de la valeur monétaire des pages consultées par un visiteur dans une session.
Special Day	Proximité du temps de visite du site à d'un jour férié (spécial).
Month	Mois de la viste
Operating Systems	Système d'exploitation que possède le visiteur..
Browser	Navigateur Web du visiteur.
Région	location du visiteur
Traffic Type	type du trafic internet par lequel le visiteur a accédé au site de la boutique.
Visitor Type	nouveau visiteur / client déjà existant.
Weekend	La visité effectuée lors d'un Weekend ou pas.
HasBought	Variable à prédire : est ce que la visite à générer du revenu ou pas (1 = Oui , 0 = Non).

TABLE 2.1 – Attributs

— **Exploration des données :**

Au cours de notre analyse exploratoire, on a pu extraire plusieurs informations nous permettant de mieux comprendre les données. On peut résumér les résulats dans ce qui suit (**une analyse détaillée est fournie dans notre notebook**) :

⋆ Relation entre les métriques Exit Rate/Bounce Rate et l'achat des produits :



On remarque que les métriques Bounce Rate et Exit Rates ont une corrélation négative avec la conversion, pour un bounce rate au delà de 0.01, la possibilité d'achat devient presque nulle. Ces deux métriques sont faibles pour les personnes effectuant un achat que par rapport aux autres.

⋆ Relation entre les valeurs monétaires des pages visités et le revenu.

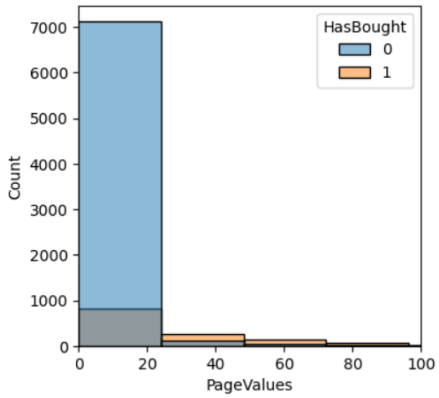


FIGURE 2.2 – Distribution de l'indice PageValues en fonction des classes.

On remarque qu'une session d'un visiteur qui dépasse le 20% en termes de valeurs moyenne des pages conduit à un taux d'achat élevé. Il existe donc une forte corrélation positive. Ceci suggère donc qu'avec un design (UI/UX) assez bon, la boutique réalisera plus de profit.

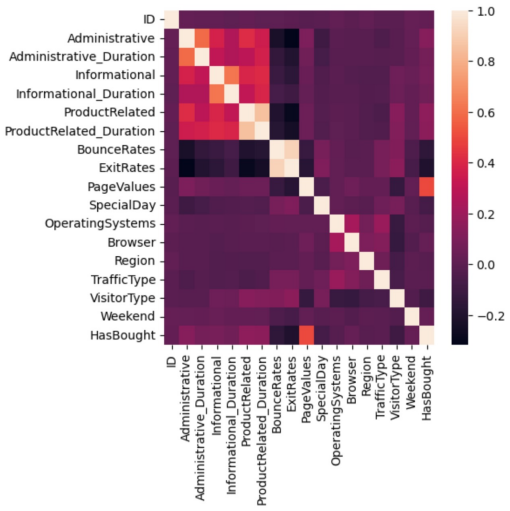
⋆ Effet des weekends et des jours fériés sur le revenu :

HasBought	0	1	conversion_rate
Weekend			
0	5623	983	0.148804
1	1664	352	0.174603

HasBought	0	1	conversion_rate
SpecialDay			
0.000000	6466	1275	0.164707
0.200000	117	12	0.093023
0.400000	153	12	0.072727
0.600000	230	21	0.083665
0.800000	221	7	0.030702
1.000000	100	8	0.074074

- En regroupant le dataset par l'attribut Weekend, on remarque que le taux de conversion dans les Weekend **augmente** de presque **4%**.
- Le nombre de visiteurs effectuant un achat est bien supérieur aux jours normaux que les jours proches des jours fériés. On constate donc que probablement la boutique n'offre pas des offres particuliers dans les jours spéciaux (black friday etc ...)

⋆ Matrice de corrélation : Les attributs Page Values, et Exit Rates ont les plus grandes corrélations avec la variable cible en comparaison avec les autres attributs.



Chapitre 3

Algorithmes d'apprentissage automatique

3.1 Preprocessing des données :

1. SMOTE (Synthetic Minority OverSampling Technique) :

   Il s'agit d'une des meilleurs techniques de sur-échantillonnage utilisée pour traiter les données déséquilibrées, comme dans notre cas, elle permet de génèrer des données synthétiques pour la classe minoritaire en utilisant des exemples existants de cette classe en utilisant le KNN. Cela permet d'équilibrer les représantants des classes, ce qui peut améliorer la performance des réduisant le biais de la classe majoritaire.

   ! Il est important de noter que parmi les bonnes pratiques du ML est d'effectuer un SMOTE après avoir effectué un **train_test_split**, car sinon, le modèle sera partiellement testé sur des exemples qu'il a bien vu dansle , la précision sera donc biaisée et ne réflète pas une bonne classification. On va donc effectuer le smote sur le X_train est le tester sur des données de test réelles, déséquilibrées et non synthétisées.

2. Feature Enginnering et Encodage :

   Pour simplifier le problème, réduire la dimension des attributs et optimiser notre tâche, on exploiter les résultats de notre **analyse exploratoire** pour supprimer les variables n'ayant aucune influence sur la conversion des visiteurs, et on a dû regrouper des attributs en une seule variable(Nombre Pages Administratives + Nombres Pages Informationelles = Nombre Pages **nonProductRelated**) On aussi effectué un recodage ordinal des variables nominales.

3. Standardisation des données :
   Pour pouvoir assurer une indépendance envers les unités de mesures, on a dû standardiser nos attributs, le choix du **RobustScaler** est justifiée par le fait que la plupart des attributs ont assez de valeurs abhérentes (d'aprés notre **EDA**), ce transformer est insensible aux outliers puisqu'il utilise la médiane et l'IRQ des variables.

4. Utilisation des chaînes de transformations (Pipelines) :
   Nous allons utiliser dans notre projet les Pipelines, des estimateurs composites dans lesquels on regroupe le RobustScaler et l'algorithme souhaité, pour centraliser le traitement et factoriser le code.

5. Grid Search (Grille des estimateurs) :
   Grid search est une méthode utilisée pour sélectionner les meilleurs hyperparamètres pour un algorithme

d'apprentissage supervisé en testant les différentes combinaisons sur un dataset de training et de validation en utilisant la méthode de **Cross-Validation avec K ensembles (K-Folds)**. En utilisant Grid search, on peut facilement identifier les paramètres optimisant la performance de nos modèles.

3.2 Algorithmes supervisés

1. **K-nearest neighbors** :
   Algorithme de classification par voisinage qui se base sur les k observations les plus proches d'une nouvelle donnée afin de prédire sa classe à partir d'un vote majoritaire.
   Les hyperparamètres optimales du KNN choisis par le GridSearch sont :
   - ⋆ Le nombre de voisins (k) = **9** .
   - ⋆ mesure de distance = **Manhattan**.
   - ⋆ les poids des voisins = **distance**.
2. **Naive Bayes :**
   Naive Bayes est un algorithme de classification probabiliste basé sur le théorème de Bayes. Il n'a pas de paramètres méritant GridSearch, ils sont donc laissées par défauts.
3. **Decision Tree :**
   Les arbres de décision sont un algorithme qui permet de modéliser les processus de décision en examinant l'influance des différents attributs sur le dataset. Les Hyper-paramètres choisies :
   - ⋆ Le nombre minimum d'exemples par feuille (min_samples_leaf) = **29**.
   - ⋆ critère de selection des attributs (criterion) = **entropy**.
   - ⋆ paramètre de pruning (ccp_alpha) = **0.001**. L'arbre de décision est visualisé dans le notebook (en annexe).
4. **Support Vector Machines :**
   SVM est un algorithme qui qui essaie de trouver un hyperplan qui sépare efficacement les données en deux classes.
   On a plotté les deux classes pour visualiser le degrès de séparation linéraire en considérant les deux attributs les plus corrélés avec la variable cible :

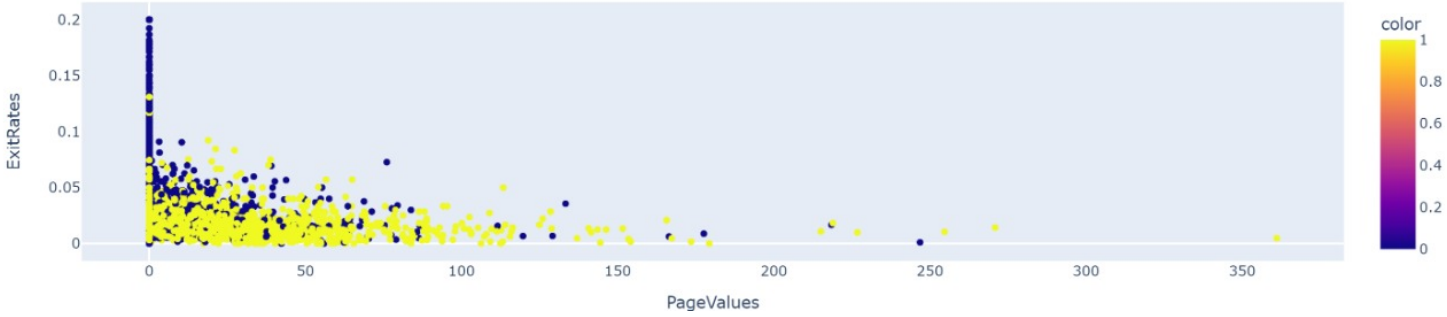


FIGURE 3.1 – Non séparation linéaire des deux classes (Nécessité du Mapping)

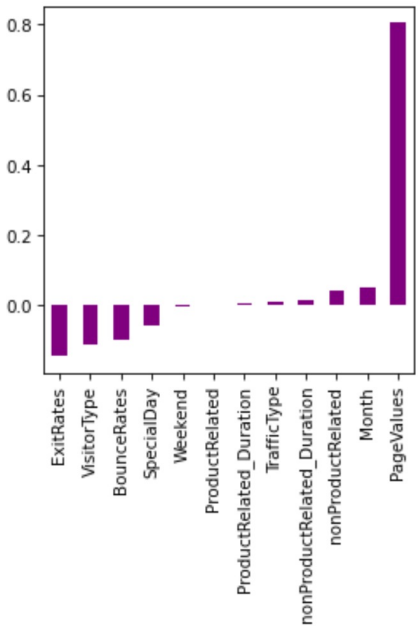
Les hyper-paramètres optimales du SVM sont :

- ⋆ kernel (Type de noyau utilisé pour transformer les données) = **rbf**.
- ⋆ Gamma (définit l'influence de chaque exemple d'entraînement sur la frontière de décision.) = **auto**.
- ⋆ Degree (degrès de la fonction Kernel) = **3**.

3.3 Algorithmes non supervisés

1. **KMeans** :
   K-Means est un algorithme non supervisé utilisé pour regrouper des données en clusters basés sur la similarité. On a dû effectuer des tests en utilisant l'**indice de Dunn** vu en cours et aussi la technique de l'Elbow pour choisir le meilleur K (voir notebook). Les hyperparamètres du K-Means qu'on a choisi sont donc :

k	indice_dunn
2	0.504051
3	0.447351
4	0.446656
5	0.442793
6	0.433363
7	0.420555
8	0.412243
9	0.237633



   A gauche sont représentés les indices de Dunn pour évaluer les performances pour différentes valeurs de K,la figure à droite représente les attributs que le K-means (K = 2) a jugé comme très importants lors du clustering.
   On remarque que l'attribut Page Values est très fortement corrélés avec les étiquettes données par le K-Means. **On peut dire qu'il a donc majoritairement distinguer entre les visiteurs ayant consultés des pages de valeurs monétaires élevée (>20%) et ceux n'ayant pas fait ceci (<20%)**.

2. **DBSCAN** :
   Dbscan est un algorithme de clustering utilisé pour découvrir des structures dans les données en s'appuyant sur la densité de points dans l'espace. Il est basé sur deux paramètres principaux :
   - ⋆ eps (rayon maximal du voisinnage d'un point = **3**,
   - ⋆ min_samples (définit le nombre minimum de points requis de points dans un cluster) = **32**.

   Géneralement on prend min_samples au minimum égal à **2 * dimension des attributs** et eps relativement petit pour s'assurer que le modèle génère **2** clusters. En considérant uniquement les attributs ayant une grande influence sur la variable cible (PageValues et ExitRates) on peut visualiser les clusters produits par notre modèle.

3. **Hierarchical Agglomerative Clustering** :
   Le modèle agglomératif consiste à considérer chaque point comme cluster et regroupe ensuite les clusters les plus proches deux à deux. Une approche comme vu au cours à été étudiée en premier temps dans le

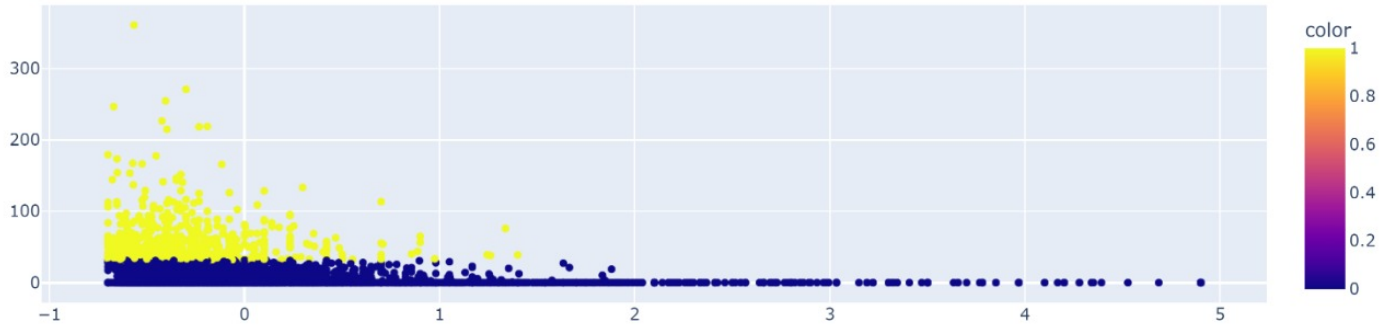


FIGURE 3.2 – Séparation par densité spatiale des points de données en deux clusters (avec détection des outliers.)

notebook, en laissant l'algorithme regrouper les 8622 clusters intiaux jusqu'au cluster global.
Ensuite, comme on souhaite générer deux clusters par ce modèle afin de le tester par la variable cible déjà connu, on a fixé les hyperparamètres aux valeurs suivantes :

- ⋆ n_clusters = **2**, on souhaite former deux clusters uniquement.
- ⋆ affinity (la mesure de distance utilisée) = **euclidean**.
- ⋆ linkage (calcul de la distance inter-clutster) = **ward**.

3.4 Multi-layer Perceptron

Comme les données ne sont pas linéairement séparables, on va utiliser un MLP pour réaliser la tâche de classification.
Déterminer une bonne architecture du réseau consiste à choisir les hyperparamètres les plus convenables pour assurer une bonne adaptation du réseau aux données.
Les couches d'entrées et sorties contiennet respectivement le nombre des attributs et le neuronne déterminant la classe (max_proba).
En utilisant la technique du GridSearch, on a pu extraire les paramètres optimaux comme suit :

- ⋆ Le nombre de couches cachées = **19**.
- ⋆ nombre de neurons par couche cachée = **16**.
- ⋆ solver = **adam** (un optimiseur basé sur la méthode de descente de gradient stochastique).
- ⋆ learning_rate_init (contrôle la taille du pas dans la mise à jour des poids) = **0.001**.
- ⋆ alpha (terme de régularisation L2) = **1e-5**.

Chapitre 4

Résultats expérimentaux et discussion

4.1 Méthodologie d'évaluation

4.1.1 Cas de l'apprentissage supervisé :

Critiques sur la métrique 'Accuracy' :
Comme nous sommes en train de traiter un ensemble de données déséquilibré, il est important de choisir la bonne métrique d'évaluation, en l'occurrence, la précision "accuracy" n'a pas autant de significativité dans notre cas, une fonction simple qui retourne '0' comme classe aura une précision de plus de 80%, alors qu'elle est loin de prédire la classe minoritaire.
La métrique ROC_AUC :
Receiver Operating Characteristic - Area Under the Curve, est en effet la meilleur métrique dans notre cas, car elle regroupe à la fois le "recall" de la classe majoritaire et le 1-recall de la classe minoritaire, et mesure donc la capacité de l'algorithme à séparer les VP et les VN, une valeur élevée signifie un bon modèle. Elle est donc fréquemment utilisée dans les tâches de classification avec un dataset déséquilibré.

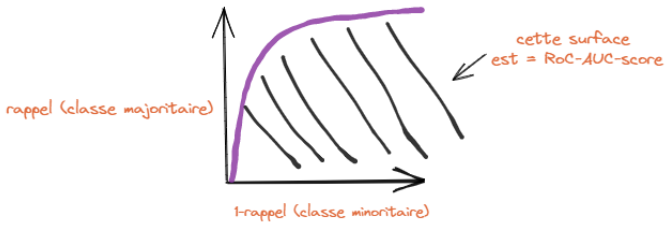


FIGURE 4.1 – Un drawing qu'on a fait à l'aide de excalidraw pour expliquer l'intuition derrière la métrique ROC_AUC.

Ceci dit, on peut comparer la précision "accuracy" des modèles comme moyen d'évaluer leurs performances globales sur les données, mais **non pas comme métrique tranchante et décisive dans la comparaison**. (Voir les Résultats expérimentaux).

4.1.2 Cas de l'apprentissage non supervisé

Critiques sur la métrique 'Accuracy' :
La métrique Accuracy, n'apporte pas, dans le cas du clustering, une significativté importante. Certe, elle permet de réveler si l'algorithme a pu distiguer les deux classes en mettant chacune dans son cluter, mais ceci n'est

réalisable que lorqu'il donne exactement les même étiquettes (labels) aux clusters que les étiquettes des classes.

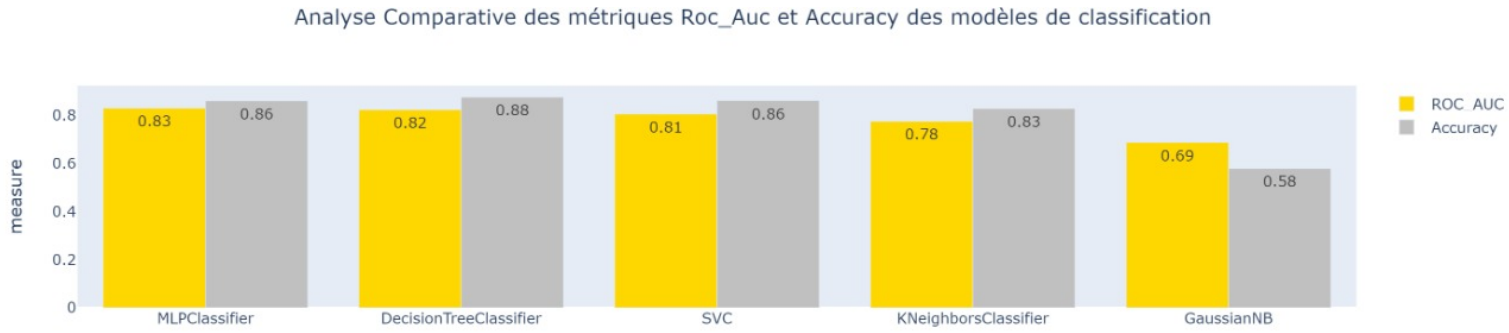
La métrique Rand_Score :
Le Rand Score est une métrique utilisée pour évaluer les performances des algorithmes de clustering. Il mesure la similitude entre les groupes formés par l'algorithme et les groupes réels dans les données. Par conséquent, il permet de réveler la capacité du modèle à minimiser la distance inter-cluster et à maximiser la distance intra-cluster.
Plus ce score est proche de 1, plus la qualité du clustering est bonne.

Ceci dit, on peut comparer la précision "**Accuracy**" des modèles comme moyen d'évaluer leurs performances globales sur les données, mais **non pas comme métrique tranchante et décisive dans la comparaison**. (Voir les Résultats expérimentaux).

4.2 Résultats expérimentaux Discussion

4.2.1 Cas de l'apprentissage supervisé

En testant les différents modèles de classification obtenus dans le chapitre précédent sur des données réelles et déséquilibrées : 2186 exemple de la classe 0 (n'a pas effectué un achat) et 401 exemple de la classe 1 (a effectué un achat). On a pu regroupé l'ensemble des résultats et métriques d'évaluation qu'on a plotté dans la figure ci-dessous.



4.2.2 Discussion et Réflexions :

⋆ Comme prévu, le perceptron à plusieurs couches (MLP) a la meilleure performance, avec un roc_auc le plus élevé, il a pu donc bien classifier la classe minoritaire.

On peut justifier ceci par le fait que les MLP ont généralement une meilleure capacité à modéliser les relations non linéaires, et surtout leur capacité à s'adapter aux données en changeant les poids des liaisons synoptiques. Le choix d'une bonne architecture (nombre de couche, learning rate...) nous a permis de réaliser ces résultats satisfaisants.

⋆ On remarque aussi que le modèle des arbres de décisions a la métrique ROC_AUC très proche de celle du MLP, il a pu donc bien classifier les deux classes.

Ceci est à un certain degré prévu également car on sait que les arbres de décisions sont généralement insensibles aux bruits et à la non linéarité du dataset, de plus, elles sont capables de gérer correctement les

interactions entre les différents attributs, en détectant ceux qui contribuent le plus à l'entropie du système, les hyper-paramètres qu'on a pu optimiser nous ont permis d'éviter les inconvénients majeures de cet algorithme, notamment l'over-fitting.

⋆ A notre surprise, le modèle SVM est le troisième meilleur classifieur, même si les données sont loin d'être linéairement séparables, chose qui affirme que le la fonction kernel a réussi à faire à mapping efficace des données.

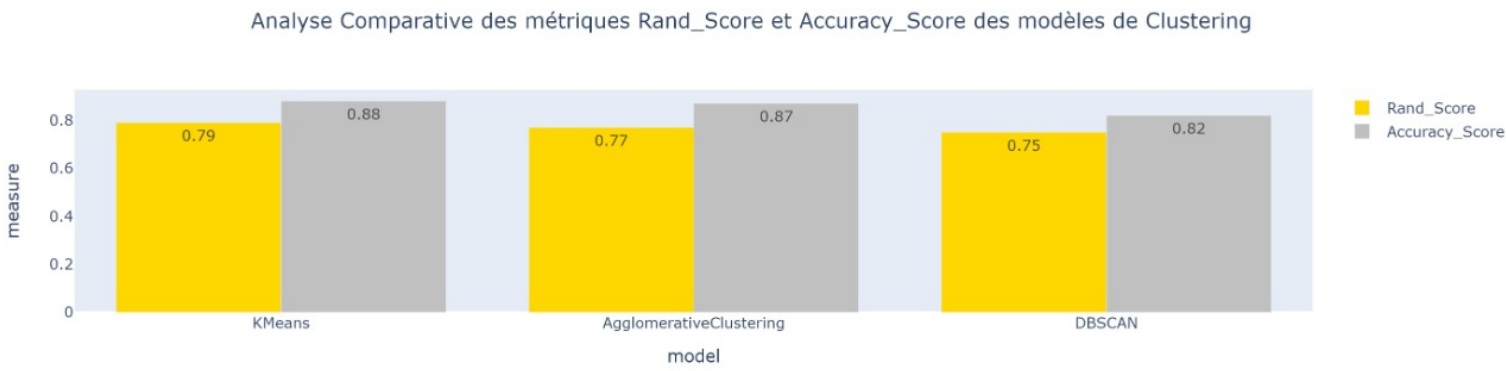
La projection des données dans un espace de dimension supérieure a pu donc garantir une bonne classification des visiteurs et bien prédire leur conversion ou non.

⋆ Quant au KNN, un ROC_AUC de **78%** indique que le modèle est capable de distinguer entre les VP et les VN, mais pas aussi bien que les DT et SVM cités précédemment.

⋆ Vient en dernière place le modèle Naive Bayes, comme vu au cours, c'est le modèle ayant généralement des performances limitées par rapport aux autres modèles, cependant, ce qui nous a attiré l'attention le plus, c'est que le NB a un roc score (**69%**) bien meilleur que la précision (**59%**) , ce qui signifie que sa capacité à distinguer les VP et les VN à été meilleure que sa capacité globale à bien classifier un point de données quelconque, chose qui nous pousse à se demander est ce que ceci revient à l'hypothèse naïve d'indépendance des attributs ou par d'autres facteurs relatifs au dataset (bruit, technique d'oversampling etc...)

4.2.3 Pour les algorithmes d'apprentissage non supervisé

4.2.4 Résultats :



4.2.5 Discussion et Réflexions :

Dans la figure ci-dessus, Nous avons mis en place les indices de Rand$score, ainsiquelesindicesdeprcision$"**Accurary K-means, le modèle hiérarchique et le dbscan.**

⋆ L'indice de Rand mesure la similarité entre les groupes prévus par le modèle et les groupes réels. Il prend en compte les correspondances entre les groupes prévus et réels, indépendamment des tailles de ces groupes, alors que L'indice de Dunn mesure l'intra-cluster et l'intercluster.

⋆ Il est toujours bon d'utiliser plusieurs métriques d'évaluation pour mieux comprendre les performances de l'algorithme de clustering, car il se peut qu'une seule métrique ne soit pas la meilleure pour évaluer ces performances.

⋆ Les résultats ont montré que le modèle K-means a une performance supérieure aux autres modèles en termes de compacité intra-cluster, de séparation inter-cluster et de correspondance avec les groupes réels.

⋆ On remarque que les modèles K-means et hiérarchique ont séparé parfaitement les deux groupes sans avoir aucune valeur aberrante, contrairement au dbscan qui a détecté 644 outliers (voir notebook), un nombre géant qui dépasse même le nombre de valeurs du cluster 1 (83 valeurs), ce qui explique sa mauvaise précision et son indice négatif de Dunn.

⋆ Les deux modèles K-means et hiérarchique sont très proches en termes de performances et la petite différence entre eux peut être expliquée par l'extensibilité de K-means envers la taille des données et sa sensibilité aux conditions initiales.

Conclusion Générale

En conclusion, ce projet a permis de développer des modèles prédictifs qui peuvent prédire avec une grande précision si un visiteur d'un site e-commerce deviendra ou non un client. Ce modèle peut être utilisé pour cibler les compagnies de marketing et maximiser les conversions en identifiant les visiteurs les plus susceptibles de devenir des clients.

En outre, il est important de mentionner les limites et les défis de ce projet. Il est nécessaire de disposer d'une quantité suffisante de données pour entraîner et valider le modèle, et il est également important de s'assurer que les données sont suffisamment représentatives de la population cible pour éviter les biais dans les résultats, ainsi de s'assurer que les données sont suffisamment propres et préparées avant de les utiliser pour entraîner le modèle.

Ce projet démontre l'importance croissante de l'intelligence artificielle et de l'apprentissage automatique dans les entreprises modernes. Les techniques de Business Intelligence basées sur l'IA peuvent aider les entreprises à comprendre les comportements de leurs clients et à prendre des décisions plus efficaces pour améliorer les résultats. Nous prévoyons de continuer à explorer et à utiliser ces techniques pour améliorer nos activités commerciales à l'avenir.

annexe

January 16, 2023

1 Analyse Comparative des algorithmes ML/DL dans une tâche de classification

1.0.1 Cas d'étude : Prédire le comportement des visiteurs d'un site de l'e-commerce pour estimer le taux de conversion des clients

- Importation des librairies nécessaires :

[497]:
```
import pandas as pd
import numpy  as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import os
import pandas as pd
from sklearn.model_selection import train_test_split␣
 ↪,GridSearchCV,StratifiedKFold
from sklearn.metrics import ConfusionMatrixDisplay,classification_report
from sklearn.metrics import f1_score , roc_auc_score, accuracy_score,␣
 ↪precision_score, recall_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from imblearn.over_sampling import SMOTE
from sklearn.naive_bayes import GaussianNB
from sklearn import svm
from imblearn.pipeline import Pipeline as imbpipeline
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler,StandardScaler,RobustScaler
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn.cluster import AgglomerativeClustering
from sklearn.neural_network import MLPClassifier
from  scipy.cluster.hierarchy import dendrogram
from scipy.cluster import hierarchy
from sklearn.metrics import rand_score
import plotly.graph_objs as go
import plotly.offline as pyo
```

- Importation de données

[482]:
```
df = pd.read_csv('input_training_data.csv')
df.rename(columns={'Revenue': 'HasBought'},inplace=True)
```

[451]:
```
df.shape
```

[451]:
```
(8622, 19)
```

[452]:
```
df.columns
```

[452]:
```
Index(['ID', 'Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',
       'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
       'Weekend', 'HasBought'],
      dtype='object')
```

1.1 Data Cleaning

[300]:
```
df.isna().sum()
```

[300]:
```
ID                         0
Administrative             0
Administrative_Duration    0
Informational              0
Informational_Duration     0
ProductRelated             0
ProductRelated_Duration    0
BounceRates                0
ExitRates                  0
PageValues                 0
SpecialDay                 0
Month                      0
OperatingSystems           0
Browser                    0
Region                     0
TrafficType                0
VisitorType                0
Weekend                    0
HasBought                  0
dtype: int64
```

- On élimine les valeurs depliqués

[6]:
```
print('no of duplicates: ',len(df[df.duplicated()]))
df.drop_duplicates(inplace=True)
```

```
no of duplicates:  0
```

[7]:
```
df.describe()
```

[7]:

	ID	Administrative	Administrative_Duration	Informational
count	8622.000000	8622.000000	8622.000000	8622.000000
mean	8015.889121	2.343192	83.405240	0.490373
std	2491.319819	3.315094	182.985925	1.258803
min	3701.000000	0.000000	-1.000000	0.000000
25%	5859.250000	0.000000	0.000000	0.000000
50%	8016.500000	1.000000	9.000000	0.000000
75%	10172.750000	4.000000	95.575000	0.000000
max	12330.000000	26.000000	3398.750000	24.000000

	Informational_Duration	ProductRelated	ProductRelated_Duration
count	8622.000000	8622.000000	8622.000000
mean	33.159578	31.872419	1192.184366
std	138.432097	44.605549	1944.264408
min	-1.000000	0.000000	-1.000000
25%	0.000000	7.000000	192.462500
50%	0.000000	18.000000	606.283333
75%	0.000000	37.000000	1450.841355
max	2549.375000	705.000000	63973.522230

	BounceRates	ExitRates	PageValues	SpecialDay	OperatingSystems
count	8622.000000	8622.000000	8622.000000	8622.000000	8622.000000
mean	0.021920	0.042457	5.704339	0.061795	2.124101
std	0.047651	0.047784	17.776590	0.199441	0.914461
min	0.000000	0.000000	0.000000	0.000000	1.000000
25%	0.000000	0.014286	0.000000	0.000000	2.000000
50%	0.003279	0.025000	0.000000	0.000000	2.000000
75%	0.017497	0.050000	0.000000	0.000000	3.000000
max	0.200000	0.200000	360.953384	1.000000	8.000000

	Browser	Region	TrafficType	Weekend	HasBought
count	8622.000000	8622.000000	8622.000000	8622.000000	8622.000000
mean	2.349803	3.154605	4.117606	0.233820	0.154836
std	1.719077	2.409020	4.070923	0.423284	0.361770
min	1.000000	1.000000	1.000000	0.000000	0.000000
25%	2.000000	1.000000	2.000000	0.000000	0.000000
50%	2.000000	3.000000	2.000000	0.000000	0.000000
75%	2.000000	4.000000	4.000000	0.000000	0.000000
max	13.000000	9.000000	20.000000	1.000000	1.000000

1.2 DATA VISUALISATION AND ANALYSIS¶

1.2.1 Analyse des variable catégoriques

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8622 entries, 0 to 8621
Data columns (total 19 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ID                       8622 non-null   int64
 1   Administrative           8622 non-null   int64
 2   Administrative_Duration  8622 non-null   float64
 3   Informational            8622 non-null   int64
 4   Informational_Duration   8622 non-null   float64
 5   ProductRelated           8622 non-null   int64
 6   ProductRelated_Duration  8622 non-null   float64
 7   BounceRates              8622 non-null   float64
 8   ExitRates                8622 non-null   float64
 9   PageValues               8622 non-null   float64
 10  SpecialDay               8622 non-null   float64
 11  Month                    8622 non-null   object
 12  OperatingSystems         8622 non-null   int64
 13  Browser                  8622 non-null   int64
 14  Region                   8622 non-null   int64
 15  TrafficType              8622 non-null   int64
 16  VisitorType              8622 non-null   object
 17  Weekend                  8622 non-null   int64
 18  HasBought                8622 non-null   int64
dtypes: float64(7), int64(10), object(2)
memory usage: 1.3+ MB
```

[9]:
```
df.groupby('HasBought').mean()
# le moyen des deux classes (0: visiteur non client , 1: vistieur Client)
```

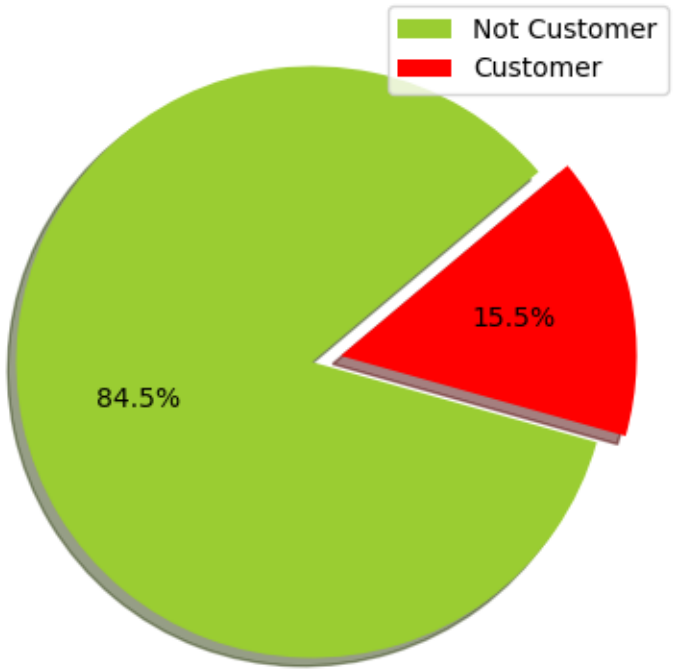
[9]:
```
                    ID  Administrative  Administrative_Duration  \
HasBought
0          7996.480170        2.160560                77.434044
1          8121.831461        3.340075               115.998579

           Informational  Informational_Duration  ProductRelated  \
HasBought
0               0.445862               29.160858       28.950322
1               0.733333               54.986298       47.822472

           ProductRelated_Duration  BounceRates  ExitRates  PageValues  \
HasBought
```

```
0                    1074.579098      0.025058   0.046707    1.916449
1                    1834.124137      0.004788   0.019256   26.380256

             SpecialDay  OperatingSystems    Browser    Region  TrafficType  \
HasBought
0              0.068533          2.132016  2.327981  3.164677     4.126252
1              0.025019          2.080899  2.468914  3.099625     4.070412

              Weekend
HasBought
0           0.228352
1           0.263670
```

[ ]:

- Les Données déséquilibrées , prédominance d'une classe par rapport à l'autre

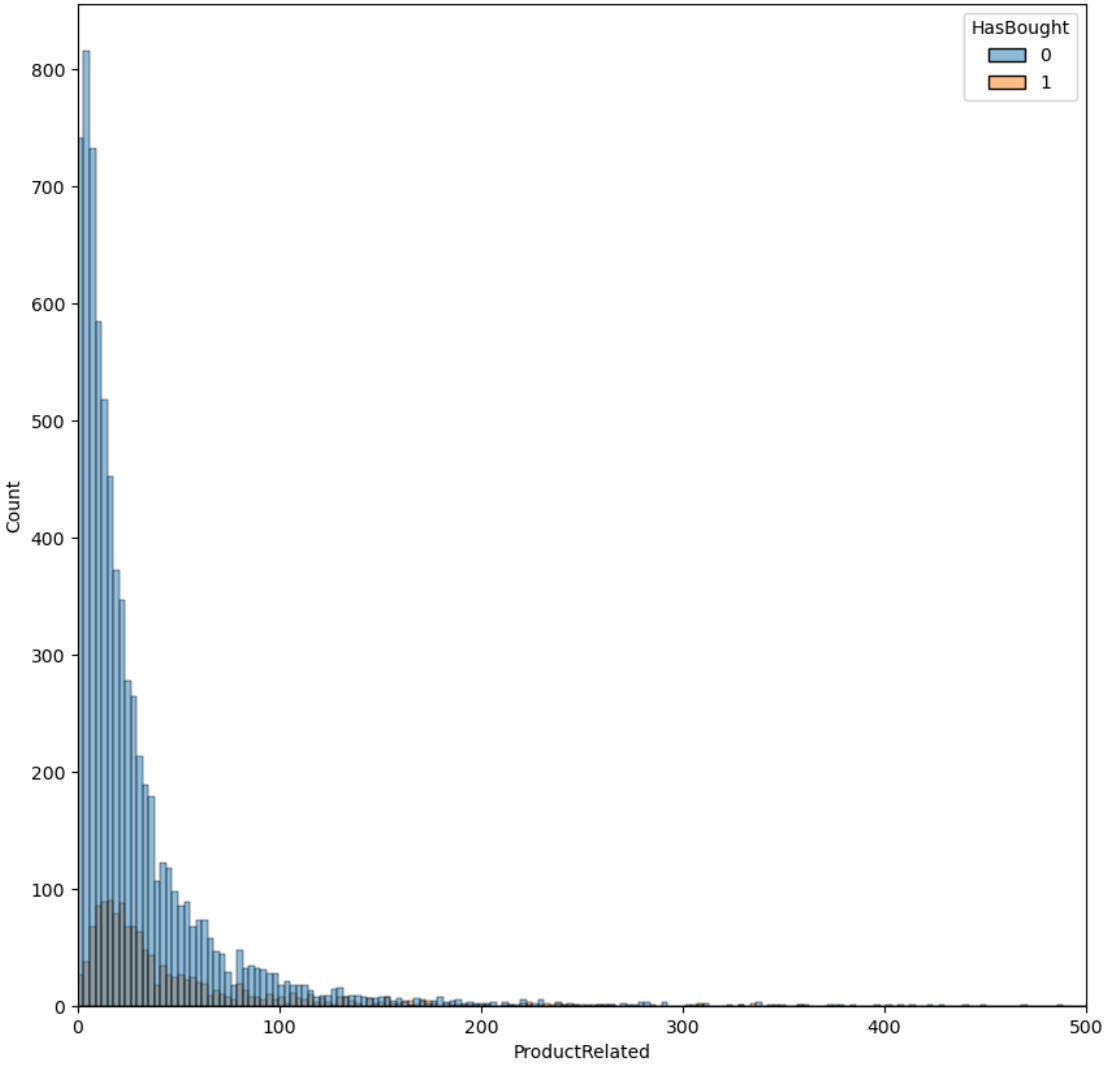
[10]:
```
plt.figure(figsize=(6,5))
plt.pie(df.HasBought.value_counts(),explode=[0,0.
↪1],colors=['yellowgreen','red'],startangle=40,autopct='%1.1f%%',shadow=True)
plt.legend(labels=['Not Customer','Customer'],loc='upper right')
plt.show()
```



On remarque donc que la classe 0 (aucune transaction n'a été effectué) est la classe prédominante

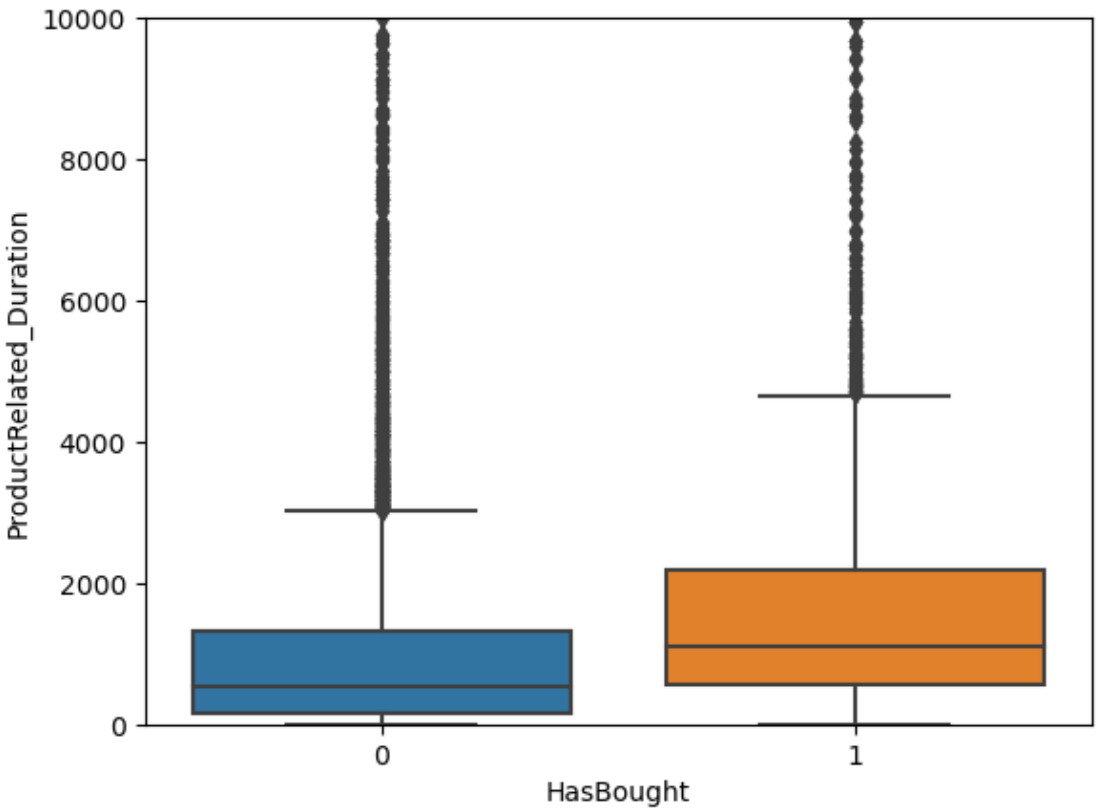
[11]:
```
plt.figure(figsize=(10,10))
plt.xlim(0,500)
sns.histplot(x='ProductRelated',data=df,hue ='HasBought',color='purple')
```

[11]: <AxesSubplot:xlabel='ProductRelated', ylabel='Count'>



- Temps passer par un visteur_non_client et un visiteur_client

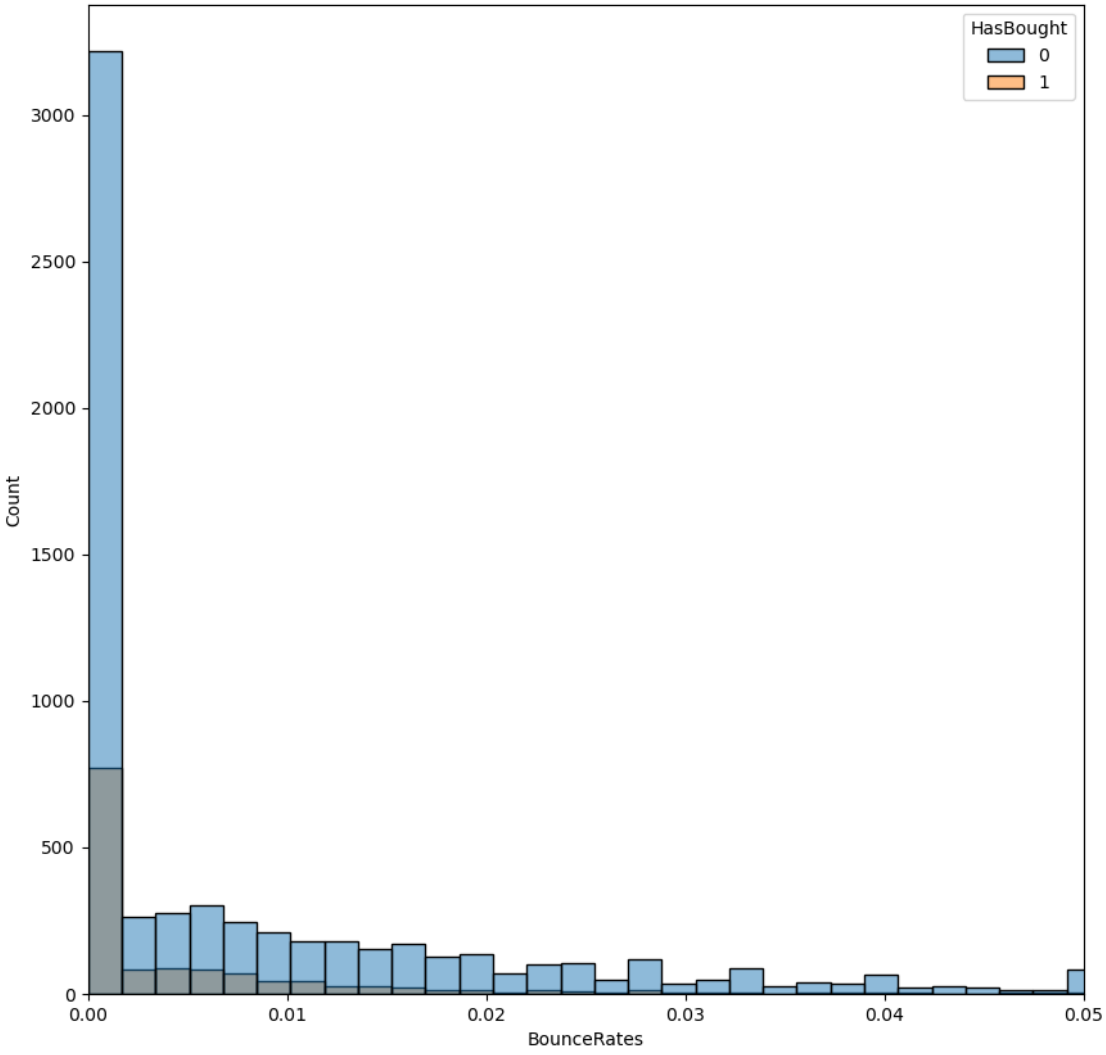
[12]:
```
plt.ylim(0,10000)
sns.boxplot( x='HasBought' , y='ProductRelated_Duration',data=df);
```



On remarque que globalement les personnes converties en clients passent assez de temps dans les pages relatives aux produits que ceux qui n'ont pas achetés de produits.

[13]:
```
plt.figure(figsize=(10,10))
plt.xlim(0,0.05)
sns.histplot(x='BounceRates',data=df,hue ='HasBought',color='brown')
```

[13]:
```
<AxesSubplot:xlabel='BounceRates', ylabel='Count'>
```

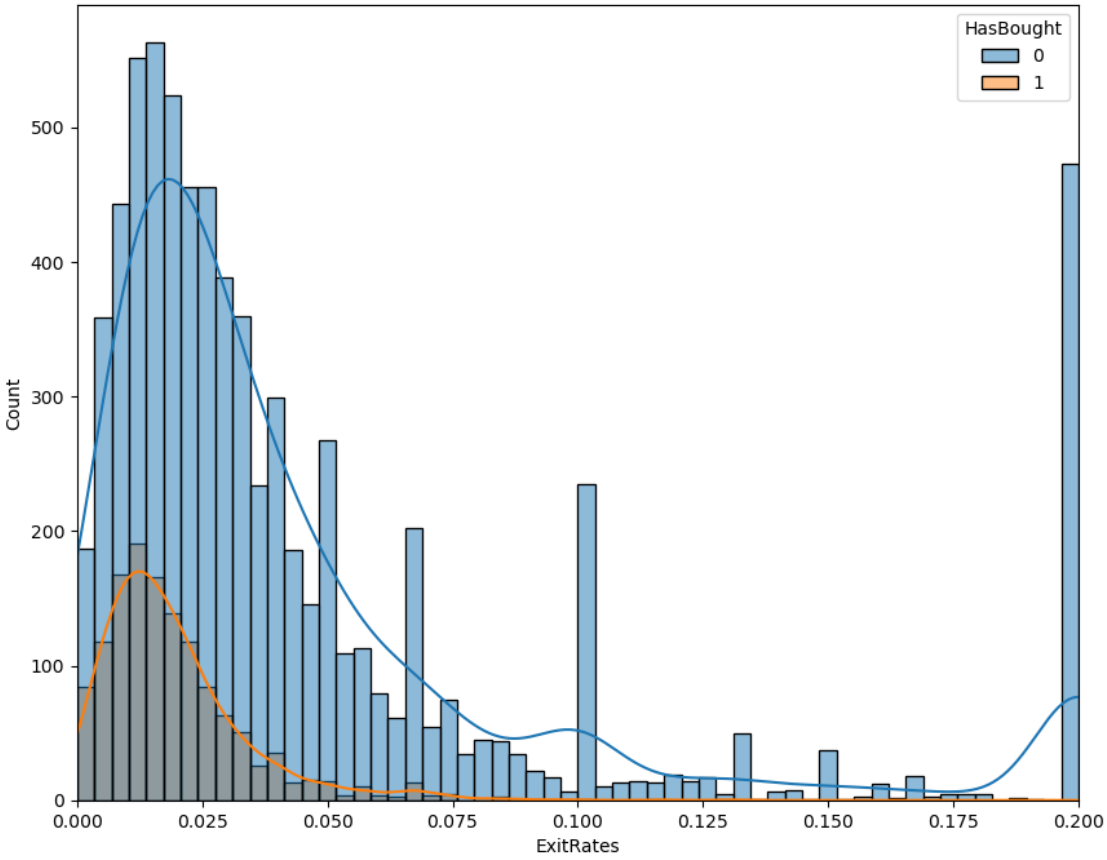


Un BounceRate faible conduit à une conversion fort probable (une corrélation négative) , un bounceRate > 0.01 conduit à un taux de conversion inférieur à la moyenne.

- Relation entre la métrique ExitRates et le taux de conversion de visiteur à client

[14]:
```
plt.figure(figsize=(10,8))
plt.xlim(0,0.2)
sns.histplot(x=df['ExitRates'],hue=df['HasBought'],color='red',kde=True)
```

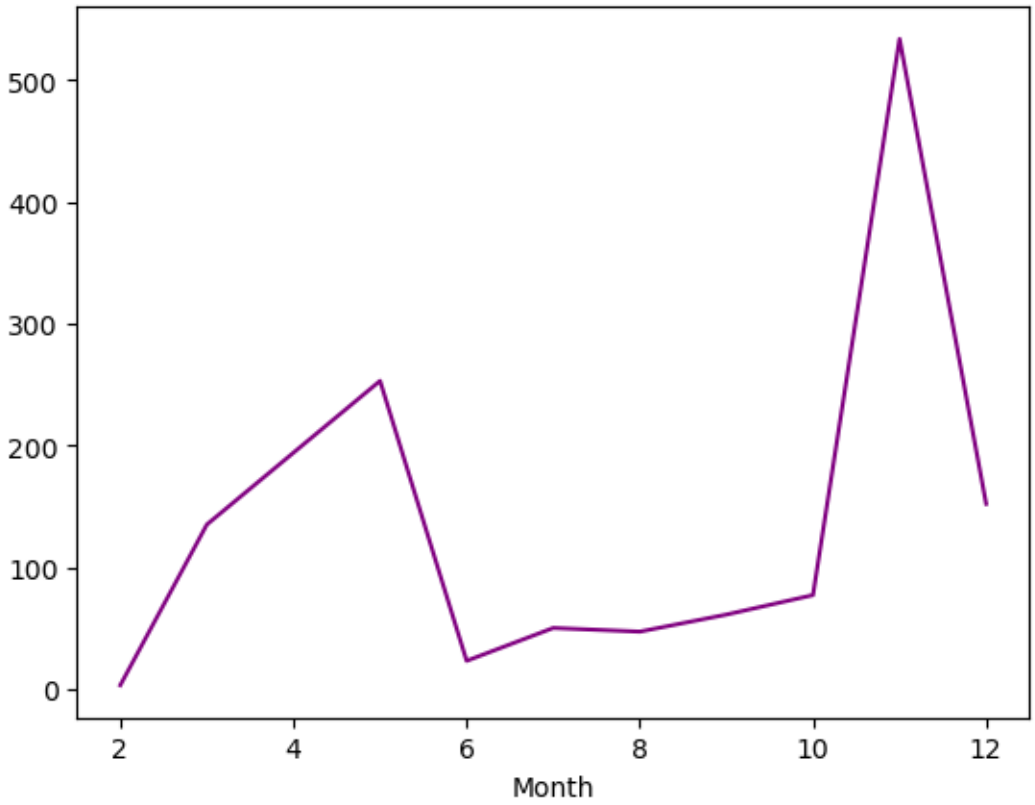
[14]:
```
<AxesSubplot:xlabel='ExitRates', ylabel='Count'>
```



Une forte corrélation négative entre le ExitRate moyen des pages visitées et la conversion du visiteur, généralement c'est entre 0.005 et 0.02

[483]:
```
months={
   'Feb' : 2 ,'Mar' : 3 ,'May' : 5 ,'June'  :6 ,'Jul' : 7 ,'Aug' :8 ,'Sep':
 ↪9,'Oct' : 10 ,'Nov' : 11 ,'Dec' : 12
}

df['Month']=df.Month.apply(lambda x: months[x])
df.groupby('Month')['HasBought'].sum().plot(kind='line',color='purple')
```

[483]:
```
<AxesSubplot:xlabel='Month'>
```



Business a fait un énorme profit en novembre.

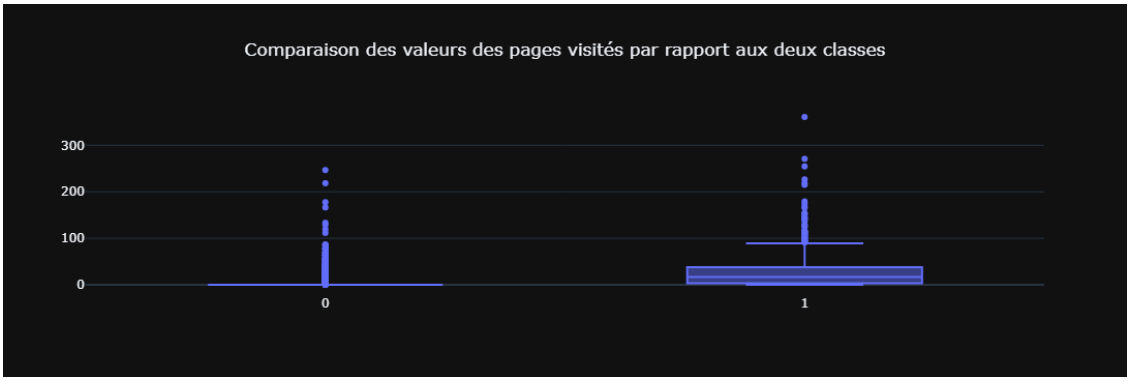
[16]:
```
df[df['HasBought'] ==1].shape[0]/df[df['HasBought'] ==0].shape[0]*100
```

[16]:
```
18.320296418279128
```

Effet de la métrique PageValues sur le taux de conversion

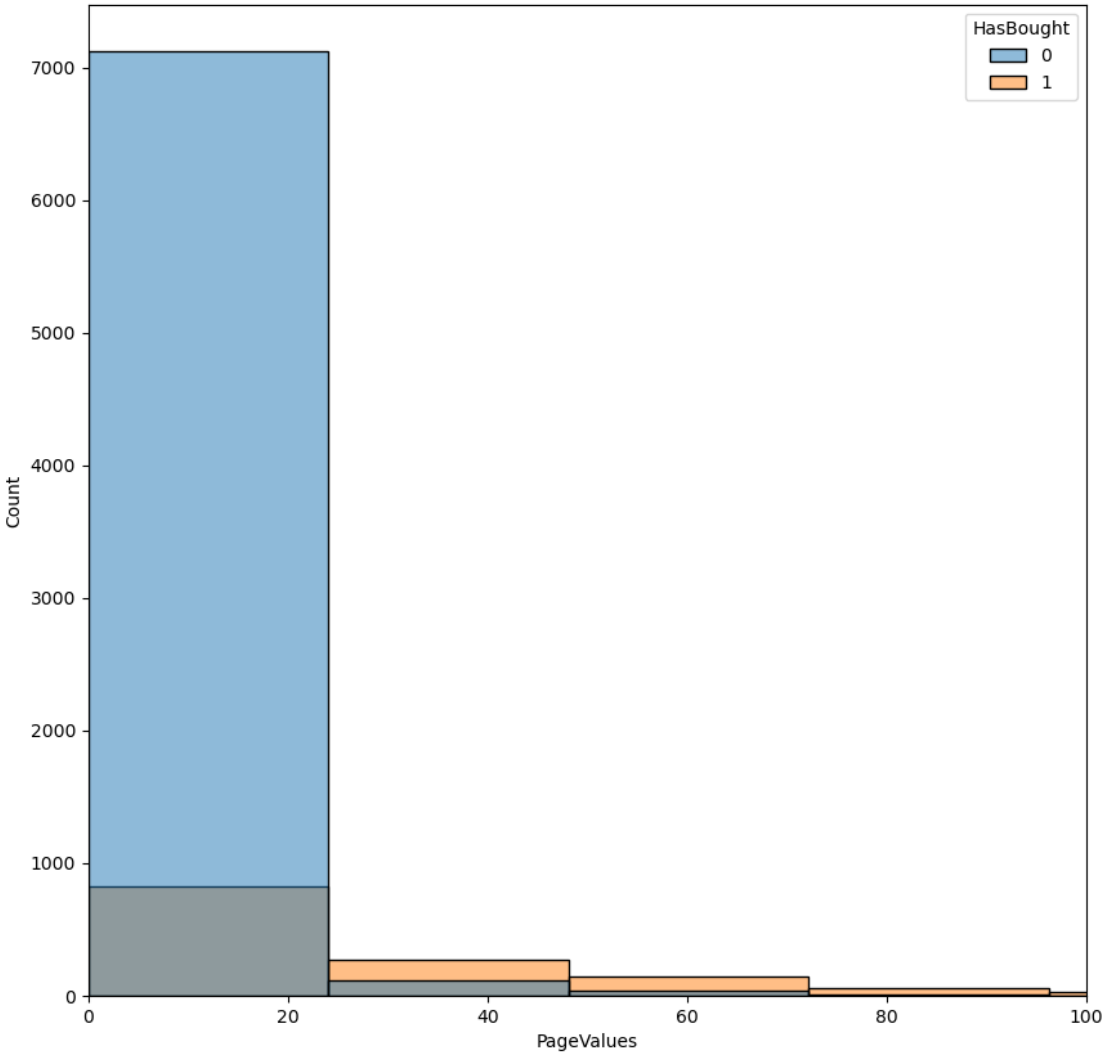
[17]:
```
#plt.figure(figsize=(10,8))
import plotly.offline as pyo
import plotly.graph_objs as go
```

[18]:
```
data = go.Box(x=df['HasBought'],y = df['PageValues'])
lay = go.Layout(title='Comparaison des valeurs des pages visités par rapport␣
 ↪aux deux classes', title_x =0.5 , template='plotly_dark')
fig = go.Figure(data= data , layout=lay)
pyo.iplot(fig)
```



[19]:
```
plt.figure(figsize=(10,10))
plt.xlim(0,100)
sns.histplot(x=df['PageValues'],hue=df['HasBought'])
```

[19]: <AxesSubplot:xlabel='PageValues', ylabel='Count'>



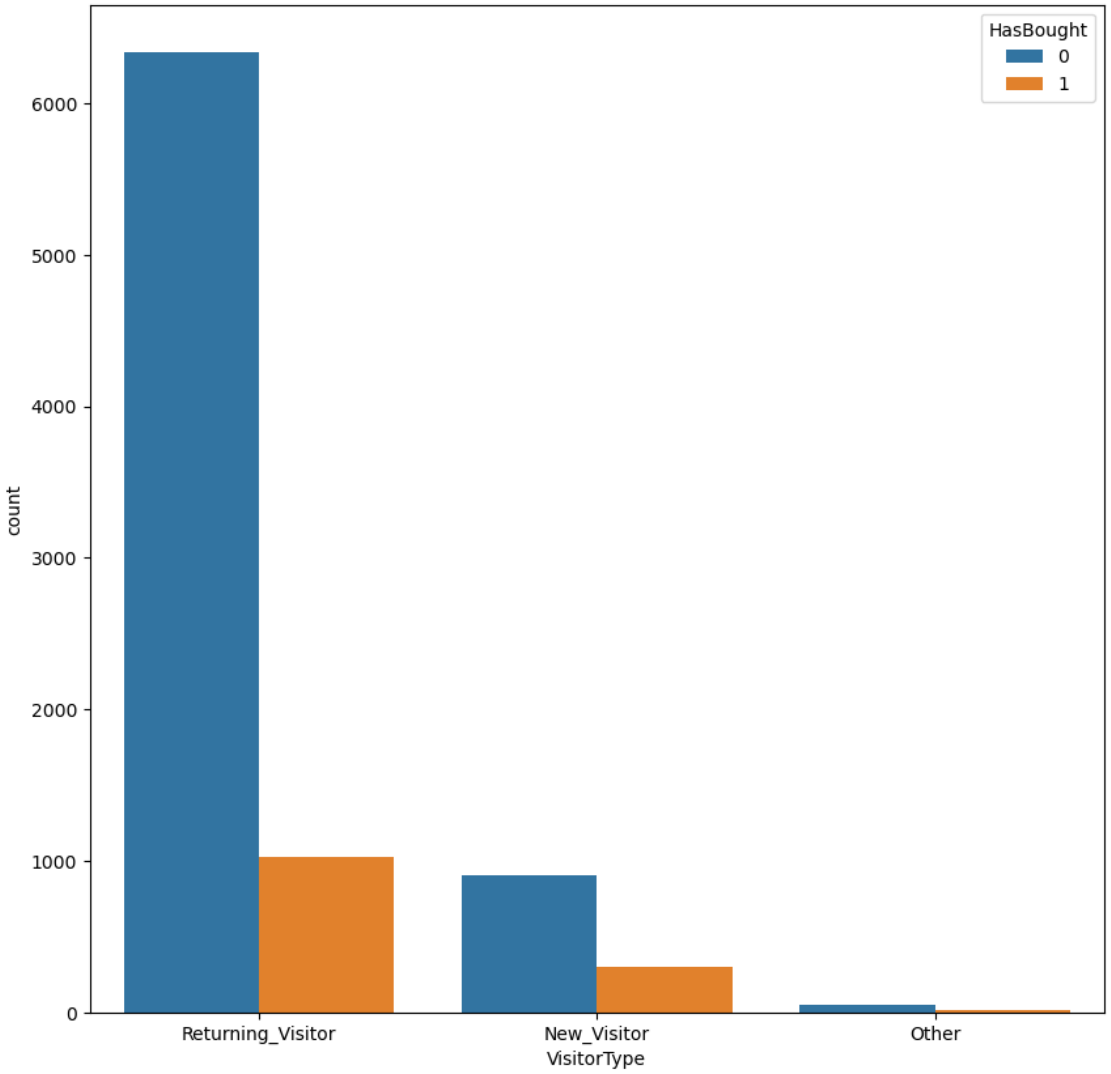
1.2.2 Type de visiteurs

[20]:
```
data = go.Bar(x=df['VisitorType'].unique().tolist(),y=df['VisitorType'].
 ↪value_counts(),marker=dict(color='gold'))
lay = go.Layout(title='Type of visitors', title_x =0.5, template='plotly_dark'␣
 ↪,yaxis=dict( range= [0,8000] ,dtick=1000 ))
fig = go.Figure(data= data , layout=lay)
pyo.iplot(fig)
```

Les visiteurs récurrents sont 7 fois plus nombreux que les nouveaux visiteurs, ce qui signifie que le site Web est beaucoup plus susceptible d'être visité après la toute première visite.

[21]:
```
plt.figure(figsize=(10,10))
sns.countplot(x=df['VisitorType'],hue=df['HasBought'],)
```

[21]:
```
<AxesSubplot:xlabel='VisitorType', ylabel='count'>
```



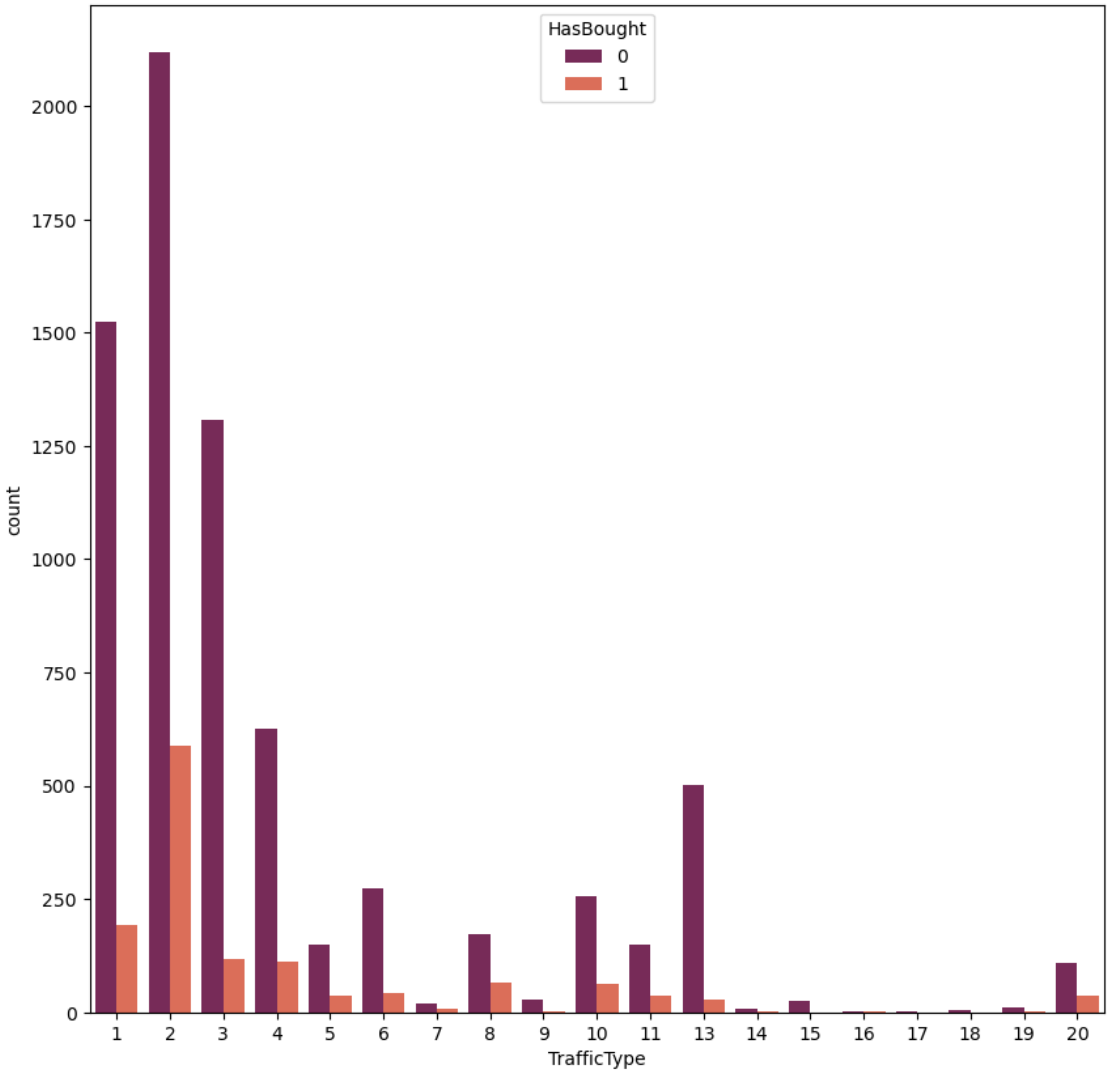
- On remarque que les visiteurs ayant déja visité le site ont un faible taux de conversion (= 10%) par rapport aux visiteurs nouveaux (new_visitors) (~25%)

[22]:
```
df['TrafficType']= df['TrafficType'].astype('category')
df['VisitorType'] = df['VisitorType'].astype('category')
```

[23]:
```
plt.figure(figsize=(10,10))
sns.countplot(df['TrafficType'],hue=df['HasBought'].astype('category'),data=␣
↪df,palette='rocket');
```

```
C:\Users\Xps\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only
```

```
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
```



- Remarquablement, le type de traffic numero 2 contient le nombre maximale de visiteurs ayant effectué des transactions. Il s'agit du trafic de référence qui concerne les visiteurs venant à un site Web à la suite de clics sur des liens sur d'autres sites Web. On peut donc éventuellement essayer de le considérer comme route majeure et unique pour vers le site si c'est possible.

[24]:
```
# on retourne au type entier pour la classification
df['TrafficType'] = df['TrafficType'].astype(int)
```

[25]:
```
weekend = pd.crosstab(df.Weekend,df.HasBought)
weekend['conversion_rate']  = weekend[1]/weekend.sum(axis=1)
```

```
weekend = weekend.style.
 ↪background_gradient(subset='conversion_rate',cmap='RdYlGn')
weekend
```

[25]: <pandas.io.formats.style.Styler at 0x2c5fce21490>

Effectivement, en weekend , le taux augmente de presque 4%

- Qu'en est -il des Special Days (les jours proches des jours fériés (spéciaux) comme Mother's Day par exemple

[26]:
```
sp = pd.crosstab(df['SpecialDay'],df.HasBought)
sp['conversion_rate'] = sp[1]/sp.sum(axis=1)
sp = sp.style.bar(subset='conversion_rate',color='yellowgreen')
sp
```

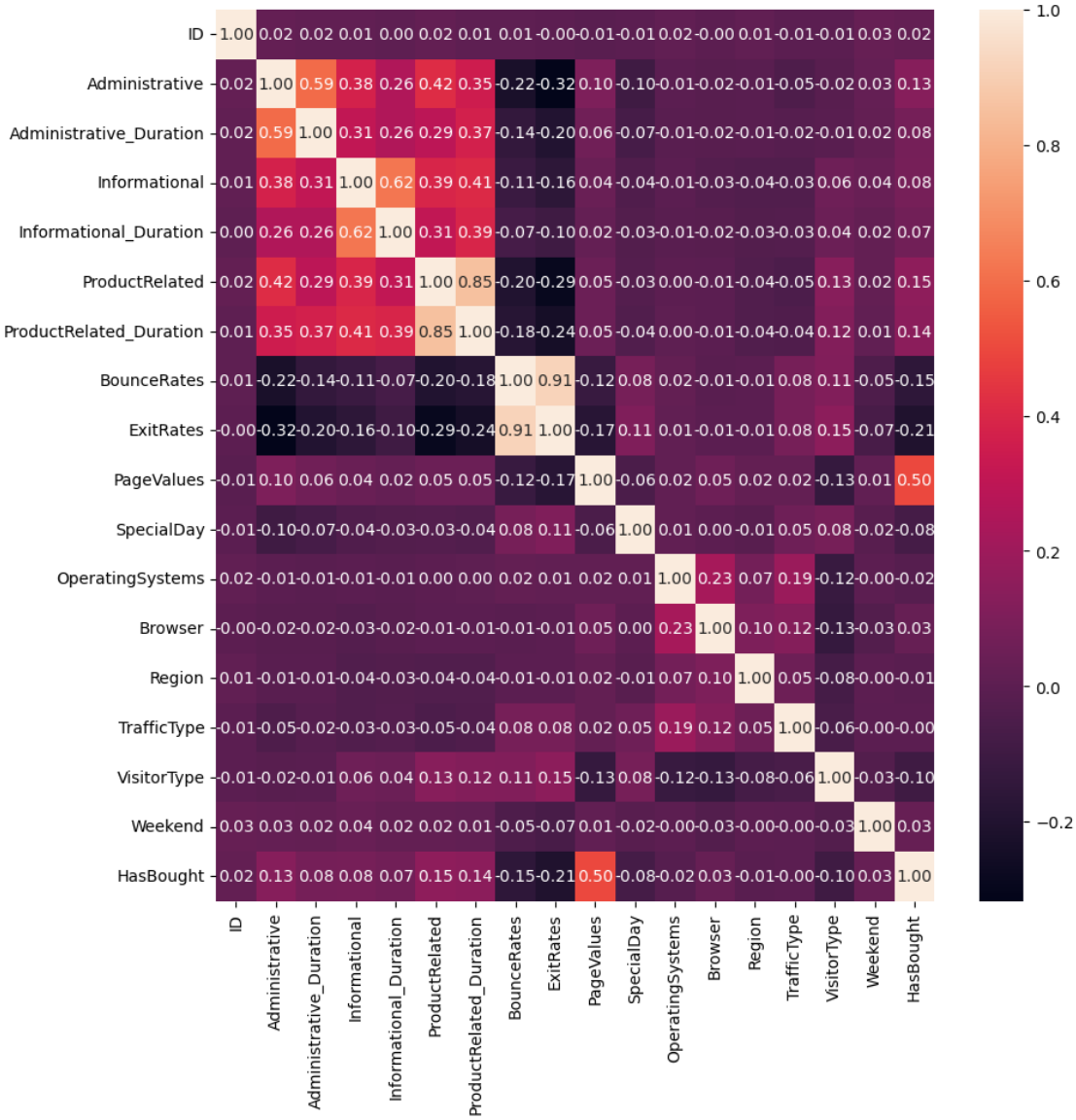
[26]: <pandas.io.formats.style.Styler at 0x2c5fcf53d00>

A notre surprise, le nombres des visiteurs effectuant un achat est bien supérieur dans les jours normaux que les jours proches des jours fériées ou les jours fériés-même. On constate donc que le taux de conversion n'est pas pas dépendant des jours fériées et que probablement le site web de l'e-commerce n'offre pas des offres particuliers dans les jours spéciaux (black friday , cyber Monday etc ...)

[453]:
```
# On va recoder les attributs catégoriques pour voir leurs corrélations avec la␣
 ↪variable output.
df_copy = df
gg={
   'Returning_Visitor' : 1,'New_Visitor' : 0 ,'Other' : -1
}

df_copy['VisitorType']=df_copy.VisitorType.apply(lambda x: gg[x])
df_copy['TrafficType'] = df_copy['TrafficType'].astype(int)
df_copy['HasBought'] =df_copy['HasBought'].astype(int)
```

[460]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df_copy.corr(),annot=True ,fmt=".2f", cmap="rocket")
```

[460]: <AxesSubplot:>



On remarque donc que les attributs PageValues et ExitRates ont les plus grandes corrélations avec la variable HasBought en comparaison avec les autres attributs.

1.3 Data Preprocessing

D'apres notre exploratory data analysis , on peut détecter les attributs qui influencent relativement ou fortement notre variable à prédire. On commence tout d'abord par enlever les colonnes insignificatives dans notre dataset.

I- Quelles sont les tâches/problèmes qu'on peut détecter d'après notre analyse exploratoire ?

- Supprimer les attributs insignificatif pour notre modèle.

- On doit procéder à générer des attributs (feature engineering) qui vont simplifier et optimiser le modèle.
- Encoder les variables nominales et catégoriques.
- Normaliser les variables continues.
- Palier au problèmes des données désequilibrées en utilisant les techniques classiques d'undersampling/oversampling.
- Il y a pas assez de valeurs abhérantes dans presque tous les attributs, on doit donc penser à utiliser une normalisation insensible aux outliers.

[484]:
```
# On regroupe les colonnes des pages Administratives et les pages␣
 ↪informationnelles en une seule colonnes qu'on nomme 'nonProductPages'
df['Administrative'] = df['Administrative'] +  df['Informational']
df.rename(columns={'Administrative' : 'nonProductRelated'},inplace=True)

#On regroupe les durées passées dans les pages Administratives et␣
 ↪infomationnelles également
df['Administrative_Duration']  = df['Administrative_Duration']  +␣
 ↪df['Informational_Duration']
df.rename(columns={'Administrative_Duration' : ␣
 ↪'nonProductRelated_Duration'},inplace=True)

#On supprime les colonnes dont nous avons pas besoin
df.
 ↪drop(['ID','Informational','Informational_Duration','OperatingSystems','Browser','Region'],

# Comme on a besoin de données numériques pour alimenter les algos ML, on␣
 ↪changer le type de données des attributs.
df['HasBought'] = df['HasBought'].astype(int)

# On va supprimer les lignes qui contiennent 'Others' dans la colonne␣
 ↪'VisitorType' , il y en a 13.
df.drop(df[df['VisitorType']=='Others'].index.tolist(),axis=0,inplace=True)

## Encoder l'attribut 'VisitorType' , recodage binaire (2 catégories).
df['VisitorType'] = np.where(df['VisitorType'] =='Returning_Visitor',1,0)
```

- Synthétiser les données de la classe 1 pour avoir des données équilibrées

```
*******   Synthetic Minority Oversampling Technique    *******
```

D'après l'exploration des différentes techniques qui permettent de palier à ce problème (undersampling , oversampling , ensemble sampling), on a opté pour la technique smote oversampling qui semble la plus rentable dans la plupart des cas.

[485]:
```
X = df.drop('HasBought',axis=1)
y = df['HasBought']
```

1.4 Données désequilibrées

[486]:
```
df.HasBought.value_counts()
```

[486]:
```
0    7287
1    1335
Name: HasBought, dtype: int64
```

[487]:
```
#splitting the train and test sets.
X_train,X_test, y_train, y_test  =  train_test_split(X,y,test_size=0.
 ↪3,random_state=69,stratify=y)
```

[488]:
```
#OverSampling
smote = SMOTE(random_state=1)
X_train, y_train = smote.fit_resample(X_train,y_train)
```

1.4.1 Données équilibrées après l'application de la technique SMOTE

[489]:
```
y_train.value_counts()
```

[489]:
```
0    5101
1    5101
Name: HasBought, dtype: int64
```

1.5 SUPERVISED LEARNING

1.5.1 KNN

[398]:
```
# nombres de découpes dans la méthode GridSearchCV
découpes = StratifiedKFold(n_splits=5,shuffle=True,random_state =0)

# construction du modèle
model = Pipeline(steps = [['scaler', RobustScaler()],
                          ['classifier', KNeighborsClassifier()]])

params =  {
          'classifier__metric' : ['euclidean','manhattan','minkowski'] ,
          'classifier__n_neighbors' : np.arange(6,15),
           'classifier__weights' :  ['uniform','distance']
}

# GridSearch  -> parametres optimales.
grid = GridSearchCV(estimator = model,
                    param_grid=params,
                    cv = découpes ,
                    scoring = 'roc_auc')
grid.fit(X_train,y_train)
```

```
knn  = grid.best_estimator_
```

[399]:
```
grid.best_params_
```

[399]:
```
{'classifier__metric': 'manhattan',
 'classifier__n_neighbors': 9,
 'classifier__weights': 'distance'}
```

[400]:
```
gg =  knn.predict(X_test)
print(roc_auc_score(y_test,gg))
```

```
0.7760373768232667
```

1.5.2 NAIVE BAYES

L'Algorithme Naive Bayes n'a pas de paramètres qui méritent un GridSearchCV

[402]:
```
## Les parameters qu'on peut jouer avec dans un "naive" bayes :/
model=GaussianNB()
model.get_params()
```

[402]:
```
{'priors': None, 'var_smoothing': 1e-09}
```

[403]:
```
# instancier le scaler.
scaler = RobustScaler()
#normaliser X_train
X_train_scaled=scaler.fit_transform(X_train)

#instancier le modèle.
naive_bayes= GaussianNB()
naive_bayes.fit(X_train_scaled,y_train)

#normaliser X_test
X_test_scaled = scaler.transform(X_test)

yhat = naive_bayes.predict(X_test_scaled)

print(classification_report(y_test,yhat))
print(roc_auc_score(y_test,yhat))
```

```
              precision    recall  f1-score   support

           0       0.95      0.53      0.68      2186
           1       0.25      0.84      0.38       401

    accuracy                           0.58      2587
```

```
   macro avg       0.60      0.69      0.53      2587
weighted avg       0.84      0.58      0.64      2587
```

```
0.6866679367455104
```

On remarque que l'algorithme NaiveBayes a une performance faible que celle du KNN.

1.5.3 DECISON TREES

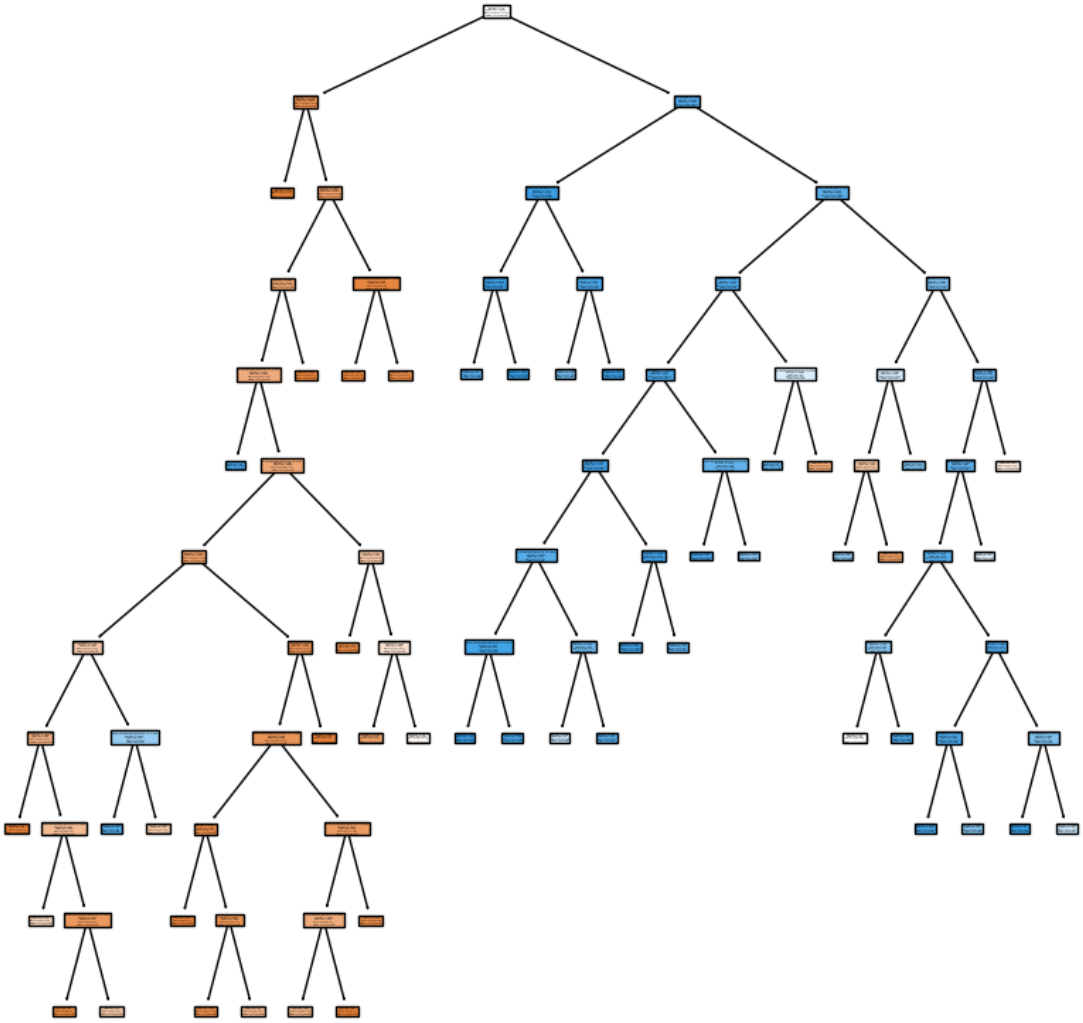
[414]:
```
# nombres de découpes dans la méthode GridSearchCV
découpes = StratifiedKFold(n_splits=5,shuffle=True,random_state =11)

# construction du modèle
pipeline_dt = Pipeline(steps = [['scaler', RobustScaler()],
                         ['classifier', DecisionTreeClassifier()]])


params_dt = {
              'classifier__ccp_alpha': [0.1, 0.01, 0.001],
              'classifier__criterion' :['gini', 'entropy','log_loss'],
              'classifier__min_samples_leaf' :   np.arange(1,30)
             }
# GridSearch  -> parametres optimales.
grid = GridSearchCV(estimator = pipeline_dt,
                    param_grid=params_dt,
                    cv = découpes ,
                    scoring = 'roc_auc')
grid.fit(X_train,y_train)

grid.best_params_
```

```
C:\Users\Xps\anaconda3\lib\site-
packages\sklearn\model_selection\_validation.py:372: FitFailedWarning:


435 fits failed out of a total of 1305.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting
error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
435 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Xps\anaconda3\lib\site-
packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
```

```
  File "C:\Users\Xps\anaconda3\lib\site-packages\sklearn\pipeline.py", line 394,
in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "C:\Users\Xps\anaconda3\lib\site-packages\sklearn\tree\_classes.py", line
937, in fit
    super().fit(
  File "C:\Users\Xps\anaconda3\lib\site-packages\sklearn\tree\_classes.py", line
352, in fit
    criterion = CRITERIA_CLF[self.criterion](
KeyError: 'log_loss'
```

[414]:
```
{'classifier__ccp_alpha': 0.001,
 'classifier__criterion': 'entropy',
 'classifier__min_samples_leaf': 29}
```

[415]:
```
from sklearn.tree import plot_tree,export_graphviz
dt = grid.best_estimator_
estimator_dt = dt.named_steps['classifier']
fig =  plt.figure(figsize=(10,10))
plot_tree(estimator_dt, filled =True , feature_names= df.drop('HasBought',axis␣
 ↪=1).columns.tolist() ,
            class_names= ["not_buying","buying"])
plt.show()
```



[416]:
```
yy = dt.predict(X_test)
print(classification_report(y_test,yy))
print(roc_auc_score(y_test,yy))
```

```
              precision    recall  f1-score   support

           0       0.95      0.90      0.93      2186
           1       0.58      0.75      0.65       401

    accuracy                           0.88      2587
   macro avg       0.77      0.82      0.79      2587
weighted avg       0.89      0.88      0.88      2587
```

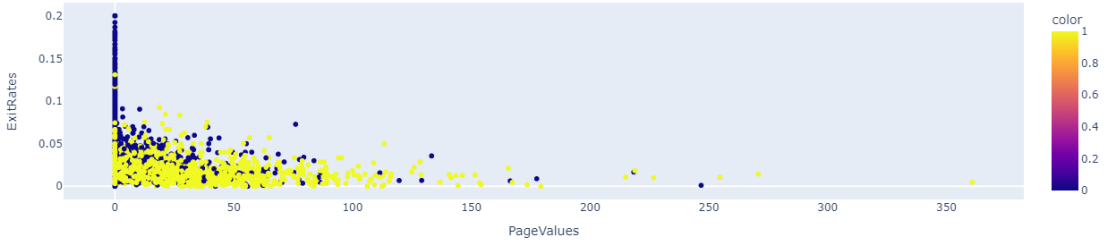
```
0.8231839203455223
```

1.5.4 SUPPORT VECTOR MACHINES

Une première analyse pour effectuer ce modèle serait de voir si les deux classes sont proches à être linéairement séparables. Comme on a un nombre d'attributs > 3, on peut par exemple prendre uniquement les deux attributs les plus corrélées avec la variable target et visualiser la séparation.

[417]:
```
import plotly.express as px
```

[472]:
```
plt.figure(figsize=(10,10))
fig =px.scatter(x=df['PageValues'],y=df['ExitRates'],color=df['HasBought'],
 ↪labels={'x':'PageValues', 'y':'ExitRates'})
fig.update_layout(xaxis=dict(showgrid=False), yaxis=dict(showgrid=False))
fig
```




```
<Figure size 1000x1000 with 0 Axes>
```

On remarque que les classes sont très loin d'être linéairement séparables. ~#Ce ne sera donc pas une tache facile pour notre SVM

[421]:
```
# cette cellule generer beacoup de warnings, du coup fallait cacher les
 ↪warnings avec ces deux lignes de codes.
import warnings
warnings.filterwarnings("ignore", category=UserWarning, module='sklearn')

# nombres de découpes dans la méthode GridSearchCV
découpes = StratifiedKFold(n_splits=5,shuffle=True,random_state =11)

# construction du modèle
svm_pipe  = Pipeline(steps = [['scaler', RobustScaler()],
                              ['classifier', svm.SVC()]])


params_svm  = {
              'classifier__kernel': ['linear', 'poly', 'rbf', 'sigmoid'], # on
 ↪a enlevé la valeur 'precomputed' because it only works on squarred matrices.
```

```
              'classifier__gamma' :['scale', 'auto'],
              'classifier__degree' : [3,4,5]
             }
# GridSearch  -> parametres optimales.
grid = GridSearchCV(estimator = svm_pipe,
                    param_grid=params_svm,
                    cv = découpes ,
                    scoring = 'roc_auc')
grid.fit(X_train,y_train)

grid.best_params_
```

[421]:
```
{'classifier__degree': 3,
 'classifier__gamma': 'auto',
 'classifier__kernel': 'rbf'}
```

[422]:
```
svm = grid.best_estimator_
pred = svm.predict(X_test)
print(classification_report(y_test,pred))
print(roc_auc_score(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.89      0.92      2186
           1       0.54      0.72      0.62       401

    accuracy                           0.86      2587
   macro avg       0.74      0.81      0.77      2587
weighted avg       0.88      0.86      0.87      2587

0.805100127083937
```

1.6 UNSUPERVISED LEARNING

On supprime la variable target qui représente la classe de chaque point de données pour appliquer les algorithmes de clustering, et avoir une idée sur les différents paradigmes avec lequels les algorithmes ont regroupés les données.

[490]:
```
# all the features's data
X  = df.drop('HasBought',axis=1)
```

[491]:
```
scaler  = RobustScaler()
X_scaled = scaler.fit_transform(X)
```

1.6.1 K-MEANS CLUSTERING

Trouvons le nombre optimale de clusters

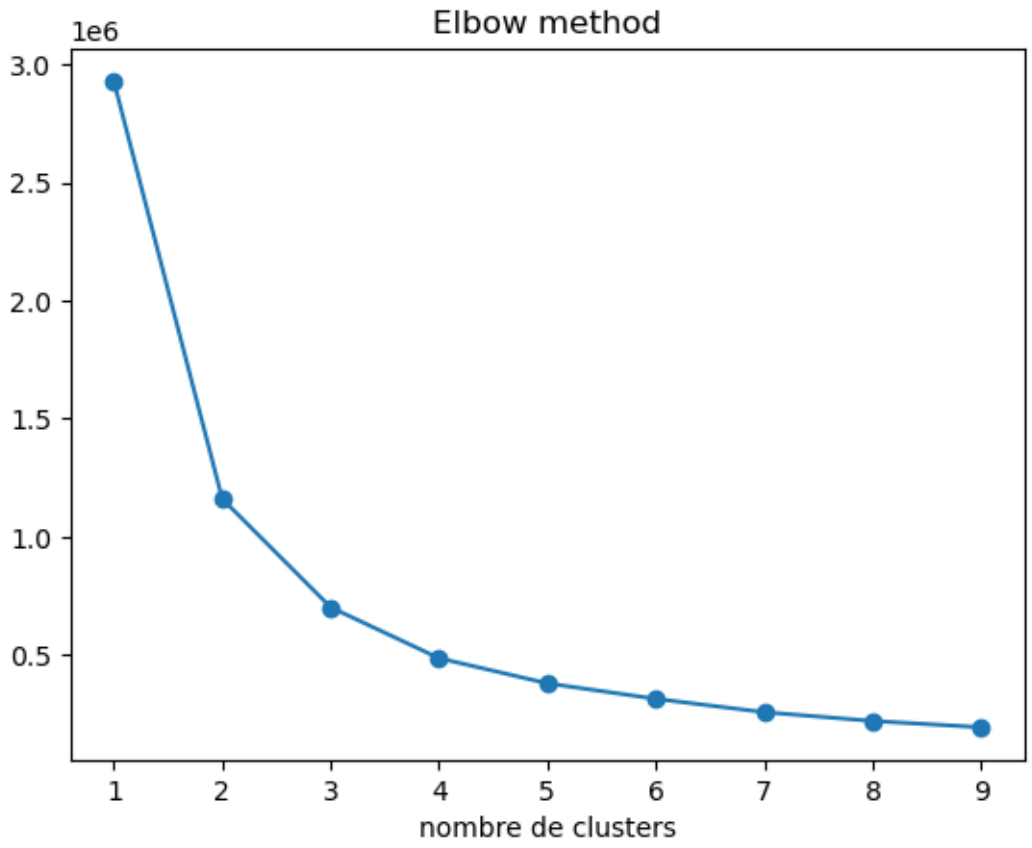
[317]:
```
from sklearn.metrics import davies_bouldin_score

indices = []
elbow = []
for  k in range(2,10) :
    kmeans = KMeans(n_clusters=k)
    labels =  kmeans.fit_predict(X_scaled)
    dunn_index = 1-davies_bouldin_score( X_scaled , labels)
    indices.append(dunn_index)
```

[318]:
```
data = pd.DataFrame({'k' : range(2,10),'indice_dunn' : indices})
data.set_index('k',inplace=True)
data =  data.style.bar(subset='indice_dunn',color='#ff6700')
data
```

[318]:
```
<pandas.io.formats.style.Styler at 0x2c5a431c760>
```

D'apres l'indice de dunn on voit que K=2 est le nombre optimal de clusters

[221]:
```
elbow = []
for  k in range(1,10) :
    kmeans = KMeans(n_clusters=k)
    labels =  kmeans.fit_predict(X_scaled)
    elbow.append(kmeans.inertia_)
plt.plot(range(1,10), elbow, marker='o')
plt.title('Elbow method')
plt.xlabel('nombre de clusters')
plt.show()
```



la méthode Elbow confirme l'optimalité de k=2

[492]:
```
kmeans_model = KMeans(n_clusters=2)
labels =  kmeans_model.fit_predict(X_scaled)

print(classification_report(df['HasBought'], labels))
```

```
              precision    recall  f1-score   support

           0       0.89      0.98      0.93      7287
           1       0.78      0.31      0.44      1335

    accuracy                           0.88      8622
   macro avg       0.83      0.65      0.69      8622
weighted avg       0.87      0.88      0.86      8622
```

[493]:
```
cg=pd.DataFrame(labels)
cg[0].value_counts()
```

[493]:
```
0    8091
1     531
Name: 0, dtype: int64
```

K-Means ne detecte pas de valeurs aberrantes

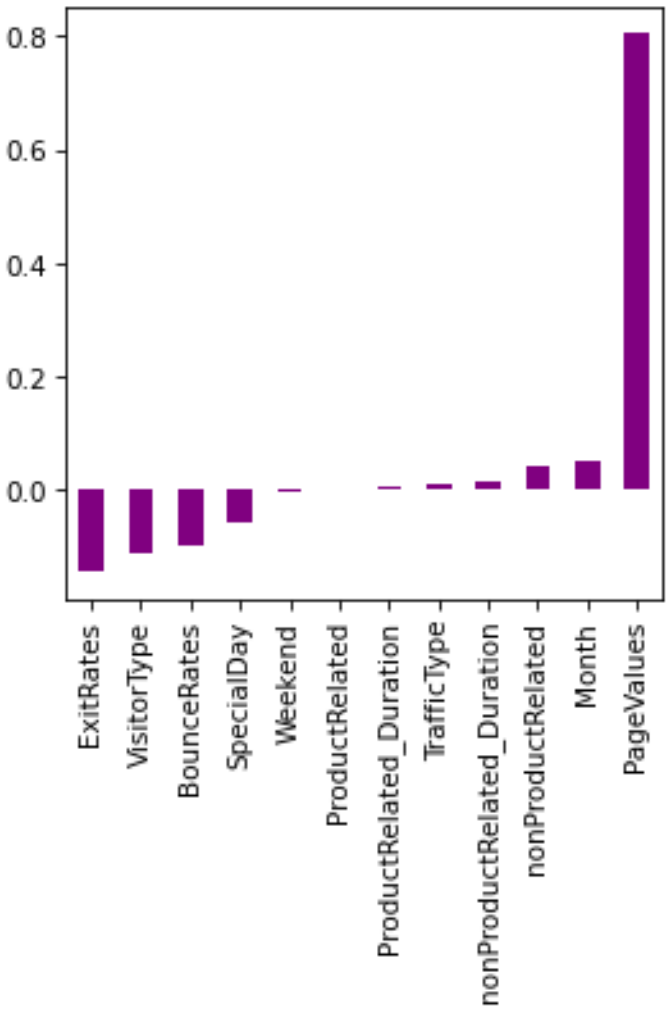
[494]:
```
Xcopy = X
Xcopy['classe'] =  labels
```

[495]:
```
Xcopy.corr()['classe'][:-2]
```

[495]:
```
nonProductRelated             0.041422
nonProductRelated_Duration    0.014659
ProductRelated                0.001252
ProductRelated_Duration       0.004449
BounceRates                  -0.099189
ExitRates                    -0.144740
PageValues                    0.805576
SpecialDay                   -0.057124
Month                         0.048209
TrafficType                   0.009191
VisitorType                  -0.114028
Name: classe, dtype: float64
```

[496]:
```
plt.figure(figsize=(4,4),dpi=75)
X.corr()['classe'].iloc[:-1].sort_values(ascending =True).
 ↪plot(kind='bar',color='purple')
```

[496]:
```
<AxesSubplot:>
```



- l'allure affiche les features prises en consideration par le modele KMeans, par exemple PageValues est le feature le plus utilisé par ce modele pour faire la separation des cluster

[388]:
```
#Qualité du K-MEANS CLUSTERING
# print(classification_report(df['HasBought'],X['classe']))
from sklearn.metrics import rand_score

dunn_kmeans = 1-davies_bouldin_score( X_scaled , labels)
rand_kmeans = rand_score(df['HasBought'], labels)
acc_kmeans=accuracy_score(df['HasBought'], labels)
print('dunn index : ', dunn_kmeans , ', rand index : ', rand_kmeans , ',␣
 ↪accuracy score : ' , acc_kmeans)
```

```
dunn index :  0.5049613975982516 , rand index :  0.787831233529084 , accuracy
score :  0.879378334493157
```

1.6.2 HIERARCHAL AGGLOMERATIVE CLUSTERING

[324]:
```
from  sklearn.cluster import AgglomerativeClustering
```

On a choisi la distance euclidienne et la distance inter-cluster sera calculée par la technique single-link.

[367]:
```
agg = ␣
 ↪AgglomerativeClustering(n_clusters=None,distance_threshold=0,affinity='euclidean',linkage=␣
 ↪'single')
cluster_labels  = agg.fit_predict(X_scaled)
```

[16]:
```
matrice_liens = hierarchy.linkage(agg.children_)
linkage_df = pd.DataFrame(matrice_liens,columns␣
 ↪=['Cluster1','Cluster2','distance','nombre_datapoints'],index=range(1,matrice_liens.
 ↪shape[0]+1))
linkage_df.rename_axis('Etape')
```

[16]:

Etape	Cluster1	Cluster2	distance	nombre_datapoints
1	56.0	103.0	1.414214	2.0
2	96.0	182.0	1.414214	2.0
3	2639.0	3335.0	1.414214	2.0
4	2820.0	6473.0	1.414214	2.0
5	957.0	976.0	1.414214	2.0
…	…	…	…	…
8616	8171.0	17235.0	743.750630	8614.0
8617	7008.0	17236.0	802.945826	8615.0
8618	17232.0	17237.0	817.469877	8618.0
8619	5580.0	17238.0	921.364206	8619.0
8620	16578.0	17239.0	967.952478	8621.0

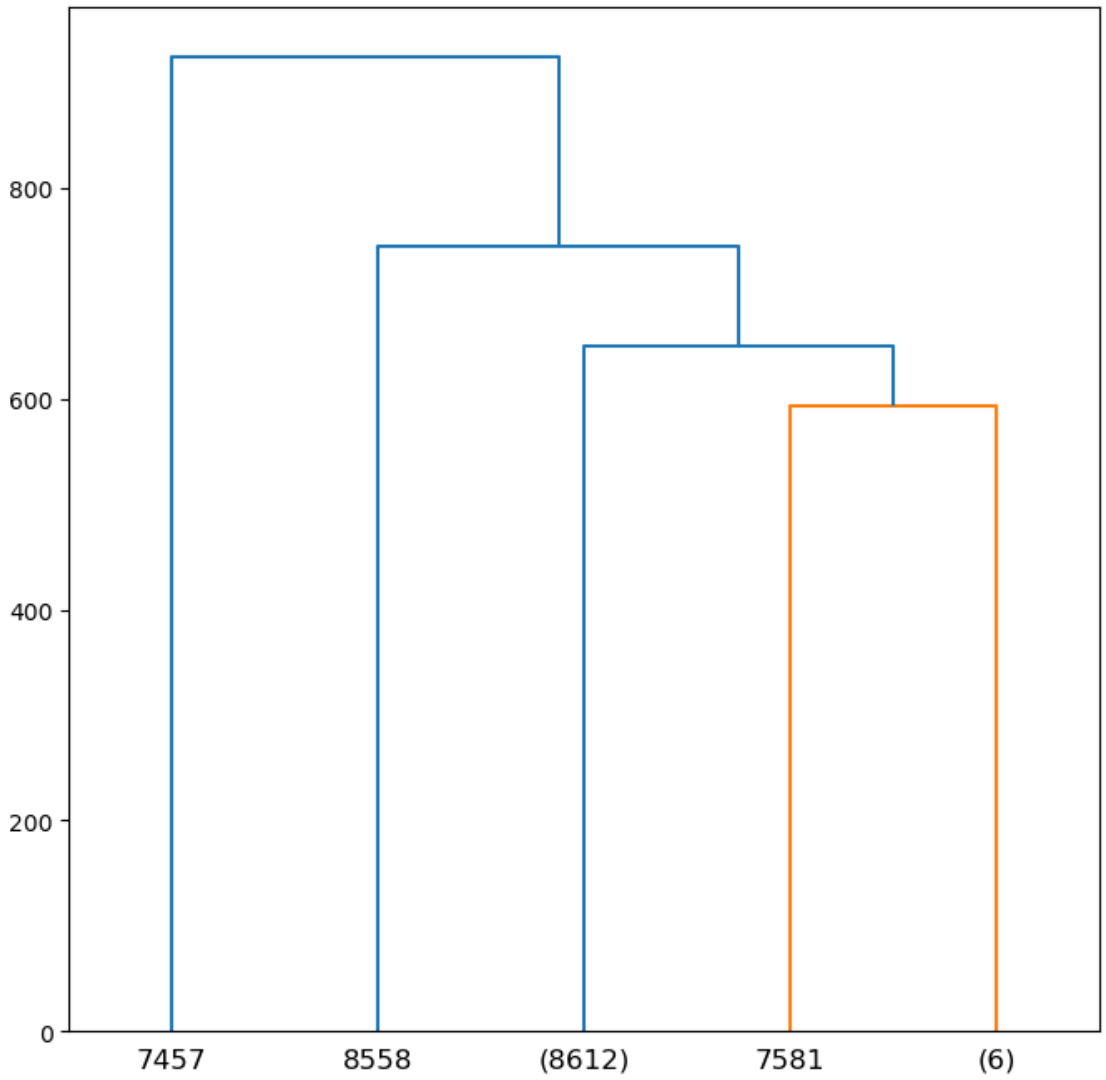
[8620 rows x 4 columns]

[ ]:
```
linkage_df.tail(5)
```

[ ]:

	Cluster1	Cluster2	distance	nombre_datapoints
8616	8171.0	17235.0	743.750630	8614.0
8617	7008.0	17236.0	802.945826	8615.0
8618	17232.0	17237.0	817.469877	8618.0
8619	5580.0	17238.0	921.364206	8619.0
8620	16578.0	17239.0	967.952478	8621.0

[229]:
```
plt.figure(figsize=(8,8))
dendro = hierarchy.dendrogram(matrice_liens,truncate_mode='lastp',p=5)
```



[370]:
```
agglomerative_model =  AgglomerativeClustering(n_clusters=2)
cluster_labels  = agglomerative_model.fit_predict(X_scaled)

print(classification_report(df['HasBought'], cluster_labels))
print(accuracy_score(df['HasBought'], cluster_labels))
```

```
              precision    recall  f1-score   support

           0       0.87      0.99      0.93      7287
           1       0.81      0.20      0.31      1335

    accuracy                           0.87      8622
   macro avg       0.84      0.59      0.62      8622
weighted avg       0.86      0.87      0.83      8622
```

0.86824402690791

[371]:
```
rand_agg = rand_score(df['HasBought'], cluster_labels)
acc_agg = accuracy_score ( df['HasBought'] , cluster_labels)
dunn_agg = 1-davies_bouldin_score(X_scaled , cluster_labels)
```

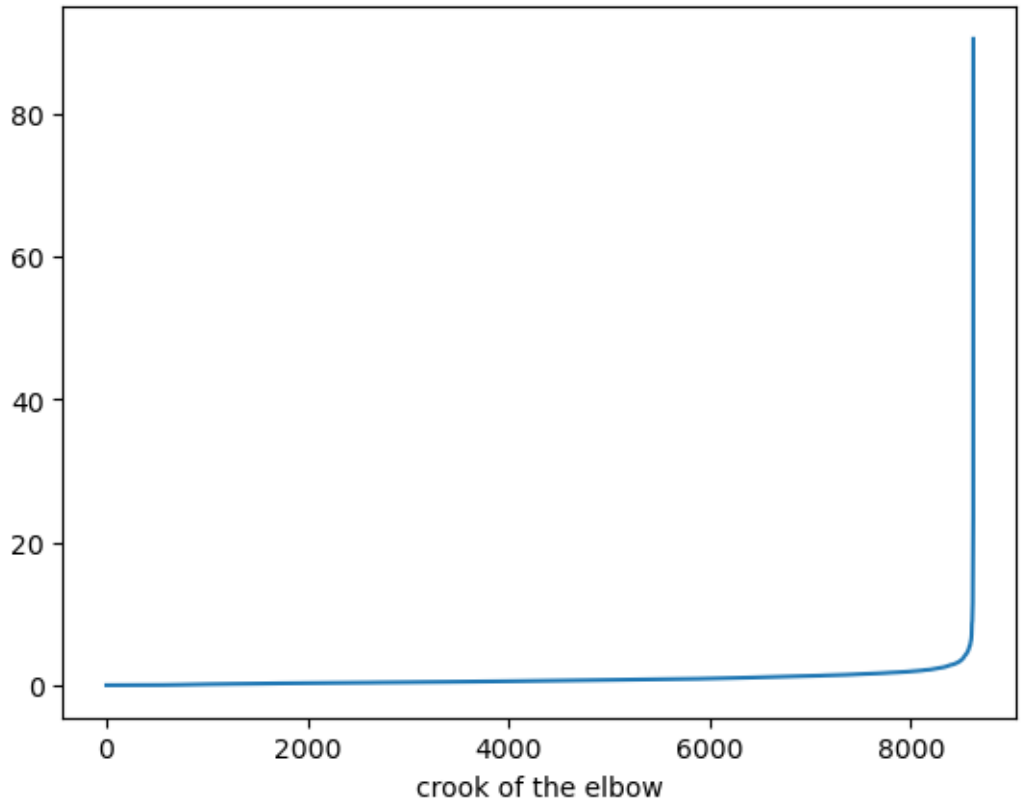
[372]:
```
pd.DataFrame(labels)[0].unique()
```

[372]:
```
array([1, 0])
```

1.6.3 DBSCAN CLUSTERING

[373]:
```
# Import Libraries
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from matplotlib import pyplot as plt

#  Calculer la distance moyenne entre chaque point des données et ses 32␣
 ↪proches voisins (32 = dim de data * 2 = 16 * 2)
voisins = NearestNeighbors(n_neighbors=32)
voisins_fit = voisins.fit(X_scaled)
distances, indices = voisins_fit.kneighbors(X_scaled)

# ploter le graphe
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)
plt.xlabel("crook of the elbow")
```

[373]:
```
Text(0.5, 0, 'crook of the elbow')
```



- d'apres la methode Crook of the elbow, eps est environ 2,3 ou 4

[498]:
```
# apres iterations sur chaque valeur de eps, on a trouvé optimum en eps = 3
dbscan = DBSCAN(eps=3, min_samples=32)
labels_db = dbscan.fit_predict(X_scaled)
```

[499]:
```
n_clusters = len(set(labels_db)) - (1 if -1 in labels_db else 0)

print("Number of clusters:", n_clusters)
```

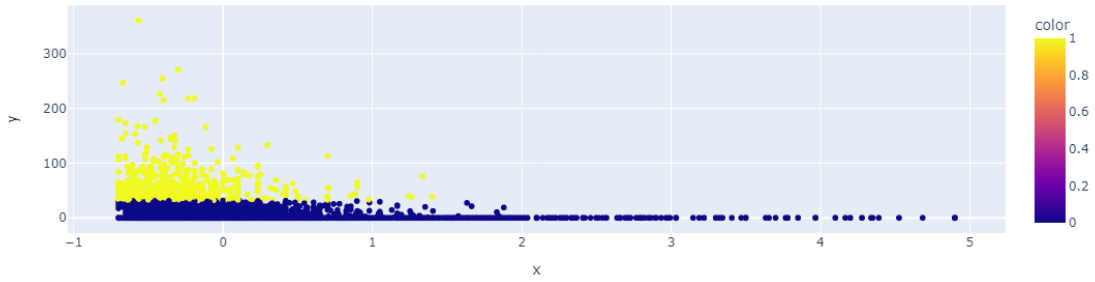
```
Number of clusters: 2
```

[500]:
```
counting=pd.DataFrame(labels_db)
counting[0].value_counts()
```

[500]:
```
 0    7881
-1     674
 1      67
Name: 0, dtype: int64
```

le modele DBScan detecte plusieur points aberrantes (674) ce qui va affecter sa precision

[501]:
```
import plotly.express as px
plt.figure(figsize=(8,8))
px.scatter(x=X_scaled[:,5],y=X_scaled[:,6],color = labels)
```




```
<Figure size 800x800 with 0 Axes>
```

[389]:
```
# Qualité du modele DBScan

rand_db = rand_score(df['HasBought'], labels_db)
acc_db = accuracy_score(df['HasBought'], labels_db)
dunn_db = 1-davies_bouldin_score(X_scaled , labels_db)

print('dunn index : ', dunn_db , ', rand index : ', rand_db , ', accuracy score␣
 ↪: ' , acc_db)
```

```
dunn index :  -0.5957614304330245 , rand index :  0.7526597175185525 , accuracy
score :  0.8159359777313848
```

1.7 Multiple Layer Perceptron

.

[134]:
```
from sklearn.model_selection import StratifiedKFold
from sklearn.pipeline import  Pipeline
from sklearn.preprocessing import RobustScaler
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from itertools import product

# cette cellule generer beacoup de warnings, du coup fallait cacher les␣
 ↪warnings avec ces deux lignes de codes.
import warnings
warnings.filterwarnings("ignore", category=UserWarning, module='sklearn')

# nombres de découpes dans la méthode GridSearchCV
```

```
découpes = StratifiedKFold(n_splits=5,shuffle=True,random_state =0)

# construction du modèle
DL_pipe  = Pipeline(steps = [['scaler', RobustScaler()],
                         ['classifier', MLPClassifier(verbose =␣
 ↪False,random_state=3, learning_rate_init=0.001, max_iter = 30)]])


params_DL  = {
              'classifier__learning_rate_init' : 10.0 ** -np.arange(1, 6),
              'classifier__hidden_layer_sizes' : [x for x in␣
 ↪product(range(15,25), range(10,20))]
             }
# GridSearch  -> parametres optimales.
grid = GridSearchCV(estimator = DL_pipe,
                    param_grid=params_DL,
                    cv = découpes ,
                    scoring = 'roc_auc')
grid.fit(X_train,y_train)

grid.best_params_
```

[134]:
```
{'classifier__alpha': 0.001,
 'classifier__hidden_layer_sizes': (19, 16),
 'classifier__random_state': 3}
```

- Apres qu'on a trouvé la combinaision des parametres optimale du modele, on essaye de le construire

[391]:
```
from sklearn.neural_network import MLPClassifier

clf = MLPClassifier(hidden_layer_sizes=(19, 16),          # 16 neurones dans␣
 ↪chaque couche (19 couches)
                    random_state=3,solver='adam' ,      # 'adam' refers to a␣
 ↪stochastic gradient-based optimizer
                    learning_rate_init=0.001)          # The initial learning␣
 ↪rate used. It controls the step-size in updating the weights
clf.fit(X_train_scaled, y_train)
# clf.predict(X_test)
```

[391]:
```
MLPClassifier(hidden_layer_sizes=(19, 16), random_state=3)
```

[392]:
```
yhat = clf.predict(X_test_scaled)

print(classification_report(y_test,yhat))
print(roc_auc_score(y_test,yhat))
```

```
              precision    recall  f1-score   support
```

```
              0       0.96      0.87      0.91      2186
              1       0.53      0.79      0.64       401

    accuracy                           0.86      2587
   macro avg       0.75      0.83      0.78      2587
weighted avg       0.89      0.86      0.87      2587

0.8308859598487769
```

2 Méthodologie d'évaluation

2.0.1 Supervisé :

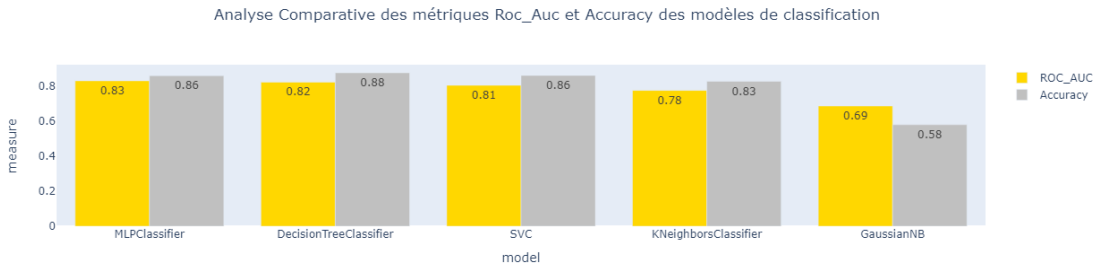
[445]:
```
evaluation =  [{'model':naive_bayes.__class__.__name__,'roc_auc_score':0.
 ↪686668,'f1_score_0':0.68,'f1_score_1':0.38,'accuracy':0.58},
               {'model':'MLPClassifier','roc_auc_score':0.830885,'f1_score_0':0.
 ↪91,'f1_score_1':0.64,'accuracy':0.86}
              ]
models = [knn,dt,svm]
for model in models :
    predictions = model.predict(X_test)
    f1_scores =  f1_score(y_test,predictions,labels=[0,1],average=None)
    evaluation.append({'model':model.named_steps['classifier'].__class__.
 ↪__name__,'roc_auc_score':roc_auc_score(y_test,predictions),'f1_score_0':␣
 ↪f1_scores[0],'f1_score_1':f1_scores[1],'accuracy' :␣
 ↪accuracy_score(y_test,predictions) })

evaluation = pd.DataFrame(evaluation)
evaluation.sort_values(by='roc_auc_score',ascending=False,inplace =True)
evaluation
```

[445]:

	model	roc_auc_score	f1_score_0	f1_score_1	accuracy
1	MLPClassifier	0.830885	0.910000	0.640000	0.860000
3	DecisionTreeClassifier	0.823184	0.925065	0.652126	0.876691
4	SVC	0.805100	0.915486	0.618337	0.861616
2	KNeighborsClassifier	0.776037	0.893209	0.558093	0.827986
0	GaussianNB	0.686668	0.680000	0.380000	0.580000

[467]:
```
### Comparaison des algorithmes d'apprentissage supervisé.

# Données à afficher dans le diagramme à barres
roc_auc_score = evaluation.roc_auc_score
accuracy = evaluation.accuracy

# Arrondir les valeurs à deux chiffres après la virgule
roc_auc_text = [round(x, 2) for x in roc_auc_score]
```

```
accuracy_text = [round(x, 2) for x in accuracy]

# Création du diagramme à barres avec les valeurs de chaque barre
data = [go.Bar(x=evaluation.model, y=roc_auc_score, marker=dict(color='gold'),␣
 ↪name='ROC_AUC', text=roc_auc_text),
        go.Bar(x=evaluation.model, y=accuracy, marker=dict(color='silver'),␣
 ↪name='Accuracy', text=accuracy_text)]

# Configuration de la figure
lay  = go.Layout(title='Analyse Comparative des métriques Roc_Auc et Accuracy␣
 ↪des modèles de classification',
                 title_x =0.5,
                 xaxis=dict(title='model',showgrid=False),
                 yaxis=dict(title='measure',showgrid=False))
fig= go.Figure(data = data ,layout=lay)

# Affichage du diagramme
pyo.iplot(fig)
```



2.0.2 Apprentissage non supervisé

[442]:
```
unsupervised = [kmeans_model,agglomerative_model,dbscan]
bilan = []
for model in unsupervised  :
    labels = model.fit_predict(X_scaled)
    bilan.append({'model':model.__class__.__name__, 'rand_score':
 ↪rand_score(df['HasBought'],labels),'dunn_index':
 ↪1-davies_bouldin_score(X_scaled , labels), 'accuracy_score' :␣
 ↪accuracy_score(df['HasBought'],labels)})
bilan = pd.DataFrame(bilan)
bilan.sort_values('rand_score',ascending = False,inplace=True)
bilan
```

[442]:

	model	rand_score	dunn_index	accuracy_score
0	KMeans	0.787831	0.504961	0.879378
1	AgglomerativeClustering	0.771181	0.553811	0.868244
2	DBSCAN	0.752660	-0.595761	0.815936

[471]:
```
## Comparaison des algorithmes d'apprentissage non supervisé.

# Données à afficher dans le diagramme à barres
rand_score = bilan.rand_score
accuracy_score = bilan.accuracy_score

# Formater les valeurs à deux chiffres après la virgule
rand_score_text = ["{:.2f}".format(x) for x in rand_score]
accuracy_score_text = ["{:.2f}".format(x) for x in accuracy_score]

# Création du diagramme à barres avec les valeurs de chaque barre
data = [go.Bar(x=bilan.model, y=rand_score, marker=dict(color='gold'),␣
 ↪name='Rand_Score', text=rand_score_text),
        go.Bar(x=bilan.model, y=accuracy_score, marker=dict(color='silver'),␣
 ↪name='Accuracy_Score', text=accuracy_score_text)]

# Configuration de la figure
lay   = go.Layout(title='Analyse Comparative des métriques Rand_Score et␣
 ↪Accuracy_Score des modèles de Clustering',
                  title_x =0.5,
                  xaxis=dict(title='model',showgrid=False),
                  yaxis=dict(title='measure',showgrid=False))
fig= go.Figure(data = data ,layout=lay)

# Affichage du diagramme
pyo.iplot(fig)
```