Groupe de statistique appliquée

Vincent Nguyen, *Baptiste* Pasquier, *Alix* Plamont *&* *Théo* Portalier

Sous la supervision d'Hallvard Geslin *et Louis* Pruvot-Caprioli

Octobre 2020 - Mai 2021

Construction d'un modèle de prédiction de clic sur des publicités en ligne avec contraintes sur la quantité de données et le temps d'évaluation

Remerciements

En premier lieu, nous tenons tout particulièrement à remercier les équipes pédagogiques de l'ENSAE Paris ainsi que la direction des études de l'école, lesquelles nous ont donné la chance à travers ce projet de mettre en pratique sur données réelles un grand nombre de compétences acquises au cours des deux premières années de notre formation. Dans le cadre de notre sujet, nous avons pu appliquer les fondements théoriques du *Machine Learning* qui nous ont été enseignés, mais aussi utiliser les outils d'analyse descriptive présentés lors des différents cours de statistique qui jalonnent le cursus. Les ressources mises à notre disposition, à l'image du Datalab de l'INSEE, nous ont également été d'un grand secours. Ce projet a aussi été pour nous le moyen de découvrir de nouveaux concepts et de mieux appréhender la poursuite de notre scolarité à l'ENSAE, tout en affinant notre projet professionnel.

Par ailleurs, il nous apparaît naturel de remercier nos deux encadrants, Hallvard Geslin et Louis Pruvot-Caprioli, pour leur disponibilité, leurs conseils toujours avisés et leur bienveillance. Les entretiens réguliers organisés avec eux ont constitué une très bonne opportunité de faire le point sur nos avancées et nos éventuelles questions et difficultés. L'expertise technique de Louis complétée par l'expérience d'Hallvard quant aux impératifs stratégiques et économiques de Criteo nous ont ainsi permis d'avancer sereinement pendant ces sept mois.

Table des matières

Introduction **4**

1 Exploration des données **5**
1.1 La base de données fournie . . . 5
1.1.1 Observations et variables . . . 5
1.1.2 Statistiques univariées . . . 5
1.2 Nettoyage et prétraitement des données . . . 7

2 Analyse descriptive **7**
2.1 Matrice des corrélations et réduction du nombre de variables . . . 7
2.2 Statistiques multivariées . . . 9
2.3 Apprentissage non supervisé . . . 10
2.3.1 Analyse en composantes principales (ACP) . . . 10
2.3.2 Clustering . . . 12

3 Métriques et modèles de classification binaire **14**
3.1 Choix et construction des métriques pertinentes . . . 14
3.2 Choix des modèles utilisés . . . 15
3.3 Méthodes de sur-échantillonnage . . . 15

4 Sélection des variables explicatives **16**
4.1 Méthode de la variance . . . 16
4.2 Univariate Feature Selection . . . 16
4.3 Recursive Feature Elimination with Cross-Validation (RFECV) . . . 16

5 Comparaison des modèles utilisés **18**
5.1 Optimisation des hyperparamètres et de la méthode de sur-échantillonnage . . . 18
5.2 Ajustement du seuil de décision . . . 19
5.3 Comparaison effective des modèles paramétrés sur la base de test . . . 19
5.4 Interprétation des modèles . . . 20
5.4.1 Régression logistique . . . 20
5.4.2 Random Forest et XGBoost . . . 21

6 Réduction des données et contraintes temporelles **22**

Conclusion **24**

Annexes **25**
A Descriptif des variables . . . 25
B Autres statistiques multivariées . . . 28
C Fondements théoriques des modèles utilisés . . . 29
C.1 Régression logistique . . . 29
C.2 Random Forest . . . 29
C.3 Boosting . . . 31
D Figures . . . 33
E Descriptif des paramètres utilisés pour chaque modèle . . . 39

Bibliographie **41**

Table des figures **42**

Liste des tableaux **43**

Introduction

Notre projet de statistique appliquée est encadré par l'entreprise Criteo, spécialiste français du ciblage publicitaire d'envergure internationale, par l'intermédiaire d'Hallvard Geslin et Louis Pruvot-Caprioli. Le cœur de métier de Criteo consiste à acheter des emplacements publicitaires via des systèmes d'enchères pour une firme cliente (Fnac, La Redoute...), laquelle rémunère ensuite Criteo en fonction du nombre de clics sur l'affichage publicitaire. Ainsi, dans la mesure où ses gains dépendent du nombre de clics, Criteo supporte l'intégralité du risque ; c'est pourquoi il lui est essentiel de mettre en place des algorithmes de prédiction de clic afin de maximiser sa rentabilité et donc de n'acheter que des affichages ayant une haute probabilité de donner lieu à un clic de la part de l'utilisateur. Notre mission vise donc à tester différents modèles de *Machine Learning* permettant de prédire ces clics à partir d'une base de données fournie par l'entreprise et correspondant à l'historique d'affichage de publicités de certains clients de Criteo (plus de 2 millions d'observations) en octobre 2020. Nous comparerons les performances de ces modèles ainsi que le temps d'exécution des algorithmes, en prêtant une attention particulière au temps de prédiction, lequel se doit d'être minimal pour que Criteo puisse enchérir en quelques millisecondes.

Après une présentation détaillée de la base de données fournie et de quelques variables clefs, puis un retour sur les divers nettoyages et prétraitements auxquels nous avons eu recours, nous réaliserons une analyse descriptive approfondie de notre table afin de dégager des variables explicatives des clics et quelques tendances remarquables. Nous procéderons ensuite à un exposé des outils utilisés, qu'il s'agisse des modèles de *Machine Learning*, des métriques d'évaluation de leur performance ou des méthodes de rééchantillonnage, indispensables dans la mesure où notre jeu de données est très déséquilibré en termes de classes. Nous pourrons alors évoquer les techniques de sélection des variables employées, puis effectuer la comparaison des différents modèles afin de déterminer les plus performants selon la métrique de référence préalablement choisie pour mesurer la qualité de la prédiction, sans pour autant oublier la nécessité de minimiser le temps de prédiction. Cette étape de comparaison sera l'occasion d'interpréter nos modèles et de faire le lien avec l'analyse descriptive préalable. Dans une dernière partie, nous étudierons l'influence de la quantité de données d'apprentissage sur les performances des modèles choisis et leurs temps d'évaluation en réduisant le nombre d'observations à différents sous-échantillons de notre base de données.

1 Exploration des données

1.1 La base de données fournie

1.1.1 Observations et variables

La base de données qui nous a été fournie par Criteo est issue de l'enregistrement de certaines données de navigation internet d'utilisateurs pour lesquels le cookie de l'entreprise a été installé sur leur navigateur, ou encore de données recueillies par des applications pour mobile. Chaque observation de la base correspond à un utilisateur naviguant sur un site internet ou une application dont Criteo a remporté un emplacement publicitaire ; une observation correspond donc à l'affichage effectif d'une publicité sur l'écran d'un appareil (ordinateur, téléphone, tablette...) pour un utilisateur donné. Les données à notre disposition ont été récoltées du mardi 13 octobre 2020 (00h00) au mercredi 21 octobre 2020 (00h32) et forment au total 2 135 241 observations. Elles ne sont pas tout à fait brutes : elles sont d'abord passées par un filtre de l'entreprise conçu pour détecter les bots (programmes pouvant cliquer automatiquement sur des publicités) et certaines données sont calculées.

Cette base nous a été transmise sous la forme de deux fichiers CSV distincts : l'un dédié à l'entraînement de nos modèles statistiques, l'autre dédié aux tests. Ils comportent respectivement 87% et 13% des données. Nous appellerons base d'entraînement les observations contenues dans le premier fichier, et base de test celles du second. La base d'entraînement contient toutes les observations jusqu'au mardi 20 octobre 2020 (00h53) et la base de test toutes les observations après cette date ; cette dernière base ne comporte donc que deux jours d'observations.

Chaque observation comporte 47 variables, dont un descriptif complet peut être trouvé en annexe A. Ces variables donnent des renseignements à la fois sur la publicité et l'enchère sous-jacente, sur l'utilisateur et sur les spécificités de la campagne publicitaire. Celles qui semblent *a priori* les plus susceptibles d'expliquer `is_display_clicked` (booléen indiquant si l'utilisateur a cliqué sur la publicité) renseignent par exemple sur :

- le degré d'engagement de l'utilisateur sur le site de l'annonceur (`contextid`) ;
- la taille de l'affichage publicitaire (`display_width` et `display_height`) ;
- la performance de la campagne publicitaire durant les dernières vingt-quatre heures (`campaignctrlast24h`) ;
- le coût de l'emplacement publicitaire (`zonecostineuro`).

Une part importante des variables comporte des informations sur l'historique d'activité de l'utilisateur (par exemple le nombre d'achats qu'il a déjà effectués auprès du site annonceur ou encore le nombre de jours écoulés depuis son dernier clic sur une publicité).

1.1.2 Statistiques univariées

Un certain nombre de variables pâtit d'une part élevée de valeurs manquantes ; c'est le cas en particulier pour les variables arborant le préfixe `ltf` pour lesquelles plus de 98% des observations n'ont pas de valeur. Parmi les autres variables lacunaires les plus notables, l'indice Google de visibilité de la page web (`googleviewability`), le nombre d'affichages de l'annonceur sur les appareils de l'utilisateur durant le dernier jour (`nbdisplaypartnerapprox_1d_sum_xdevice`) ou encore le nombre de fois que l'utilisateur a pu voir la publicité durant la dernière heure (`nbdisplay_1hour`) accusent respectivement d'une proportion de valeurs manquantes de 69%, 12% et 10%.

Les affichages décrits dans la base d'entraînement sont le résultat de la navigation de 1 138 732 utilisateurs (soit 1.6 affichage en moyenne par utilisateur). Les publicités proviennent de 10 annonceurs distincts et s'inscrivent dans 78 campagnes publicitaires différentes.

Sans surprise, le jeu de données est très déséquilibré : très peu d'affichages publicitaires sont cliqués (6%) ; la classe des observations comportant un clic est largement minoritaire.

Nous avons au début de notre travail – afin de nous approprier les données – tracé la distribution empirique de la plupart des variables de la base. Nous reportons ici la distribution des variables, énumérées dans la partie 1.1.1, ayant le plus fort potentiel explicatif.

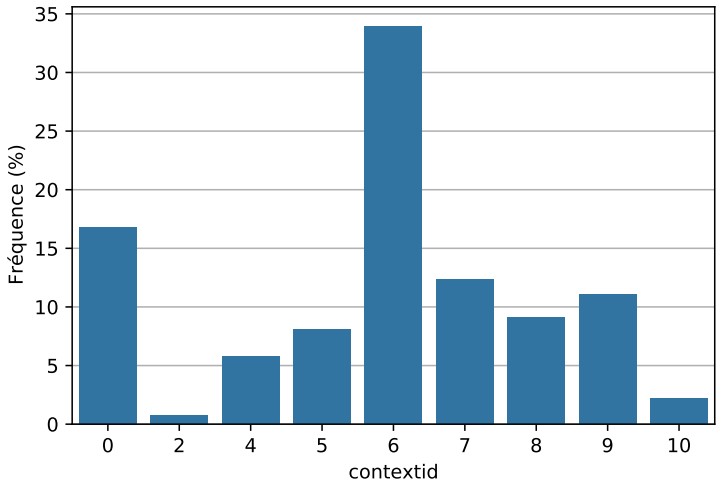


FIGURE 1 – Distribution empirique du degré d'engagement de l'utilisateur sur le site de l'annonceur.

Voir l'annexe A pour plus de détails concernant ces modalités.

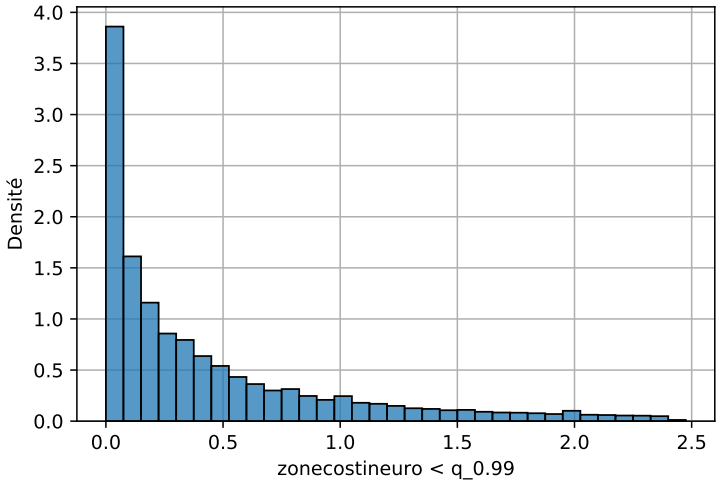
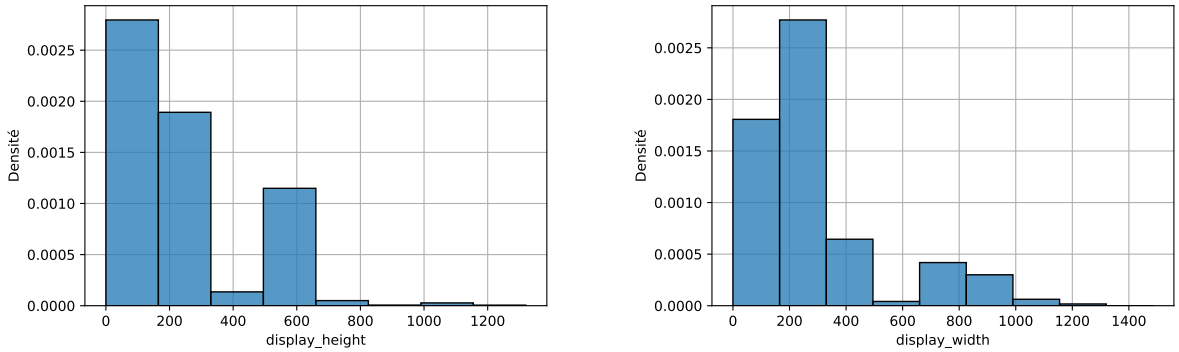


FIGURE 2 – Densité empirique du coût de l'emplacement publicitaire.

Le 1% des valeurs les plus élevées est omis : cette densité a un profil d'exponentielle décroissante.



(a) Hauteur en pixels de l'affichage.

(b) Largeur en pixels de l'affichage.

FIGURE 3 – Densités empiriques des deux variables renseignant sur la taille de la publicité.

1.2 Nettoyage et prétraitement des données

Afin de limiter par la suite les transformations sur les données, nous avons entrepris la création de deux nouvelles bases de données : les bases d'entraînement et de test prétraitées. Ce sont sur ces bases prétraitées que nous allons ensuite travailler.

Ces bases comprennent les modifications suivantes :

- Confrontés au nombre important de valeurs manquantes pour certaines variables, nous en avons référé à nos encadrants, qui nous ont alors suggéré de remplacer ces vides par des 0. Ces variables lacunaires correspondant à un nombre d'affichages, de ventes, etc., possiblement égal à 0 lorsque la valeur est manquante, ce choix nous a paru cohérent. Cette opération concerne les variables de préfixe `ltf`, `campaignctrlast24h` ainsi que les variables commençant par `nbdisplay`.
- Afin de pouvoir considérer la variable `contextid` comme quantitative, nous l'avons recodée de telle sorte à ce que le degré d'engagement d'un utilisateur sur le site de l'annonceur soit effectivement croissant avec les modalités de la variable (voir annexe A).

En outre, nous avons trouvé plus pertinent de travailler avec la taille totale de l'affichage publicitaire, plutôt que ses hauteur et largeur, et avons remplacé ces deux dernières variables par leur produit, donnant ainsi naissance à `display_size`. Cette variable n'est pas la seule que nous ayons créée : grâce à l'horodatage de l'affichage, nous avons codé `weekday` et `hour`, fournissant le jour et l'heure de l'affichage. Ces nouvelles variables ont été ajoutées au descriptif en annexe A.

Enfin, nous nous sommes aperçus que le filtre anti-bot de Criteo avait laissé passer des observations issues d'un utilisateur cliquant 16741 fois (bases d'entraînement et de test réunies). Nous avons supprimé les observations correspondantes.

2 Analyse descriptive

2.1 Matrice des corrélations et réduction du nombre de variables

Préalablement à la réalisation de statistiques descriptives multivariées, il nous est apparu essentiel de raisonner sur la matrice des corrélations (figure 4), laquelle représente les coefficients de corrélation de toutes les paires de variables quantitatives en valeur absolue. Ainsi, au sein d'un

groupe de variables très corrélées entre elles, nous avons fait le choix de n'en conserver qu'une seule. Ce cas de figure s'est présenté pour tous les groupes de variables dont les coefficients de corrélation absolus de chaque association étaient supérieurs à 0.8. Cela concerne essentiellement les variables de préfixe `ltf` : en effet, les variables associées à un nombre d'affichages, de ventes ou de clics sur les 4 dernières semaines (autrement dit de suffixe `4w`) sont généralement très corrélées aux mêmes variables sur les 90 derniers jours (de suffixe `90d`). C'est le cas par exemple des variables `ltf_nbpartnerdisplay_4w` et `ltf_nbpartnerdisplay_90d`, dont la corrélation absolue est proche de 1 : ici, nous avons arbitré en faveur de `ltf_nbpartnerdisplay_90d` dans la mesure où le nombre d'affichages publicitaires provenant de l'annonceur pour un utilisateur est plus conséquent sur 90 jours que sur 4 semaines. Nous avons donc pu, en étudiant les corrélations de chaque variable, réduire la taille de notre base.

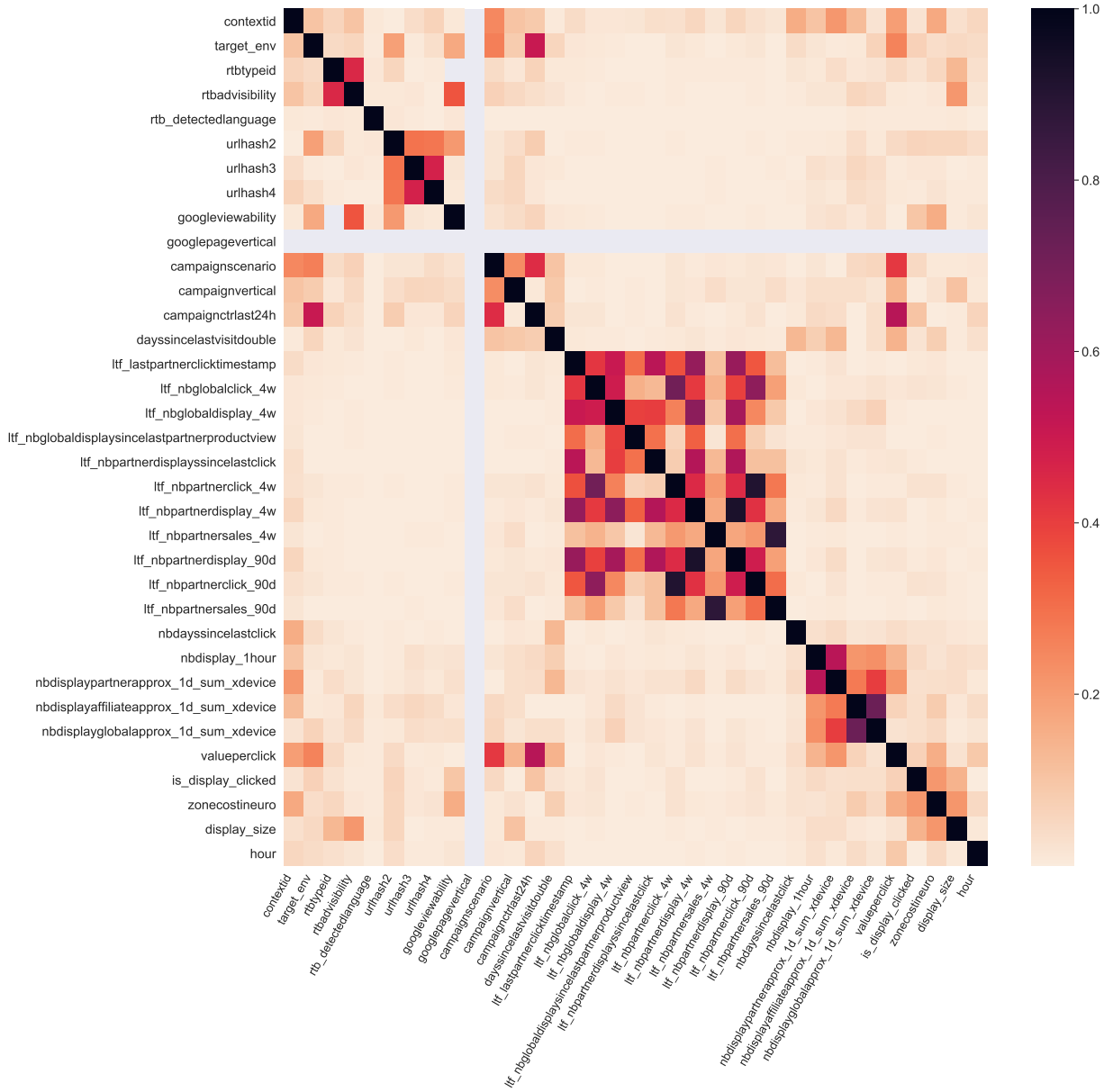


FIGURE 4 – Matrice des corrélations de toutes les variables quantitatives de la base.

En ce qui concerne la variable à prédire `is_display_clicked`, nous constatons qu'elle est très peu corrélée avec l'ensemble des variables quantitatives de la base (coefficients de corrélation

absolus inférieurs à 0.2), à l'exception de la taille de la publicité `display_size` et du coût de l'emplacement publicitaire `zonecostineuro`, dans une proportion tout de même assez peu significative. La matrice des corrélations ne nous permet donc pas de dégager à ce stade de grandes tendances explicatives des clics.

Variables quantitatives	**Variables catégorielles**
display_size	display_env
nbdayssincelastclick	target_env
campaignctrlast24h	campaignscenario
nbdisplay_1hour	campaignvertical
zonecostineuro	dayssincelastvisitdouble
dayssincelastvisitdouble	is_interstitial
ltf_nbglobaldisplay_4w	device_type
ltf_nbpartnerdisplay_90d	weekday
ltf_nbpartnerclick_90d	hour
ltf_nbpartnersales_90d	
ltf_nbpartnerdisplayssincelastclick	
contextid	
nbdisplayglobalapprox_1d_sum_xdevice	

TABLE 1 – Variables présélectionnées après analyse des corrélations et des valeurs manquantes.

Parallèlement, nous avons également pris la décision de retirer certaines variables qui nous semblaient peu pertinentes, notamment en raison d'une faible quantité de données, à l'instar du type de l'annonceur (`googlepagevertical`) ou de l'information sur la visibilité de la page (`googleviewability`), très souvent non fournie par Google. Concernant les variables catégorielles, nous n'avons retiré que le pays de l'utilisateur (`user_country`), dans la mesure où l'immense majorité des utilisateurs sont français et où un certain nombre de pays n'apparaissent qu'une seule fois. La table 1 ci-dessus recense les variables que nous avons retenues à la suite de cette analyse.

2.2 Statistiques multivariées

Afin de visualiser plus finement les éventuelles corrélations entre les variables explicatives et la variable à prédire `is_display_clicked`, nous avons réalisé des statistiques bivariées en traçant le taux de clic observé en fonction des variables quantitatives présélectionnées de la table 1. Pour avoir une bonne estimation de ce taux de clic en fonction des variables quantitatives continues, nous avons discrétisé ces dernières en leur attribuant dix modalités croissantes (il s'agit d'un découpage de l'ensemble des valeurs observées des variables en dix intervalles de même amplitude). Ces nouvelles modalités sont numérotées de 0 à 9 ; lorsque le taux de clic est représenté en fonction d'une seule variable, le numéro de la modalité est remplacé par la valeur centrale de l'intervalle de discrétisation.

Les graphiques les plus éloquents ont été obtenus pour les variables citées dans la partie 1.1.1 ; on constate effectivement un lien positif entre le taux de clic et ces quatre variables, même si le lien avec `contextid`, comparativement aux autres, semble faible (voir figure 5).

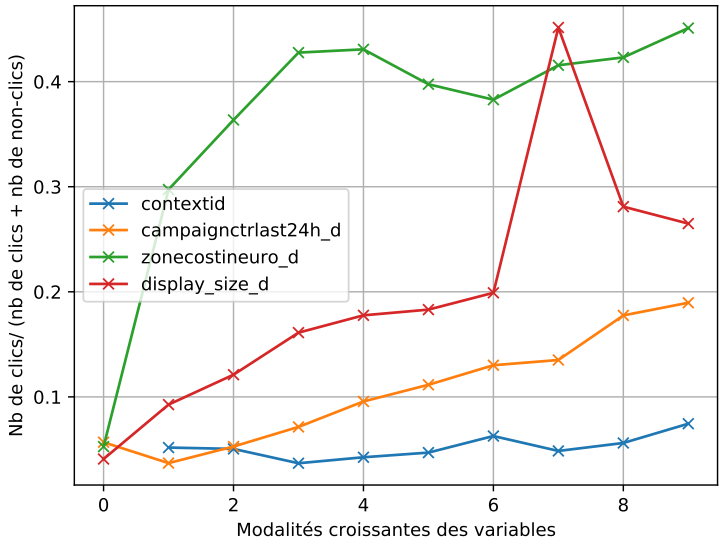


FIGURE 5 – Tracé du taux de clic observé en fonction des modalités croissantes de quatre variables quantitatives discrète (`contextid`) ou discrétisées (`campaignctrlast24h`, `zonecostineuro` et `display_size`).

Pour les autres variables, le profil des courbes est plus ambigu et laisse supposer des liens plus complexes. C'est le cas par exemple des variables de la table 1 indiquant le nombre des derniers affichages de l'utilisateur, et dont les courbes présentent un motif en « U » : on peut supposer que l'utilisateur est d'abord lassé par la publicité, puis que l'effet de répétition reprend le dessus et augmente la probabilité de clic. Ces graphiques sont disponibles en annexe D (figure 20).

Ces tracés nous auront cependant confortés dans notre présélection de variables : le taux de clic varie *a priori* significativement avec chacune d'elles. Nos variables semblent donc contenir un pouvoir explicatif intéressant de `is_display_clicked`, que nous allons pouvoir exploiter dans nos futurs modèles.

D'une façon plus marginale, nous avons poursuivi l'analyse multivariée en produisant d'autres types de graphiques. Un complément est disponible en annexe B.

2.3 Apprentissage non supervisé

2.3.1 Analyse en composantes principales (ACP)

Dans le but premier de visualiser les observations, et le but second d'étudier la possibilité d'obtenir une variable synthétisant la propension à cliquer, nous avons entrepris une analyse en composantes principales (ACP). Nous nous sommes concentrés pour cette analyse sur les variables quantitatives prétraitées, présélectionnées en table 1 et préalablement centrées et réduites – afin d'éviter les effets d'échelle.

Nous avons ainsi créé treize nouvelles variables (les axes factoriels), décorrélées entre elles et expliquant chacune une part de la variance totale. Le graphique donnant la décomposition de cette variance sur chacun des axes (figure 6) montre que le premier axe se distingue largement des autres tandis que la variance expliquée par les axes suivants semble décroître linéairement jusqu'au treizième. Cette constatation est cohérente avec ce que nous avons vu dans la matrice des corrélations : le premier axe reprend en fait l'information contenue dans les variables de préfixe

`ltf` (voir figure 21 en annexe D), qui forment un bloc de variables très corrélées, alors que les variables quantitatives restantes sont globalement peu corrélées entre elles.

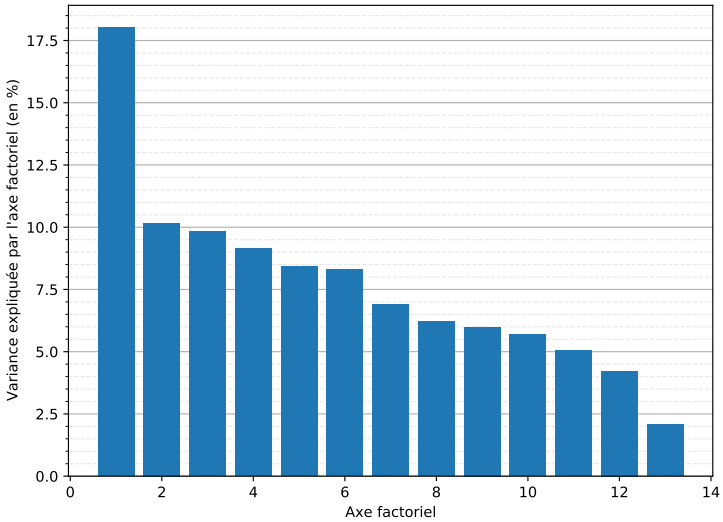


FIGURE 6 – Proportion de la variance expliquée par chacun des axes factoriels obtenus par ACP sur les variables quantitatives de la table 1.

Les variables `ltf` comportent plus de 98% de 0 : étant donnée l'importance de l'axe 1 par rapport aux autres, il en résulte que les observations sont toutes entassées autour de la droite d'équation $axe1 = 0$ sans que l'on puisse discriminer les clics des non-clics (voir figure 22 en annexe D). L'axe 1 ne nous intéressera donc pas pour notre étude.

Pour déterminer les axes qui nous intéressent, et ensuite projeter les observations dans les plans correspondants, nous avons entrepris de tracer la distribution des observations sur différents axes selon le clic, afin d'ensuite nous concentrer uniquement sur les axes dont les deux distributions ne se superposent pas. La différence la plus marquée entre les deux distributions se situe au niveau de l'axe 2 (voir figure 7) : la queue de distribution est plus épaisse pour les clics, qui en moyenne ont une valeur plus élevée sur cet axe. Dans une moindre mesure, l'axe 7 et l'axe 11 partagent les mêmes propriétés que l'axe 2 (voir figure 23 en annexe D) – d'une façon nettement atténuée pour l'axe 11. Pour les autres axes, les deux courbes se superposent.

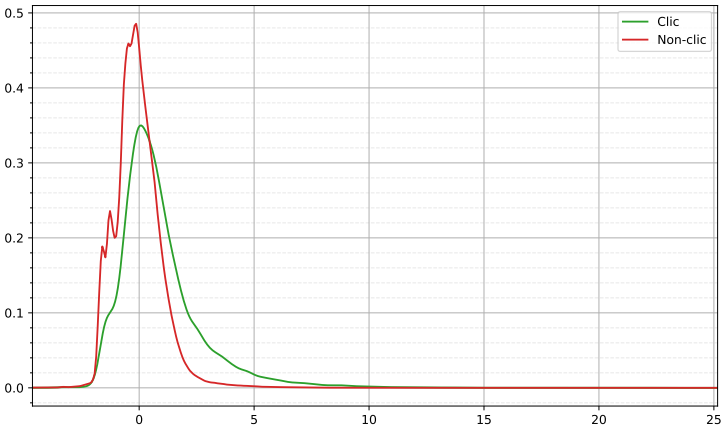


FIGURE 7 – Distributions des données sur l'axe 2 obtenu par ACP sur les variables quantitatives de la table 1 selon le clic.

Nous nous sommes donc concentrés sur les trois plans factoriels (axe 2, axe 7), (axe 2, axe 11) et (axe 7, axe 11). Pour plus de clarté, nous y avons projeté un nombre restreint d'observations (choisies aléatoirement) égal à 20% du nombre total de clics : la première moitié des points correspondant à un clic, l'autre à un non-clic. Ces projections sont présentées en figure 8 et en figure 24 (annexe D).

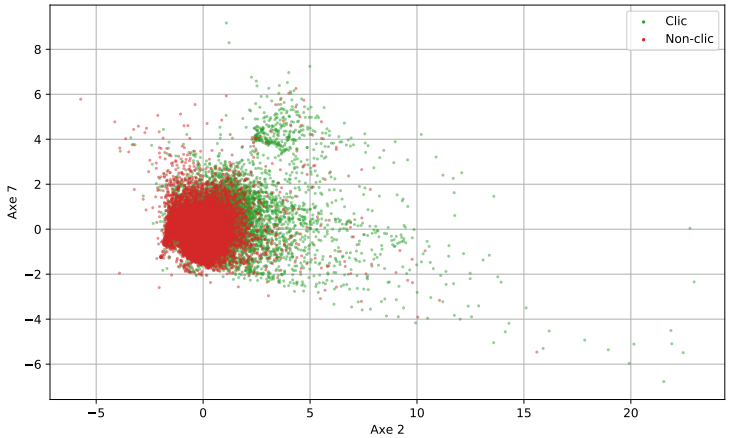


FIGURE 8 – Projection des observations dans le plan factoriel (axe 2, axe 7) obtenu par ACP sur les variables quantitatives de la table 1.

Pour chaque classe, un nombre de points égal à 10% des clics totaux est représenté.

Globalement, dans les plans (axe 2, axe 7) et (axe 2, axe 11), les clics se distinguent au nord-est et au sud-est ; dans le plan (axe 7, axe 11) c'est au nord-ouest et au sud-est. L'analyse du cercle des corrélations pour le premier de ces plans (voir figure 9) montre que ces directions sont assez fortement liées (positivement) aux variables `campaignctrlast24h`, `display_size`, `zonecostineuro` et `contextid`, variables déjà évoquées dans la partie 1.1.1. En analysant les cercles des deux autres plans (voir figure 25 en annexe D) on remarque en plus l'intervention – plus marginale – des variables `nbdisplay_1hour` et `nbdisplayglobalapprox_1d_sum_xdevice` qui semblent avoir un lien positif avec le clic. L'utilisateur aurait donc tendance à cliquer sur des publicités qu'il a déjà souvent vues.

Cette ACP, même si elle permet d'observer un lien entre certaines variables et la propension à cliquer, ne permet pas facilement d'en obtenir une variable synthétique. L'axe 2 discrimine plutôt bien les clics des non-clics mais cela reste insuffisant : les deux distributions présentées en figure 7 ne sont pas suffisamment différentes et ne semblent pas plus intéressantes que les variables originelles pour notre étude. Néanmoins, cette analyse nous aura confirmé l'importance des variables évoquées en partie 1.1.1. On s'attend à ce que ces variables soient utiles pour nos modèles et sélectionnées par les différentes méthodes que nous envisagerons.

2.3.2 Clustering

En complément de l'analyse en composantes principales précédente, nous avons également procédé à un clustering basé sur la méthode des k-moyennes, appliquée aux variables quantitatives centrées réduites de la table 1. Cette étape vise à discriminer les clics des non-clics et ainsi à dégager des caractéristiques qui seraient propres à chacune des deux classes.

Dans un premier temps, il nous a fallu déterminer le nombre optimal de clusters pour nous

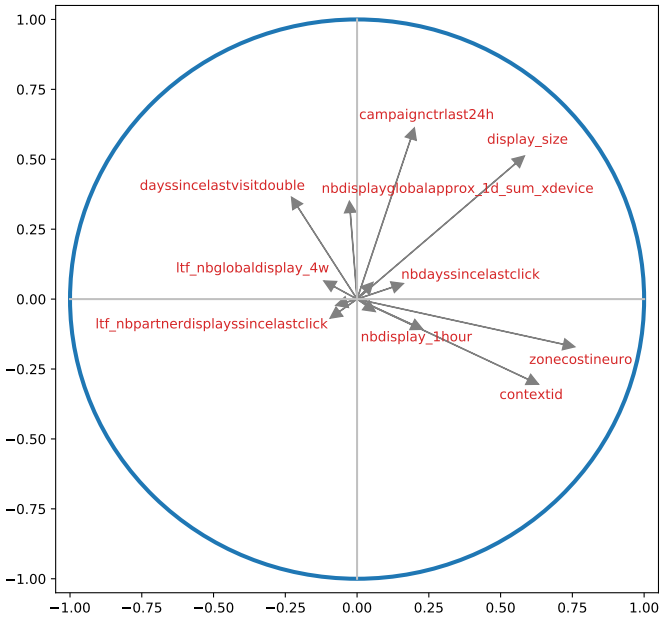


FIGURE 9 – Cercle des corrélations du plan factoriel (axe 2, axe 7) obtenu par ACP sur les variables quantitatives de la table 1.

assurer que le partitionnement en deux clusters était bien le plus judicieux. C'est pourquoi nous avons tracé le score moyen de silhouette en fonction du nombre de clusters (en faisant varier celui-ci dans un intervalle réaliste, compris ici entre 2 et 10) ; le nombre optimal de clusters étant associé au maximum de ce score. D'un point de vue théorique, la métrique silhouette évalue grâce à un coefficient (compris entre -1 et 1) propre à chaque point la différence entre la distance moyenne de ce point avec les points du même cluster et la distance moyenne de ce même point avec les points des autres clusters. Si le point est bien placé, il doit donc être plus proche en moyenne des points de son cluster ; ceci se matérialise par un coefficient de silhouette proche de 1. A l'inverse, un coefficient proche de -1 témoigne d'un mauvais partitionnement. Dans notre cas, le score de silhouette est une moyenne du coefficient pour l'ensemble des observations. Le graphique ci-dessous confirme que le nombre optimal de clusters est bien 2.

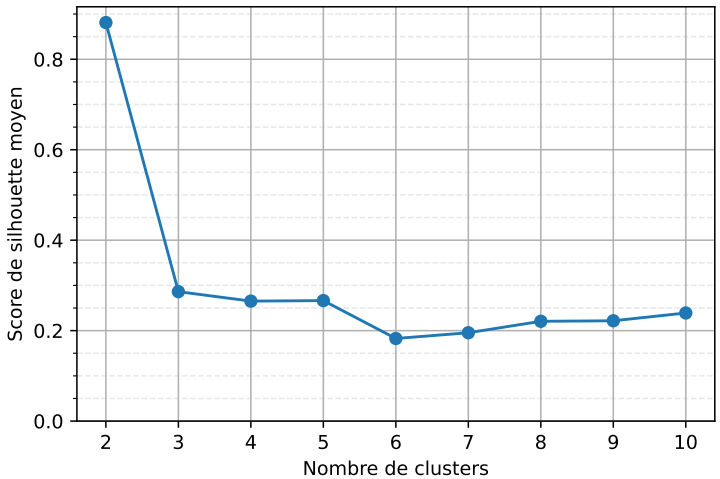


FIGURE 10 – Coefficient de silhouette moyen en fonction du nombre de clusters désiré.

Nous avons alors appliqué la méthode des k-moyennes pour 2 clusters, et avons obtenu des groupes de tailles respectives 6 055 et 1 604 439. Cette répartition ne semble ainsi pas distinguer les clics des non-clics. En effet, dans chacun des clusters, la proportion de clics s'élève à environ 6%, soit le même taux que dans la base initiale. Plus largement, quel que soit le nombre de clusters, aucune configuration n'a vraiment permis de discriminer les observations selon la valeur de la variable `is_display_clicked`. Il n'est donc pas possible de dresser un profil type d'observation caractérisant les clics au moyen de cette méthode de clustering.

3 Métriques et modèles de classification binaire

Préalablement à la sélection des variables explicatives dans le cadre de l'implémentation de nos modèles, il nous apparaît essentiel de définir les fondements théoriques des différents modèles auxquels nous aurons recours et de définir une métrique de référence, d'autant plus que certaines méthodes de sélection dépendent des modèles et de la métrique choisie pour mesurer la performance.

3.1 Choix et construction des métriques pertinentes

Afin de comparer les modèles que nous utiliserons en termes de performances brutes et de temps d'exécution, il est fondamental de déterminer préalablement une métrique de référence. Nous allons pour cela nous appuyer sur la matrice de confusion associée aux modèles de classification binaire.

		Classe prédite	
		Non-clic	Clic
Classe réelle	Non-clic	Vrai négatif TN	Faux positif FP
	Clic	Faux négatif FN	Vrai positif TP

TABLE 2 – Matrice de confusion, mesurant la qualité d'un classifieur.

Pour une entreprise comme Criteo, rémunérée au clic, prédire un faux positif (on perd alors le coût de l'emplacement publicitaire) est moins coûteux que la prédiction d'un faux négatif (on perd la rémunération du clic de l'utilisateur) ; autrement dit, il faut surtout faire en sorte d'éviter de prédire des non-clics qui seraient en réalité des clics, quitte à prédire trop de clics. Pour cela, on prêtera une grande attention au *recall*, afin de limiter les erreurs de type II. Sa formule est donnée par

$$recall = \frac{TP}{TP + FN}$$

et indique donc la proportion de clics effectivement prédits parmi l'ensemble des « vrais » clics.

La prédiction de faux positifs ayant tout de même un coût, et un classifieur ne prédisant que des 1 ayant un *recall* de 100%, il apparaît pertinent de s'appuyer sur une autre métrique. La *precision*, elle, va donner plus d'importance à la véracité de la prédiction. Elle est en effet définie comme la proportion de clics bien prédits parmi l'ensemble des clics prédits.

Néanmoins, nous ne pouvons pas non plus nous appuyer sur le simple score de *precision* car,

comme dit plus haut, prédire un clic à tort est moins grave qu'en omettre un.

Le F_1 score, combinaison du *recall* et de la *precision*, conviendrait mieux à notre étude, mais celui-ci accorde une importance égale aux deux métriques. C'est la raison pour laquelle nous avons finalement opté pour le F_β score, défini par

$$F_\beta = \left(1 + \beta^2\right) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

et qui permet d'assigner plus ou moins d'importance au *recall* par rapport à la *precision* en contrôlant le paramètre β.

L'entreprise Criteo a ainsi requis que nous utilisions le score F_3, lequel confère trois fois plus d'importance au *recall* qu'à la *precision* et répond bien aux impératifs économiques. Il s'agira désormais de notre métrique de référence.

3.2 Choix des modèles utilisés

Afin de réaliser nos prédictions, nous utiliserons dans la suite trois modèles de *Machine Learning* et comparerons leurs performances et leurs temps d'évaluation. Il s'agit de la régression logistique, des forêts aléatoires et du modèle XGBoost, qui repose sur le principe du *gradient boosting*. Nous présentons les fondements théoriques de ces modèles en annexe C.

3.3 Méthodes de sur-échantillonnage

Nous faisons face à un jeu de données extrêmement déséquilibré, au sein duquel la classe minoritaire (modalité 1 de la variable cible, associée au clic) représente à peine 6% des observations de la base. Ce type de déséquilibre, au-delà du ciblage publicitaire, est également présent dans des problématiques telles que la détection de fraudes. Dans une telle configuration, les modèles de *Machine Learning* que nous allons utiliser tendent à classer naïvement une immense majorité des observations dans la classe majoritaire et présentent ainsi une faible performance pour les métriques pertinentes d'un point de vue métier, à savoir dans notre cas le score F_3. Le rééchantillonnage est alors l'une des méthodes les plus régulièrement utilisées pour résoudre ce problème. Nous appliquerons ici deux procédures de sur-échantillonnage (ou *oversampling*) classiques afin d'augmenter la proportion de clics dans notre base.

La plus simple d'entre elles, implémentable via la méthode `RandomOverSampler` du package `imbalanced-learn`, consiste à répliquer des observations de la classe minoritaire afin d'atteindre un certain ratio de clics par rapport aux non-clics, choisi grâce au paramètre `sampling_strategy`. Cette méthode, naïve dans le sens où elle ne tient pas compte des données en elles-mêmes, présente parfois un risque de surapprentissage (ou *overfitting*) que des approches plus complexes peuvent limiter. C'est le cas de la méthode SMOTE (Synthetic Minority Oversampling TEchnique), qui opère en sélectionnant une observation de la classe minoritaire et un nombre fixé de ses plus proches voisins (généralement compris entre 3 et 5). En fonction du ratio de clics désirés, la différence entre chaque caractéristique de l'observation sélectionnée et la caractéristique correspondante de certains de ses voisins sera calculée, puis multipliée par un nombre entre 0 et 1. Le résultat est ensuite ajouté à la caractéristique de l'observation choisie et on obtient alors un nouveau vecteur de caractéristiques et donc une nouvelle observation synthétique de la classe minoritaire. Le processus est répété pour toutes les observations de la classe minoritaire. En pratique, nous nous appuierons

sur la méthode SMOTENC, laquelle permet de traiter simultanément les variables quantitatives et catégorielles. Nous avons ainsi pu par exemple créer une base sur-échantillonnée contenant autant de clics que de non-clics (voir figure 27 en annexe D).

4 Sélection des variables explicatives

Avant de pouvoir implémenter nos modèles, il est essentiel de procéder à une phase de sélection des variables explicatives afin de réduire le volume de calculs à réaliser. Il existe différentes méthodes de sélection, certaines pouvant dépendre des modèles. Nous en étudierons ici trois et retiendrons la plus performante. Cette partie a ainsi vocation à compléter les premiers résultats fournis par l'analyse descriptive, à l'issue de laquelle nous avons pu déjà éliminer un certain nombre de variables de notre étude, notamment via la matrice des corrélations. La sélection se fait désormais sur les variables restantes.

4.1 Méthode de la variance

Nous recourons dans un premier temps à la méthode de la variance, qui constitue l'une des méthodes de sélection de variables les plus basiques : elle consiste en effet à retirer l'ensemble des variables ayant une variance inférieure à un certain seuil, proche de 0. En fixant celui-ci à 0.01, l'immense majorité de nos variables quantitatives et catégorielles (recodées sous forme de variables indicatrices) présente une variance significative, à l'exception notable de la variable `campaignctrlast24h` et de quelques rares modalités de variables catégorielles. Le nombre de variables retenues resterait encore assez conséquent. Par ailleurs, les graphiques issus de l'analyse descriptive suggèrent que `campaignctrlast24h` a plutôt un fort pouvoir prédictif de la variable cible `is_display_clicked` : il semble donc pertinent de la conserver. Cette intuition sera confirmée par les méthodes suivantes.

4.2 Univariate Feature Selection

Cette méthode repose sur l'utilisation de tests statistiques. Nous avons choisi ici de nous appuyer sur le test de corrélation de Pearson, lequel renvoie indépendamment pour chaque variable son coefficient de corrélation avec la variable cible `is_display_clicked` ainsi que la p-valeur du test associé. Aux seuils usuels de 5% et 1%, il apparaît que seules les variables `ltf_nbglobaldisplay_4w` et `display_env_other` (modalité de la variable `display_env` recodée sous forme de variable indicatrice) ainsi que quelques modalités des variables `hour` et `weekday` ne sont pas significatives. Cette méthode de sélection semble trop parcimonieuse, il apparaît nécessaire de recourir à des méthodes dépendantes des modèles utilisés.

4.3 Recursive Feature Elimination with Cross-Validation (RFECV)

Contrairement aux méthodes précédentes, la méthode Recursive Feature Elimination (RFE) permet quant à elle de sélectionner les n meilleures variables pour un modèle donné. L'algorithme va récursivement trouver le meilleur sous-ensemble de variables, en enlevant progressivement des variables jusqu'à atteindre le nombre demandé. A chaque étape, le modèle est entraîné, puis les variables sont classées par importance, les variables les moins importantes étant écartées. Le score d'importance des variables provient soit directement du modèle (par exemple un arbre de décision)

soit d'une méthode statistique. Par ailleurs, l'utilisation de la validation croisée (méthode RFECV) permet dans un premier temps de déterminer le nombre de variables optimal en s'appuyant sur un score d'importance moyen reposant sur une métrique de référence. Ce nombre optimal peut alors être lu grâce à la règle du coude en traçant le score issu de cette validation croisée en fonction du nombre de variables. Nous pouvons alors appliquer la fonction `RFE` en indiquant en paramètre ce nombre optimal afin d'obtenir la liste des variables à sélectionner.

Nous appliquons donc cette méthode à nos trois modèles, toujours avec le score F_3 comme métrique de référence. Le graphique donné en figure 11 nous permet de faire l'hypothèse, via la règle du coude, que le nombre optimal de variables est 9 pour XGBoost, tandis que les graphiques des figures 28 et 29 disponibles en annexe D conduisent à 11 variables pour la régression logistique contre 9 pour Random Forest.

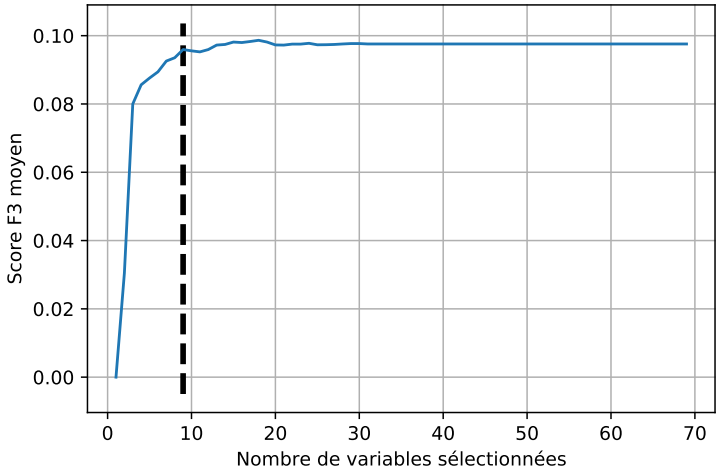


FIGURE 11 – Représentation du score F_3 moyen en fonction du nombre de variables sélectionnées par `RFECV` pour le modèle XGBoost.

On détermine ensuite ces variables par `RFE` et on obtient les sélections présentées en table 3.

Régression logistique	**Random Forest**	**XGBoost**
display_size	display_size	nbdayssincelastclick
nbdayssincelastclick	nbdayssincelastclick	campaignctrlast24h
campaignctrlast24h	campaignctrlast24h	nbdisplay_1hour
nbdisplay_1hour	nbdisplay_1hour	contextid
zonecostineuro	dayssincelastvisitdouble	zonecostineuro
device_type_Desktop	nbdisplayglobalapprox_1d_sum_xdevice	is_interstitial_True
display_env_app_ios	contextid	device_type_iPad
campaignscenario_13	zonecostineuro	device_type_Desktop
device_type_iPhone	is_interstitial_True	device_type_Android
target_env_2		
is_interstitial_True		

TABLE 3 – Variables sélectionnées par `RFECV` pour chaque modèle.

5 Comparaison des modèles utilisés

5.1 Optimisation des hyperparamètres et de la méthode de sur-échantillonnage

Les trois modèles que nous avons choisis possèdent de nombreux hyperparamètres, notamment les modèles qui utilisent des arbres : Random Forest et XGBoost. Le choix des meilleurs hyperparamètres est effectué par une méthode de *grid search* qui consiste à parcourir un ensemble de combinaisons de paramètres puis à sélectionner la meilleure combinaison selon le score F_3. La *grid search* est par ailleurs combinée à une méthode de validation croisée (`GridSearchCV`) pour évaluer de manière robuste la généralisation des modèles sur de nouvelles données.

Les paramètres utilisés pour chaque modèle figurent en annexe E. Nous avons par ailleurs calculé plusieurs `GridSearchCV` pour chaque modèle avec utilisation ou non des variables sélectionnées par la méthode RFECV et avec utilisation ou non d'une méthode de sur-échantillonnage. La méthode SMOTENC n'a été calculée que dans le cas de la sélection RFECV pour des raisons de temps de calcul car la méthode RFECV réduit considérablement le nombre de variables et donc la complexité du sur-échantillonnage.

La table 4 présente les performances des meilleures combinaisons de paramètres obtenues pour chaque modèle et chacune des leurs variations (RFECV et sur-échantillonnage). Les couleurs permettent de visualiser la répartition des scores (le rouge correspond aux cinq plus mauvais scores dans chaque colonne, le jaune aux cinq scores intermédiaires, et le vert aux cinq meilleurs scores). Les trois lignes du tableau en caractères gras correspondent aux trois variations des modèles que nous utiliserons dans la suite de l'étude, sélectionnées selon un compromis entre le score F_3 et les temps de prédiction et d'apprentissage.

Modèle	**RFECV**	**Sur-échantillonnage**	**Score F_3**	**Temps d'apprentissage (s)**	**Temps de prédiction (s)**
Régression logistique	Non	Non	0.483	10.63	0.03
		RandomOver	0.484	18.55	0.03
	Oui	**Non**	**0.481**	**2.30**	**0.01**
		RandomOver	0.482	3.70	0.01
		SMOTENC	0.482	305.54	0.01
Random Forest	Non	Non	0.519	89.55	1.04
		RandomOver	0.517	123.85	1.13
	Oui	**Non**	**0.510**	**67.21**	**0.81**
		RandomOver	0.508	97.61	0.89
		SMOTENC	0.503	296.63	0.77
XGBoost	Non	Non	0.546	18.32	0.42
		RandomOver	0.545	41.85	0.43
	Oui	**Non**	**0.538**	**12.15**	**0.09**
		RandomOver	0.540	27.28	0.10
		SMOTENC	0.524	282.82	0.09

TABLE 4 – Meilleurs scores F_3 et durées d'apprentissage et de prédiction associées obtenus par `GridSearchCV` sur différents modèles.

5.2 Ajustement du seuil de décision

Pour augmenter la performance de nos algorithmes, nous avons cherché à ajuster le seuil de décision. En effet, tous les algorithmes utilisés permettent de calculer la probabilité associée à chacune des deux classes (0 ou 1) pour chaque observation, et les classifient par rapport au seuil de décision fixé à 0.5 par défaut. Cependant, il est intéressant de faire varier le seuil de décision pour obtenir un meilleur compromis *recall*/*precision*, augmentant ainsi le score F_3.

Nous obtenons les résultats suivants, lisibles sur la figure 12 et la table 5, pour les meilleurs paramètres de chaque modèle déterminés lors des `GridSearchCV` précédentes. Concernant la régression logistique, faire varier le seuil de décision n'augmente pas significativement la performance. En effet, le seuil de décision à 0.5 est théoriquement optimal si la régression est bonne. De même, la différence de performance est mineure pour le modèle XGBoost. En revanche, elle est significative pour le modèle Random Forest, avec un seuil bien supérieur à 0.5. Finalement, dans l'optique de comparer effectivement nos modèles paramétrés sur la base de test, on choisit le seuil 0.49 pour la régression logistique, contre 0.47 pour le modèle XGBoost et 0.64 pour le modèle Random Forest.

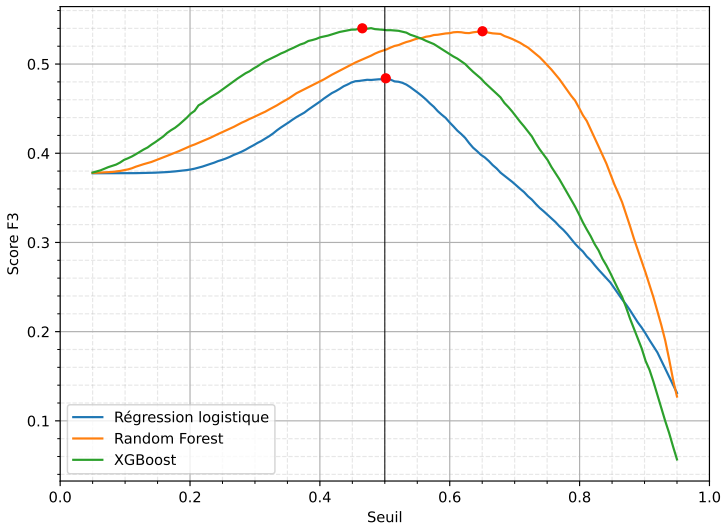


FIGURE 12 – Représentation du score F_3 en fonction du seuil de décision pour chacun des trois modèles choisis.

Modèle	Score F_3	Score F_3 ajusté
Régression logistique	0.482	0.482
Random Forest	0.514	0.535
XGBoost	0.535	0.536

TABLE 5 – Scores F_3 bruts et scores F_3 ajustés au moyen du seuil de décision pour les trois modèles choisis.

5.3 Comparaison effective des modèles paramétrés sur la base de test

Nous appliquons les trois modèles choisis sur la base de test avec les meilleures combinaisons de paramètres et les seuils ajustés déterminés dans les sous-parties précédentes. Les résultats sont présentés en table 6. Nous rejetons le modèle Random Forest, car il n'apporte pas de gain de

performance significatif permettant de compenser un temps de prédiction très long. En revanche, les deux autres modèles sont intéressants pour notre étude : la régression logistique offre une performance décente avec un score F_3 de 0.486 et un temps de prédiction très court de 0.01 seconde, tandis que le modèle XGBoost donne une bonne performance avec 0.535 pour le score F_3 et un temps de prédiction très correct de 0.06 seconde. La régression logistique présente cependant l'avantage d'être assez économe en ce qui concerne le temps d'apprentissage. Par ailleurs, nous constatons que les scores obtenus pour les trois modèles sont très proches de ceux observés pour la base d'entraînement (voir table 5), ce qui suggère l'absence de surapprentissage.

Modèle	**Score F_3**	**Temps d'apprentissage (s)**	**Temps de prédiction (s)**
Régression logistique	0.486	2.88	0.01
Random Forest	0.538	115.16	0.64
XGBoost	0.535	13.91	0.06

TABLE 6 – Comparaison effective des trois modèles choisis sur la base de test.

5.4 Interprétation des modèles

À présent que nous avons sélectionné les trois meilleurs modèles pour notre étude, nous allons donner des éléments d'interprétation de chaque modèle. Cela permettra de faire le lien avec l'analyse descriptive effectuée en section 2, de vérifier la cohérence des modèles et surtout, pour un projet d'entreprise comme le nôtre, de gagner en transparence.

5.4.1 Régression logistique

Notre modèle de régression logistique étant régularisé et pondéré (voir l'équation (1) en annexe C.1), il n'est pas possible d'utiliser les statistiques habituelles afin de construire des tests de significativité ou des intervalles de confiances sur les coefficients. Nous nous contentons ici d'interpréter qualitativement et de comparer l'effet des différentes variables sur le clic.

Les coefficients de notre meilleur modèle de régression logistique sont donnés en table 7. Nous pouvons ici interpréter leur signe et les comparer entre eux ; nous nous gardons de toute interprétation causale (certaines variables présentant un problème d'endogénéité évident) et regardons ces coefficients sous l'angle de l'importance des variables sur la probabilité de prédiction de clic.

Variable	`zonecostineuro`	`campaignctrlast24h`	`is_interstitial_True`
Coefficient	0.762348	0.318018	0.241982
`display_size`	`display_env_app_ios`	`target_env_2`	`nbdayssincelastclick`
0.152171	0.072239	-0.073403	-0.092570
`campaignscenario_13`	`device_type_iPhone`	`nbdisplay_1hour`	`device_type_Desktop`
-0.133118	-0.143667	-0.151742	-0.257959

TABLE 7 – Coefficients du modèle choisi de régression logistique.

En vert, les coefficients des variables ayant un effet positif sur la probabilité de prédiction de clic ; en rouge, ceux des variables ayant un effet négatif.

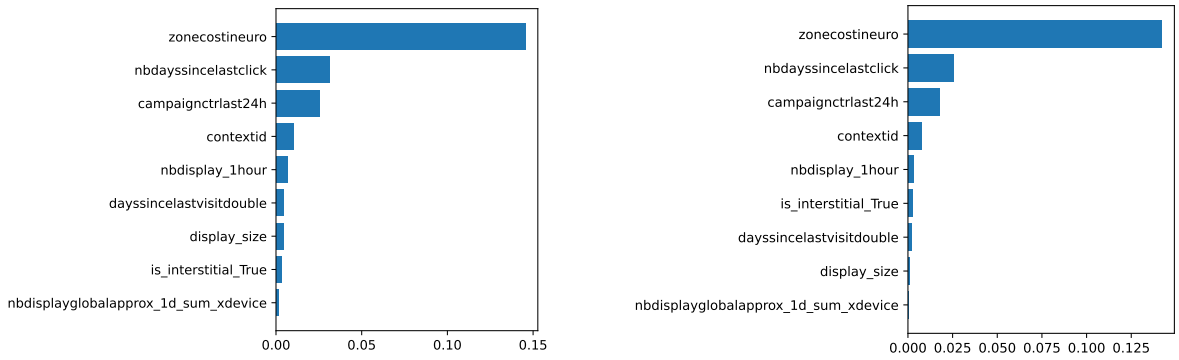
Sans surprise, les variables citées dans la partie 1.1.1 et sur lesquelles la figure 5 se concentre, à l'exception de `contextid` qui n'a pas été retenue par la méthode RFECV, ont un effet positif sur la probabilité de prédiction de clic. Le coût de l'emplacement publicitaire ainsi que la performance de la campagne durant les dernières vingt-quatre heures ont un pouvoir prédictif important, tout comme la taille de l'affichage. Ces résultats vont dans le sens des constatations de la section 2, même si `display_size` a finalement un effet moindre que `campaignctrlast24h` pour ce modèle.

Parmi les modalités des variables catégorielles les plus notables, on constate qu'afficher une publicité sur une application pour iPhone permet de prédire une probabilité de clic plus importante, alors même que l'affichage d'une publicité en général sur un iPhone (sur le navigateur internet Safari par exemple) ou sur un ordinateur permet de prédire le contraire. À travers les coefficients de `nbdayssincelastclick` et `nbdisplay_1hour`, le modèle tranche clairement pour un effet négatif de ces variables qui présentaient pourtant des ambiguïtés.

5.4.2 Random Forest et XGBoost

Les modèles de type Random Forest ou XGBoost sont par nature beaucoup plus complexes à interpréter puisqu'ils combinent de nombreux arbres. Différentes méthodes permettent cependant de classifier les variables selon leur niveau d'importance dans le modèle. Nous détaillerons ici les résultats obtenus sur le modèle Random Forest, ceux du modèle XGBoost étant très similaires (voir figures 30 et 31 en annexe D).

La *permutation importance* permet de comparer les performances d'un modèle avec et sans chacune des variables. Pour éviter d'entraîner de nombreux modèles (un pour chaque variable), l'astuce consiste à mélanger aléatoirement toutes les observations d'une variable pour supprimer son information. On peut alors comparer la performance du modèle sur le score F_3.



(a) Données d'entraînement.

(b) Données de test.

FIGURE 13 – Représentation de la *permutation importance* avec le score F_3 pour les 9 variables du modèle Random Forest associé à la RFECV.

D'après la figure 13, quatre variables ont une grande importance pour la performance des modèles : l'engagement de l'utilisateur (`contextid`), le taux de clic pour la campagne sur les dernières 24h (`campaignctrlast24h`), le nombre de jours écoulés depuis le dernier clic de l'utilisateur (`nbdayssincelastclick`) et le prix de l'emplacement publicitaire (`zonecostineuro`), par ordre croissant d'importance. En effet, l'ordre d'importance pour les autres variables varie entre la base d'entraînement et la base de test : l'importance mesurée pour ces variables est beaucoup trop soumise aux légères différences entre les deux bases, nous amenant à dire que seules les quatre

premières variables sont très significatives dans le modèle Random Forest. En particulier, la variable `zonecostineuro` se démarque beaucoup des autres, ce qui est cohérent avec les conclusions de l'analyse descriptive et l'interprétation de la régression logistique.

La méthode SHAP (SHapley Additive exPlanations) offre une perspective moins globale en se plaçant à l'échelle de l'observation. Pour chaque observation, cette méthode calcule les contributions de chaque variable à la prédiction. Le graphique 14 montre ces contributions pour plusieurs observations, chaque point correspondant à une observation.

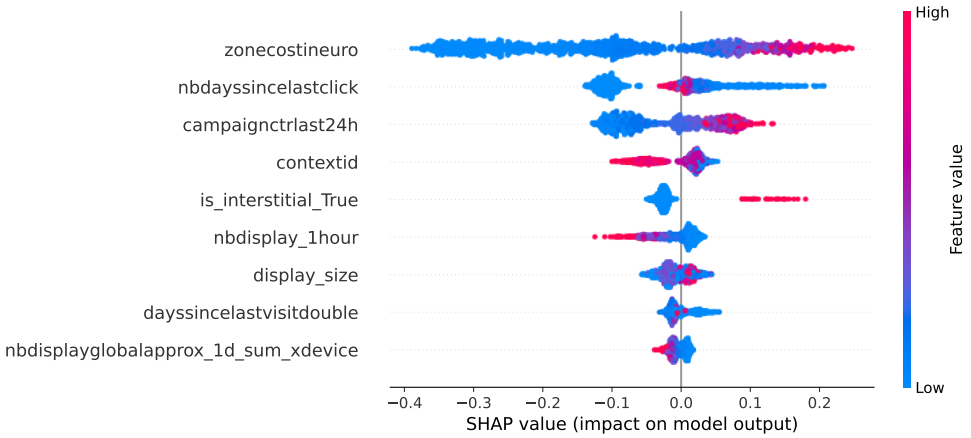


FIGURE 14 – Représentation de l'impact des 9 variables du modèle Random Forest associé à la RFECV sur la probabilité de prédiction d'un clic en fonction de leurs valeurs.

Lecture : Pour chaque variable en ordonnée, les points roses correspondent à des observations telles que la valeur de la variable est élevée, et les points bleus à des observations telles que la valeur de la variable est faible. Un point situé à gauche sur l'axe des abscisses signifie que pour cette observation, la variable en question impacte négativement la probabilité de prédiction d'un clic alors qu'un point situé à droite indique que la variable l'impacte positivement.

Le modèle Random Forest a donc plus de chance de prédire un clic lorsque le prix de l'emplacement publicitaire (`zonecostineuro`) et le taux de clics pour la campagne sur les dernières 24h (`campaignctrlast24h`) sont élevés ou lorsque le nombre d'affichages de la publicité sur la dernière heure (`nbdisplay_1hour`) est faible.

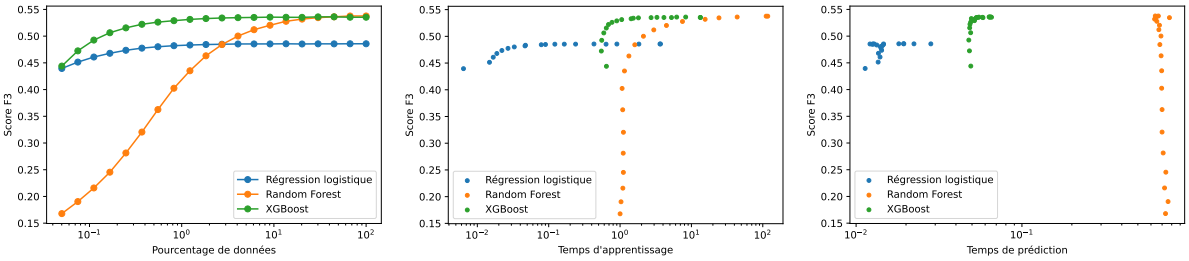
6 Réduction des données et contraintes temporelles

En vue de tester la robustesse de nos modèles, nous avons cherché à mesurer l'influence de la réduction des données du jeu d'entraînement sur la performance et les temps de calculs. Nous avons donc tracé dans un premier temps le score F_3 en fonction du pourcentage de données utilisées (les observations étant choisies aléatoirement) sur une échelle logarithmique pour les trois modèles (figure 15a). Les trois courbes sont bien évidemment croissantes, mais le premier graphique montre en outre un autre inconvénient du modèle Random Forest, qui est beaucoup moins robuste à la réduction des données que les deux autres modèles.

Il est aussi intéressant de tracer le score F_3 en fonction du temps d'apprentissage (figure 15b) pour savoir à quel point on peut diminuer ce dernier sans trop affecter la performance. Les courbes ont des concavités plus ou moins marquées, ce qui traduit le fait que le gain de performance est de

moins en moins élevé au fur et à mesure que le temps d'apprentissage augmente. Nous retrouvons le fait que le modèle Random Forest possède un temps d'apprentissage élevé, alors qu'il présente une chute vertigineuse du score F_3 lorsque ce temps est légèrement réduit. Au contraire, le temps d'apprentissage peut être réduit de 3 à 0.1 seconde pour la régression logistique et de 10 à 1 seconde pour le modèle XGBoost sans trop compromettre le score F_3.

Enfin, nous avons tracé le score F_3 en fonction du temps de prédiction (figure 15c), dont la minimisation est capitale pour Criteo dans la mesure où les enchères doivent se faire en quelques millisecondes. Il apparaît que l'ordre de grandeur du temps de prédiction reste le même quand la quantité de données augmente, et qu'il dépend surtout de l'algorithme utilisé : un ordre de grandeur de 0.01 seconde pour la régression logistique, 0.1 seconde pour XGBoost et 1 seconde pour Random Forest.



(a) Pourcentage de données. (b) Temps d'apprentissage. (c) Temps de prédiction.

FIGURE 15 – Représentation du score F_3 en fonction des critères (a), (b) et (c) pour chacun des trois modèles (échelle logarithmique en abscisse).

Dans les deux graphiques (b) et (c), les points sont associés aux différents pourcentages de données étudiés dans le graphique (a). On lit donc ici en abscisse, pour chaque pourcentage de données, les temps d'apprentissage et de prédiction correspondants.

Conclusion

En conclusion, conformément aux objectifs fixés au commencement du projet, nous avons mis en place une démarche scientifique visant à construire pas à pas un modèle de prédiction de clic robuste aux contraintes liées à la quantité de données disponibles et au temps d'évaluation requis par Criteo.

Dans un premier temps, un travail préparatoire sur les données nous a permis de nous familiariser avec les variables, de nettoyer et réorganiser la base, mais aussi de dégager les premières variables potentiellement explicatives des clics. Nous avons ensuite mis en œuvre les méthodes d'analyse descriptive étudiées lors de nos deux premières années d'études à l'ENSAE, de manière à affiner nos premières observations. Ainsi, l'analyse de la matrice des corrélations nous a permis d'éliminer un certain nombre de variables, tandis que des techniques d'apprentissage non supervisé comme l'ACP, complétée par des statistiques multivariées analysant la proportion de clics en fonction des variables explicatives, ont conforté les tendances que nous avions dégagées en amont. Il nous est également apparu nécessaire de rendre compte dans ce rapport des initiatives moins concluantes que nous avons été amenés à prendre, à l'instar du clustering, qui n'a pas permis de discriminer les clics des non-clics et ainsi de brosser un portrait caractéristique des observations associées au clic.

En second lieu, et avant de pouvoir implémenter nos modèles, il nous a fallu déterminer une métrique de référence permettant de mesurer efficacement la performance de nos algorithmes et de respecter les impératifs économiques de l'entreprise. Après quelques tâtonnements pour arbitrer entre le *recall* et la *precision*, nous avons opté pour un score hybride, en l'occurrence le score F_3, lequel donne tout de même trois fois plus d'importance au *recall* qu'à la *precision*.

Cette étape franchie, nous avons pu débuter la phase de modélisation, en choisissant trois modèles distincts dans l'optique de les comparer : la régression logistique, très utilisée chez Criteo, mais également les forêts aléatoires et le modèle de *gradient boosting* XGBoost, dont nous avons présenté les fondements théoriques. Préalablement à l'implémentation de nos modèles, nous avons procédé à une phase de sélection des variables explicatives ; la plus efficace étant la RFECV, dépendante des modèles et que nous avons donc retenue. Il a également été essentiel de tenir compte du caractère déséquilibré de notre jeu de données, dans la mesure où les clics ne représentent que 6% des observations. C'est la raison pour laquelle nous avons eu recours à des méthodes de sur-échantillonnage, dont nous avons pu tester la pertinence lors de l'optimisation des hyperparamètres des trois modèles par *grid search* et validation croisée. À ce stade, nous avons alors conservé la meilleure combinaison de paramètres pour chaque modèle, en tâchant de minimiser le temps de prédiction, puis nous avons entrepris d'améliorer la performance de nos algorithmes en jouant sur le seuil de décision. Lors de la comparaison effective réalisée sur la base de test, nous avons finalement constaté que la régression logistique présente des temps de prédiction extrêmement faibles comparativement aux autres modèles, tandis que le modèle XGBoost offre de très bonnes performances en termes de score F_3 pour un temps de prédiction raisonnable.

En réduisant la quantité de données dans le jeu d'entraînement, la tendance se confirme : les deux modèles cités ci-dessus conservent de bonnes performances de classification malgré des pourcentages de données disponibles très faibles et des temps de prédiction réduits, alors que les forêts aléatoires rencontrent une chute vertigineuse de leurs performances à mesure que le volume de données décroît. La régression logistique et le modèle XGBoost semblent finalement proposer un compromis intéressant dans le cadre de notre étude, qu'il serait intéressant de compléter en construisant un modèle prédisant l'intention d'un clic.

Annexes

A Descriptif des variables

Nom	Type	Description
day	Texte	Date du jour de l'affichage.
hashed_partner_id	Texte	Identifiant haché du client.
hashed_campaign_id	Texte	Identifiant haché de la campagne publicitaire.
contextid	Entier	Engagement de l'utilisateur à une date donnée et sur un site donné, réestimé à chaque visite sur le site. Cet engagement est forcément croissant dans cet ordre : $0 < 2 < 10 < 4 < 5 < 6 < 7 < 8 < 9$. Les modalités de cette variable ont la signification suivante : 0 correspond à aucune visite sur le site ; 2 correspond à une visite sur la page d'accueil ; 4 à un article vu sur la page ; 5 à deux ou trois articles vus sur la page ; 6 à plus de quatre articles vus sur la page ; 7 à un achat effectué ; 8 à deux ou trois achats effectués ; 9 à plus de quatre achats effectués ; 10 à une visite sur la liste des produits disponibles.
display_env	Texte	Environnement d'affichage de la publicité (IOS, Android, Safari...).
target_env	Entier	Environnement cible de la campagne publicitaire : 1 pour le web ; 2 pour les applications.
rtbtypeid	Entier	Identifiant de la plateforme d'enchère en temps réel.
rtbadvisibility	Entier	Valeur transmise par la plateforme d'enchère en temps réel indiquant la position et la visibilité de la publicité. L'encodage n'est pas le même selon les plateformes : cette variable doit être croisée avec rtbtypeid.
rtb_detectedlanguage	Entier	Valeur transmise par la plateforme d'enchère en temps réel indiquant la langue détectée. L'encodage n'est pas le même selon les plateformes : cette variable doit être croisée avec rtbtypeid.
urlhash2	Entier	Partie de l'URL hachée du site internet.
urlhash3	Entier	Partie de l'URL hachée du site internet.
urlhash4	Entier	Partie de l'URL hachée du site internet.
user_country	Texte	Pays de l'utilisateur.
hashed_affiliateid	Texte	Identifiant haché de l'annonceur.
hashed_app_id	Texte	Identifiant haché de l'application.
googleviewability	Entier	Information, lorsqu'elle est donnée par Google, sur la visibilité de la page.
googlepagevertical	Entier	Type de l'annonceur.
campaignscenario	Entier	Identifiant correspondant à un type (marketing) de campagne publicitaire.

campaign-vertical	Entier	Identifiant correspondant au type des produits concernés.
campaign-ctrlast24h	Réel	Taux de clic calculé pour la campagne durant les dernières vingt-quatre heures.
is_interstitial	Booléen	Information sur le processus d'enchère.
dayssincelastvisitdouble	Réel	Nombre de jours écoulés depuis la dernière visite de l'utilisateur.
ltf_lastpartnerclicktimestamp	Entier	Dernier horodatage auquel l'utilisateur a cliqué sur une publicité de l'annonceur.
ltf_nbglobalclick_4w	Entier	Nombre total de clics pour cet utilisateur sur les quatre dernières semaines.
ltf_nbglobaldisplay_4w	Entier	Nombre total d'affichages publicitaires pour cet utilisateur sur les quatre dernières semaines.
ltf_nbglobaldisplaysincelastpartnerproductview	Entier	Nombre total d'affichages publicitaires pour cet utilisateur depuis sa dernière vue d'un produit de l'annonceur.
ltf_nbpartnerdisplayssincelastclick	Entier	Nombre d'affichages publicitaires de l'annonceur pour cet utilisateur depuis son dernier clic.
ltf_nbpartnerclick_4w	Entier	Nombre de clics de cet utilisateur sur les publicités de l'annonceur sur les quatre dernières semaines.
ltf_nbpartnerdisplay_4w	Entier	Nombre d'affichages publicitaires de l'annonceur pour cet utilisateur sur les quatre dernières semaines.
ltf_nbpartnersales_4w	Entier	Nombre de ventes de l'annonceur pour cet utilisateur sur les quatre dernières semaines.
ltf_nbpartnerdisplay_90d	Entier	Nombre d'affichages publicitaires de l'annonceur pour cet utilisateur sur les quatre-vingt-dix derniers jours.
ltf_nbpartnerclick_90d	Entier	Nombre de clics de cet utilisateur sur les publicités de l'annonceur sur les quatre-vingt-dix derniers jours.
ltf_nbpartnersales_90d	Entier	Nombre de ventes de l'annonceur pour cet utilisateur sur les quatre-vingt-dix derniers jours.
nbdayssincelastclick	Entier	Nombre de jours écoulés depuis le dernier clic de l'utilisateur.
nbdisplay_1hour	Entier	Nombre d'affichages de la publicité pour cet utilisateur sur la dernière heure.
nbdisplaypartnerapprox_1d_sum_xdevice	Réel	Approximation du nombre d'affichages publicitaires de l'annonceur sur tous les appareils de l'utilisateur sur le dernier jour.
nbdisplayaffiliateapprox_1d_sum_xdevice	Réel	Approximation du nombre d'affichages publicitaires de l'annonceur affilié sur tous les appareils de l'utilisateur sur le dernier jour.

nbdisplay-globalapprox-_1d_sum_-xdevice	Réel	Approximation du nombre total d'affichages publicitaires sur tous les appareils de l'utilisateur sur le dernier jour.
hashed_uid	Texte	Identifiant haché de l'utilisateur.
hashed_xd_id	Texte	Identifiant inter-appareils d'un utilisateur (peut rassembler plusieurs appareils d'un même utilisateur).
valueperclick	Réel	Rémunération en euros obtenue si la publicité est cliquée.
device_type	Texte	Type d'appareil.
display_width	Entier	Largeur en pixel de la publicité.
display_height	Entier	Hauteur en pixel de la publicité.
display_size	Entier	*Variable créée.* Aire (nombre de pixels) de l'affichage publicitaire (display_height × display_width).
display_timestamp	Entier	Horodatage de la publicité.
hour	Entier	*Variable créée.* Heure de l'affichage publicitaire.
weekday	Entier	*Variable créée.* Jour de l'affichage publicitaire.
is_display_clicked	Booléen	Indique si l'utilisateur a cliqué sur la publicité.
zonecostin-euro	Réel	Prix de l'emplacement publicitaire.

TABLE 8 – Descriptif de toutes les variables de la base fournie par Criteo et des variables créées pour l'étude.

B Autres statistiques multivariées

Pour compléter la partie 2.2, nous avons également entrepris d'étudier la distribution des variables les plus pertinentes évoquées en partie 1.1.1 en fonction du nombre de clics des utilisateurs. Nous nous focalisons ici sur la variable `contextid`, réévaluée à chaque visite et qui traduit l'engagement de l'utilisateur à une date donnée et sur un site donné. La boîte à moustaches (également connue sous le nom de boîte de Tukey) ci-dessous semble ainsi dégager une légère tendance confirmant nos premières hypothèses. En effet, nous constatons que l'engagement moyen de l'utilisateur, lequel correspond à la somme des valeurs prises par la variable `contextid` pour chacune de ses visites sur un site référencé divisée par ce nombre de visites, présente une valeur médiane globalement croissante avec le nombre de clics de l'utilisateur. Ceci suggère que les utilisateurs les plus engagés en moyenne seraient un peu plus enclins à cliquer que les autres. Néanmoins, les boîtes associées aux modalités 7, 8 et 9, correspondant à des utilisateurs cliquant beaucoup, présentent des premiers quantiles plus bas, probablement en raison d'un très faible nombre d'utilisateurs adeptes des clics massifs. Les boîtes de Tukey ne donnent ainsi qu'un aperçu des grandes tendances explicatives des clics et confortent nos observations précédentes, mais ne sauraient cependant se suffire à elles-mêmes.

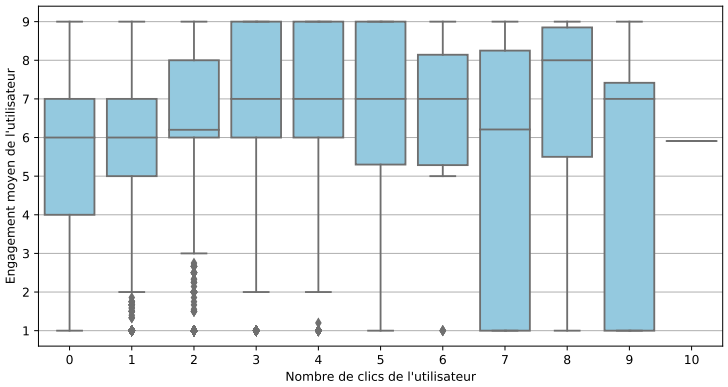


FIGURE 16 – Boîtes de Tukey représentant l'engagement moyen de l'utilisateur en fonction de son nombre de clics cumulé.

C Fondements théoriques des modèles utilisés

C.1 Régression logistique

La régression logistique est un modèle binaire : elle cherche à expliquer $Y \in \{0, 1\}$ avec une famille de régresseurs X (incluant la constante). Elle s'inscrit dans le cadre des modèles linéaires généralisés (modèle de la forme $h(\mathbb{E}[Y|X]) = X^T\beta_0$), où la fonction de lien (fonction h) est l'inverse de la fonction de répartition de la loi logistique $\Lambda : x \mapsto 1/(1 + e^{-x})$. Le modèle s'écrit donc

$$\mathbb{E}[Y|X] = \mathbb{P}(Y = 1|X) = 1/(1 + e^{-X^T\beta_0}).$$

Pour être identifié, ce modèle exige que la matrice $\mathbb{E}[XX^T]$ soit inversible : cela exclut la possibilité d'une colinéarité parfaite entre les régresseurs ; en particulier, il ne faudra pas conserver l'indicatrice de la première modalité de nos variables catégorielles pour ce modèle.

L'estimation du paramètre β_0 grâce à notre jeu de données $(Y_i, X_i)_{i\in[\![1,n]\!]}$ (supposé i.i.d) s'effectue au moyen du maximum de vraisemblance (ou plutôt de log-vraisemblance) :

$$\hat{\beta} = \arg\max_{\beta} \underbrace{\sum_{i=1}^{n} Y_i \ln\left(\Lambda(X_i^T\beta)\right) + (1 - Y_i)\ln\left(1 - \Lambda(X_i^T\beta)\right)}_{= \ell_n(Y|X,\beta)}.$$

Ce maximum est bien unique et existe bien du moment que le modèle est identifié et qu'aucune variable n'est constante.

Le problème d'optimisation n'admet pas, en général, de solution analytique simple. L'estimateur est obtenu numériquement, par exemple au moyen d'un algorithme de Newton-Raphson. Celui-ci va converger extrêmement rapidement vers $\hat{\beta}$. C'est un point fort de la régression logistique : la durée d'apprentissage est très faible, tout comme la durée de prédiction qui est quasiment instantanée.

En pratique, afin de limiter le phénomène de surapprentissage, on pénalise la log-vraisemblance (dans L^2) en lui retirant $\lambda\|\beta\|^2$, où λ est un réel positif et $\|\cdot\|$ est la norme euclidienne. Il peut également être utile, en présence d'un jeu de données déséquilibré, d'assigner des poids selon la classe des observations (un poids w_0 pour la classe des observations telles que $Y = 0$; w_1 pour la classe telle que $Y = 1$) dans l'expression de la log-vraisemblance. L'estimateur de la régression logistique régularisée et dont les classes sont pondérées vérifie alors :

$$\hat{\beta} = \arg\max_{\beta} \sum_{i=1}^{n} w_1 Y_i \ln\left(\Lambda(X_i^T\beta)\right) + w_0(1 - Y_i)\ln\left(1 - \Lambda(X_i^T\beta)\right) - \lambda\|\beta\|^2. \tag{1}$$

Une fois $\hat{\beta}$ obtenu, on prédit Y au moyen de $\hat{Y}(X) = \mathbb{1}\left[\Lambda(X^T\hat{\beta}) \geqslant s\right]$ où le seuil de décision s est usuellement fixé à $1/2$ (seuil de décision optimal lorsque $\Lambda(X^T\hat{\beta}) = \mathbb{E}[Y|X]$, i.e. lorsque la régression est parfaite).

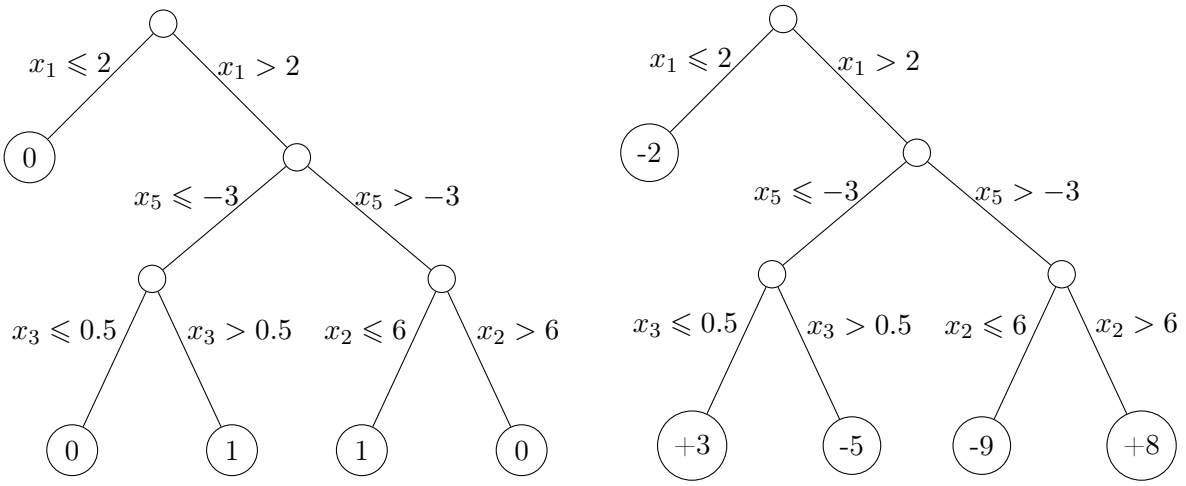
C.2 Random Forest

Classification And Regression Tree (CART) Un arbre de décision CART peut prendre en entrée un vecteur de données $x = (x_1, \ldots, x_n)$ de n'importe quel type :

- scalaire : $x_j \in \mathbb{R}$
- catégorielle : $x_j \in \{0; 1; 2; \ldots\}$

avec pour contrainte que chaque nœud de décision correspond à une condition binaire $\{x_j \leqslant a\}$ contre $\{x_j > a\}$.

Les nœuds terminaux correspondent soit à une classe dans un modèle de classification soit à une valeur numérique dans un modèle de régression.



(a) Arbre de classification.

(b) Arbre de régression.

FIGURE 17 – Exemples d'arbres de classification et de régression (CART).

Nous expliquons par la suite la construction d'un arbre de classification ; le cas de la régression est assez proche mais diffère dans la procédure de séparation des branches. La construction d'un arbre de classification se fait par itérations successives. À chaque itération, on choisit une variable x_j puis on fixe un seuil θ tel que les 2 ensembles des données $\{x_j \leqslant \theta\}$ et $\{x_j > \theta\}$ soient les plus différents possibles :

- On considère un ensemble de données $\mathcal{D} = \{(X_i, Y_i)\}$ avec $\forall i, (X_i, Y_i) \in \mathbb{R}^d \times \{0, 1\}$.
- Pour chaque variable $j = 1, \ldots, d$ et chaque seuil $\theta \in \mathbb{R}$:
  - On découpe les données en deux sous-ensembles $\mathcal{D}_\leqslant$ et $D_>$ tels que

$$\mathcal{D}_\leqslant = \{(X_i, Y_i) \in \mathcal{D}; X_{i,j} \leqslant \theta\} \quad \text{et} \quad \mathcal{D}_> = \{(X_i, Y_i) \in \mathcal{D}; X_{i,j} > \theta\}$$

  - On calcule ensuite les fréquences

$$p_\leqslant = \frac{\{(X_i, Y_i) \in \mathcal{D}_\leqslant; Y_i = 1\}}{\#\mathcal{D}_\leqslant} \quad \text{et} \quad p_> = \frac{\{(X_i, Y_i) \in \mathcal{D}_>; Y_i = 1\}}{\#\mathcal{D}_>}$$

  - La qualité du découpage est mesurée par l'incertitude

$$\frac{\#\mathcal{D}_\leqslant}{\#\mathcal{D}_\leqslant + \#\mathcal{D}_>} I(p_\leqslant, 1 - p_\leqslant) + \frac{\#\mathcal{D}_>}{\#\mathcal{D}_\leqslant + \#\mathcal{D}_>} I(p_>, 1 - p_>)$$

    où I est, dans le cadre du modèle CART, l'indice de diversité de Gini défini par $I(p, 1-p) = 2p(1-p)$ et dont le tracé figure en annexe D (figure 26).
- On choisit finalement le découpage ayant la plus petite incertitude, puis on réitère l'algorithme sur les ensembles $D_\leqslant$ et $D_>$.

Lors d'une prédiction sur un ensemble d'observations, chaque observation passe dans l'arbre, puis on retourne la classe la plus prédite (vote majoritaire) ou bien la fréquence de chaque classe sur l'ensemble des observations si l'on souhaite des probabilités. Dans le cas d'une prédiction sur une seule observation, les probabilités seront donc binaires.

Forêt aléatoire Une forêt aléatoire consiste à entraîner plusieurs arbres de décision pour ensuite effectuer une « moyenne ».

- A partir du dataset $\mathcal{D}_n$ on tire B nouveau datasets avec remise $\mathcal{D}_n^{(1)}, \ldots, \mathcal{D}_n^{(B)}$.
- On entraîne B arbres de décision $f(\mathcal{D}_n^{(1)}, \cdot), \ldots, f(\mathcal{D}_n^{(B)}, \cdot)$.
- Pour un modèle de classification, on retourne la classe la plus prédite par les B arbres. Pour un modèle de régression, on réalise la moyenne suivante :

$$f_{bagging}(\cdot) = \frac{1}{B} \sum_{b=1}^{B} f(\mathcal{D}_n^{(b)}, \cdot)$$

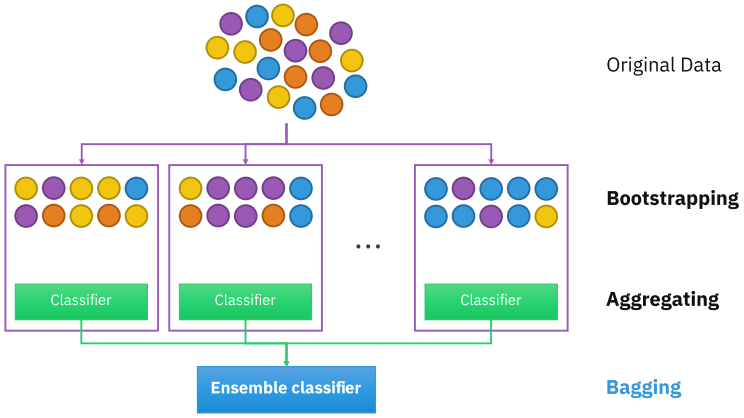


FIGURE 18 – Illustration du concept de *bootstrap aggregating*.[1]

L'objectif de cette méthode est de rendre le modèle plus robuste : un petit changement dans une des observations peut conduire à une modification importante des paramètres d'un arbre de décision. Faire une moyenne de plusieurs arbres permet ainsi de réduire cette instabilité, au prix d'une perte en interprétabilité.

Dans le cas d'un modèle de classification, l'algorithme fonctionne par un vote majoritaire et retourne la classe la plus prédite par les arbres. L'algorithme peut aussi retourner les probabilités de chaque classe calculées par une moyenne sur l'ensemble des arbres des probabilités de chaque classe.

C.3 Boosting

Les méthodes de *boosting* reposent sur une approche ensembliste, laquelle consiste à créer un classifieur fort à partir de classifieurs plus faibles de manière itérative, chaque modèle visant à corriger les faiblesses des précédents. Traditionnellement en confrontation avec les méthodes de *bagging*, qui considèrent les différents modèles parallèlement avant de les combiner, le *boosting*

1. Source : image par Sirakorn / CC BY-SA 4.0

présente l'avantage de s'appuyer sur des modèles très simples et améliore la précision des prédictions, malgré un risque accru de surapprentissage. Les classifieurs faibles utilisés dans le cadre du *boosting* sont généralement des arbres de décision (CART) de faible profondeur, présentant une erreur légèrement inférieure à 0.5. L'objectif est alors de fabriquer un nouveau classifieur pour lequel cette erreur serait significativement réduite.

L'algorithme AdaBoost (Adaptive Boosting) est un bon moyen de présenter le principe de l'algorithme de *boosting*. Introduit en 1996 par Yoav Freund et Rob Shapire, cet algorithme construit un premier arbre de décision à partir des données, en conférant à chacune des observations un poids égal. En fonction de la classification issue du premier arbre, les observations les plus difficiles à prédire (autrement dit mal classifiées dans l'exemple ci-dessous) se voient attribuer un poids plus important tandis que le poids des autres observations est mécaniquement diminué. Un nouvel arbre est alors construit à partir de ces données pondérées, et ainsi de suite à chaque itération, dans l'objectif permanent d'améliorer les prévisions du modèle précédent et d'abaisser l'erreur de classification. En sortie, le classifieur final, dit fort, est une combinaison linéaire des classifieurs faibles construits au fil des itérations de l'algorithme de façon séquentielle. Le caractère ensembliste de la méthode se retrouve donc dans le renforcement de modèles individuels qui agrégés créent un modèle plus fort et plus puissant. Nous retrouvons notamment ce principe dans les algorithmes de *gradient boosting*, lesquels ont recours à une descente de gradient sur l'erreur.

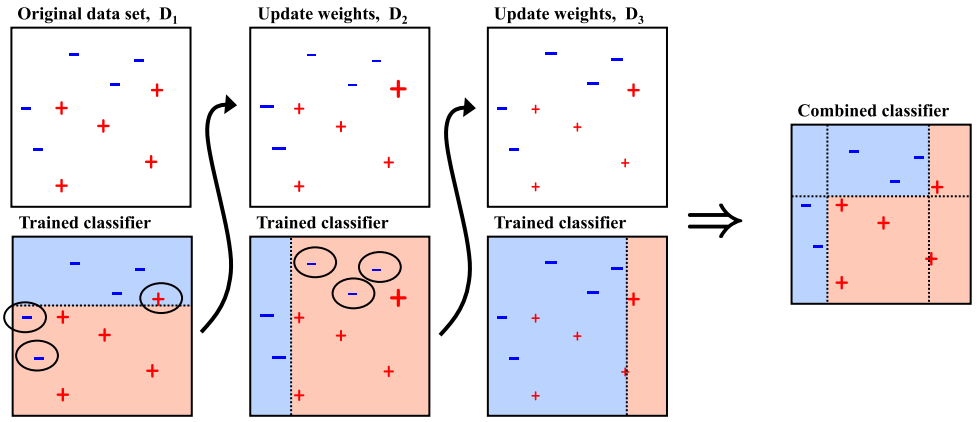
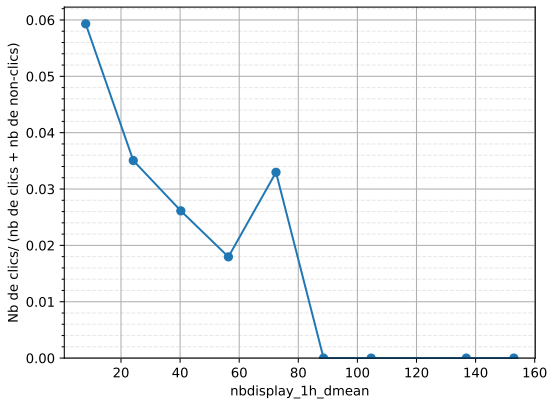


Figure 19 – Illustration des itérations de la méthode de *boosting* AdaBoost. [2]

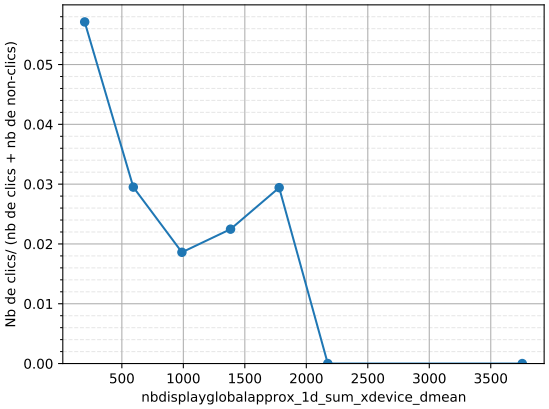
2. Source : image par Alexander Ihler

D Figures

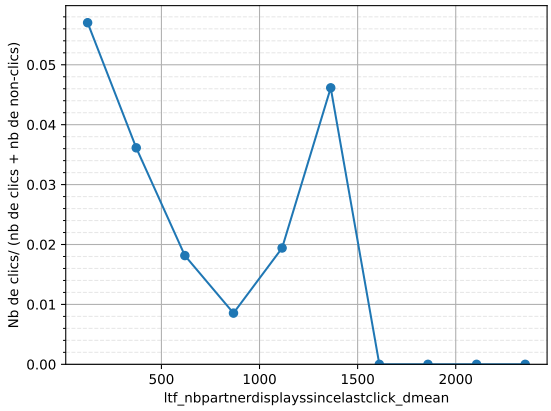
Proportion de clics en fonction des variables explicatives



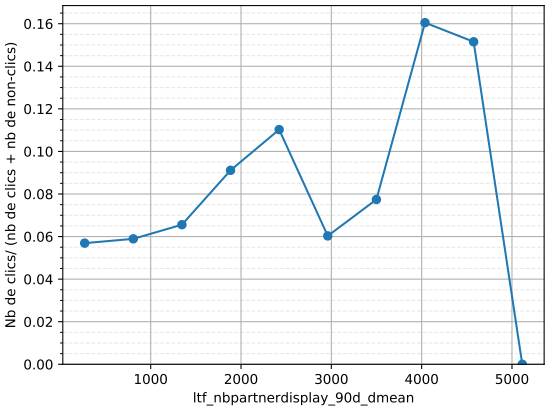
(a) Variable `nbdisplay_1hour`.



(b) Variable `nbdisplayglobalapprox_1d_sum_xdevice`.



(c) Variable `ltf_nbpartnerdisplayssincelastclick`.



(d) Variable `ltf_nbpartnerdisplay_90d`.

FIGURE 20 – Tracés du taux de clics en fonction de variables indiquant le nombre des derniers affichages publicitaires.

Ces variables ont été discrétisées pour comporter dix modalités croissantes. Chaque modalité a pour valeur le centre de l'intervalle de discrétisation. On constate un double effet de ces variables avec le motif en « U » que comportent ces courbes.

Analyse en composantes principales

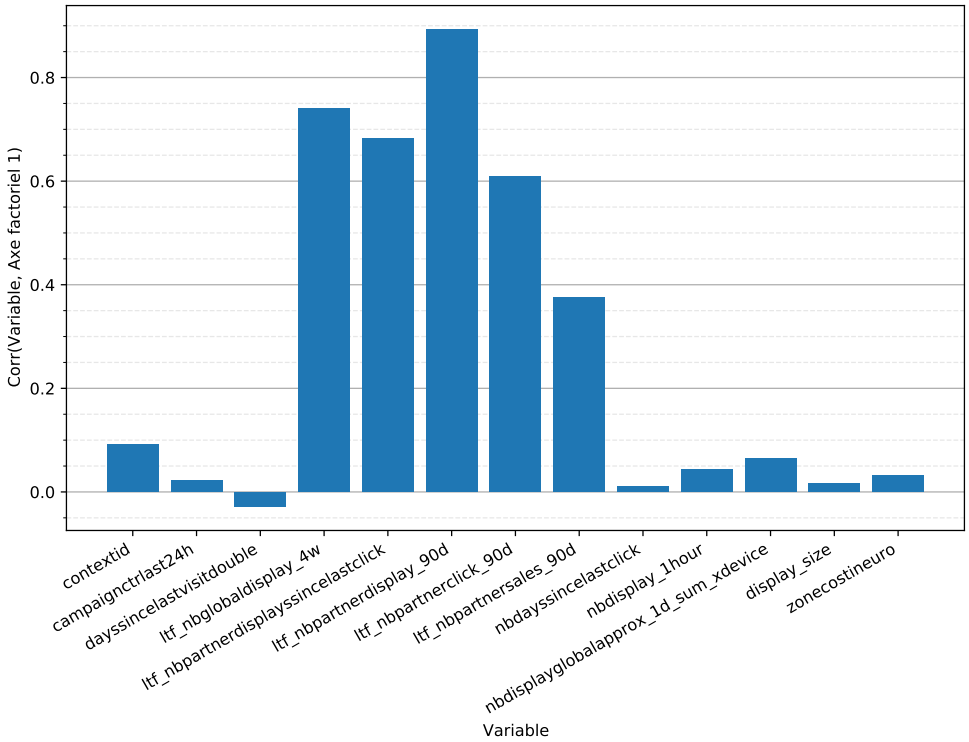


FIGURE 21 – Corrélations entre les variables quantitatives présélectionnées de la table 1 et le premier axe de l'ACP effectuée sur celles-ci.

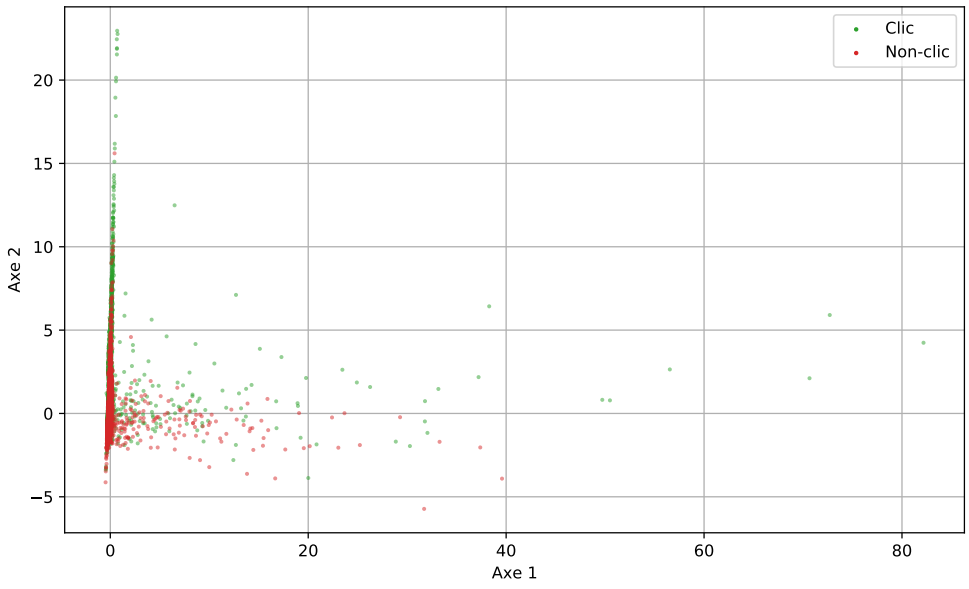
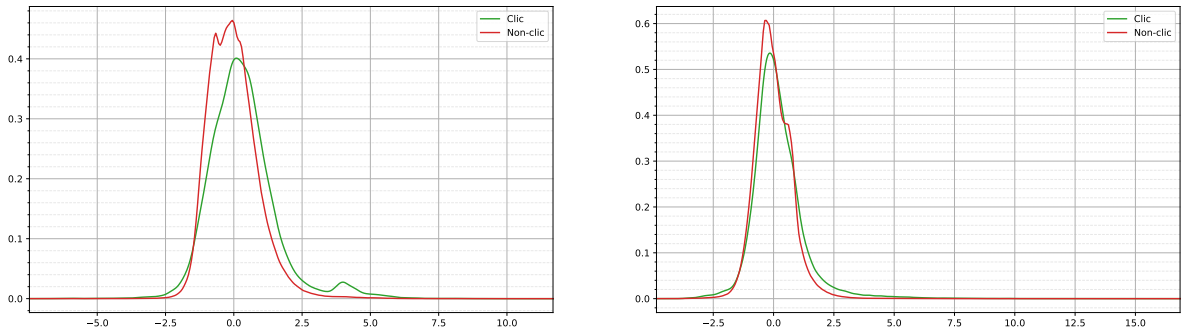


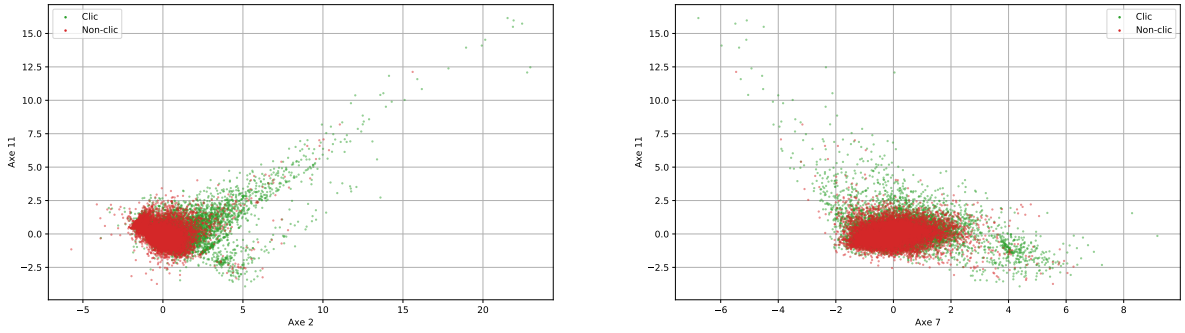
FIGURE 22 – Projection des observations dans le plan factoriel (axe 1, axe 2) obtenu par ACP sur les variables quantitatives de la table 1.

Pour chaque classe, un nombre de points égal à 10% des clics totaux est représenté.



(a) Axe 7. (b) Axe 11.

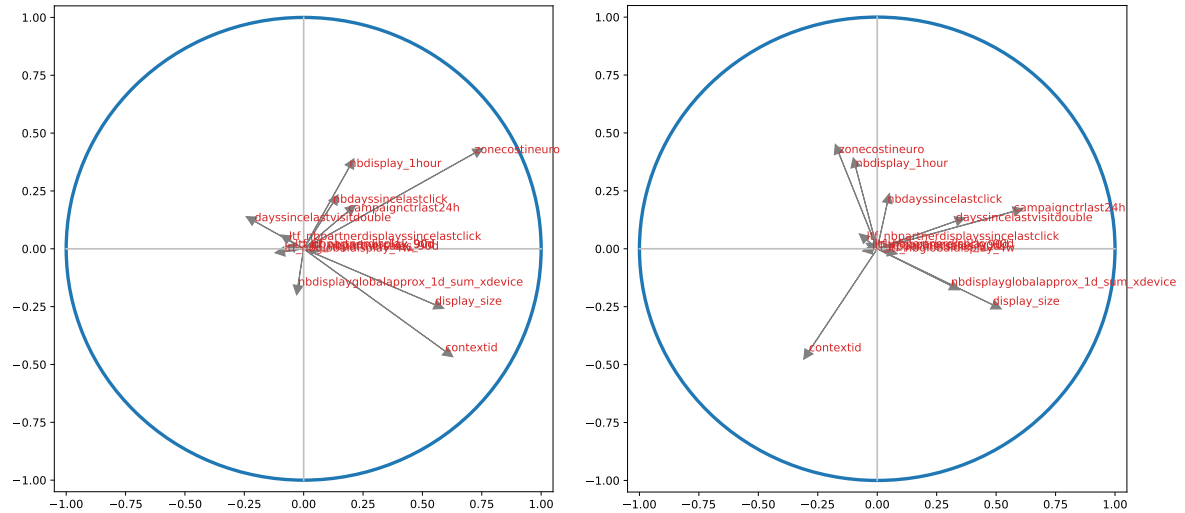
FIGURE 23 – Distributions des données sur les axes 7 et 11 obtenus par ACP sur les variables quantitatives de la table 1 selon le clic.



(a) Plan factoriel (axe 2, axe 11). (b) Plan factoriel (axe 7, axe 11).

FIGURE 24 – Projections des observations dans les plans factoriels (axe 2, axe 11) et (axe 7, axe 11) obtenus par ACP sur les variables quantitatives de la table 1.

Pour chaque classe (clic ou non-clic), un nombre d'observations égal à 10% des clics totaux est représenté.



(a) Plan factoriel (axe 2, axe 11). (b) Plan factoriel (axe 7, axe 11).

FIGURE 25 – Cercles des corrélations des plans factoriels (axe 2, axe 11) et (axe 7, axe 11) obtenus par ACP sur les variables quantitatives de la table 1.

Random Forest

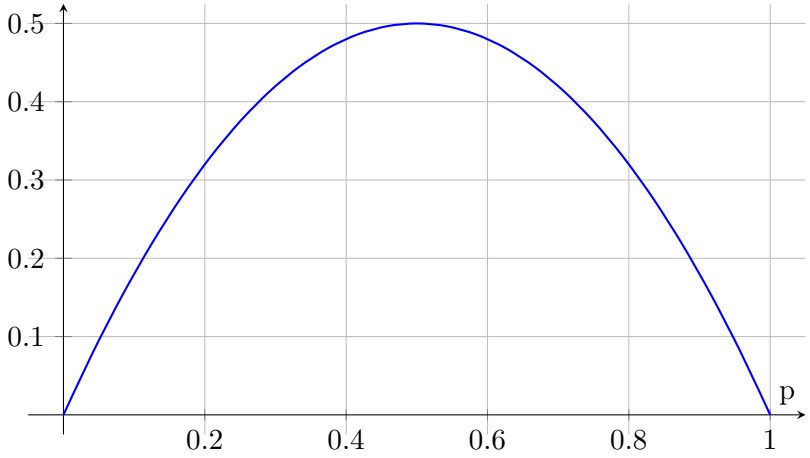
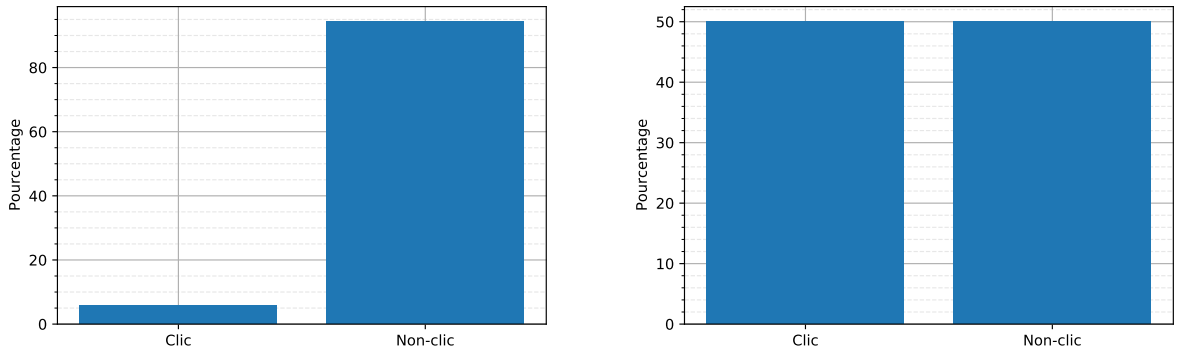


FIGURE 26 – Tracé de l'indice de diversité de Gini.

Sur-échantillonnage



(a) Base de données originale.

(b) Base de données sur-échantillonnée.

FIGURE 27 – Répartitions des clics et non-clics dans les bases de données originale et sur-échantillonnée.

Sélection des variables explicatives

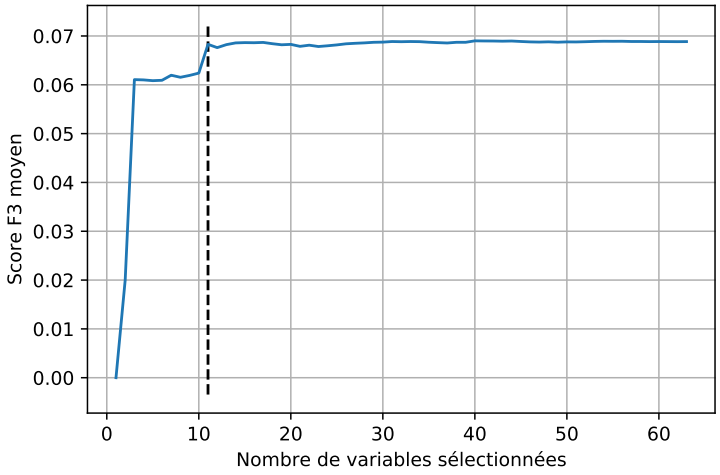


FIGURE 28 – Représentation du score F_3 moyen en fonction du nombre de variables sélectionnées par `RFECV` pour la régression logistique.

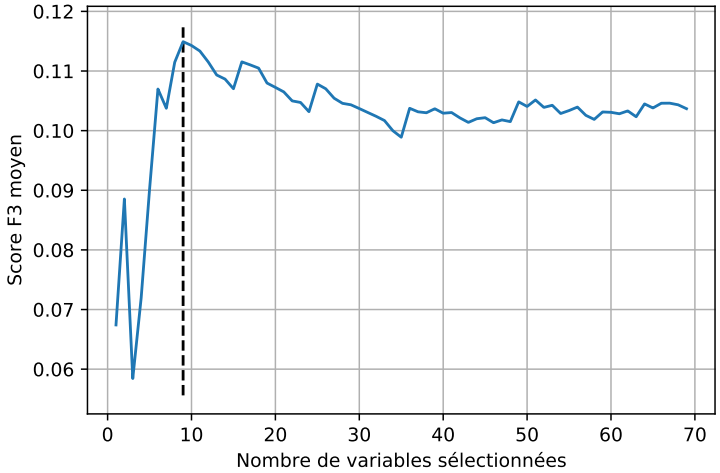
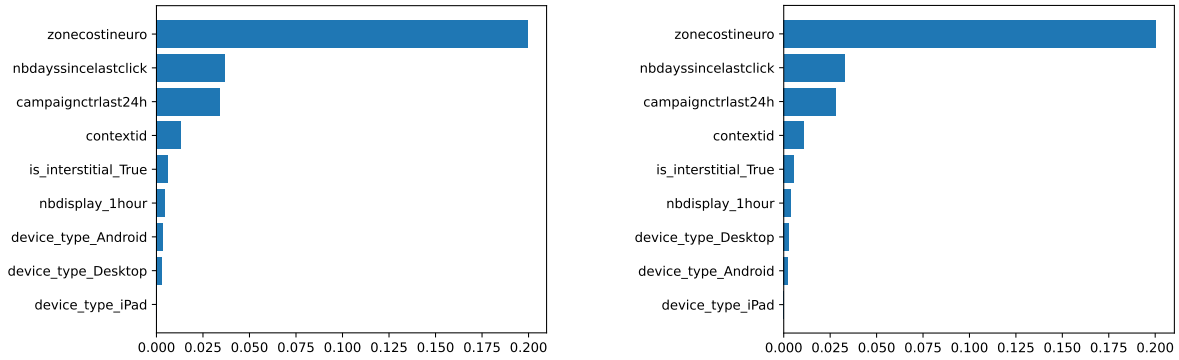


FIGURE 29 – Représentation du score F_3 moyen en fonction du nombre de variables sélectionnées par `RFECV` pour le modèle Random Forest.

Interprétation des modèles



(a) Données d'entraînement.

(b) Données de test.

FIGURE 30 – Représentation de la *permutation importance* avec le score F_3 pour les 9 variables du modèle XGBoost associé à la RFECV.

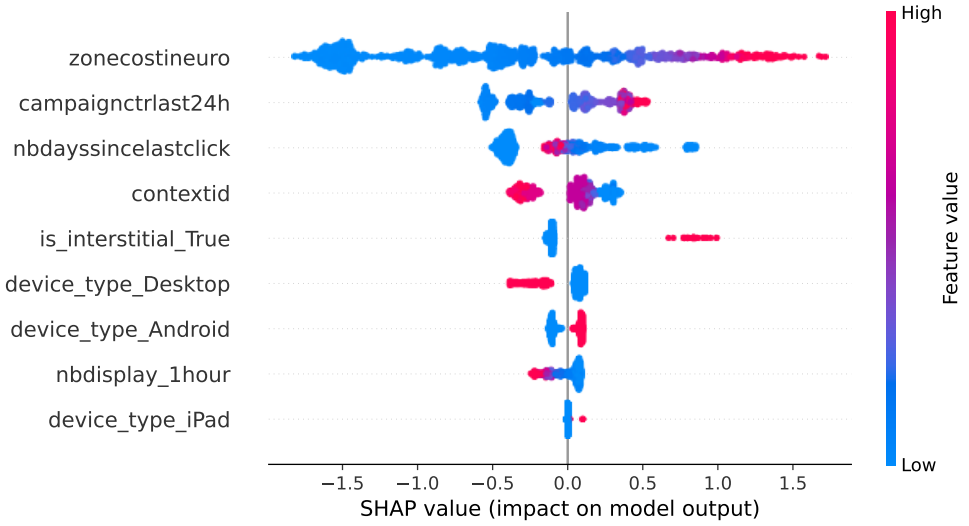


FIGURE 31 – Représentation de l'impact des 9 variables du modèle XGBoost associé à la RFECV sur la probabilité de prédiction d'un clic en fonction de leurs valeurs.

Lecture : Pour chaque variable en ordonnée, les points roses correspondent à des observations telles que la valeur de la variable est élevée, et les points bleus à des observations telles que la valeur de la variable est faible. Un point situé à gauche sur l'axe des abscisses signifie que pour cette observation, la variable en question impacte négativement la probabilité de prédiction d'un clic alors qu'un point situé à droite indique que la variable l'impacte positivement.

E Descriptif des paramètres utilisés pour chaque modèle

Nom	Type	Description
`C`	Réel strictement positif ; par défaut $C = 1$.	Inverse de la constante de régularisation. Spécifie λ dans l'équation 1 : $\lambda = 1/C$.
`class_weight`	"balanced" ou dictionnaire de la forme $\{0 : w_0, 1 : w_1\}$ où w_0 et w_1 sont des réels strictement positifs ; par défaut $w_0 = w_1 = 1$.	Les réels w_0 et w_1 sont les mêmes que dans l'équation 1, ils associent un poids à chacune des classes 0 et 1. La valeur "balanced" ajuste les poids pour accorder autant d'importance aux deux classes : $w_0 = \frac{n}{2n_0}$ et $w_1 = \frac{n}{2n_1}$, où n_0, n_1 et n désignent respectivement l'effectif de la classe 0, l'effectif de la classe 1 et l'effectif total.

TABLE 9 – Paramètres du modèle `LogisticRegression` de Scikit-Learn utilisés dans nos modèles. [3]

Nom	Type	Description
`bootstrap`	Booléen ; par défaut : `True`	Indique si des échantillons doivent être utilisés pour construire les arbres. Si `False` l'ensemble du dataset est utilisé.
`max_depth`	Entier ; par défaut : `None`	Profondeur maximale des arbres. Si `None`, l'arbre croît jusqu'à ce que chaque nœud soit pur (c'est-à-dire que toutes les observations d'un nœud ont le même label) ou contienne moins d'observations que `min_samples_split`.
`max_features`	"auto", "sqrt", "log2", entier ou réel ; par défaut : "auto"	Nombre de variables à considérer lorsqu'on cherche la meilleure séparation.
`min_samples_leaf`	Entier ou réel ; par défaut : 1	Nombre d'observations minimum arrivant dans chaque nœud terminal (feuille) des arbres.
`min_samples_split`	Entier ou réel ; par défaut : 1	Nombre d'observations minimum pour effectuer une séparation sur un nœud interne des arbres. Une séparation ne sera par ailleurs envisagée que si elle laisse au moins `min_samples_leaf` observations sur les branches gauche et droite.
`n_estimators`	Entier ; par défaut : 100	Nombre d'arbres dans la forêt.
`class_weight`	"balanced", "balanced_subsample", dictionnaire de la forme $\{0 : w_0, 1 : w_1\}$ ou liste de dictionnaires de cette forme ; par défaut : `None`	Poids associés à chaque classe.

TABLE 10 – Paramètres du modèle `RandomForestClassifier` de Scikit-Learn utilisés dans nos modèles. [4]

3. Source : site internet de la librairie Scikit-Learn, page du modèle de régression logistique.

Nom	Type	Description
`n_estimators`	Entier ; par défaut : 100	Nombre d'arbres dans la forêt.
`max_depth`	Entier ; par défaut : 6	Profondeur maximale de l'arbre.
`learning_rate`	Réel de l'intervalle [0,1] ; par défaut : 0.3	A chaque étape, les poids du nouvel arbre sont atténués par ce coefficient pour réduire le surapprentissage.
`gamma`	Réel positif ; par défaut : 0	Réduction minimale de la fonction de perte requise pour créer une partition supplémentaire sur un noœud interne de l'arbre. Plus cette valeur est élevée, plus l'algorithme est prudent.
`min_child_weight`	Réel positif ; par défaut : 1	Somme minimale du poids des occurrences requise dans un nœud fils.
`subsample`	Réel de l'intervalle]0,1] ; par défaut : 1	Ratio du sous-échantillon des observations entraînées. Choisir 0.5 conduira XGBoost à sélectionner aléatoirement la moitié des observations entraînées et empêchera le surapprentissage.
`colsample_bytree`	Réel de l'intervalle]0,1] ; par défaut : 1	Ratio du sous-échantillon de colonnes utilisées pour construire chaque arbre.
`scale_pos_weight`	Réel ; par défaut : 1	Contrôle le rapport entre les classes positives et négatives.

TABLE 11 – Paramètres du modèle `XGBClassifier` de XGBoost utilisés dans nos modèles. [5]

4. Source : site internet de la librairie Scikit-Learn, page du modèle de forêts aléatoires.

5. Source : site internet de la librairie XGBoost, pages des paramètres et de l'interface de programmation Python.

Bibliographie

[1] *1.13. Feature selection*. URL : https://scikit-learn.org/stable/modules/feature_selection.html.

[2] Jason Brownlee. *Recursive Feature Elimination (RFE) for Feature Selection in Python*. Août 2020. URL : https://machinelearningmastery.com/rfe-feature-selection-in-python/.

[3] Benoit Cayla. *La star des algorithmes de ML : XGBoost*. Oct. 2020. URL : https://www.datacorner.fr/xgboost/.

[4] N. V. Chawla et al. « SMOTE : Synthetic Minority Over-sampling Technique ». In : *Journal of Artificial Intelligence Research* 16 (juin 2002), p. 321-357. ISSN : 1076-9757. DOI : 10.1613/jair.953. URL : http://dx.doi.org/10.1613/jair.953.

[5] Marco Cuturi. *Régression Logistique + Tests du χ_2*. Notes du cours Introduction Stats Econométrie. ENSAE Paris, 2017. URL : http://marcocuturi.net/Teaching/ENSAE/2017/ISE/ISE10.pdf.

[6] *Introduction to Boosted Trees*. URL : https://xgboost.readthedocs.io/en/latest/tutorials/model.html.

[7] Will Koehrsen. *Random Forest in Python*. Jan. 2018. URL : https://towardsdatascience.com/random-forest-in-python-24d0893d51c0.

[8] Jean-Marc Lasgouttes. *Introduction au boosting*. Notes du cours Méthodes de boosting. Inria Paris, 2019-2020. URL : http://marcocuturi.net/Teaching/ENSAE/2017/ISE/ISE10.pdf.

[9] Aurélia Léon. *Interprétabilité des modèles de Machine Learning*. Oct. 2020. URL : https://www.aquiladata.fr/insights/interpretabilite-des-modeles-de-machine-learning/.

[10] Vianney Perchet. *Decision Trees & Random Forests*. Notes du cours Theorical Foundations of Machine Learning. ENSAE Paris, fév. 2021.

[11] Dario Radečić. *Feature Selection in Python - Recursive Feature Elimination*. Sept. 2019. URL : https://towardsdatascience.com/feature-selection-in-python-recursive-feature-elimination-19f1c39b8d15.

[12] Ricco Rakotomalala. *Régression régularisée - Ridge, Lasso, Elasticnet*. Notes de cours. Université Lumière Lyon 2. URL : https://eric.univ-lyon2.fr/~ricco/cours/slides/regularized_regression.pdf.

[13] *SMOTENC*. URL : https://imbalanced-learn.org/dev/references/generated/imblearn.over_sampling.SMOTENC.html.

[14] Michael Visser. *Modèles binaires*. Notes du cours Économétrie 2. ENSAE Paris, 2021.

[15] Eugenio Zuccarelli. *Performance Metrics in Machine Learning - Part 3 : Clustering*. Jan. 2021. URL : https://towardsdatascience.com/performance-metrics-in-machine-learning-part-3-clustering-d69550662dc6.

[16] Mohd Zuhaib. *Demystifying the Confusion Matrix Using a Business Example*. Déc. 2019. URL : https://towardsdatascience.com/demystifying-confusion-matrix-29f3037b0cfa.

Table des figures

1 Distribution empirique du degré d'engagement de l'utilisateur 6
2 Densité empirique du coût de l'emplacement publicitaire 6
3 Densités empiriques de la hauteur et de la largeur de l'affichage 7
4 Matrice des corrélations 8
5 Taux de clic en fonction de quatre variables notables 10
6 Proportion de la variance expliquée par chacun des axes factoriels 11
7 Distributions des observations sur l'axe 2 selon le clic 11
8 Projection des observations dans le plan factoriel (axe 2, axe 7) 12
9 Cercle des corrélations du plan factoriel (axe 2, axe 7) 13
10 Score de silhouette moyen en fonction du nombre de clusters 13
11 Score F_3 moyen en fonction du nombre de variables sélectionnées pour XGBoost . . 17
12 Scores F_3 des modèles en fonction des seuils de décision 19
13 *Permutation importance* pour le modèle Random Forest 21
14 Impact des variables sur la probabilité de prédiction d'un clic pour le modèle Random Forest 22
15 Variation du score F_3 selon la quantité de données et les durées de prédiction et apprentissage 23
16 Engagement moyen de l'utilisateur selon son nombre de clics cumulé 28
17 Exemples d'arbres CART 30
18 Concept du *bootstrap aggregating* 31
19 Illustration de la méthode de *boosting* AdaBoost 32
20 Tracés du taux de clics en fonction d'autres variables 33
21 Corrélations entre les variables quantitatives présélectionnées et le premier axe de l'ACP 34
22 Projection des observations dans le plan factoriel (axe 1, axe 2) 34
23 Distributions des observations sur les axes factoriels 7 et 11 selon le clic 35
24 Projections des observations dans les plans factoriels (axe 2, axe 11) et (axe 7, axe 11) 35
25 Cercles des corrélations des plans factoriels (axe 2, axe 11) et (axe 7, axe 11) 35
26 Indice de diversité de Gini 36
27 Répartition des clics selon l'échantillonnage 36
28 Score F_3 moyen en fonction du nombre de variables sélectionnées pour la régression logistique 37
29 Score F_3 moyen en fonction du nombre de variables sélectionnées pour le modèle Random Forest 37
30 *Permutation importance* pour le modèle XGBoost 38
31 Impact des variables sur la probabilité de prédiction d'un clic pour le modèle XGBoost 38

Liste des tableaux

1 Variables présélectionnées 9

2 Matrice de confusion 14

3 Sélections des variables par `RFECV` 17

4 Résultats des différentes `GridSearchCV` 18

5 Scores F_3 ajustés par seuil de décision pour les modèles choisis 19

6 Comparaison effective des modèles choisis 20

7 Coefficients du modèle choisi de régression logistique 20

8 Descriptif des variables 27

9 Paramètres du modèle `LogisticRegression` de Scikit-Learn 39

10 Paramètres du modèle `RandomForestClassifier` de Scikit-Learn 39

11 Paramètres du modèle `XGBClassifier` de XGBoost 40