



PROJET FINAL

---

# Advance Machine Learning

---

DATA ET IA  
PROMOTION 2022  
December 2021

ETUDIANTS : : SOULAÏMANE JOUHRI, CHEICK KANOUTE, KHALIL  
SAGOUMI

ENCADRANT : DENIS OBLINS

## Remerciements

Merci à Mr Denis OBLIN d'avoir assuré de nous avoir transmis ses conseils, son expérience et son savoir quand à l'implémentation et l'optimisation des modèles de machine learning.

# Table des matières

<b>0</b>	<b>Consignes</b>	<b>3</b>
<b>1</b>	<b>Exploratory Data Analysis &amp; Processing</b>	<b>3</b>
1.1	Étude sur les colonnes du dataset . . . . .	3
1.2	Focus sur les valeurs nulles . . . . .	3
1.3	Focus sur les valeurs uniques . . . . .	4
1.4	Focus sur les types des colonnes . . . . .	4
<b>2</b>	<b>Data Visualisation</b>	<b>4</b>
2.1	Distribution de la variable revenue . . . . .	5
2.2	Total Vs Moyenne . . . . .	5
<b>3</b>	<b>Modélisation, tuning &amp; hyperparamétrage</b>	<b>6</b>
3.1	Le modèles random forest et ses paramètres . . . . .	6
3.2	methode de la grille aléatoire d'hyperparamètres . . . . .	6
<b>4</b>	<b>Conclusion</b>	<b>7</b>

## 0 Consignes

L'objectif de ce projet est de réussir à modéliser les revenus par utilisateur qui ont visités un "Google Merchandise Store". Néanmoins il est important de noter que dans notre cas, la démarche est plus importante que les résultats. Le challenge réside dans la réflexion autour du problème et non dans la performance des résultats.

## 1 Exploratory Data Analysis & Processing

Le dataset étudié est relativement grand, il contient 903 653 lignes et 55 colonnes. La valeur à prédire dans ce dataset est le revenu (transactionRevenue) généré par utilisateur (fullVisitorId). Nous retrouvons donc dans ce dataset des informations très précises sur l'utilisateur. Par exemple, la date à laquelle il a visité le site en question, sa géolocalisation, l'OS ou encore le type d'appareil (tablette, téléphone, dekstop) qui a permit à l'utilisateur de se rendre sur le site.

### 1.1 Étude sur les colonnes du dataset

Afin de faciliter l'exploration de nos dataset nous avons implémenté une fonction permettant d'avoir pour chaque colonnes :

- Le nom de la colonne
- Le type (Int, date, object..)
- le nombre de valeurs uniques
- Les trois premières valeurs de la colonne
- L'entropie afin de mesurer le désordre au sein de chaque colonne

Voici les quelques premières lignes de ce tableau.

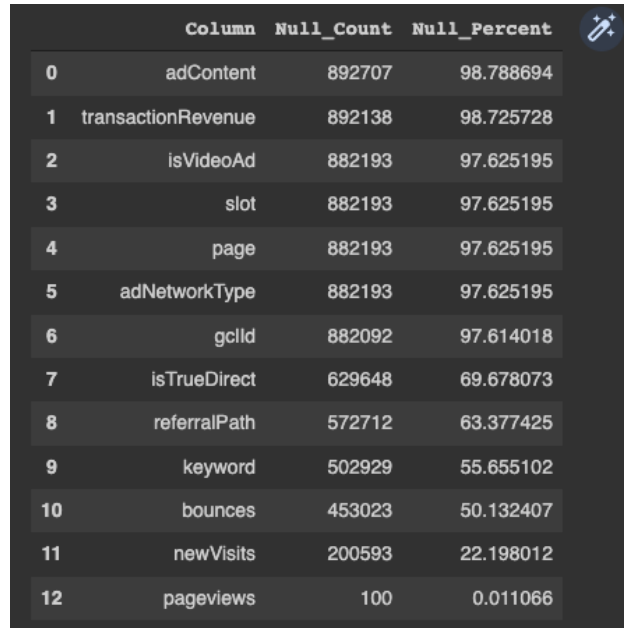
Name	dtype	Missing	Unique	First Value	Second Value	Third Value	Entropy
channelGrouping	object	0	8 Organic Search	Organic Search	Organic Search	Organic Search	2.11
date	int64	0	366 20160902	20160902	20160902	20160902	6.46
fullVisitorId	object	0	722903 113186044078586503	37209020877927880	389546263509774593	389546263509774593	16.22
sessionId	object	0	607365 113186044078586503_1472830385	377306020577927880_1472880147	389546263509774593_1472865396	389546263509774593_1472865396	19.76
socialEngagementType	object	0	1 Not Socially Engaged	Not Socially Engaged	Not Socially Engaged	Not Socially Engaged	0.0
visitId	int64	0	886303 1472830385	1472880147	1472880147	1472865396	19.76
visitNumber	int64	0	364 1	1	1	1	1.43
visitStartTime	int64	0	867189 1472830385	1472880147	1472880147	1472865396	19.76

FIGURE 1 – Description de quelques colonnes du dataset

Ainsi, nous pouvons conclure que plusieurs colonnes contiennent une seul valeur unique et n'apportent donc pas d'information. D'autres colonnes contiennent énormément de valeurs manquantes, la plus part d'entre elles seront supprimées. Enfin, notre target (transactions-Revenue) possède 892138 valeurs manquantes. Cela peut paraître a priori étrange mais en réalité une valeur non renseignée dans cette colonne signifie que l'utilisateur n'a pas généré de revenu. Dès lors, les valeurs manquantes de cette colonne seront remplacées par 0.

### 1.2 Focus sur les valeurs nulles

Regardons plus précisément le nombre de valeurs nulles par colonnes.



	Column	Null_Count	Null_Percent
0	adContent	892707	98.788694
1	transactionRevenue	892138	98.725728
2	isVideoAd	882193	97.625195
3	slot	882193	97.625195
4	page	882193	97.625195
5	adNetworkType	882193	97.625195
6	gcId	882092	97.614018
7	isTrueDirect	629648	69.678073
8	referralPath	572712	63.377425
9	keyword	502929	55.655102
10	bounces	453023	50.132407
11	newVisits	200593	22.198012
12	pageviews	100	0.011066

FIGURE 2 – Liste des colonnes contenant des valeurs manquantes

Nous décidons donc de remplacer les valeurs manquantes de notre target par 0. Puis pour les autres colonnes, nous avons essayé, en fonction de notre compréhension de la signification de ces dernières, de compléter les valeurs nulles. Par exemple pour la colonne "NewVisits" qui nous indique si c'est la première visite pour un utilisateur ou non, nous considérons que c'est en effet la première visite pour l'utilisateur si cette donnée est manquante.

### 1.3 Focus sur les valeurs uniques

Comme énoncé dans la section 1.1 certaines colonnes contiennent plusieurs valeurs uniques quelque soit l'utilisateur. Par exemple la colonne "visits" vaut toujours 1. Les autres colonnes sont à valeur unique pour des raisons éthiques notamment pour protéger certaines informations privées des utilisateurs. Nous décidons donc de supprimer toutes ces colonnes.

### 1.4 Focus sur les types des colonnes

Afin de ne pas se limiter en terme de modèles applicables, nous catégorisons toutes nos variables de type "object".

## 2 Data Visualisation

Une fois que nos données ont été traitées, nous continuons quelques data visualisation, et en particulier un peu d'analyse mathématique. Cela nous permet de mieux comprendre sur

quoi et avec quoi nous allons travailler.

## 2.1 Distribution de la variable revenue

Notre premier plot permet de voir que le log de notre target suit une loi normale. Cela est cohérent puisque nous avons énormément de données. De plus ces deux graphs illustre bien le principe de pareto (aussi connu sous le nom de la règles des 80/20) qui établit que dans beaucoup de cas 80% sont du à 80% des causes. Dans notre cas, 80% des utilisateurs ont généré 20% des revenus

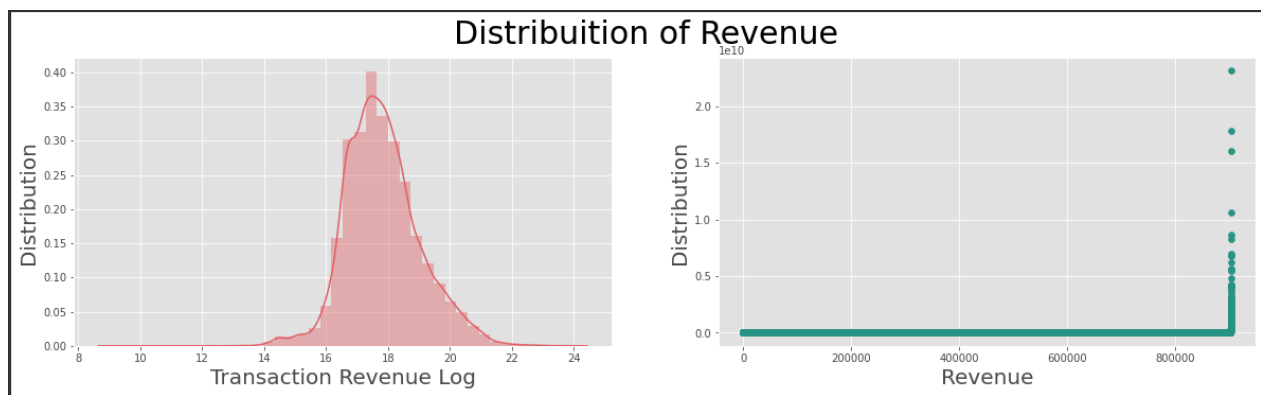


FIGURE 3 – Distribution de la target

## 2.2 Total Vs Moyenne

Dans cette section nous utilisons la fonction 'countmean' afin comparer le total par colonnes et calculer le revenue moyen pour chaque valeurs distinct de cette colonne. Par exemple d'après la figure 7. Nous remarquons que les utilisateurs de Google Chrome sont beaucoup plus nombreux a visiter le site, mais en revanche se sont les utilisateurs de Firefox qui génère le plus de revenus.

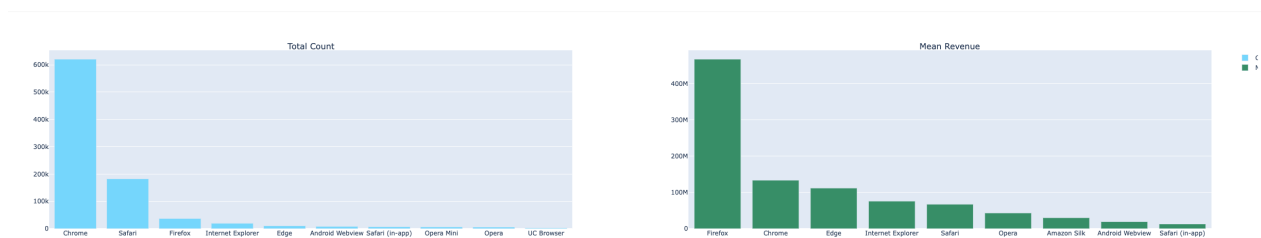


FIGURE 4 – Nombre d'utilisateurs par navigateur VS Revenu moyen par navigateur

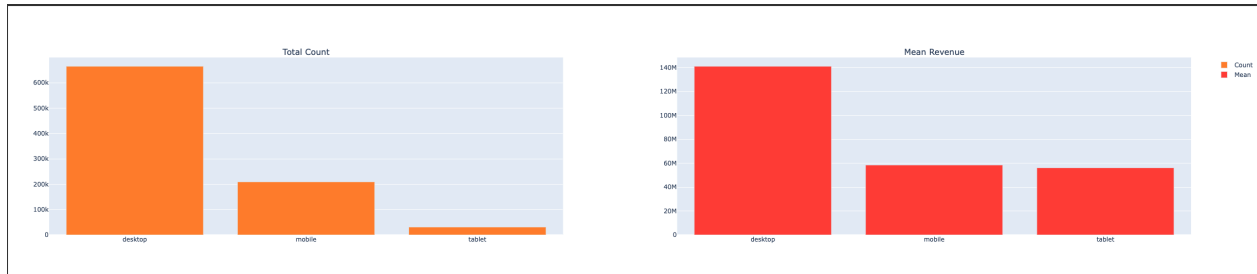


FIGURE 5 – Nombre d'utilisateurs par type d'appareils VS Revenu moyen par type d'appareils

### 3 Modélisation, tuning & hyperparamétrage

On commence par établir nos prédictions avec 3 modèles : une régression linéaire, random forest et xgboosting. Dans un premier temps on lance nos modèles avec les paramètres par défaut. Le modèle qui nous donne la meilleure RMSE est le modèle random forest (rmse : 1.66204). Nous décidons donc dans la suite de cette section d'améliorer ce modèle. Dans le cas d'une forêt aléatoire, les hyperparamètres comprennent le nombre d'arbres de décision dans la forêt et le nombre de features prises en compte par chaque arbre lors de la division d'un nœud.

#### 3.1 Le modèles random forest et ses paramètres

Les modèles random forest utilisent 16 hyperparamètres. Il peut donc être très compliqué et fastidieux de trouver la bonne combinaison de paramètre. En se documentant nous sélectionnons 6 paramètres qui peuvent potentiellement avoir le plus d'impact sur la précision du modèle :

- `n_estimators` : nombre d'arbres dans la forêt
- `max_features` : nombre maximum de features utilisées pour splitter un noeud
- `max_depth` : taille maximum dans chaque arbre de décision
- `min_samples_split` : nombre minimum de données dans un noeuds avec que celui ci soit splitter
- `min_samples_leaf` : nombre minimum de données autorisées dans un noeud
- `bootstrap` : méthode pour sampler les données

#### 3.2 methode de la grille aléatoire d'hyperparamètres

Pour avoir une idée précise de l'impacte de chaque hyperparamètre il pourrait être intéressant d'aller lire les papiers de recherche sur le sujet. Mais une première approche serait de générer des valeurs aléatoires et regarder quelles sont les combinaisons de valeurs qui donnent les meilleurs résultats. Voici les différentes valeurs de paramètres que nous allons tester :

Nous utilisons ensuite la fonction `RandomizedSearchCV()` qui permet de 'fitter' plusieurs fois le même modèle avec des combinaisons de paramètres différentes. La fonction `Randomi-`

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
```

FIGURE 6 – Valeurs de paramètres générées aléatoirement

zedSearchCv() prend en argument deux arguments important : 'n\_iter' indique le nombre de combinaisons à tester. Et 'cv' qui détermine le nombre de folds dans la cross validation.

Après avoir testé 10 combinaisons différentes d'hyperparamètres, notre code nous retourne le set de paramètre optimal pour notre prédiction.

```
[42] rf_random.best_params_

{'bootstrap': True,
 'max_depth': 50,
 'max_features': 'sqrt',
 'min_samples_leaf': 2,
 'min_samples_split': 10,
 'n_estimators': 200}
```

FIGURE 7 – Paramètres optimaux pour le modèle random forest

Comparons désormais les résultats de notre premier modèle avec celui qui utilise le set de paramètres optimaux.

Nous obtenons une diminution de la rmse de 2.64%

## 4 Conclusion

Finalement, nous n'avons pas pu augmenter considérablement la précision de nos prédictions. Cela est probablement dû à notre puissance de calcul limitée. En effet, nous avons pu tester un nombre de combinaisons d'hyperparamètres très limité. Notre entraînement s'est aussi effectué sur environ 250 000 lignes soit environ 25% du dataset.