1 INTRODUCTION

L'algorithme XGBoost, acronyme pour Extreme Gradient Boosting, représente une avancée significative dans le domaine de l'apprentissage automatique, en particulier dans le contexte des modèles basés sur les arbres de décision. Cet algorithme appartient à la famille du boosting, une technique qui combine plusieurs modèles faibles pour former un modèle plus puissant et robuste. L'approche de gradient boosting, utilisée par XGBoost, consiste à minimiser la fonction de coût en ajustant itérativement les modèles pour corriger les erreurs résiduelles des itérations précédentes.

Réputé pour sa performance élevée, XGBoost présente des avantages tels que la régularisation intégrée, la capacité à traiter des ensembles de données complexes, et une grande flexibilité pour s'adapter à diverses tâches en apprentissage automatique, de la classification à la régression.

La popularité croissante de XGBoost a conduit à son adoption dans une variété de domaines, y compris l'économie et les sciences sociales. Des applications de XGBoost dans ces domaines ont été étudiées et documentées dans la littérature scientifique. Les travaux de Tianqi Chen, créateur de XGBoost, ainsi que des publications telles que "XGBoost: A Scalable Tree Boosting System" (2016) ont établi des bases solides pour comprendre l'algorithme.

En économie, XGBoost a été utilisé pour des tâches telles que la prévision de séries temporelles, la modélisation du risque financier, et la prédiction des comportements économiques. Dans les sciences sociales, il a trouvé des applications dans la modélisation du comportement des consommateurs, l'analyse des données démographiques, et d'autres domaines connexes.

Cette introduction dépeint brièvement l'algorithme XGBoost, soulignant ses caractéristiques clés, et met en lumière son impact dans les domaines de l'économie et des sciences sociales, en référençant des travaux de recherche notables dans la littérature scientifique.

2 Matériel et Méthodes

2.1 Problématique

Bob, un jeune entrepreneur qui a créé sa propre entreprise de téléphonie mobile, veut mener un combat acharné contre les grandes entreprises comme Apple, Samsung, etc.

Cependant, Il ne sait pas comment estimer le prix des mobiles créés par son entreprise et ne peut absolument pas se baser sur de simples suppositions sur ce marché concurrentiel de la téléphonie mobile. Pour résoudre son problème, il collecte les données de vente des téléphones portables de diverses entreprises et fait appelle à nous pour notre expertise dans l'apprentissage automatique.

Notre travail sera donc de mettre en évidence la relation existante entre les caractéristiques d'un téléphone portable (par exemple : - RAM, mémoire interne, etc.) et son prix de vente sur le marché.

Dans ce problème, nous n'allons pas prédire le prix réel des mobiles, mais une fourchette de prix indiquant à quel point le prix peut être élevé.

Pour ce faire nous allons faire appel à un modèle de machine learning appelé XGBoost.

2.2 Présentation de la base de données

Notre jeu de données présente 21 variables dont 6 sont des variables qualitatives (blue, dual_sim, four_g, three_g, touch_screen, wifi) et 15, des variables quantitatives. La variable dépendante à analyser est nommée Price_range. C'est une variable qualitative multinomiale regroupant 4 differentes categories que sont low cost, medium cost, high cost and very high cost. (presenter l'encodage et l'analyse univariée de la var dependante)

Concernant les prédicteurs on distingue :

battery_power : Total energy a battery can store in one time measured in mAh

blue : Has bluetooth or not

clock_speed : speed at which microprocessor executes instructions

dual_sim : Has dual sim support or not

fc : Front Camera mega pixels

four_g : Has 4G or not

int_memory : Internal Memory in Gigabytes

m_dep : Mobile Depth in cm

mobile_wt : Weight of mobile phone

n_cores : Number of cores of processor

pc : Primary Camera mega pixels

px_height : Pixel Resolution Height

px_width : Pixel Resolution Width

ram : Random Access Memory in Mega Bytes

sc_h : Screen Height of mobile in cm

sc_w : Screen Width of mobile in cm

talk_time : longest time that a single battery charge will last when you are

three_g : Has 3G or not

touch_screen : Has touch screen or not

wifi : Has wifi or not

2.3 Présentation XGBoost

XGBoost un modèle de machine learning basé sur un apprentissage séquentiel et les arbres de décision. Comme son nom l'indique utilise le boosting de gradient, alors commençons par brièvement présenter cet algorithme.

Le gradient boosting, est une méthode d'ensemble de « weak learners », créés les uns après les autres, formant un strong learner. De plus, chaque « weak learner » est entraîné pour corriger les erreurs des « weak learners » précédents. Le premier « weak learner » (w1) est très basique, il s'agit tout simplement de la **moyenne** des observations. Il est donc très peu efficace, mais il va servir de base au reste de l'algorithme.

Par la suite, on calcule l'écart entre cette moyenne et la réalité que l'on appelle premier résidu. La particularité de Gradient Boosting est qu'il va essayer de prédire à chaque étape non pas les données elles-mêmes mais les résidus. Ainsi donc, le second « weak learner » est entraîné pour prédire le premier résidu. Les prédictions du second « weak learner » sont ensuite multipliées par un facteur inférieur à 1. On obtient donc de nouvelles prédictions, souvent légèrement meilleure que les précédentes.

L'idée derrière cette multiplication est que plusieurs petits pas sont plus précis que quelques grands pas. La multiplication réduit donc la taille des « pas » pour augmenter la précision.

XGBoost est en fait **une version particulière de l'algorithme de Gradient Boost.** En effet, il s'agit également d'un assemblage de « weak learners » qui prédisent les résidus, et corrigent les erreurs des "weak learners" précédents. Cependant, La particularité d'XGBoost est qu'il utilise Les « weak learners » de type arbres décisionnels. Les arbres qui ne sont pas assez bons sont "élagués", c'est à dire qu'on leur coupe des branches, jusqu'à ce qu'ils soient suffisamment performant. Sinon ils sont complètement supprimés. Cette méthode est appelée le « pruning » (élagage).

Ainsi, XGBoost s'assure de ne conserver que de bons weak learners. Pour détailler son implémentation considérons un classifieur faible initial $f0$. Apres l'avoir optimisé, le modèle va chercher à construire un nouveau classifieur faible $f1$ à partir de $f0$ en introduction un terme de résidu h tel que :

$$f1(x) = f0(x) + h(x)$$

Ans i le résultat $f1$ obtenu représente donc un classifieur plus performant qui se base sur les erreurs de $f0$. En répétant l'opération un certain nombre de fois, disons q, on construit un classifieur final F complexe qui est une combinaison linéaire des fi, où chacun des fi est associé à un poids αi tel que :

$$F(x) = i=1\sum n\ \alpha ifi(x)$$

De plus, XGBoost est informatiquement optimisé pour rendre les différents calculs nécessaires à l'application d'un Gradient Boosting rapides. Il propose également un panel d'hyperparamètres très important permettant d'avoir un control total sur l'implémentation du gradient boosting.

Dans le cadre de notre analyse, nous utiliserons la méthode suivante pour calibrer nos hyperparamètres afin d'éviter des temps de calculs phénoménaux :

- Tout d'abord, nous allons fixer un taux d'apprentissage Learning rate assez élevé (entre 0.1 et 1). Chaque nouvel arbre est donc multiplié par le taux d'apprentissage : plus ce taux est faible, plus la convergence vers un optimal sera lente. On fixera de même un nombre d'arbres plus ou moins important en fonction de la complexité des données.

- Pour définir les hyperparamètres relatifs à la structure des arbres max_depth (profondeur max d'un arbre) et min_child_weight (seuil minimal nœud), nous allons effectuer une recherche par grille relativement petite (environ une vingtaine de points).

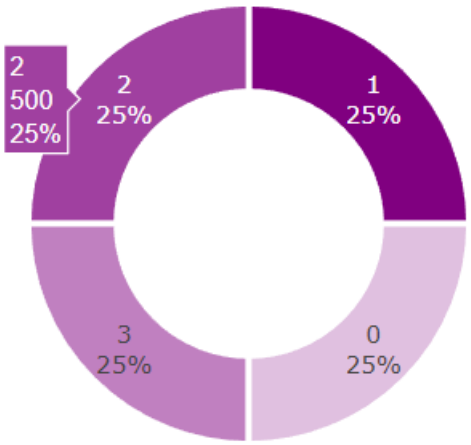
- Ensuite, nous nous attarderons sur le coefficient gamma pour régulariser les profondeurs des arbres. Plus il a une valeur élevée, plus la pénalisation empêche de construire des arbres trop profonds si l'apport en performance n'est pas aussi élevé. On peut l'optimiser avec une recherche par grille simple avec un test de valeurs $\gamma \in [0, +\infty[$.

- Concernant les hyperparamètres d'échantillonnage colsample bytree et subsample (situé entre 0 et 1) on précèdera également par une recherche par grille, mais que pour les valeurs inférieures à 50%.

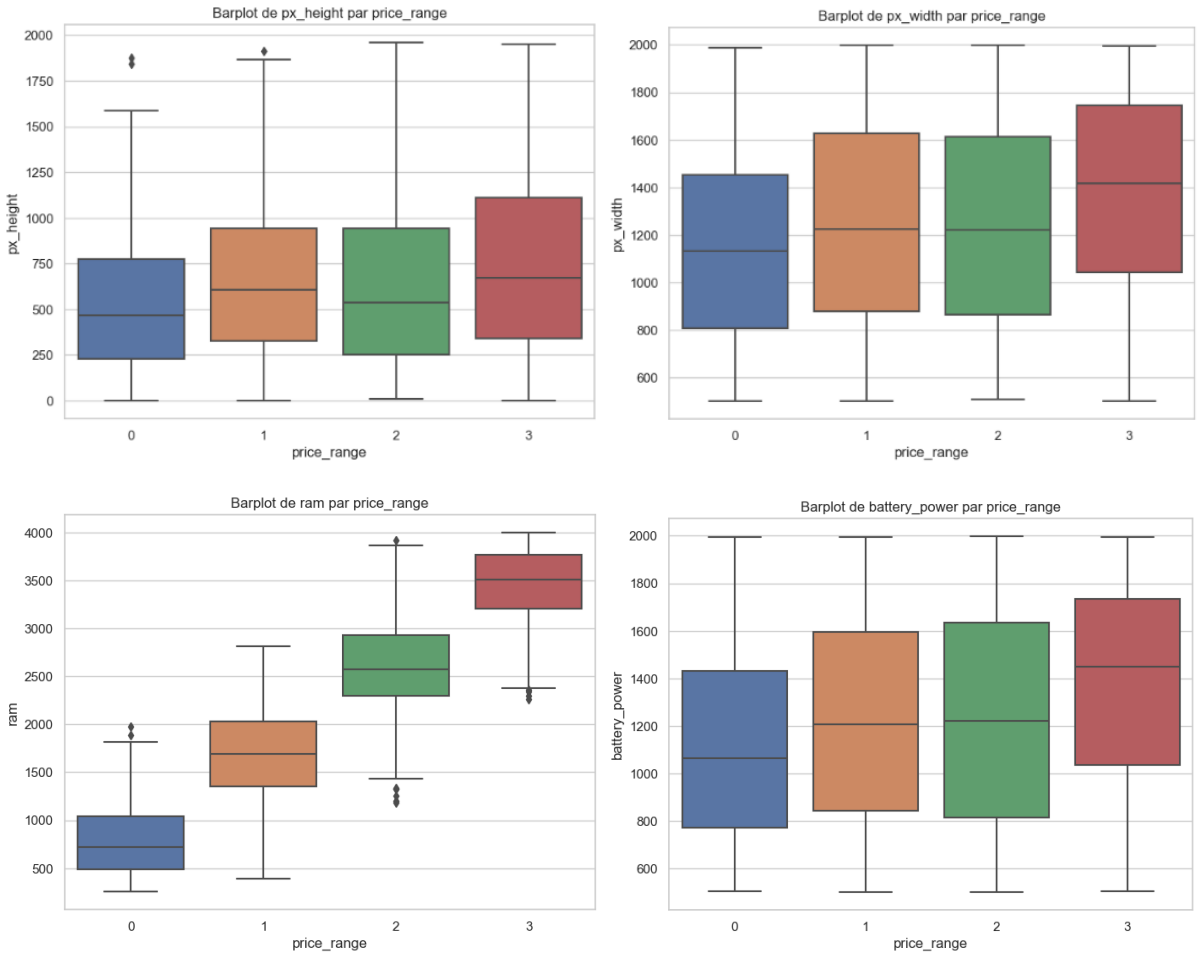
- Enfin dans le cas ou nos données seraient peu volumineuses, il ne reste plus qu'à réduire le taux d'apprentissage Learning rate (0.01 puis 0.001) tout en augmentant le nombre d'arbres (n_estimators) avec les hyperparamètres optimisés des étapes précédentes. Il est fréquent d'utiliser un early stopping dans cette dernière étape lorsque l'on ne sait pas exactement combien d'arbres utiliser. Dans notre cas, puisque nos données sont assez volumineuses (500k lignes), nous n'avons pas le luxe de pouvoir inférer sur ces valeurs.

3 Discussion et Résultats

On note une répartition equitable des différentes modalités de notre variable d'interet.



Pour analyser la relation entre nos variables quantitatives et notre variable d'interet on effectue un test parametrique one-way ANOVA. On observe à partir des p-values que les variables indépendantes présentant une relation très significative avec la variable dépendante sont battery_power, px_height, px_width et ram. Ces relations sont illustrées par les Boxplots suivants :



Il parait donc très judicieux d'analyser la répartition moyenne de ces variables selon les 4 différentes catégories de notre variable dépendante. On peut observer les résultats dans le tableau ci-dessous :

Price range	**px_width**	**px_height**	**px_width**	**ram**
Low cost	1150.270	536.408	1150.270	785.314
Medium cost	1251.908	666.892	1251.908	1679.490
Hight cost	1234.046	632.284	1234.046	2582.816
Very High cost	1369.838	744.848	1369.838	3449.232

Le graphique ci-dessous illustre parfaitement les résultats de ce tableau.

Par contre ayant calculé le khi – deux afin de quantifier la relation entre notre variables d'interet et les variables indépendantes quantitatives, il ressort qu'aucune d'elles ne présentent une relation significative avec notre variable d'interet.

Afin de prédire les différentes classes de prix pour chaque telephone, nous avons effectue un modèle d'apprentissage automatique en utilisant la technique de XGBoost. Des résultats de ce tableau, en examinant les différentes valeurs de AUC-ROC, il ressort que le modèle a un fort taux de vrai positif contre un très faible taux de faux positif pour les telephones de bas prix en d'autres termes le modèle est très efficace pour discriminer les téléphones à bas prix tandis que le modèle prédit moins bien les téléphones à prix élevé en d'autres il est moins efficace pour discriminer les téléphones à prix élevé.

De plus les variables qui ont une importance capitale dans la prediction de la categorie du prix d'un telephone dans notre cas d'usage sont : la RAM, la capacité de batterie et la longueur et la largeur de pixels.

- La RAM est un composant important qui affecte la capacité du téléphone à exécuter plusieurs applications simultanément et à gérer des tâches plus complexes. Plus la quantité de RAM est élevée, plus le téléphone peut gérer efficacement les applications gourmandes en ressources et effectuer des tâches complexes rapidement. Cela peut influencer la perception de la performance globale du téléphone.
- La capacité de la batterie est généralement mesurée en mAh (milliampères-heures) et reflète la quantité d'énergie que la batterie peut stocker. Les téléphones dotés de batteries ayant une capacité plus élevée ont tendance à offrir une meilleure autonomie. Une autonomie plus longue peut être un critère important pour certains consommateurs, notamment ceux qui utilisent intensivement leur téléphone.
- La longueur et la largeur de pixels (résolution) d'un écran de téléphone peuvent influencer la catégorie de prix du téléphone, bien que cela dépende également d'autres caractéristiques et facteurs. Une résolution d'écran plus élevée est généralement associée à une meilleure qualité d'affichage, offrant des images plus nettes et des détails plus fins. Les téléphones avec des écrans de haute résolution peuvent être positionnés dans des catégories de prix supérieures, surtout s'ils offrent également d'autres fonctionnalités haut de gamme.

4 Conclusion

Le modèle d'Extreme Gradient Boosting, se démarque comme une solution de pointe dans le domaine de l'apprentissage automatique, particulièrement en ce qui concerne les modèles basés sur les arbres de décision. Ses avantages incluent une performance exceptionnelle, une capacité à traiter des ensembles de données complexes, et une régularisation intégrée qui prévient le surajustement. Sa flexibilité lui permet de s'adapter à diverses tâches, de la classification à la régression en passant par la détection d'anomalies.

Cependant, l'utilisation de XGBoost n'est pas sans défis. La sensibilité aux paramètres exige une attention particulière lors du réglage des hyperparamètres, et son entraînement peut nécessiter du temps, surtout sur des ensembles de données volumineux. Les exigences en mémoire peuvent également être significatives.

Relativement à d'autres méthodes basées sur les arbres de décision, comme les forêts aléatoires, XGBoost se distingue par sa performance supérieure grâce à son approche de boosting et à la régularisation intégrée. C'est une extension du gradient boosting standard, et donc le XGBoost surpasse le GBM dans de nombreuses situations en termes de vitesse et de performance. alternatives modernes comme LightGBM et CatBoost, XGBoost offre une combinaison unique de rapidité, de polyvalence et de capacité à gérer des ensembles de données de grande dimension.

En conclusion, XGBoost demeure un choix puissant pour diverses applications en apprentissage automatique, mais son utilisation efficace requiert une compréhension approfondie de ses paramètres et une attention particulière aux spécificités de la tâche à accomplir.