Rapport data challenge: problème de classification

Said OUGOUADFEL, Simon LEGRIS, François SLAWNY

November 2025

Introduction

Dans le cadre du cours *Apprentissage supervisé et Data challenge*, il nous a été demandé de participer à une *compétition* sur *Kaggle*, visant à tester ce qu'on a appris en cours. Ce rapport résume ce qu'on a pu implémenter: les différents modèles utilisés, les différentes approches pour tester leurs *efficacité*, le traitement des données et la méthode de sélection du meilleur modèle.

Le *challenge* est structuré en deux problèmes d'apprentissage supervisé:

- Un problème de *classification*: prédire, suite à une réservation d'une chambre dans un hôtel, si le client concerné *viendra*, *ne va pas se présenter* le jour *j*, ou *annulera*-t-il sa réservation.
- Un problème de régression, *prédire la popularité* d'un son, sur *spotify*.

Dans ce qui va suivre, on s'intéresse au premier problème, le problème de *classification*. On premier lieu, on s'intéressera aux données, afin d'y extraire de l'information qui nous sera utile pour entraîner des modèles. Puis, on s'intéressera à la procédure utilisée pour le choix des modèles d'apprentissage et puis on commentera sur certaines difficultés rencontrées durant le *data challenge*.

Les données:

Pour le problème de classification, on nous a fourni des données tabulaires ayant des données *numériques* et des données *catégorielles*.

Une première remarque est de voir que les *classes* - (0 pour *check-out* présent), (1 pour *cancel* annulé) et (2 pour *no-show* ne s'est pas présenté) - ne sont pas équilibrées. Il y a une sur-représentation de la classe 0 - plus que la classe 1, qui est plus représentée que la classe 2.

Classes	Proportion
Check-out	62.9409 %
Cancel	36.0664 %
No-show	0.9928 %

Table 1: La proportion des classes dans le jeu de données fourni

Ce que ce tableau nous fait comprendre est que dans la plupart des cas, les clients honorent leurs réservations - dans ce jeu de données. Ceci dit, le manque d'exemples - en proportion - sur *no-show* rend le jeu de données biaisais, une information qui sera prise en compte dans la suite de notre projet.

En se restreignant aux données numériques, on remarque qu'elles ne sont pas corrélées. Ce qui implique, dans un premier temps, qu'on n'a pas d'autre choix que de les garder. Le manque de corrélation fait que chaque colonne peut nous apporter une information supplémentaire sur le problème.

Quant aux données catégorielles, plusieurs choses peuvent être dites. D'abord, plusieurs modèles d'apprentissage ne traitent pas les données catégorielles automatiquement. Ces modèles-là nécessitent un pré-traitement des données. Ce *pre-processing* s'est avéré être très chronophage, et devient un *hyperparamètre* qui doit être optimisé. Dans ce qui va suivre on discutera de quels choix de *pré-processing* qu'on a pris et les motivations derrières ces choix.

Remarques sur les données catégorielles:

Durant le traitement des données catégorielles, on a remarqué que certaines valeurs étaient sur-représentées dans le jeu de données. Un exemple frappant est la colonne *country*. En regardant attentivement cette catégorie, on se rend compte de la chose suivante: *country = 'PRT'* est sur-représentée dans le jeu de données avec une proportion de 40 %. Bien d'autres catégories sont sur-représentées. Certaines ont une explication rationnelle, comme dans le cas de *meal* où *BB* est sur-représenté. C'est tout à fait naturel, étant donné que les réservations d'hôtels sont dans la plupart des cas pour des vacances et donc beaucoup de déplacements - pas de déjeuners dans les hôtels. Évidemment cette sur-représentativité de certaines catégories ne nous donne pas spécialement des informations sur le statuts réservations.

Dans le cas de *meal*, on voit clairement que *BB* est dans tous les statuts de réservations le premier. Ceci dit, en regardant de plus près, on voit bien que la probabilité estimée $\hat{\mathbb{P}}(\text{meal} = \text{'SC'}|\text{reservation-status} = 2) \geq \hat{\mathbb{P}}(\text{meal} = \text{'HB'}|\text{reservation-status} = 2)$ sachant quand-même que la classe *HB* est plus représentée que la classe *SC*.

Problème avec les variables catégorielles:

On remarque qu'on a beaucoup de valeurs possibles pour certaines variables catégorielles. Un *one-hot encoding* serait excessivement encombrant et couteux d'un point de vue dimensionnel et algorithmique. Pour y remedier, une solution possible était de trouver une codification qui puisse à la fois donner une information sur les variables encodées, et qui en plus permet de réduire la dimension de nos données. Le dernier point étant extrêmement important pour la vitesse l'entraînement des modèles. On a décidé d'utiliser plusieurs types d'encodages possibles suivant ce critère: si la colonne a plus ($\geq$) de trois valeurs, l'encoder, sinon faire un *one-hot encoding*.

Codification possible des variables catégorielles:

Une première approche a été de regarder comment codifier des variables catégorielles en gardant le maximum d'information qu'elles nous rapportent sur le *status des réservations* au moyens d'estimations de probabilités conditionnelles.

Soit X une colonne catégorielle qui peut prendre des valeurs $\{x_1, \ldots, x_p\}$ où $p \geq 3$. Si $X[i]$ est la i^{me} valeur dans la colonne X, on estime les probabilités suivantes:

$$\hat{X}_{\text{check\_out}}[i] = \hat{\mathbb{P}}(\text{check\_out}|X = X[i])$$
$$\hat{X}_{\text{cancel}}[i] = \hat{\mathbb{P}}(\text{cancel}|X = X[i])$$
$$\hat{X}_{\text{no\_show}}[i] = \hat{\mathbb{P}}(\text{no\_show}|X = X[i])$$

Et on estime aussi le nombre suivant:

$$\hat{n}[i] = \sum_{j=1}^{N} \mathbf{1}_{X[j]=X[i]}$$

Il suffit alors de remplacer la colonne X par le vecteur de ces probabilités estimées, en ajoutant $\hat{n}$.

Avantages:

- Encodage informatif et supervisé: on capture des relation statistiques entre la variable catégorielle et la variable cible.
- Réduction de dimension: au lieu d'utiliser un *one-hot encoding* qui risque de donner plus de 200 modalités, cet encodage permet de réduire considérablement la dimension, indépendamment du nombres de modalités dans chaque catégorie.
- Une codification interprétable.
- Compatibilité avec plusieurs modèles.

Inconvénients:

- Possibilité de sur-apprentissage: une modalité rare peut impliquer une probabilité de 1.
- Manque de robustesse: une seule observation peut donner un signal fort et faux
- Pas de pondérations sur les catégories rares.
- Dépend énormement du jeu de données.

Pour palier aux inconvénients, on a estimé les probabilités suivantes:

$$\hat{p}_{\text{check\_out}} = \hat{\mathbb{P}}(\text{check\_out})$$
$$\hat{p}_{\text{cancel}} = \hat{\mathbb{P}}(\text{cancel})$$
$$\hat{p}_{\text{no\_show}} = \hat{\mathbb{P}}(\text{no\_show})$$

Et puis on a transformé nos données comme ceci:

$$\hat{X}_{\text{check\_out}}[j] \leftarrow \frac{n[j] \times \hat{X}_{\text{check\_out}}[j] + K \times \hat{p}_{\text{check\_out}}}{\hat{n}[j] + K}$$

$$\hat{X}_{\text{cance}}[j] \leftarrow \frac{n[j] \times \hat{X}_{\text{cancel}}[j] + K \times \hat{p}_{\text{cancel}}}{n[j] + K}$$

$$\hat{X}_{\text{no\_show}}[j] \leftarrow \frac{n[j] \times \hat{X}_{\text{no\_show}}[j] + K \times \hat{p}_{\text{no\_show}}}{\hat{n}[j] + K}$$

où K est un ***hyperparamètre*** de lissage.

Une autre codification fréquentiste possible est de faire le même processus mais cette fois-ci avec les estimateurs suivants:

$$\hat{X}_{\text{check\_out}}[i] = \hat{\mathbb{P}}(X = X[i] | \text{ check\_out})$$
$$\hat{X}_{\text{cancel}}[i] = \hat{\mathbb{P}}(X = X[i] | \text{ cancel})$$
$$\hat{X}_{\text{no\_show}}[i] = \hat{\mathbb{P}}(X = X[i] | \text{ no\_show})$$

avec le même lissage qu'avant. Le tableau suivant représente la grosse différence entre les deux codifications.

Type de codage	**Ce que ça représente**	**Avantage**	**Inconvénient**
$P(y \mid x)$	Influence de la variable sur la cible	Directement utile pour la prédiction	Risque de fuite de cible si mal géré
$P(x \mid y)$	Distribution des catégories pour chaque classe	Peut mieux capturer la structure des classes	Moins directement discriminant pour la prédiction

Table 2: Comparaison entre les deux types de codage probabiliste des variables catégorielles.

Il reste alors les catégories ayant moins ($<$) de 3 valeurs possibles qu'on codifie avec du *one-hot encoding*.

Entraînement:

Première idée:

Ce qu'on a fait, dès lors où on a choisi la codification, est de sélectionner des modèles pour l'entraînement. On en a vu pas mal en cours. On a crée une liste de plusieurs modèles, et on les a entrainé sur 80 % des données *train.csv*, avec une *validation croisée* à 5 blocs.

Pour réduire les biais, à chaque étape de la *cross-validation*, dans le bloc d'entrainement, on calcule la codification mentionnée plus haut, on transforme nos données d'entraînement, ainsi que les données de validation, ***en utilisant uniquement la codification calculée dans le bloc d'entraînement*** avec l'hypothèse $\mathbb{P}_{\text{entraînement}}(Y|X) = \mathbb{P}_{\text{validation}}(Y|X)$.

Maintenant grâce aux blocs de validation, on enregistre les erreurs *OOB* pour chaque model et chaque étape de la *CV*. Avec ces erreurs *OOBs*, on entraîne un *meta-learner*, pour une aggregation d'experts. Pourquoi ? En effet, la raison vient du fait qu'on veuille choisi le meilleur *meta-learner* qui sache mieux combiner entre les ***types*** de modèles, en se basant sur leurs prédictions réelles après des entraînements différents. Ainsi on capture "universellement" les meilleurs combinaisons sur des types de modèles différents. Autrement dit, si une forêt aléatoire entrainée à chaque étape sur différentes données d'entraînement, et qu'elle généralise bien sur l'ensemble de la *CV*, elle aura un coeifficient élevé pour l'aggregation d'experts.

On a remarqué, en implémentant l'idée, que les modèles types fôrets aléatoire avaient plus de capacité à bien généraliser.

Table 3: Résultat de log-loss par modèle et par fold

Modèle	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
HistGradientBoostingClassifier	0.3124	0.3159	0.3139	0.3167	0.3044
ExtraTrees	0.4331	0.4346	0.4331	0.4440	0.4348
LogisticRegression	0.4244	0.4310	0.4276	0.4329	0.4288
RandomForest	0.3750	0.3759	0.3762	0.3800	0.3735
CatBoost	0.3338	0.3371	0.3386	0.3383	0.3293
Moyenne	**0.3757**	**0.3789**	**0.3779**	**0.3824**	**0.3742**

Meta-Learner	Log-loss	Weighted F1
HistGradientBoostingClassifier	0.2846	0.8711

Table 4: Performance du meta-learner sur les prédictions OOF

Fold	Log-loss	Weighted F1
1	0.3320	0.8478
2	0.3362	0.8485
3	0.3401	0.8465
4	0.3329	0.8505
5	0.3084	0.8656
Mean	0.3299	0.8518
Std	0.0111	0.0070

Table 5: Performance du meta-learner sur les folds de test et statistiques globales

De ce qui a précédé, on peut déduire que les méthodes *boosting* sont plus appropriés pour nos données. On en déduit aussi que l'agrégation d'experts fait mieux en moyenne sur la log-loss que chacun des modèles.

Deuxième idée:

Ici, on va chercher à garder la codification que pour le modèle *HistGradientBoostingClassifier* puisqu'il ne gère pas les données catégorielles de manière *"native"*. Et on utilise les modèles *LightGBM* et *CatBoost* en laissant les données catégorielles originelles.

95 % des données d'entraînement serviront à entraîner les trois modèles. La moitié des 5% restantes, serviront à faire du *online learning* avec l'algorithme *EWA - Exponential Weighted Algorithm* pour l'aggregation d'experts:

Algorithm 1 Exponential Weights Algorithm (EWA) pour classification

Require: Nombre d'experts N, learning rate $\eta > 0$, nombre de rounds T

Initialiser les poids des experts : $w_{i,1} = 1$ pour $i = 1, \ldots, N$
for $t = 1$ **to** T **do**
  Recevoir les prédictions de chaque expert : $\hat{y}_{i,t}$ pour $i = 1, \ldots, N$
  Normaliser les poids : $p_{i,t} = \frac{w_{i,t}}{\sum_{j=1}^{N} w_{j,t}}$
  Agréger les prédictions : $\hat{y}_t = \sum_{i=1}^{N} p_{i,t} \hat{y}_{i,t}$
  Observer la vraie étiquette y_t
  Calculer la perte de chaque expert : $\ell_{i,t} = \ell(y_t, \hat{y}_{i,t})$
  Mettre à jour les poids : $w_{i,t+1} = w_{i,t} \cdot \exp(-\eta \cdot \ell_{i,t})$
end for
return Prédictions agrégées $\hat{y}_1, \ldots, \hat{y}_T$

D'abord le résulatat de l'entraînement et test sur la moitié des 5 % de données de test:

Modèle	**Log-loss**	**F1-weighted**
CatBoost	0.3085	0.8660
LGBM	0.3148	0.8602
HistGradientBoosting	0.3100	0.8620
Méthode agrégée (Test final)	0.2994246203229059	0.8704161752594727

Table 6: Performances des modèles sur la première moitié le jeu de test

On voit que l'aggregation d'experts fait mieux que chacun des modèles sur la première moitié du jeu de données test - 5 %.

On test l'aggregation d'experts sur la seconde moitié des données tests:

Modèle	**Log-loss**	**F1-weighted**
Aggregation EWA	0.30402657019077556	0.8641351183805687

Table 7: Performances de l'aggregation d'experts par *EWA* sur la seconde moitié des données test

Conclusion:

On a réussi à implémenter plusieurs modèles différents d'apprentissages supervisé, on a su gérer les données avec des codifications - qu'on n'a pas vu en cours-, on a implémenté des méthodes d'online learning pour pouvoir entraîner au mieux l'aggrégation d'experts.

Ce qu'on aurait pu faire: essayer de trouver les meilleurs hyperparamètres pour nos codifications de données, les meilleurs hyperparamètres pour les modèles et peut-être voir certaines architectures.

SVM

Motivations et Approche

Lors de cet exercice, nous avions également particulièrement envie d'expérimenter un peu avec les SVM. Cette envie provenait des travaux que l'un d'entre nous avait réalisés pour son TIPE en classe préparatoire, qui l'avaient introduit à cette architecture et lui avait particulièrement plu. Nous n'avions que peu d'illusions sur les résultats que des SVM seraient capables d'atteindre face à des architectures plus modernes ou plus performantes, comme des Random Forests, mais nous voulions voir d'où nous pouvions partir et jusqu'où nous pouvions aller.

Notre approche a donc été de partir d'un SVM le plus basique possible et de réaliser diverses expériences pour essayer d'améliorer le premier résultat et de comprendre pourquoi cela fonctionnait ou non.

Traitement des données

Les différentes expériences menées nous ont conduits à adapter progressivement le traitement des données. Nous avons ainsi considéré des features temporelles comme les week-end, la saisonnalité ou les jours de la semaine. Il se trouve que l'utilisation de ces données n'a pas donné de résultats probants.

Nous avons également créé quelques variables dérivées des variables d'origine et qui auraient pu être intéressantes dans le contexte hôtelier. Par exemple, est-ce que le séjour dure uniquement une nuit, un week-end ou un séjour prolongé ? Combien de personnes comprises dans la réservation ou quel est le tarif moyen par personne. Ces variables nous ont permis d'avoir de légers gains de performance.

Le traitement qui s'est avéré le plus utile a été de nettoyer les variables extrêmes. Nous avons ainsi mis en place un clipping quantile à 1% et 99%. Dans les cas où de très grands écarts existaient, nous avons appliqué une transformation logarithmique afin de resserrer l'écart. Nous avons obtenu les meilleurs modèles grâce à ces traitements.

Pour éviter toute erreur sur les données utilisées lors de l'entraînement et du test, le classifieur est inséré dans un pipeline unique qui comprend tous les pré-traitements.

0.1 Méthode d'apprentissage et choix des hyperparamètres

Nous avons fait le choix de partir d'un SVM linéaire. Ce modèle était en effet suffisamment rapide pour tester un maximum de dispositions et d'approches tout en étant robuste.

Nous avons rencontré, lors de certaines expériences, des problèmes de convergence du modèle qui nous ont conduits à accepter un nombre d'itérations plus élevé que celui par défaut et à relaxer les conditions de convergence.

Le choix des hyperparamètres s'est fait simplement, via un split 80/20 : on entraîne sur 80% et on mesure le F1 pondéré sur 20%. On retient la variante

qui fait le meilleur score en validation, puis on entraîne à nouveau sur 100% des données d'entraînement avant de prédire le test. Nous avons préféré cette manière de faire à une validation croisée car elle était plus rapide tout en restant fiable.

Concrètement, nous avons exploré quelques réglages simples. Par exemple, nous avons fait varier la force de régularisation C. Nous avons également repondéré certaines classes minoritaires, mais cela a été détrimental du point de vue du score.

Enfin, nous avons fait des tentatives en utilisant des modèles non linéaires, tels qu'un noyau de fonction de base radiale. Afin d'éviter le coût élevé d'un tel modèle, nous avons testé une approximation avec des Random Fourier Features. Cependant, le temps de compilation, supérieur à 10 heures, nous a découragés de poursuivre cette approche.

Comparaison des méthodes et résultats

Toutes les variantes ont été comparées avec le même protocole : un split 80/20 stratifié, sélection au F1 pondéré sur la partie validation, puis réentraînement sur 100% des données d'apprentissage avant de produire les prédictions de soumission. Ce cadre nous a permis de juger de l'effet réel de chaque idée indépendamment des autres. Le premier modèle entraîné, un SVM linéaire le plus simple possible a servi de témoin.

La variante E4, ajustements de convergence et choix automatique de dual offrent un petit gain par rapport au témoin. Comme il était peu coûteux, nous avons gardé cette configuration par la suite. La meilleure amélioration vient de E9, qui combine clipping des extrêmes et log1p sélectif sur les variables numériques. Cette étape stabilise les distributions et aide le modèle linéaire à séparer les classes de façon plus nette. Les variables introduites en E11, telles que les nuits totales, les clients totaux, les coûts dérivés, etc; apportent un gain modéré, mais restent inférieures à E9 lorsqu'elles sont utilisées seules. Nous espérions qu'en combinant les résultats de E9 et E11 nous obtiendrions un résultat meilleur, mais ils étaient moins bons qu'E9 simple. Les réglages successifs pour améliorer cette combinaison, E13 et E14 n'ont pas permis de progresser. À l'inverse, certaines pistes se sont révélées décevantes : les variables temporelles exploitées en E11 n'apportent pas de signal exploitable au-delà des autres variables. Les approches consistant à pondérer certaines classes, comme E5, ont dégradé la performance. Enfin, l'ajout de non-linéarité par approximation RBF lors de E15 s'est révélé inexploitable à cause du temps de compilation trop élevé.

Au vu de ces observations, la solution retenue pour la soumission est le modèle E9 : un SVM linéaire avec pipeline de clipping + log1p sélectif sur les variables numériques. Ce choix est justifié par son meilleur F1 sur notre validation et sur le leaderboard public, sa vitesse d'entraînement, et sa robustesse. Les pistes plus complexes n'ont en outre pas permis d'améliorer le résultat. Le score de E9 sur le leaderboard privé est 0.80233 contre 0.79440 pour le modèle témoin.

Conclusion

Au terme de ces expériences, les résultats sont plutôt contrastés. Comme nous nous y attendions, les résultats des SVM n'étaient pas exceptionnels. Là où nous avons été un peu déçus, c'est par le peu de progrès permis par nos différentes expériences. Nous pouvons néanmoins en tirer la leçon que les modèles simples ne sont pas forcément beaucoup moins bons que des modèles plus complexes.

Rapport Data Challenge d'Apprentissage supervisé - Problème de régression

Said OUGOUADFEL, Simon LEGRIS, François SLAWNY
November 2025

1. Présentation des problèmes

L'objectif de ce projet est de prédire la popularité d'un morceau de musique à partir de ses caractéristiques audio et métadonnées issues d'un jeu de données Spotify. La variable cible est « popularity », celle-ci est continue, ce qui en fait un problème de régression supervisée. L'enjeu est d'obtenir le meilleur score R^2 possible tout en justifiant rigoureusement les choix méthodologiques. Les modèles testées ont premièrement été ceux vus en cours afin de valider la maîtrise de ces modèles simplifier : SVM, Forêts Aléatoires, Gradient Boosting, AdaBoost etc. Puis pour optimiser notre score nous avons ensuite étendu nos tests à des modèles plus industriels de Boosting comme LightLGBM, CatBoost et XGBoost.

2. Prétraitements éventuels sur les données (+0.08 à +0.12 de score R^2 public)

Premièrement des tests statistiques ont été réalisés pour étudiées les types de corrélations entre nos variables. Mais avant d'appliquer les tests statistiques d'association et de dépendance, les hypothèses fondamentales nécessaires à leur validité ont été systématiquement vérifiées. Ces vérifications préalables ont permis de s'assurer que les méthodes statistiques employées reposent sur des hypothèses empiriquement satisfaites et que les conclusions qui en découlent sont fondées.

L'hypothèse d'indépendance des observations a été considérée comme respectée, les données correspondant à des morceaux distincts sans duplication d'artiste ou d'album. Cette condition garantit la validité des tests statistiques employés, qui reposent tous sur l'hypothèse d'indépendance entre les unités d'observation.

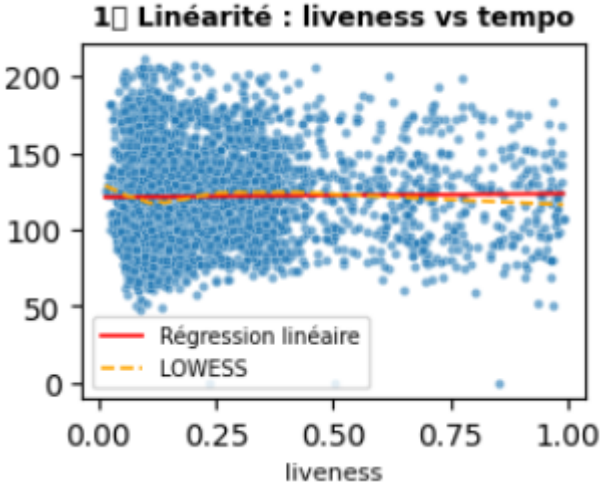
Enfin un enrichissement à partir de données externes à été réalisé pour optimiser le score final.

Tests statistiques et Préprocesseurs (+0.05 à +0.08 de score R^2 public)

Itération 1 sur les corrélations entre variables continues

Vérification des hypothèses préalables:

Il à été vérifié pour chaque couple si ils étaient reliés linéairement de manière significative.

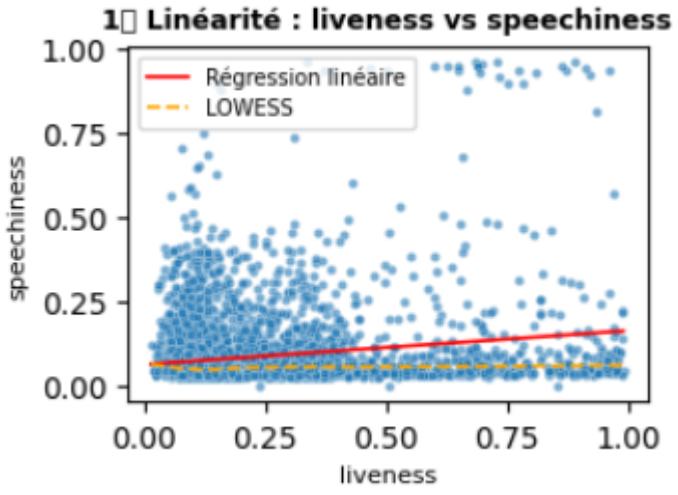


Ce nuage de points montre la relation brute entre les deux variables continues.

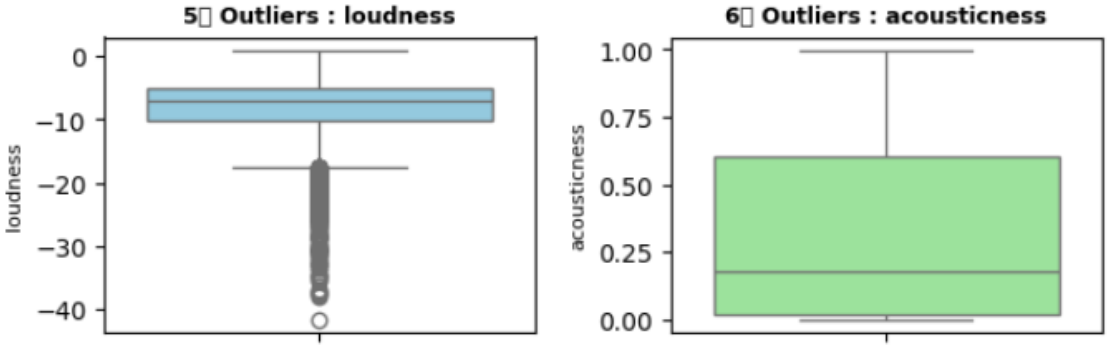
La droite rouge représente la régression linéaire et la ligne orange la courbe LOWESS (lissage non linéaire).

Les deux lignes doivent être proches, indiquant une relation à peu près linéaire entre valence (émotion positive du morceau) et danceability (caractère dansant).
Si la courbe LOWESS s'écarte fortement de la droite, cela signifie une relation non linéaire, et le test de Pearson ne serait plus approprié (il faudrait plutôt Spearman).

Ici, la tendance est globalement linéaire et croissante, donc l'hypothèse de linéarité est vérifiée.



Ici, les droites ne se superposent pas, la relation n'est pas linéaire.



Ces graphiques permettent d'évaluer la symétrie des distributions et la présence d'outliers (valeurs extrêmes).

Ce qu'on cherche :
Des distributions à peu près symétriques et peu d'outliers marqués.
Des points isolés (cercles sous le boxplot) indiquent des valeurs atypiques qui peuvent influencer la corrélation.

Ici la variable acousticness semble assez symétrique, sans outliers importants mais la variable loudness présente beaucoup d'outliers (ronds sous la boîte).

Ces outliers font qu'un test de Pearson n'est plus envisageable car la corrélation de Pearson est très sensible aux valeurs extrêmes.

Quelques points aberrants peuvent complètement modifier la pente de la droite de régression et donc gonfler ou inverser le coefficient de corrélation.

Tests statistiques appliqués:

Les corrélations entre variables continues ont été étudiées à l'aide du coefficient de Pearson lorsque la relation paraissait linéaire, et du coefficient de Spearman lorsqu'elle semblait monotone sans suivre de structure strictement linéaire.
Ces analyses ont mis en évidence des corrélations fortes, atteignant des valeurs proches de 0,7 en valeur absolue, notamment entre les variables energy, loudness et acousticness, suggérant une redondance d'information acoustique.

Conséquences sur les choix de prétraitement:

Les variables energy, loudness et acousticness, fortement corrélées entre elles, ont premièrement été combinées en une seule composante synthétique nommée sound_profile à l'aide d'une réduction de dimension par PCA.
Cette approche permet de réduire la redondance tout en préservant la dimension sonore la plus représentative des morceaux mais celle-ci réduisait légèrement le R^2 score et n'a donc pas été gardée pour le script final

Itération 2 sur les corrélations entre variables continues et catégorielles

Vérification des hypothèses préalables:

La normalité des distributions a d'abord été testée pour l'ensemble des variables continues telles que energy, loudness, danceability, acousticness, valence, tempo ou encore duration_ms.

Le test de Shapiro–Wilk a été appliqué sur chaque couple de variables continue - catégorielle et les résultats ont indiqué que la majorité des distributions pour chaque variable continue dans chaque sous-groupe des variables catégorielles ne suivaient pas une loi normale (p-valeurs inférieures à 0,05).

Ce constat remet en cause l'utilisation de tests paramétriques classiques qui ont déjà été vus en M1 comme le test de Pearson ou l'ANOVA, et a conduit à privilégier des approches non paramétriques, plus robustes face à des distributions asymétriques.

L'homogénéité des variances a ensuite été examinée à l'aide du test de Levene. Les résultats ont révélé une hétérogénéité importante des variances entre groupes, ce qui confirme la nécessité d'utiliser des tests non paramétriques pour les comparaisons inter-catégories, tels que le test de Kruskal–Wallis.

Tests statistiques appliqués:

L'influence des variables catégorielles sur les variables continues a été évaluée par le test de Kruskal–Wallis, plus approprié que l'ANOVA dans un contexte de non-normalité et d'hétéroscédasticité.
Les résultats ont montré des différences significatives de distribution pour la majorité des couples de variables testés (p-valeur proches de zéro), indiquant que les variables catégorielles telles que key, mode, time_signature, explicit et track_genre exercent une influence statistiquement significative sur les variables continues.

Itération 3 sur les corrélations entre variables catégorielles

Pas besoin de vérification des hypothèses préalables

Tests statistiques appliqués:

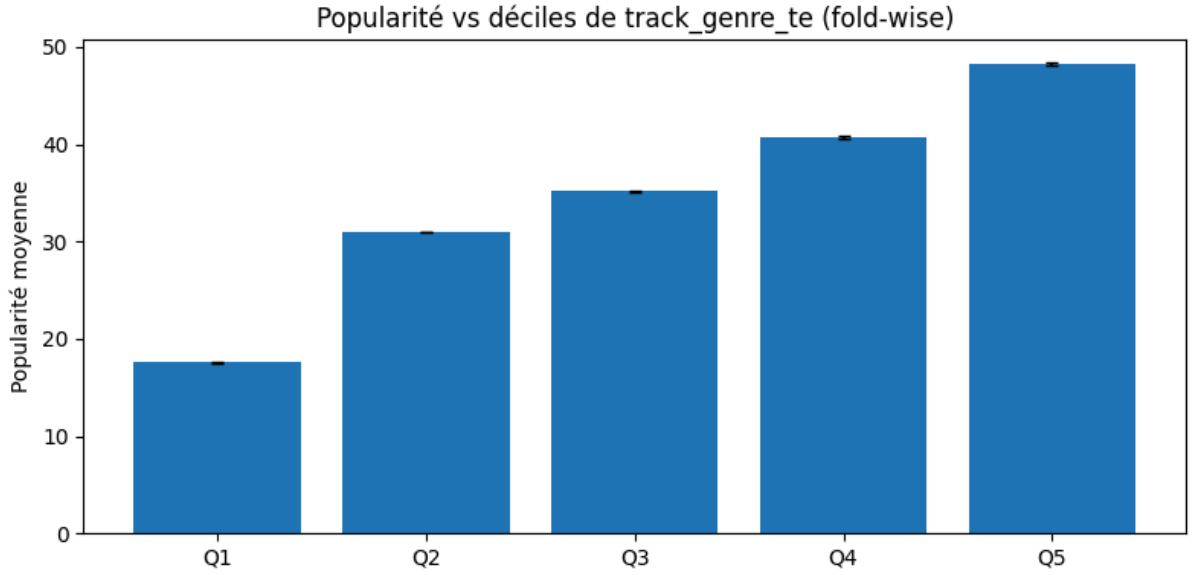
Les relations entre variables catégorielles ont été explorées à l'aide du test du Khi² d'indépendance, et la force de ces associations a été mesurée par le coefficient de Cramér's V.
Les résultats, généralement inférieurs à 0,4, ont mis en évidence des dépendances modérées entre certaines variables qualitatives, suggérant des interactions sans véritable redondance structurelle.

Conséquences final sur les choix de prétraitement pour les variables catégorielles

Le préprocesseur final, combine standardisation, imputation robuste et encodages adaptés à la nature des variables.

Les variables continues sont standardisées après imputation par la médiane, tandis que les variables catégorielles à faible cardinalité (key, mode, time_signature, explicit) sont encodées par one-hot encoding.

La variable track_genre, de très forte cardinalité, a été encodée par une moyenne lissée (target encoding) calculée de manière fold-wise afin d'éviter toute fuite de la cible entre les ensembles d'apprentissage et de validation.



Comme on peut le voir ici la popularité moyenne varie en fonction des différents genre, d'où l'utilité de modéliser cette cette variation par une nouvelle variable track_genre_te.

Les variantes plus expérimentales (encoding CatBoost, décomposition en tokens ou réduction de dimension par PCA) ont été testées mais finalement écartées, n'apportant pas d'amélioration significative de la performance.

Synthèse

Enfin, toutes les variables continues ont été centrées et réduites à l'aide d'un StandardScaler afin de garantir une échelle comparable entre variables.
Les valeurs manquantes ont été imputées par la médiane pour les variables continues, et par le mode pour les variables catégorielles, afin de limiter l'influence des valeurs extrêmes et de préserver la cohérence des distributions.

L'ensemble de ces traitements s'appuie sur des vérifications statistiques rigoureuses.
Les hypothèses de normalité et d'homogénéité ont été testées avant tout choix de méthode, et les tests statistiques retenus – Pearson, Spearman, Kruskal–Wallis, Khi² et Cramér's V – ont été sélectionnés en fonction du type et du comportement empirique des variables.
Le prétraitement qui en résulte permet de réduire la redondance, de limiter la complexité computationnelle et de préserver l'information la plus discriminante pour l'apprentissage supervisé.

Ce protocole assure la validité statistique du traitement des données et garantit la robustesse des modèles construits à partir d'un jeu de 85 000 observations.

Enrichissement des données (+0.03 à +0.04 R^2 score public)

Le jeu de données initial, constitué d'environ 85 000 morceaux, a été enrichi à partir de deux sources externes : data.csv et dataset.csv. Trouvés respectivement à partir de ces sources: Music Recommendation System using Spotify Dataset et Spotify Music Recommendation system. Celles-ci ont été vérifiées, pas de leak de label et aucun leaks d'informations temporelle ou causale n'a été trouvée.

Le premier fichier apportait des variables structurelles relatives aux artistes et albums, comme le nombre de titres par album (album_track_count) ou par artiste (artist_track_count), ainsi que des indicateurs de popularité antérieure des genres (ext_prior_pop_ds).

Le second, issu d'une base open-source, fournissait des informations temporelles telles que l'année de sortie (year) et la date de publication de l'album (release_date), qui ont permis d'introduire une dimension chronologique.

Les deux sources ont été intégrées après vérification de l'absence de fuite entre le jeu d'entraînement et le jeu de test. Les jointures ont été réalisées sur les variables communes comme track_genre et track_id lorsqu'elle était disponible. Les colonnes entièrement vides ont été supprimées, et les variables temporelles normalisées pour assurer leur compatibilité avec le reste des features.

Les tests menés sur d'autres sources (notamment spotify.csv trouvée aussi sur la plateforme Kaggle) n'ont pas montré d'amélioration du score R^2. Ces données ont donc été écartées afin de ne pas alourdir inutilement le modèle.

Mise à jour du préprocesseur final (+0.01 R^2 score public)

La dernière version de notre préprocesseur a été développée prenant en compte toutes les précédentes modifications. Elle conserve la structure initiale tout en introduisant un traitement automatique des distributions continues. Chaque variable est désormais analysée afin d'appliquer la transformation la plus adaptée à sa forme statistique (logarithmique, troncature, Yeo-Johnson ou quantile). Cette adaptation fine permet de réduire l'influence des valeurs aberrantes et d'améliorer la normalité des variables d'entrée.

Ainsi, V3 Plus ne modifie pas la logique du pipeline mais en renforce la robustesse et la stabilité numérique. Cette version a été retenue pour l'ensemble des modèles finaux, car elle permet une meilleure cohérence du prétraitement et une légère amélioration du score R^2 sans complexifier le flux d'apprentissage.

3. Méthodes d'apprentissage

L'entraînement des modèles s'est déroulé en plusieurs phases successives, en partant des méthodes classiques pour aboutir à des approches plus élaborées et adaptées à la structure du jeu de données.

Dans une première étape, les modèles étudiés au cours du semestre ont été évalués. Le SVM et AdaBoost se sont révélés trop rigides pour capturer la complexité du signal, avec des scores R^2 très faibles (inférieurs à 0.3). Le Gradient Boosting a présenté un sous-apprentissage marqué. La Forêt Aléatoire et le GradientBoosting, en revanche, ont rapidement montré une capacité supérieure à modéliser les interactions non linéaires, atteignant respectivement un R^2 proche de 0.38 et 0.42 sur le jeu de test.

Une seconde étape a consisté à explorer des méthodes de boosting plus modernes : LightGBM, CatBoost et XGBoost. Après une série de tests, XGBoost s'est distingué comme le meilleur compromis entre performance et stabilité. L'ajustement progressif des hyperparamètres a conduit à la configuration suivante : un taux d'apprentissage de 0.032, une profondeur maximale de 12, 3000 arbres, un sous-échantillonnage de 0.8 et une régularisation modérée (λ=0.9, α=0.05).

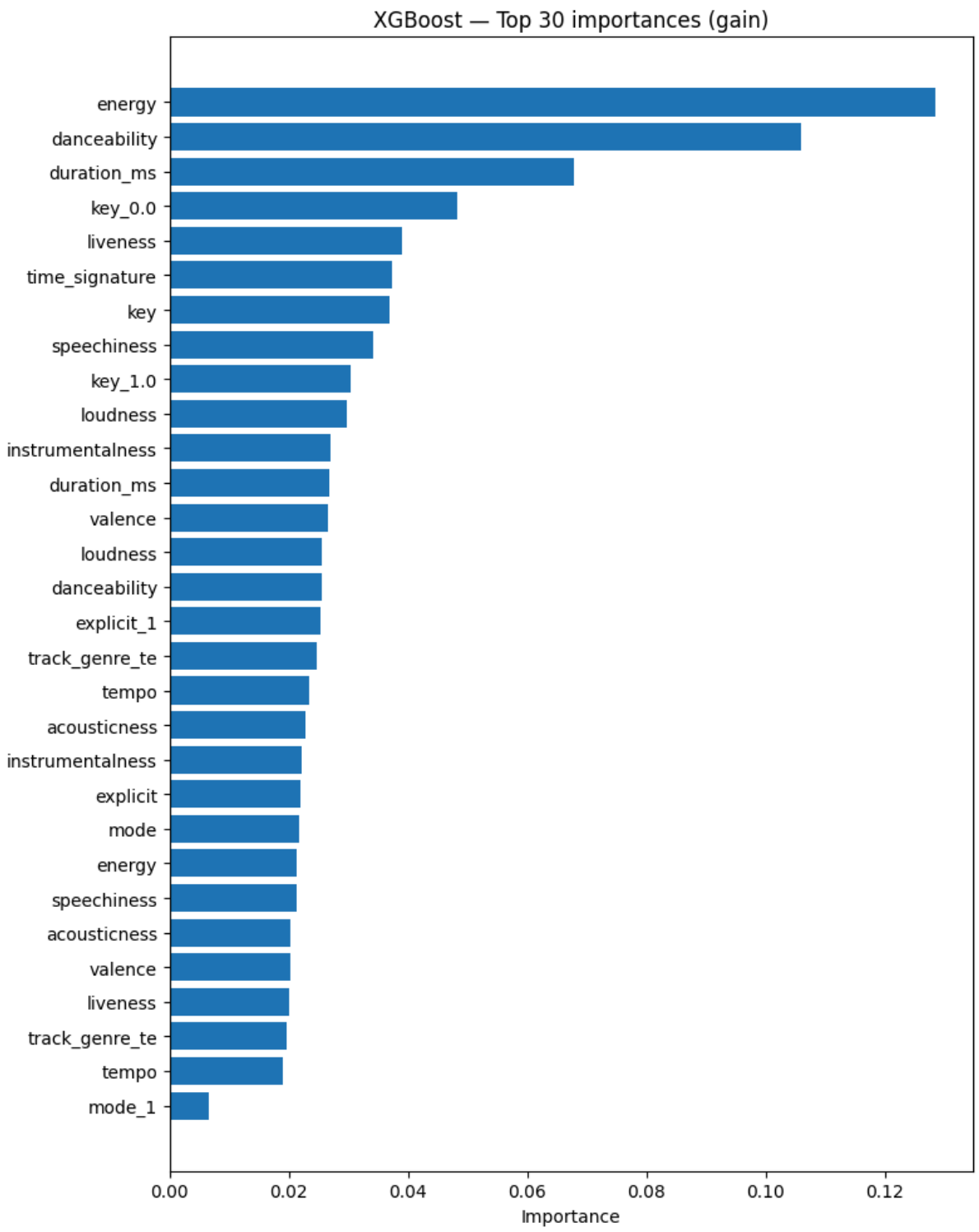
Cette version a atteint un score local de $R^2 = 0.725$ sur le jeu de test interne, confirmant sa supériorité sur les autres modèles.

Enfin, une architecture de type *Mixture of Experts (MoE)* a été introduite afin de capturer les sous-structures acoustiques latentes du dataset. Un *Gaussian Mixture Model* (GMM) a d'abord segmenté les morceaux selon leurs profils sonores (énergie, valence, tempo, etc.), puis un modèle XGBoost a été entraîné séparément pour chaque cluster, les prédictions finales étant obtenues par pondération selon les probabilités d'appartenance. Ce modèle n'a pas augmenté radicalement le R^2 (gain d'environ 0.005 à 0.008), mais il a permis d'améliorer la stabilité des résultats entre validation locale et soumission Kaggle.

4. Comparaison des méthodes et solution retenue

La comparaison finale a mis en évidence la nette supériorité des méthodes de boosting sur les approches traditionnelles. Les modèles linéaires ou faiblement non linéaires, tels que le SVM ou l'AdaBoost, sous-apprennent fortement. La Forêt Aléatoire, bien que robuste, atteint rapidement un plafond de performance.

Les modèles LightGBM et CatBoost se sont montrés compétitifs, mais leur comportement était plus variable selon les splits de validation. XGBoost a finalement été retenu comme modèle principal pour son équilibre entre performance, stabilité et maîtrise des hyperparamètres.



Ce graphique illustre la contribution relative des variables à la prédiction de la popularité musicale dans le modèle XGBoost final.

Les variables acoustiques dominent largement : energy et danceability apparaissent comme les déterminants principaux, suivis par duration_ms et certaines composantes tonales (key, time_signature). Cela confirme que les morceaux perçus comme dynamiques, rythmés et structurés de façon régulière tendent à être plus populaires.

On observe également que des indicateurs tels que speechiness, liveness ou valence participent modestement à la variance expliquée, traduisant des effets secondaires liés à la texture sonore ou à l'émotion du morceau.

La présence de variables encodées comme track_genre_te et explicit_1 parmi les facteurs explicatifs montre que le genre musical et certains aspects textuels ou de contenu ont aussi un rôle, bien que secondaire face aux caractéristiques acoustiques pures.

Globalement, cette hiérarchie confirme que la popularité est mieux prédite par l'énergie et la rythmicité du morceau que par des métadonnées plus abstraites, ce qui valide empiriquement la structure du modèle et la pertinence du préprocesseur.

5. Conclusion

Ce travail montre que la popularité d'un morceau peut être modélisée de manière satisfaisante à partir de ses caractéristiques musicales et structurelles. La démarche adoptée, fondée sur des tests statistiques rigoureux et des choix méthodologiques justifiés, a permis d'élaborer un pipeline cohérent allant de l'analyse exploratoire à la modélisation avancée.

Le modèle final, associant un préprocesseur optimisé (*v3 plus*), un XGBoost affiné et une calibration postérieure, capture plus de 70 % de la variance expliquée dans les données. L'enrichissement via *data.csv* et *dataset.csv* s'est révélé essentiel, apportant une meilleure représentation des tendances temporelles et des effets de production.

Les perspectives d'amélioration concernent désormais l'intégration de variables contextuelles (collaborations, tendances de genre, temporalité fine) et l'introduction de contraintes monotones dans XGBoost pour stabiliser la hiérarchie des variables explicatives. Une exploration plus poussée du stacking ou du fine-tuning adaptatif pourrait également conduire à un léger gain supplémentaire de performance.