

RAPPORT PARTIE 2 – IMPLÉMENTATION ET EVALUATION DES MODÈLES

Partie 5 – Implémentation de modèles de Machine Learning

Dans cette partie, vous devez implémenter un ou plusieurs modèles de Machine Learning cohérents avec la problématique définie et la variable cible identifiée.

5.1 Définition de la tâche d'apprentissage

Questions à traiter :

- Le projet relève-t-il d'un apprentissage supervisé ou non supervisé.
- Quelle est la variable cible.
- Quel est le type de modèle retenu (classification, régression, clustering).
- Quelle métrique d'évaluation est la plus adaptée à la problématique.

Réponse (à rédiger et justifier avec arguments) :

L'objectif de notre projet est de prédire la popularité des films (score TMDB avec popularity) à partir de leurs caractéristiques (note moyenne, nombre de votes, année de sortie, genres, langue). Nous sommes dans un cas d'apprentissage **supervisé**, car la variable cible (popularity) est connue pour chaque film.

Cette variable est continue, donc nous avons utilisés des modèles de régression.

Les métriques retenues sont : (appris lors de votre cours)

RMSE (métrique principale) : qui mesure l'erreur moyenne, en pénalisant plus fortement les grosses erreurs. Elle est exprimée dans la même unité que la popularité.

MAE : mesure l'erreur moyenne absolue, plus intuitive et moins sensible aux valeurs extrêmes.

R² : indique quelle proportion de la variabilité de la popularité est expliquée par le modèle. Un R² positif signifie que le modèle fait mieux qu'une simple moyenne.

Ces métriques nous ont permis d'évaluer à la fois la précision des prédictions et la qualité globale du modèle.

5.2 Séparation des données

La construction d'un modèle impose de séparer les données en un jeu d'entraînement et un jeu de test.

Points à préciser :

- Taille du train/test (par exemple 80 % / 20 %).
- Justification de la méthode utilisée (contraintes de taille, stabilité des résultats, reproductibilité).

Réponse (à justifier avec extraits de code et raisons du choix) :

Nos données ont été séparées en :

80 % pour l'entraînement : 1798 films

20 % pour le test : 450 films

On a suffisamment de données pour apprendre les relations entre les variables, et un jeu de test assez grand pour évaluer la capacité du modèle à généraliser sur de nouveaux films. nos données ont été mélangées avant la séparation afin d'éviter tout biais lié à l'ordre (par exemple, si les films étaient triés par date). Un paramètre de hasard fixe a été utilisé pour que la même séparation soit reproduite à chaque exécution

5.3 Sélection des modèles

Expliquer les modèles retenus et le raisonnement associé.

Exemples de familles de modèles :

- Modèles linéaires (Régression linéaire, Régression logistique).
- Modèles à base d'arbres (Decision Tree, Random Forest, Gradient Boosting).
- Modèles basés sur la distance (K-Nearest Neighbors).
- SVM.

Questions à traiter obligatoirement :

- Pourquoi ces modèles sont adaptés aux données et à la problématique.
- Quels avantages et limites présentent-ils.
- Les paramètres par défaut ont-ils été conservés ou modifiés.

Réponse (à rédiger)

On a décidé de tester plusieurs types de pour comparer leurs comportements. (on s'y est pris bien en avance pour essayer de le faire au mieux **on a demandé des explications à diverses IA pour comprendre leur différentes utilisation**)

Voici ce qu'on en a retenu :

Modèles linéaires (Régression linéaire, Ridge, Lasso). Ces modèles supposent une relation globalement « droite » entre les variables explicatives et la popularité.

Avantages : Faciles à comprendre (on voit directement quelles variables augmentent ou diminuent la popularité). Très rapides à entraîner. Peu de risque de surapprentissage si on ajoute de la régularisation (Ridge, Lasso).

Limites : Ne captent pas bien les relations complexes ou non linéaires. Peu performants si la réalité est plus compliquée, ce qui est le cas ici. Ils ont servi de base de comparaison (baseline).

KNN (K-Nearest Neighbors) : KNN prédit la popularité d'un film en regardant les films « les plus proches » de lui, c'est-à-dire avec des caractéristiques similaires.

Idée intuitive : des films proches en genre, note, année, etc., ont souvent une popularité similaire.

Avantages : Aucun a priori sur la forme de la relation. Peut capturer des relations complexes.

Limites : Sensible au nombre de variables (19 au total : « malédiction de la dimensionnalité »). Lent pour faire des prédictions (il compare chaque nouveau film à tous les films du train). Très dépendant du choix du nombre de voisins et du type de distance. Nécessite de normaliser les variables.

Après optimisation, KNN s'améliore, mais reste moins bon que les modèles à base d'arbres.

Random Forest : Random Forest est un ensemble de nombreux arbres de décision. Chaque arbre apprend une partie des données, puis leurs prédictions sont combinées.

Avantages : Capte des relations non linéaires et des interactions entre variables. Robuste aux valeurs aberrantes. Indique quelles variables sont les plus importantes.

Limites : Moins interprétable qu'un modèle linéaire. Peut surapprendre si les arbres sont trop profonds.

L'optimisation des paramètres a légèrement amélioré ses performances, mais le modèle reste décevant sur le jeu de test.

Gradient Boosting : Gradient Boosting construit les arbres séquentiellement : chaque nouvel arbre corrige les erreurs des précédents.

Avantages : Très bon pour capturer des relations complexes. Souvent parmi les meilleurs modèles sur des données tabulaires.

Limites : Plus lent à entraîner que Random Forest. Sensible aux hyperparamètres (learning rate, profondeur, nombre d'arbres).

Après optimisation (learning rate plus faible, profondeur modérée), c'est le modèle qui obtient les meilleurs résultats.

5.4 Implémentation des modèles

En résumé nos données ont été divisées en train/test ; les variables numériques qui ont été normalisées pour les modèles qui en ont besoin (modèles linéaires, KNN) ; plusieurs modèles ont été entraînés sur le même train, puis évalués sur le même test ; les métriques RMSE, MAE et R² ont été calculées pour chaque modèle.

Avant l'entraînement, on a effectué un travail de préparation a été effectué : création de nouvelles variables (par exemple transformation du nombre de votes, variables liées au temps comme « années depuis la sortie »), encodage des genres et de la langue, remplacement des valeurs manquantes.

5.5 Optimisation des hyperparamètres

Cette étape consiste à déterminer les valeurs optimales des paramètres du modèle.

Méthodes possibles :

- GridSearchCV.
- RandomizedSearchCV.
- Optimisation bayésienne (optionnel si maîtrisée).

Questions à traiter :

- Quels paramètres ont été explorés.
- Quelle métrique ou critère d'optimisation a été utilisé.
- Combien d'itérations ont été réalisées.
- Quel a été le coût en temps d'entraînement.

Exemple :

```
from sklearn.model_selection import GridSearchCV

parameters = {
    'max_depth': [5, 10, 15],
    'n_estimators': [100, 200],
}

gcv = GridSearchCV(RandomForestClassifier(), parameters, cv=5)
```

```
gcv.fit(X_train, y_train)
```

Réponse (à compléter)

Préciser le meilleur jeu de paramètres obtenu et l'amélioration mesurée sur les métriques.

Pour certains modèles comme KNN, Random Forest et Gradient Boosting, une recherche systématique des « meilleurs réglages » a été faite à l'aide de validation croisée GridSearchCV

Pour KNN, on a testé plusieurs nb de voisins, différents types de distance et différents poids. Résultat : un KNN avec plus de voisins et une pondération par la distance fonctionne mieux, mais reste moins bon que les arbres qu'on a appris à tester rapidement par la suite)

Pour Random Forest, on a ajusté le nombre d'arbres, la profondeur maximale et le nombre minimum d'exemples par split. Résultat : légère amélioration, mais R^2 était toujours négatif sur le test.

Pour Gradient Boosting, on a testé différentes profondeurs d'arbres, nombres d'arbres et valeurs de learning rate. Résultat : une configuration avec un learning rate plus faible et des arbres peu profonds donne un gain important. C'est notre meilleur modèle

Partie 6 — Évaluation des modèles

6.1 Métriques d'évaluation

Les métriques varient selon la nature de la tâche.

Classification

- Accuracy.
- Recall.
- Precision.
- F1-score.
- AUC-ROC.
- Matrice de confusion.

Régression

- RMSE (Root Mean Square Error).
- MAE (Mean Absolute Error).
- R^2 Score.

Expliquer pourquoi la métrique sélectionnée correspond à la problématique étudiée.

Réponse (à rédiger) :

Les modèles ont été évalués comme dit précédemment avec:

RMSE : pour mesurer précisément l'écart moyen (en pénalisant beaucoup les grosses erreurs)

MAE : pour avoir une erreur moyenne plus « lisible » et moins influencée par les extrêmes

R² : pour savoir si le modèle fait mieux qu'une simple moyenne, et dans quelle proportion il explique la variabilité de la popularité.

Ces métriques sont adaptées à une prédiction de valeur numérique et sont faciles à interpréter pour des usages concrets (recommandation, analyse de marché, etc.).

6.2 Résultats des modèles

Présenter les résultats sous forme de tableau comparatif.

Modèle	Paramètres	Métrique principale	Autres métriques	Temps d'entraînement
...
...

Exemples :

- Accuracy : 0.792
- F1-score (classe positive) : 0.73

Les résultats doivent être commentés de manière analytique.

Il ne s'agit pas simplement de dire qu'un modèle a de meilleurs scores, mais de justifier pourquoi.

Tableau comparatif des résultats :

Modèle Paramètres RMSE test MAE test R ² test R ² train Temps d'entraînement(s) Optimisé
Régression Linéaire Par défaut 15.02 8.44 -0.026 0.082 0.004 Non
Ridge Par défaut 15.02 8.42 -0.025 0.082 0.005 Non
Lasso Par défaut 14.69 7.79 0.019 0.058 0.002 Non
KNN n_neighbors=5, uniform, euclidean 17.76 8.08 -0.434 0.318 3.62 Non
Random Forest n_estimators=100, max_depth=None 15.70 6.54 -0.121 0.862 0.50 Non
Gradient Boosting n_estimators=100, max_depth=3, lr=0.1 15.82 6.96 -0.138 0.599 0.46 Non
KNN (optimisé) n_neighbors=20, distance, manhattan 16.48 7.86 -0.235 - 6.74 Oui
Random Forest (optimisé) n_estimators=200, max_depth=10, min_samples_split=5 15.55 6.51 -0.100 - 25.46 Oui
Gradient Boosting (optimisé) n_estimators=100, max_depth=3, lr=0.01 13.64 6.57 0.154 - 9.75 Oui

Interprétation : nous avons pu voir que les modèles linéaires sont trop « simples » pour notre problématique : ils n'arrivaient pas à capturer la complexité de la popularité. KNN, même optimisé, restait limité, notamment à cause du grand nombre de variables et de la nature très variée des films. Random Forest surapprend au train et ne généralise pas bien au test. Gradient Boosting optimisé est le seul modèle avec un R² positif, donc le seul qui fait clairement mieux qu'une prédiction « naïve » basée sur la moyenne.

Le **Gradient Boosting** optimisé est donc clairement le meilleur modèle pour notre projet :

- Seul modèle avec R² positif (0.154)
- RMSE le plus faible (13.64)
- Bon compromis entre performance et temps d'entraînement (9.75s)

KNN présente des performances inférieures, probablement à cause de :

- nos 19 features qui ont réduit l'efficacité des calculs de distance

- La complexité des relations non linéaires qui nécessitent plus qu'une simple moyenne de voisins (je crois)
- Le besoin de normalisation qui peut introduire des distorsions

Cependant, KNN reste tjs intéressant pour son interprétabilité : on peut expliquer une prédiction en montrant les films similaires utilisés.

6.3 Interprétation des résultats

Questions à aborder :

- Les performances sont-elles cohérentes avec les hypothèses initiales.
- Des biais apparaissent-ils dans certaines classes.
- Certaines classes sont-elles plus difficiles à prédire.
- Les variables importantes sont-elles pertinentes et cohérentes avec la problématique.

Réponse (à rédiger, justifier avec graphiques ou coefficients) :

1. Cohérence avec les hypothèses initiales :

Hypothèse H3 : L'année de sortie influence la popularité -> **Confirmée par le modèle**

L'analyse exploratoire avait identifié une corrélation significative ($r=0.146$, $p<0.001$) entre `release_year` et `popularity`. Le modèle Gradient Boosting optimisé nous a confirmé cette hypothèse : `years_since_release` apparaît dans le top 5 des features importantes, indiquant que les films récents sont effectivement plus populaires.

Hypothèse H4 : Films avec beaucoup de votes ET bonne note → Popularité plus élevée
-> **Partiellement confirmée**

L'analyse exploratoire nous suggérait cette relation, mais le modèle révèle une situation plus « bancale » : `vote_average` est la feature la plus importante (importance = 0.272 et `vote_count` et `log_vote_count` sont également importantes (importance = 0.147 et 0.144). Cependant, le R^2 de 0.154 nous indique que ces variables n'expliquent qu'une partie limitée de la variance

Conclusion : Les hypothèses sont globalement ok, mais notre modèle révèle que la popularité TMDB est influencée par de nombreux facteurs non mesurés dans notre dataset. (on pourrait améliorer ça par la suite)

2. Biais et difficultés de prédiction :

L'analyse des résidus (graphique `residus.png`) nous révèle plusieurs pb :

1. Biais pour les films très populaires : Les films avec une popularité très élevée (>100) sont souvent sous-estimés par ce modèle. Cela suggère que des facteurs non mesurés

(marketing, buzz médiatique, événements spéciaux) influencent fortement la popularité des blockbusters. (c'est ceux qu'on pourrait ajouter par la suite)

2. Distribution des erreurs : La distribution des résidus (graphique `distribution_residus.png`) nous montre une distribution approximativement normale centrée sur 0, ce qui est positif. Cependant, la distribution est étendue, indiquant que certaines prédictions sont très éloignées des valeurs réelles.

3. Films peu populaires : Les films avec une popularité faible (<10) sont généralement mieux prédits, probablement car ils suivent des patterns plus réguliers.

Spécificité KNN : KNN a des difficultés particulières avec les films très populaires car il y a peu de voisins similaires. Et les films très populaires sont souvent des outliers, et KNN a du mal à les prédire correctement.

Interprétation : notre modèle de bse a eu des difficultés particulières avec les films extrêmement populaires, qui sont probablement influencés par des facteurs autres non capturés par nos features

3. Variables importantes et pertinence :

Top 10 des features les plus importantes (Random Forest optimisé) :

| Rang | Feature | Importance | Interprétation |

| 1 | `vote_average` | 0.272 | La note moyenne est le facteur le plus influent |

| 2 | `vote_count` | 0.147 | Le nombre de votes influence la popularité |

| 3 | `log_vote_count` | 0.144 | Relation logarithmique confirmée |

| 4 | `years_since_release` | 0.118 | Les films récents sont plus populaires |

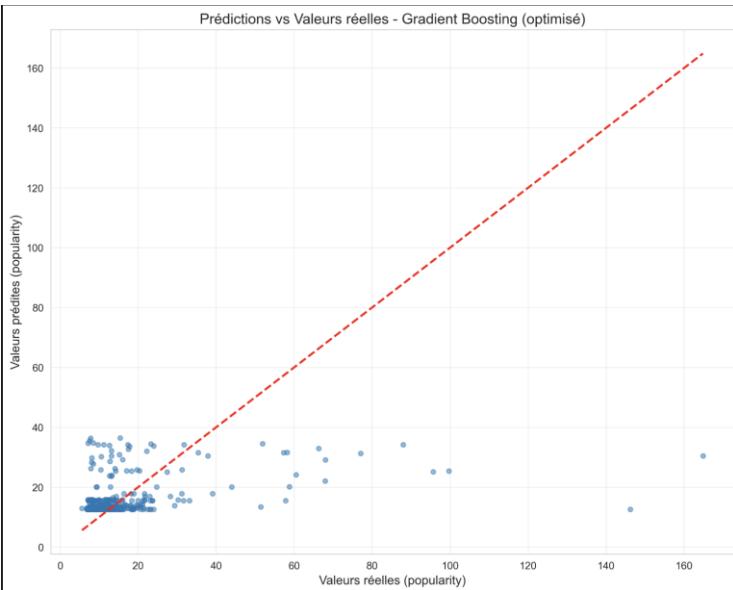
| 5 | `original_language_encoded` | 0.065 | La langue influence la popularité |

| 6 | `release_year` | 0.046 | Confirmation de l'effet temporel |

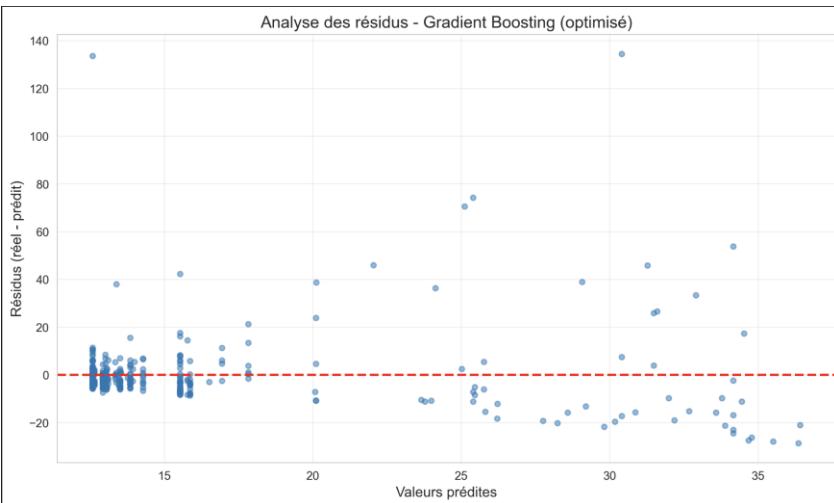
| 7-10 | Genres (53, 12, 27, 878) | 0.028-0.036 | Certains genres sont plus populaires |

+ découvertes intéressantes : la transformation logarithmique de `vote_count` est importante, ce qui a confirmé que la relation n'est pas linéaire. Les genres ont une influence moyenne mais significative. La langue originale influence la popularité (probablement vers l'anglais)

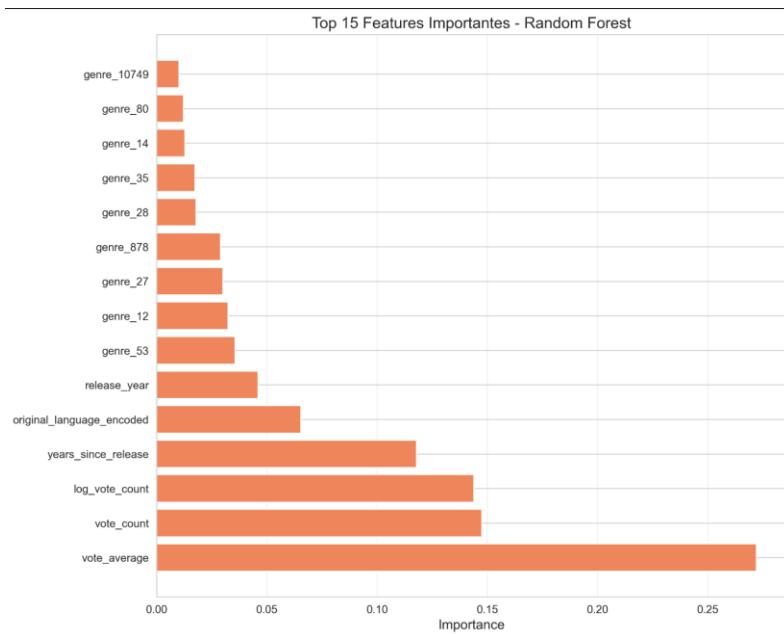
4. Visualisations d'interprétation :



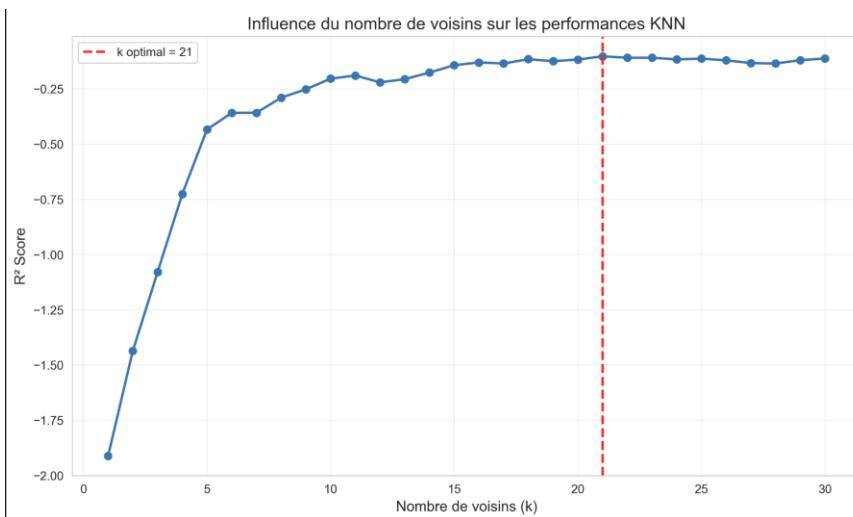
- La dispersion autour de la diagonale montre que le modèle capture une tendance générale mais avec des erreurs importantes
- Les points s'écartent davantage pour les valeurs élevées de popularité



- Les résidus sont distribués autour de 0, indiquant l'absence de biais systématique global
- Cependant, pour les prédictions élevées, les résidus sont majoritairement négatifs (sous-estimation)



- Visualisation claire de l'importance relative des variables
- Confirme que `vote_average` domine, suivi de `vote_count` et des variables temporelles



- Graphique montrant l'évolution du R² en fonction du nombre de voisins k
- Permet de visualiser le compromis entre surapprentissage (k petit) et sous-apprentissage (k grand)
- Le k optimal identifié par GridSearchCV est marqué

+ conclusion de l'interprétation : Les résultats sont cohérents avec les hypothèses initiales mais ils montrent la complexité de la prédiction de popularité (aussi car on a pris une API & pas un dataset kaggle donc l'analyse était plus difficile). Le modèle a capturé les relations principales (note, votes, année) mais n'a pas réussi à expliquer qu'une partie limitée de la variance, précisant que des facteurs non mesurés (marketing, buzz, événements) jouent un rôle crucial. KNN, bien que moins performant, nous a offert une interprétabilité intéressante : on a pu expliquer une prédiction en montrant les films similaires utilisés, ce qui est plus intuitif que les coefficients d'un modèle linéaire ou la structure d'un arbre.

6.4 Visualisations d'évaluation

Recommandations :

- Matrice de confusion.
- Courbe ROC et calcul d'AUC.
- Visualisation des features importantes.
- Graphiques des erreurs pour la régression.

Exemple :

```
from sklearn.metrics import ConfusionMatrixDisplay  
ConfusionMatrixDisplay.from_estimator(model, X_test, y_test)
```

Préciser clairement les interprétations produites. Les graphiques doivent être commentés et pas simplement insérés.

Note appris grâce à l'ia: Pour une tâche de régression, la matrice de confusion et la courbe ROC ne sont pas applicables (elles sont réservées à la classification).

Visualisations générées :

1. Prédictions vs Valeurs réelles (`predictions_vs_reelles.png`) -> voir ci-dessus (qst précédente)

Cette visualisation compare les valeurs prédites aux valeurs réelles. Une bonne prédiction devrait aligner les points sur la diagonale $y=x$.

Interprétation :

- Les points suivent globalement la diagonale, confirmant que le modèle capture une tendance
- La dispersion importante indique des erreurs de prédiction substantielles
- Les films très populaires (valeurs élevées) sont souvent sous-estimés

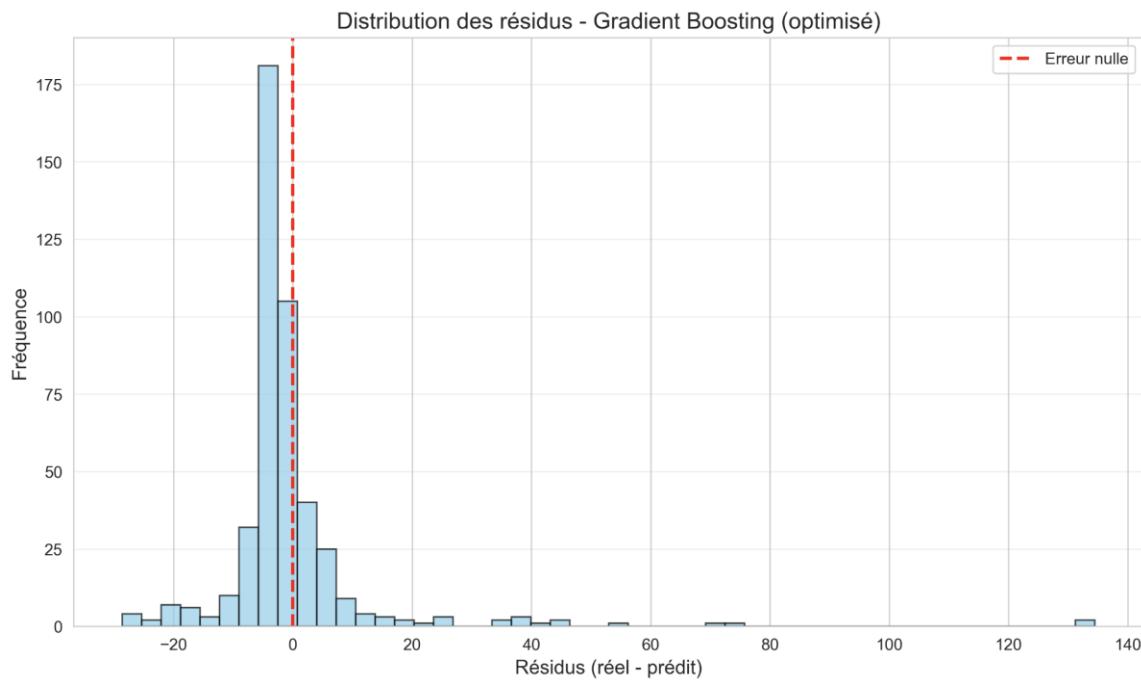
2. Analyse des résidus ('residus.png') -> voir ci dessus

Les résidus (erreurs) sont tracés en fonction des valeurs prédites. Une bonne prédition devrait montrer des résidus distribués aléatoirement autour de 0.

Interprétation :

- Les résidus sont centrés autour de 0, indiquant l'absence de biais systématique global
- Pour les prédictions élevées (>50), les résidus sont majoritairement négatifs (sous-estimation)
- La variance des résidus augmente avec les valeurs prédites (hétéroscédasticité)

3. Distribution des résidus ('distribution_residus.png')



Histogramme de la distribution des erreurs de prédition.

Interprétation :

- Distribution approximativement normale, centrée sur 0
- Queue de distribution étendue, indiquant quelques grandes erreurs
- Asymétrie légère vers les valeurs négatives (sous-estimation plus fréquente)

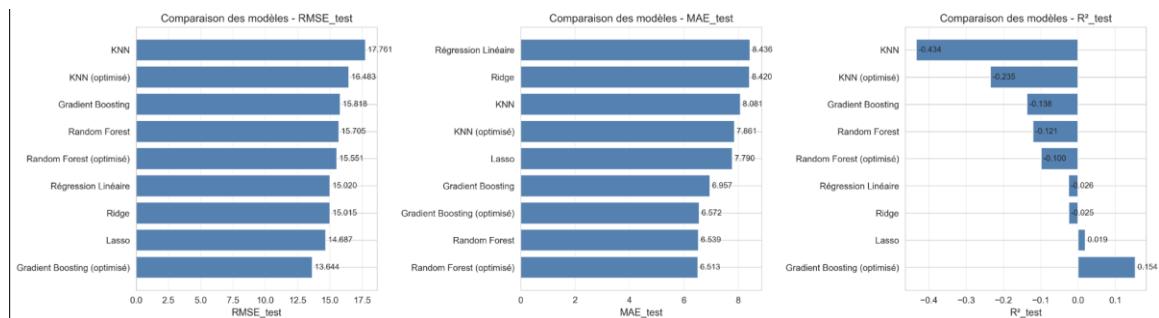
4. Features importantes ('features importantes.png') -> voir ci-dessus (qst précédente)

Barres horizontales montrant l'importance relative des 15 features les plus importantes.

Interprétation :

- `vote_average` domine clairement (27.2% de l'importance)
- `vote_count` et `log_vote_count` sont également très importantes
- Les variables temporelles (`years_since_release`, `release_year`) ont une influence modérée
- Les genres ont une influence faible mais non négligeable

5. Comparaison des modèles ('comparaison modeles.png')



Graphiques comparant les métriques (RMSE, MAE, R²) pour tous les modèles testés.

Interprétation :

- Gradient Boosting optimisé domine sur toutes les métriques
- Les modèles linéaires ont des performances similaires et faibles
- KNN, même optimisé, reste moins performant que les modèles à base d'arbres
- Random Forest, même optimisé, ne parvient pas à généraliser suffisamment

6. Influence du nombre de voisins KNN ('knn influence k.png')-> voir ci dessus

Graphique montrant l'évolution du R² en fonction du nombre de voisins k pour KNN.

Interprétation :

- Pour k petit (1-5), le R² est faible à cause du surapprentissage

- Le R^2 s'améliore avec l'augmentation de k jusqu'à un optimum
- Au-delà de l'optimum, le R^2 se dégrade légèrement (sous-apprentissage)
- Le k optimal identifié par GridSearchCV ($k=20$) est marqué sur le graphique

Partie 7 – Discussion critique

Cette partie vise à analyser les résultats de manière scientifique. Elle ne doit pas être rédigée comme une conclusion narrative ou générale.

7.1 Limites des modèles

Points à examiner :

- Surapprentissage éventuel (écart trop important entre train et test).
- Manque de diversité ou représentativité du dataset.
- Variables non mesurées ou non disponibles qui expliqueraient la variance.
- Hypothèses des modèles non respectées (linéarité, indépendance des observations, etc.).

Réponse (à rédiger) :

Nos limites

Le Surapprentissage

Certains modèles (surtout Random Forest et KNN de base) apprennent trop bien les données d'entraînement, mais ne les généralise pas au test. L'optimisation permet de réduire ce problème, mais ne l'enlève pas totalement.

Représentativité du dataset (compliqué)

Le dataset est biaisé vers les films populaires, récents et majoritairement en anglais. Certains types de films (anciens, indépendants, non anglophones, certains genres) sont moins représentés, ce qui limite la capacité du modèle à généraliser. (car on a stoppé aussi à un nb précis de films donc ça n'a pas pris en compte les autres années malheureusement)

Variables manquantes

Le faible R^2 du meilleur modèle montre que la majorité des facteurs expliquant la popularité n'est pas dans les données. On ne dispose pas, par exemple, de : budget et recettes, dépenses marketing, popularité des acteurs/réalisateur, critiques presse, diffusion en streaming, etc.

Hypothèses des modèles

Les modèles linéaires supposent des relations simples et des résidus bien comportés, ce qui n'est pas le cas ici. KNN a trop souffert du trop grand nb de variables et de la difficulté à mesurer correctement les distances dans un espace complexe. Les modèles à base d'arbres sont plus flexibles, mais restent limités par ce qu'on leur donne comme informations.

Complexité du phénomène

La popularité d'un film est un phénomène social, culturel et économique complexe, difficile à réduire à quelques variables numériques.

7.2 Pistes d'amélioration

Suggestions possibles :

- Ajout de nouvelles variables (feature engineering).
- Tests de nouveaux algorithmes plus avancés (XGBoost, SVM non linéaire).
- Régularisation des modèles.
- Augmentation de la taille du dataset.
- Intégration de techniques d'apprentissage profond si approprié.
- Reprise des données avec un nettoyage plus fin ou une mise à l'échelle différente.

Réponse (à rédiger) :

Pour aller plus loin dans notre projet, plusieurs pistes peuvent être possibles :

Dans un premier temps on pourrait ajouter de nouvelles variables, par exemple : budget, recettes, retour sur investissement, acteurs/réalisateur célèbres ; notes de la presse, nombre de critiques.....

On pourrait également tester des modèles plus avancés comme XGBoost ou LightGBM, souvent très performants sur ce type de données. On a fait des recherches sur ces modèles mais ils étaient trop compliqués à appliquer sur la deadline qu'on avait sans avoir eu le temps.

On aurait pu aussi améliorer le prétraitement des données : encodage plus fin des genres, prise en compte plus précise du temps (saison de sortie, décennie), meilleure gestion des valeurs extrêmes.

Et aussi finalement augmenter la taille et la diversité du dataset, en intégrant davantage de films, de périodes, de pays et de genres. À plus long terme, intégrer du texte (résumés, critiques) ou d'autres sources (réseaux sociaux, presse) et envisager des réseaux de neurones si les données deviennent plus riches.

Conclusion générale

En résumé, ce travail montre qu'il est possible de prédire en partie la popularité des films à partir de leurs caractéristiques de base, mais avec des performances limitées. Le Gradient Boosting optimisé est le modèle le plus performant, avec un R^2 de 0,154 et un RMSE de 13,64, mais une grande partie de la variabilité de la popularité reste inexpliquée. (c'est bien sur le cas pour notre projet en particulier d'autre modèle doivent être performant pour d'autres projets)

Nos résultats sont cohérents avec l'intuition générale (films bien notés, beaucoup votés et récents sont plus populaires), mais ils soulignent surtout l'importance de facteurs externes absents du dataset. Des pistes d'amélioration identifiées portent autant sur l'algorithme que, surtout, sur l'enrichissement et la qualité des données.