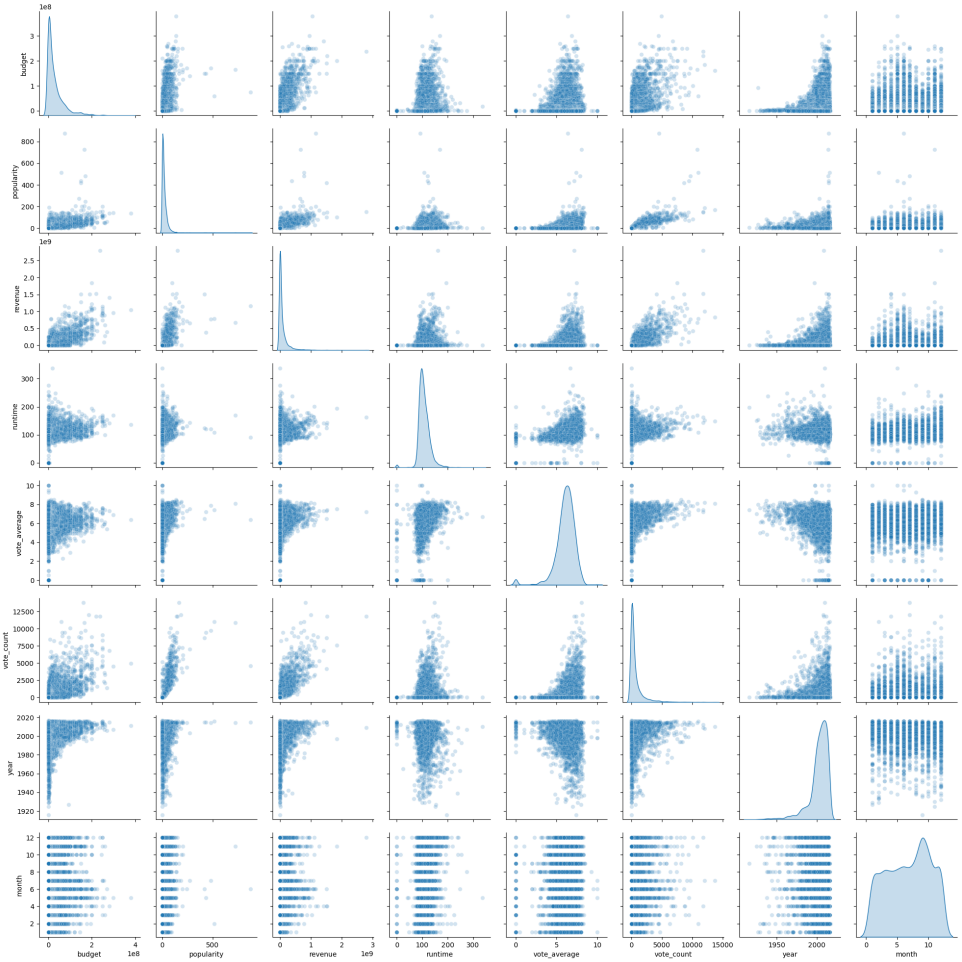
Rapport de Projet : Apprentissage Automatique pour la Prédiction des Notes de Films

Mouhcine MAIMOUNI Theo CONDAMIN

Introduction

Ce projet s'inscrit dans le cadre de l'analyse prédictive appliquée au domaine du cinéma. L'objectif principal est de construire un modèle capable de prédire la note d'appréciation publique d'un film en fonction de plusieurs caractéristiques. Cette démarche repose sur l'exploitation d'un dataset contenant diverses informations sur les films, telles que leur budget, leur genre, ou encore leur popularité. L'approche adoptée comprend une analyse mathématique approfondie, une préparation rigoureuse des données et l'utilisation de modèles de machine learning avancés.

Présentation du Dataset



Lien du Dataset

Le lien vers le dataset utilisé est disponible ici :
https://www.kaggle.com/datasets/anandshaw2001/movie-rating-dataset/data

Lien du GitHub

Le projet complet peut être consulté sur GitHub :
https://github.com/MouhcineMM/Data-ML-DL/tree/mains

Intérêt pour le Dataset

Nous avons choisi pour ce projet un dataset portant sur l'appréciation publique des films. Notre objectif est de prédire les notes d'appréciation des films en fonction de certaines caractéristiques, afin d'anticiper quels films sont les plus susceptibles de rencontrer un succès auprès du public.

Informations sur le Dataset

Le dataset contient **19 dimensions (colonnes)**, décrites dans le tableau ci-dessous :

Dimensions finales après préparation des données

Suite à la préparation du dataset pour la mise en place des modèles de prédiction, les dimensions les plus pertinentes sont les suivantes :

```
['budget', 'genres', 'keywords', 'original_language', 'original_title', 'overview', 'p
'production_companies', 'production_countries', 'release_date', 'revenue', 'runtime',
'spoken_languages', 'title', 'vote_average', 'vote_count', 'year', 'month']
```

Détails supplémentaires

- **Date de parution du dataset :** 09/03/2024.
- **Nombre de lignes :** 4804 (dont 2 supprimées pour manque de contenu).

Partie Machine Learning : Random Forest Regressor

Présentation du modèle Random Forest Regressor

Proposé par Leo Breiman en 2001, le Random Forest est un algorithme d'apprentissage automatique basé sur l'assemblage de multiples arbres de décision pour effectuer des prédictions. Ce modèle repose sur une technique appelée *bagging* (Bootstrap Aggregating), qui combine plusieurs modèles faibles pour obtenir un modèle robuste et performant.

Autrement dit, un arbre de décision n'aura que très peu de résultats positifs car trop rigide. Cependant, la combinaison de plusieurs arbres apporte de la diversité de résultats (une certaine nuance), et la moyenne de ces résultats apporte souvent de bonnes performances.

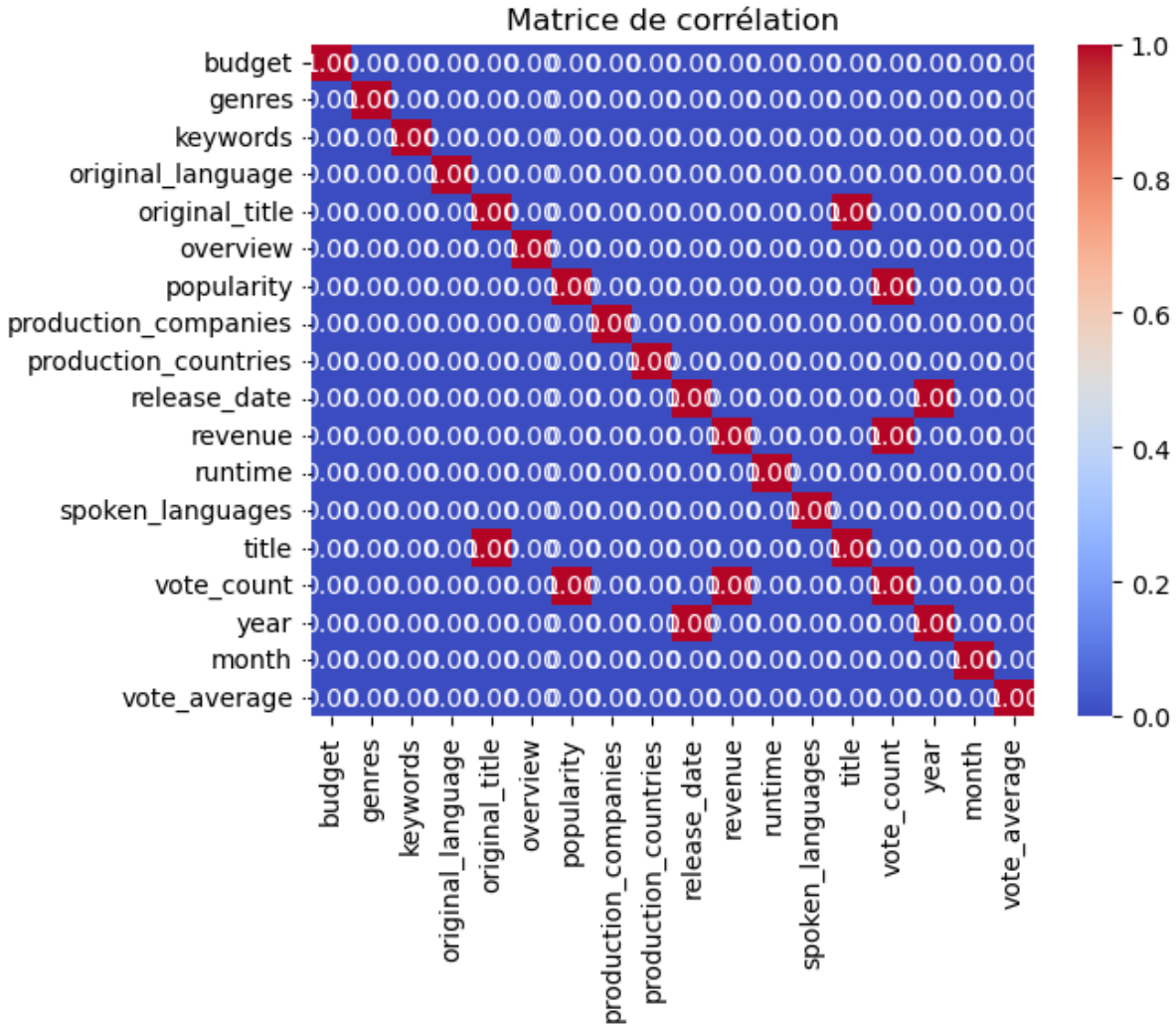
#	Colonne	Type de Donnée	Description
0	budget	int64	Budget du film en dollars.
1	genres	object	Liste des genres associés au film sous forme d'objets JSON.
2	homepage	object	URL de la page web officielle du film, si disponible.
3	keywords	object	Liste de mots-clés associés au film sous forme d'objets JSON.
4	original_language	object	Langue d'origine du film au format ISO 639-1.
5	original_title	object	Titre original du film.
6	overview	object	Résumé ou synopsis du film.
7	popularity	float64	Score de popularité du film.
8	production_companies	object	Liste des sociétés de production associées sous forme d'objets JSON.
9	production_countries	object	Liste des pays de production au format JSON contenant les codes et noms ISO.
10	release_date	object	Date de sortie du film au format YYYY-MM-DD.
11	revenue	int64	Revenus générés par le film au box-office mondial en dollars.
12	runtime	float64	Durée du film en minutes.
13	spoken_languages	object	Liste des langues parlées sous forme d'objets JSON contenant les codes ISO et les noms.
14	status	object	Statut du film (par exemple, "Released" ou "Post Production").
15	tagline	object	Phrase d'accroche utilisée pour promouvoir le film.
16	title	object	Titre officiel du film.
17	vote_average	float64	Note moyenne donnée par les utilisateurs.
18	vote_count	int64	Nombre de votes recueillis par le film.

Table 1: Description des dimensions du dataset brut.

Pourquoi ce modèle ?

Pour rappel, l'objectif de ce projet est de prédire la note d'un film en fonction de plusieurs variables. Cela implique l'utilisation d'un modèle de machine learning supervisé capable de réaliser une tâche de régression. Nous avons initialement envisagé un modèle de régression linéaire, qui est adapté lorsqu'il existe une relation linéaire forte entre la variable cible (la note) et les caractéristiques des données. Cependant, notre analyse préliminaire a révélé une absence de corrélation linéaire marquée entre ces variables, comme le montre la matrice de corrélation. Cela a exclu la régression linéaire comme option optimale.

Nous avons donc opté pour le modèle *Random Forest Regressor*, qui est mieux adapté pour capturer des relations complexes et non linéaires entre les variables. Ce modèle est également robuste face aux variances des données et peut gérer efficacement des interactions entre plusieurs caractéristiques, même lorsque celles-ci ne suivent pas une structure strictement linéaire.



Nous avons hésité entre un modèle de régression linéaire et un modèle de Random Forest Regressor, que nous avons finalement choisi. Le modèle de régression linéaire est pertinent lorsqu'il existe une relation linéaire forte entre l'output que nous souhaitons estimer et les caractéristiques des données. Cependant, après une étude de corrélation entre les données, nous avons remarqué qu'il n'y avait pas de corrélation apparente entre elles et l'output.

Nous nous sommes donc tournés vers un modèle de Random Forest Regressor, qui est plus pertinent dans le cas où les interactions entre les variables ne sont pas purement linéaires. De plus, ce modèle est connu pour être robuste face aux variances des données grâce à son approche globale qui exploite la diversité des systèmes de résolution.

Principe mathématique

1. Construction des arbres

La construction des arbres de décision dans un Random Forest repose sur deux mécanismes de tirage aléatoire : *Tree Bagging* et *Feature Sampling*.

1.1.a Tree Bagging (Bootstrap Aggregating): Le *Tree Bagging* consiste à générer des échantillons aléatoires des données d'entraînement pour construire chaque arbre de la forêt. Pour un arbre t, on crée un sous-ensemble D_t de taille N en tirant des

échantillons avec remplacement (*bootstrap*). Cela signifie que certains échantillons peuvent être sélectionnés plusieurs fois, tandis que d'autres peuvent ne pas être sélectionnés.

$$D_t = \text{Bootstrap}(D, N)$$

où D est l'ensemble des données d'entraînement, N est la taille du sous-ensemble, et D_t est un sous-ensemble échantillonné de D.

1.1.b Feature Sampling (Sélection de caractéristiques): Le *Feature Sampling* consiste à sélectionner un sous-ensemble aléatoire de caractéristiques (ou variables) pour chaque arbre. Cette sélection permet de réduire la corrélation entre les arbres, rendant la forêt plus robuste.

- Pour un *Random Forest Regressor* : On utilise environ un tiers des variables disponibles.
- Pour un *Random Forest Classifier* : On utilise la racine carrée du nombre total de variables.

1.2 Critère de division

Lors de la construction d'un arbre, chaque nœud est divisé en fonction d'une caractéristique (x) et d'un seuil (S) choisis pour minimiser la variance des valeurs cibles dans les sous-ensembles créés. Le critère de division repose ainsi sur la réduction de la variance intra-groupe.
Pour chaque nœud, on cherche donc à minimiser la variance définie par :

$$\text{Variance}(S) = \frac{1}{|S|} \sum_{i=1}^{n} (y_i - \bar{y})^2$$

où S est l'ensemble des échantillons associés au seuil S :

$$S = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots\}$$

$\bar{y}$ est la moyenne des y_i pour lesquelles leur x répond au seuil. Cette opération est effectuée séparément pour les échantillons du côté gauche du nœud (par exemple, $x \leq S$) et pour ceux du côté droit (par exemple, $x > S$).

2. Prédiction d'un arbre individuel

Une fois l'arbre construit, la prédiction pour une nouvelle observation x suit un parcours dans l'arbre :

À chaque nœud, une règle de décision est appliquée (par exemple, $x \leq S$), et l'observation est dirigée vers la branche gauche ou droite selon le cas. Ce processus se poursuit jusqu'à atteindre une feuille terminale. La prédiction de l'arbre est alors la moyenne des valeurs cibles y des données d'entraînement présentes dans cette feuille terminale :

$$\hat{y}_t(x) = \frac{1}{|D_t|} \sum_{i \in D_t} y_i$$

où D_t est l'ensemble des échantillons atteignant la feuille terminale pour l'arbre t.

3. Agrégation des prédictions

Pour obtenir la prédiction finale de la forêt, on effectue une moyenne des prédictions de tous les arbres :

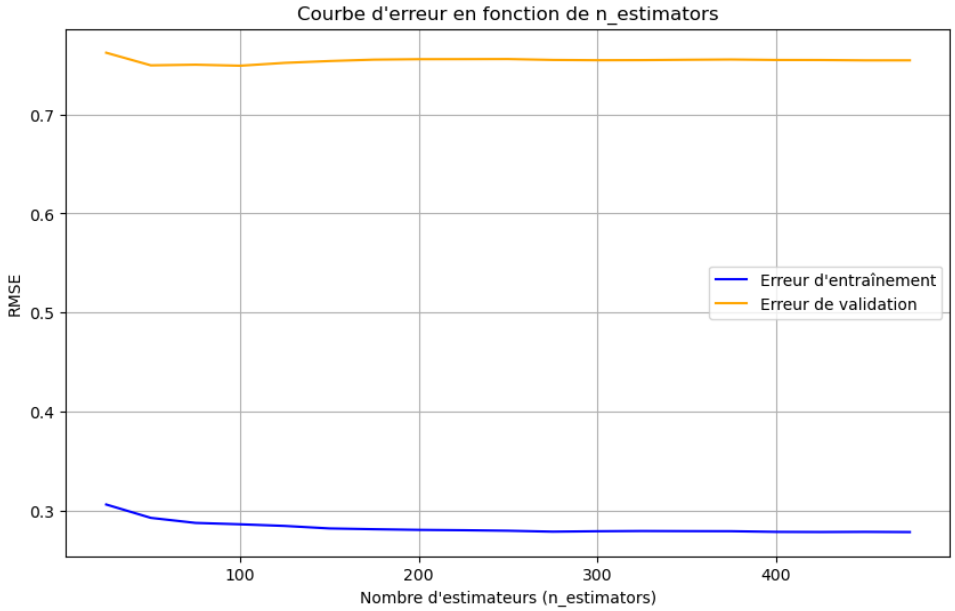
$$\hat{y}(x) = \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t(x)$$

où T est le nombre total d'arbres dans la forêt.

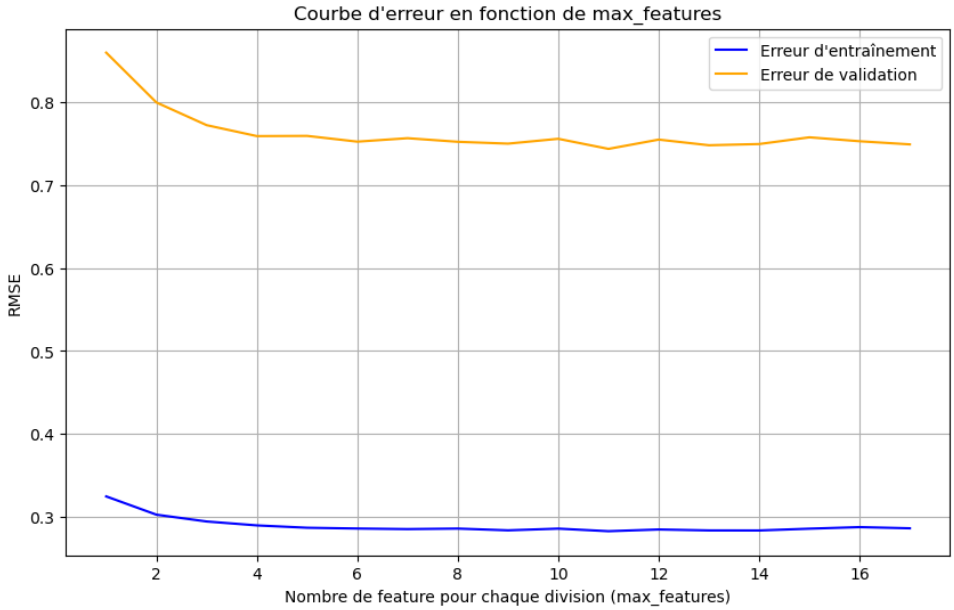
Justification dans l'optimisation des paramètres

Nous avons jugé que les hyperparamètres clés pour ce modèle étaient :

- Le nombre d'arbres utilisés pour la forêt (T).
- Le nombre maximal de caractéristiques utilisées pour chaque division.



On remarque que plus le nombre d'arbre est grand plus le RMSE est bas, il est donc important de choisir un nombre important d'arbre pour la forêt. Cependant on remarque que plus d'arbre sont choisi moins l'ajout marginal d'arbre a une importance significative sur le modèle. Nous avons donc fait le choix pour ce paramètre d'utiliser 500 arbres pour notre random forest.



On remarque que la performance du modèle n'est pas proportionnelle au nombre maximum de features autorisées par division. Par conséquent, nous ferons le choix de limiter ce paramètre à une valeur optimale pour chaque division.

Examinons maintenant les impacts de ces changements de paramètre.

Nous avions les scores suivants pour les paramètres standard, soit 100 arbres et toutes les features par division :

- Mean Squared Error : 0.5614646628511968
- R^2 Score : 0.5851203799960016
- RMSE : 0.7493094573346828
- RMSE relatif : 7.493094573346829 %

En optimisant les paramètres suite à notre étude, nous obtenons les scores suivants :

- Mean Squared Error : 0.5567359616233087
- R^2 Score : 0.5886145300259894
- RMSE : 0.7461474127967668
- RMSE relatif : 7.461474127967668 %

La différence n'est donc pas très importante.

Prédictions pour les 4 lignes retirées

Ces exemples illustrent les performances du modèle en termes de prédiction. Les valeurs prédites sont globalement proches des valeurs réelles, montrant que le modèle capture efficacement les relations dans les données.

ID	Valeurs Réelles	Valeurs Prédites
19995	7.2	7.18
285	6.9	6.93
206647	6.3	6.74
49026	7.6	7.04

Table 2: valeurs réelles et les valeurs prédites.

Partie Deep Learning : Gradiant

1 Prétraitement des Données

1.1 Normalisation des Données Numériques

La normalisation des données numériques est réalisée à l'aide de `StandardScaler`, qui transforme les données pour qu'elles aient une moyenne de 0 et une variance de 1. Cela est important car les modèles de machine learning, notamment les réseaux de neurones, sont sensibles à l'échelle des données.

Sans normalisation, les caractéristiques avec des valeurs beaucoup plus grandes (par exemple le budget d'un film) domineraient le processus d'apprentissage, empêchant le modèle de capturer des informations pertinentes provenant de caractéristiques avec des échelles plus petites. Le `StandardScaler` suit la formule :

$$X' = \frac{X - \mu}{\sigma}$$

où μ est la moyenne et σ l'écart-type. Cela garantit que les données sont centrées autour de 0 avec une variance unitaire, facilitant l'entraînement du modèle.

1.2 Encodage des Variables Catégorielles

L'encodage `OneHot` est utilisé pour transformer des variables catégorielles (comme la langue et le pays de production) en vecteurs binaires. Par exemple, la langue originale d'un film peut être anglais, français, etc. Pour qu'un réseau de neurones comprenne ces informations, chaque langue est représentée par un vecteur binaire où une seule position est '1' et le reste est '0'.

Mathématiquement, cet encodage est équivalent à une projection des catégories dans un espace euclidien de grande dimension où chaque catégorie occupe une coordonnée orthogonale. Le `OneHotEncoder` est préféré ici par rapport à d'autres méthodes (comme l'encodage d'entiers) car il ne présuppose aucune relation d'ordre entre les catégories.

1.3 Prétraitement du Texte

Les colonnes textuelles (genres, mots-clés) sont tokenisées, c'est-à-dire que chaque mot est converti en un entier représentant sa position dans le vocabulaire appris. Ensuite, ces séquences d'entiers sont remplies (*padded*) pour assurer une longueur constante.

Le *tokenization* est un prétraitement essentiel car les réseaux de neurones exigent que les données d'entrée soient numériques. De plus, le *padding* permet d'assurer que les entrées du modèle aient une dimension constante, facilitant les calculs matriciels dans le réseau de neurones.

2 Construction et Entraînement du Modèle

2.1 Réseau de Neurones (Couches Denses)

Les couches denses (*Dense layers*) sont des couches entièrement connectées où chaque neurone d'une couche est connecté à chaque neurone de la couche précédente. Le modèle de réseau est composé de plusieurs couches de neurones pour capturer des relations complexes entre les caractéristiques d'entrée et la sortie (la note du film).

Mathématiquement, les connexions d'une couche dense se traduisent par des multiplications matricielles. Si X est l'entrée, W les poids et b les biais, la sortie Z d'une couche dense est donnée par :

$$Z = W \cdot X + b$$

Les fonctions d'activation ReLU (*Rectified Linear Unit*) sont utilisées pour introduire de la non-linéarité dans le modèle. ReLU est définie comme :

$$f(x) = \max(0, x)$$

Ce choix est fait car ReLU permet de surmonter le problème du *vanishing gradient* rencontré avec d'autres fonctions d'activation comme *sigmoid* ou *tanh*.

2.2 Dropout et Régularisation

Le *dropout* est une méthode de régularisation où un certain pourcentage de neurones est désactivé de manière aléatoire lors de chaque itération d'entraînement. Cela aide à éviter le sur-apprentissage en forçant le réseau à ne pas trop dépendre d'une seule fonctionnalité.

Mathématiquement, cela correspond à multiplier les sorties des neurones par une variable aléatoire de Bernoulli (avec probabilité p de désactiver un neurone) pendant l'entraînement.

2.3 Optimisation avec Adam

L'optimiseur *Adam* (*Adaptive Moment Estimation*) combine la descente de gradient stochastique avec des taux d'apprentissage adaptatifs. Contrairement à la méthode de descente de gradient classique, Adam ajuste les taux d'apprentissage individuellement pour chaque paramètre, en utilisant des moyennes mobiles des gradients et de leurs carrés.

Les équations de mise à jour d'Adam sont les suivantes :

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{m_t}{\sqrt{v_t} + \epsilon}$$

où g_t est le gradient, m_t et v_t sont les premières et secondes estimations des moments, α est le taux d'apprentissage, et β_1 et β_2 sont des hyperparamètres.

3 Évaluation et Optimisation

3.1 Fonction de Perte (Mean Squared Error)

La fonction de perte utilisée pour ce modèle est la *Mean Squared Error* (MSE), qui mesure l'écart quadratique moyen entre les valeurs réelles et les prédictions. Elle est particulièrement adaptée pour les tâches de régression où l'on souhaite minimiser les erreurs de prédiction.

Mathématiquement, la MSE est donnée par :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

où y_i est la vraie valeur, $\hat{y}_i$ est la valeur prédite, et n est le nombre de données.

3.2 Optimisation des Hyperparamètres

La recherche d'hyperparamètres consiste à tester différentes combinaisons d'epochs et de *batch size* afin d'optimiser la performance du modèle. L'Erreur Absolue Moyenne (*Mean Absolute Error*, MAE) est utilisée ici comme métrique d'évaluation :

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

En optimisant ces hyperparamètres, nous cherchons à minimiser la MAE tout en évitant le sur-apprentissage grâce à des techniques comme l'arrêt anticipé.

Résultats Obtenus

Les performances du modèle, après entraînement, ont été évaluées à l'aide de plusieurs métriques sur un ensemble de test. Les résultats obtenus sont détaillés ci-dessous :

- **Erreur Absolue Moyenne (MAE)** : 0.6663
  Cette métrique indique une erreur moyenne absolue de 0.6663 unités entre les prédictions et les valeurs réelles.
- **Erreur Quadratique Moyenne (MSE)** : 0.9956
  Cette valeur reflète l'erreur moyenne quadratique, pénalisant davantage les grandes erreurs par rapport au MAE.
- **Racine de l'Erreur Quadratique Moyenne (RMSE)** : 0.9978
  Cette métrique, exprimée dans les mêmes unités que la variable cible, montre que l'écart moyen entre les prédictions et les valeurs réelles est d'environ 0.9978 unités.
- **Coefficient de Détermination (R^2)** : 0.3576
  Cette valeur indique que le modèle explique environ 35,76% de la variance des données.

En résumé, ces résultats mettent en évidence la capacité du modèle à capturer les principales tendances présentes dans les données.

Prédictions pour les 4 lignes retirées

Ces exemples illustrent les performances du modèle en termes de prédiction. Les valeurs prédites sont globalement proches des valeurs réelles, montrant que le modèle capture efficacement les relations dans les données. Cependant, certaines prédictions, comme pour l'ID 3371, présentent un écart plus important.

ID	Valeurs Réelles	Valeurs Prédites
596	5.2	5.39
3371	3.8	5.47
3049	5.3	5.71
45	5.5	5.52

Table 3: valeurs réelles et les valeurs prédites.

Conclusion

Ce rapport a permis d'explorer et de comparer deux approches complémentaires pour la prédiction de notes de films : le machine learning avec un modèle de type Random Forest Regressor et le deep learning via un réseau de neurones denses. Ces deux techniques ont montré leurs forces et leurs limites, qui offrent des pistes d'amélioration pour les travaux futurs.

Le Random Forest Regressor, en tant que modèle d'apprentissage supervisé classique, s'est avéré efficace pour capturer des relations complexes et non linéaires entre les variables, tout en restant robuste aux données bruyantes ou incomplètes. Cependant, il est limité par sa dépendance à des hyperparamètres spécifiques et peut montrer des performances stagnantes lorsqu'il est confronté à des structures de données plus abstraites ou fortement dimensionnelles, comme le traitement du texte ou des données non structurées.

De son côté, le modèle de deep learning s'est distingué par sa capacité à modéliser des relations hautement complexes et à intégrer diverses sources d'informations, comme les variables catégorielles encodées et les textes tokenisés. Néanmoins, cette flexibilité se paie par une plus grande sensibilité à la qualité et à la quantité des données, ainsi qu'à un coût computationnel plus élevé. De plus, malgré sa puissance, le modèle a montré des performances relativement modestes sur ce dataset, en partie dues à une variance difficile à maîtriser et à la nécessité d'un ajustement plus fin des hyperparamètres.

Les résultats obtenus suggèrent que chacune de ces approches pourrait bénéficier des forces de l'autre. Par exemple :

Le Random Forest Regressor pourrait être utilisé comme outil d'analyse exploratoire ou comme base pour des méthodes hybrides, intégrant ses prédictions comme features dans un réseau de neurones. Le deep learning, avec des ajustements supplémentaires, pourrait être enrichi par des mécanismes tels que des réseaux récurrents (RNN) ou des transformeurs pour capturer des relations temporelles ou contextuelles plus riches, notamment dans les colonnes textuelles. Perspectives d'amélioration :

Enrichissement des données : Introduire des données supplémentaires (comme les avis des spectateurs ou les tendances des réseaux sociaux) pourrait améliorer la qualité des prédictions. Approches hybrides : Combiner des techniques classiques et avancées, comme utiliser le Random Forest pour une sélection initiale des caractéristiques ou intégrer ses prédictions dans un pipeline de deep learning. Optimisation des hyperparamètres

: Mettre en place une recherche approfondie par optimisation bayésienne ou algorithmes évolutionnaires pour maximiser les performances. Gestion de la variance : Pour le deep learning, explorer des techniques comme les ensembles de modèles ou des méthodes de régularisation plus avancées. En conclusion, ce projet met en lumière les forces et les limites des approches de machine learning et de deep learning, tout en ouvrant la voie à des travaux futurs visant à tirer parti des complémentarités entre ces deux paradigmes pour des prédictions encore plus performantes et robustes.