

# Projet MRR - Méthodologie du projet

LE BOUEDEC Mathias

MALFOY Lilou

## 1 Introduction

Le projet vise à prédire les évaluations et la popularité d'un nouvel animé, ainsi qu'à identifier les utilisateurs susceptibles de l'apprécier afin de recommander efficacement de nouveaux contenus.

Pour cela, nous utiliserons des méthodes de régression régularisée qui permettent de faire des prédictions fiables tout en évitant le surapprentissage.

Nous avons trié un dataset qui contient 13 fichiers CSV couvrant différents aspects de l'écosystème animé, collectés depuis MyAnimeList et l'API Jikan (1917–2025).

Le dataset est disponible sur Kaggle à l'adresse suivante : <https://www.kaggle.com/datasets/neelagiriaditya/anime-dataset-jan-1917-to-oct-2025>.

Nous disposons désormais de deux datasets : le premier visant à prédire les variables `y_score` (note moyenne sur 10 d'un animé) et `y_members` (nombre d'utilisateurs ayant un animé dans sa liste, meilleur indicateur de popularité d'un animé), le second à prédire `user_score`.

## 2 Prédire l'évaluation et la popularité d'un animé

### 2.1 Dataset trié sans NaN, pour les animés publiés depuis 2010

#	Colonne	Non-Null	Type	Uniques
0	mal_id	4645	int64	4645
1	type	4645	object	9
2	genres	4645	object	496
3	studios	4645	object	738
4	themes	4645	object	630
5	demographics	4645	object	7
6	source	4645	object	17
7	rating	4645	object	6
8	episodes	4645	float64	N/A
9	season	4645	object	4
10	main_characters	4645	int64	N/A
11	supporting_characters	4645	int64	N/A
12	recommendation_mal_id	4645	object	N/A
13	staff	4645	object	N/A
14	y_score	4645	float64	N/A
15	y_members	4645	int64	N/A

#### Colonnes dérivées :

- `main_characters` : nombre de personnages principaux, issu de `characters_anime_works.csv`.
- `supporting_characters` : nombre de personnages secondaires, issu du même fichier.
- `recommendation_mal_id` : liste d'animés recommandés via `recommendations.csv`.
- `staff` : liste des IDs du personnel (réalisateurs, scénaristes, doubleurs), obtenu via `person_anime_works.csv`.

#### Encodage des variables catégorielles :

- Colonnes simples (`type`, `source`, `rating`, `season`, `demographics`) : transformées par *one-hot encoding*.
- Colonnes multi-label (`genres`, `studios`, `themes`) : converties en listes puis encodées par *multi-hot encoding*.

TABLE 1 – Dataset final avant encodage utilisé pour la prédiction.

### 2.2 Traitement des listes `recommendation_mal_id` et `staff`

Pour chaque animé, les recommandations (`recommendation_mal_id`) sont d'abord converties en listes d'animés référencés. Pour chaque liste, on calcule la moyenne, le maximum et le minimum des variables à prédire scores et members et des statistiques des watchers : `watching`, `completed`, `on_hold`, `dropped`, `plan_to_watch` et `total`, puis ces valeurs sont agrégées pour produire une seule ligne par anime.

Pour les staffeurs (`staff`), chaque staffeur est associé à tous ses autres projets : on calcule d'abord la moyenne des variables à prédire par staffeur ainsi que le nombre d'animés sur lequel il a travaillé, puis lors du merge sur le dataset on exclut l'animé courant pour corriger le biais (on soustrait score et members divisés par le nombre d'animés du staffeur pour chaque staffeur). Ensuite, pour chaque anime, on prend la moyenne de ces moyennes corrigées pour obtenir les statistiques finales des staffeurs.

### 2.3 Data processing

Le dataset final est chargé et les colonnes de type objet, comme `staff`, ainsi que la clé primaire `mal_id` sont supprimées pour ne conserver que des variables numériques. Les features et les cibles (`y_score`, `y_members`) sont séparées. Les données sont divisées en ensembles d'entraînement et de test (80/20) et les features numériques sont standardisées avec `StandardScaler`. Pour l'évaluation des modèles, une K-Fold Cross-Validation est appliquée sur l'ensemble d'entraînement afin d'estimer la performance de manière robuste avant le test final.

### 2.4 Modèles de régressions linéaires régularisés

Nous avons évalué quatre modèles linéaires : **LinearRegression**, **Ridge**, **Lasso** et **ElasticNet**. Les modèles ont été entraînés séparément pour `y_score` et `y_members`, afin de tenir compte de la nature et de l'échelle différentes des deux cibles. Pour `y_members`, une transformation logarithmique a été nécessaire afin de réduire l'impact des valeurs extrêmes pour les régressions Lasso et ElasticNet. Le modèle **LinearRegression** sert de référence, puisqu'il ne comporte aucune régularisation. Le modèle **Ridge** ( $L_2$ ) permet de réduire la variance des coefficients tout en conservant l'ensemble des variables, ce qui est pertinent lorsque chaque feature paraît informative. Le modèle **Lasso** ( $L_1$ ) introduit une sélection implicite de variables, certains coefficients pouvant être contraints à zéro ; bien que nos variables soient majoritairement utiles, il a été inclus pour vérifier l'absence de redondance. Enfin, **ElasticNet**, qui combine les pénalités  $L_1$  et  $L_2$ , permet de bénéficier simultanément des avantages de la régularisation et d'une éventuelle sélection de variables, notamment en présence de features corrélées.

Cette combinaison de modèles permet de comparer différentes stratégies : conserver toutes les variables (**Ridge**), réduire la complexité via sélection (**Lasso/ElasticNet**) et mesurer l'impact sur la performance avec une validation robuste par **K-Fold Cross-Validation**.

### 2.5 Évaluation des modèles

Les modèles sont évalués à l'aide de métriques classiques de régression : MAE, RMSE et  $R^2$ .

Une K-Fold Cross-Validation a été appliquée sur l'ensemble d'entraînement afin d'obtenir des métriques fiables avant le test final.

Des visualisations sont produites pour comparer les valeurs réelles et prédites, incluant un scatter plot avec la diagonale idéale et un histogramme des erreurs pour analyser la distribution des résidus et les comparer à une gaussienne. Ces graphiques permettent d'identifier visuellement les biais ou les écarts importants dans les prédictions.

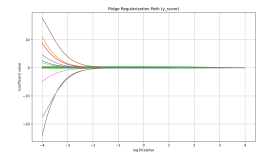
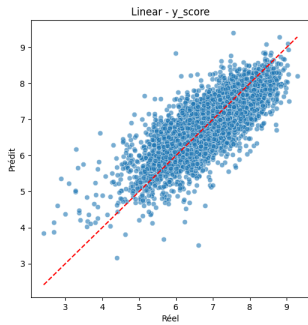
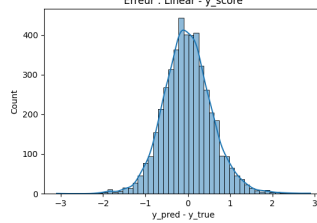


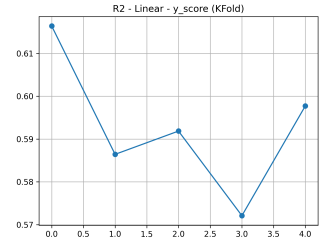
FIGURE 1 – Ridge – coefficients vs  $\alpha$



(a) Régression



(b) Distribution des erreurs



(c) Évolution du  $R^2$  (K-fold)

FIGURE 2 – Résultats du modèle **LinearRegression** pour la prédiction de  $y\_score$ . De gauche à droite : nuage de points des valeurs prédites, histogramme des erreurs, et évolution du coefficient de détermination  $R^2$  au fil des folds.

## 3 Prédire la note d'un utilisateur pour un nouvel animé

### 3.1 Profilage des utilisateurs

Chaque utilisateur est représenté à partir des notes qu'il a attribuées aux animés, utilisées pour construire un vecteur de préférences décrivant son comportement de notation par genre. Afin de garantir la qualité des profils, seuls les utilisateurs dont l'âge était renseigné (non manquant) ont été conservés. De plus, nous avons retenu les utilisateurs figurant parmi ceux ayant complété le plus d'animés dans le dataset Kaggle, de manière à disposer de suffisamment d'historique pour estimer leurs préférences et à éviter les profils trop peu renseignés ou dominés par des notes identiques.

### 3.2 Clustering

Chaque utilisateur est représenté à partir de ses notes attribuées aux animés. Pour chaque genre (par exemple *Action*, *Comedy*, *Romance*, etc.), nous calculons la moyenne des notes données par l'utilisateur, ce qui permet d'obtenir un vecteur de préférences reflétant son comportement de notation. Afin d'assurer une échelle comparable entre utilisateurs, ces profils sont ensuite standardisés grâce à une normalisation (*StandardScaler*). Sur ces vecteurs normalisés, nous appliquons l'algorithme *K-means* avec  $k=5$  clusters. Chaque utilisateur se voit ainsi assigner un identifiant de cluster correspondant au groupe auquel son profil est le plus proche. Ce regroupement permet de distinguer différents types de comportements de notation (par exemple, utilisateurs appréciant fortement les genres d'action ou ceux favorisant les œuvres dramatiques) et constitue la base pour l'entraînement de modèles spécialisés adaptés aux préférences de chaque groupe.

### 3.3 Préparation des caractéristiques

Pour la prédiction des notes, seules les caractéristiques associées aux animés sont utilisées, de manière à éliminer toute fuite d'information liée à l'historique des utilisateurs. Les features retenues comprennent notamment le score global de l'animé, le nombre de membres, l'année de sortie et le nombre d'épisodes. Des transformations (*logarithme*) et interactions ( $members \times episodes$ ,  $score \times members$ ) sont également intégrées afin de mieux modéliser les relations non linéaires et l'échelle très variable de certaines variables.

### 3.4 Modèles par cluster

Pour chaque cluster d'utilisateurs, nous commençons par entraîner des modèles de régression pénalisés (Ridge et Lasso) afin de prédire le  $user\_score$ . Ces méthodes permettent de contrôler la complexité du modèle et de réduire le surapprentissage, tout en identifiant l'importance relative des différentes features. Les performances de ces modèles, mesurées notamment par le coefficient de détermination  $R^2$ , sont reportées pour chaque cluster.

Après analyse et recherches complémentaires, nous avons exploré des modèles plus flexibles, en particulier les arbres de décision et **Random Forest**, afin de mieux capturer les relations non linéaires entre les caractéristiques des animés et les préférences des utilisateurs. Chaque cluster dispose ainsi d'un modèle dédié, permettant d'adapter les prédictions aux comportements spécifiques des groupes identifiés.

### 3.5 Évaluation

Chaque modèle est évalué avec le **MAE** et le  **$R^2$** , ce qui permet d'identifier la qualité de la prédiction dans chaque cluster.

### 3.6 Prédiction $user\_score$

Pour prédire la note qu'un utilisateur attribuera à un animé, on commence par identifier son cluster, reflétant ses préférences par genre. On applique ensuite le modèle de régression dédié. La sortie est une prédiction continue du  $user\_score$ , enregistrée pour analyses et pouvant être transformée en recommandation binaire. Cette approche adapte les prédictions aux comportements typiques de chaque cluster tout en restant centrée sur les animés.

### 3.7 Recommandations

À partir des notes prédites par les modèles de régression, nous créons une variable binaire *recommendations*, qui prend la valeur 1 lorsque la note prédite dépasse un seuil défini (ici fixé à 7), et 0 sinon. Cette variable permet d'identifier rapidement les animés que l'utilisateur est susceptible d'apprécier.

Pour chaque utilisateur, les animés correspondant à une prédiction supérieure au seuil sont regroupés dans une liste personnalisée de recommandations. Afin de garantir la pertinence et la sécurité des recommandations, un filtrage supplémentaire est appliqué sur l'âge : seuls les animés pour lesquels l'âge de l'utilisateur est supérieur ou égal à l'âge minimum requis sont conservés.

Cette approche assure que chaque recommandation est à la fois adaptée aux préférences de l'utilisateur — telles que capturées par le modèle et par le cluster auquel il appartient — et conforme aux critères d'âge, ce qui améliore la qualité et la fiabilité du système de recommandation.

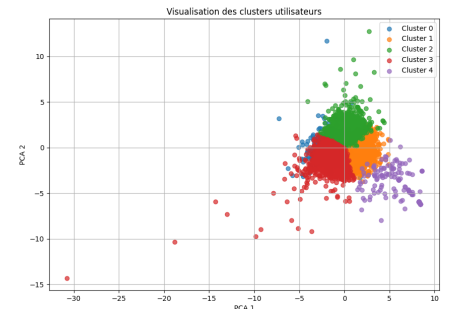


FIGURE 3 – Différents clusters d'utilisateurs

Cluster 0: Ridge	alpha=233.57, MAE=2.37, $R^2=0.06$
Cluster 0: Lasso	alpha=0.0026, MAE=2.37, $R^2=0.06$
Cluster 1: Ridge	alpha=233.57, MAE=1.87, $R^2=0.09$
Cluster 1: Lasso	alpha=0.0010, MAE=1.87, $R^2=0.09$
Cluster 2: Ridge	alpha=162.38, MAE=1.86, $R^2=0.08$
Cluster 2: Lasso	alpha=0.0016, MAE=1.86, $R^2=0.08$
Cluster 3: Ridge	alpha=112.88, MAE=1.83, $R^2=0.10$
Cluster 3: Lasso	alpha=0.0026, MAE=1.83, $R^2=0.10$
Cluster 4: Ridge	alpha=1000.00, MAE=3.62, $R^2=0.01$
Cluster 4: Lasso	alpha=0.0785, MAE=3.63, $R^2=0.01$

FIGURE 4 – Résultats Ridge et Lasso

username	anime_id	recommendation
0 Karniale	23311	0
1 Karniale	21	0
2 Karniale	41457	1
3 Karniale	48569	1
4 Karniale	32998	0
5 Karniale	35839	1
6 Karniale	33337	0
7 Karniale	34984	0
8 Karniale	31580	0
9 Karniale	33253	0

FIGURE 5 – Exemple de dataset montrant les recommandations pour l'utilisateur