Thomas Handschumacher   
Nehdim Osmani

RAPPORT DE PROJET :

Groupe 5

Prédiction des notes des films sur le site IMDB



Projet Fin d’étude 2020/2021

Encadrant : Ala Bayoudh

Introduction

IMDB est un site répertoriant une très grande quantité de films, des informations sur ceux-ci ainsi que des commentaires. Chaque film est noté entre 0 et 10 en fonction de certains critères, c’est l’équivalent de Allo Ciné à l’internationale. Les films sont notés par les utilisateurs et par les critiques de journalistes.

Notre projet consiste à créer un modèle basé sur les techniques de Machine Learning ayant pour but la prédiction du score des films sur le site internet IMDB. Pouvoir déterminer statistiquement la note que notre film pourra obtenir permettra aux réalisateurs de produire le meilleur film possible. L’outil permet également de faciliter le travail du site IMBD dans sa notation. Il est également utile pour les utilisateurs de connaître les meilleurs films du moment. La prédiction des scores pourra prendre en compte de nombreuses informations comme la date de réalisation, le réalisateur, le budget, etc.

Le base de données train et test sont déjà données. Nous allons dont utilisé la base train afin de trouver le meilleur modèle possible puis l’appliquer sur la base test.

*Comment obtenir un programme le plus performant possible qui permettra de prédire les scores des films ?*

# Architecture du projet

Nous allons commencer par analyser nos données, puis les traiter afin qu’on puisse mettre en place nos algorithmes de machine learning, puis mettre au point un réseaux de neurones. Nous allons ensuite comparer tous ces modèles et enfin appliquer le meilleur modèle aux données test.

Découverte, analyse des données manquantes …

Données Train

Traitement des données manquantes et textuels

Mise au point d’algorithmes de   
machine learning

Mise au point d’un  
réseau de neurones

Appliquer le meilleur modèle aux données Test

Comparaison et sélection du meilleur modèle

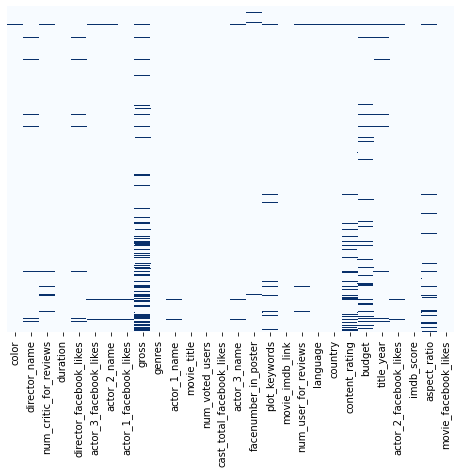
# Analyse des données

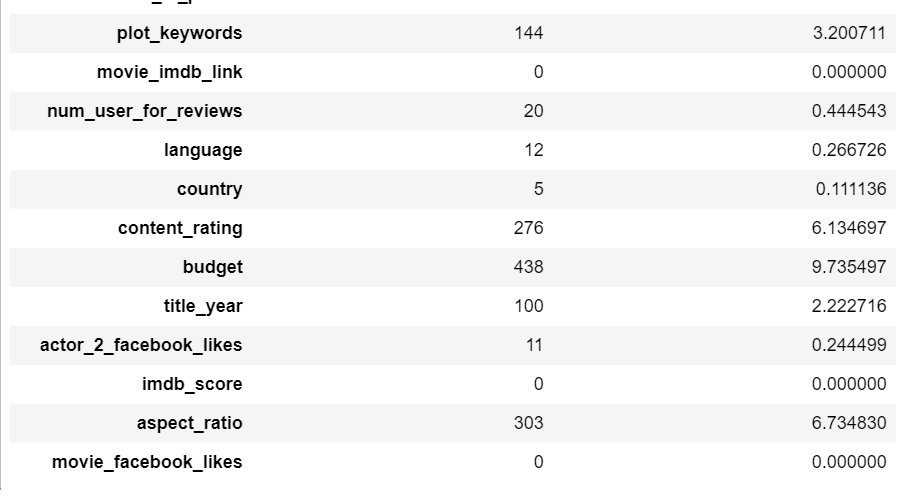
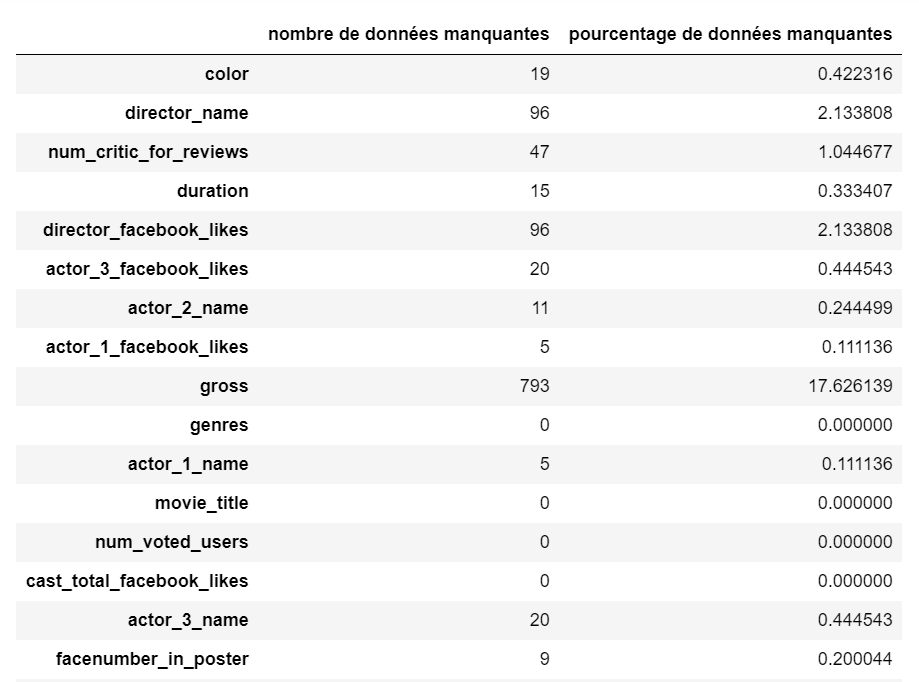
## Découverte

La première étape du projet consiste à découvrir et analyser les données que nous avons. Nous avons donc une base de données de 4500 films et 28 variables explicatives sur chaque film. Nous appliquerons ensuite le modèle que nous aurons choisit sur 500 films ayant les mêmes variables explicatives.

Les variables sont les suivantes :

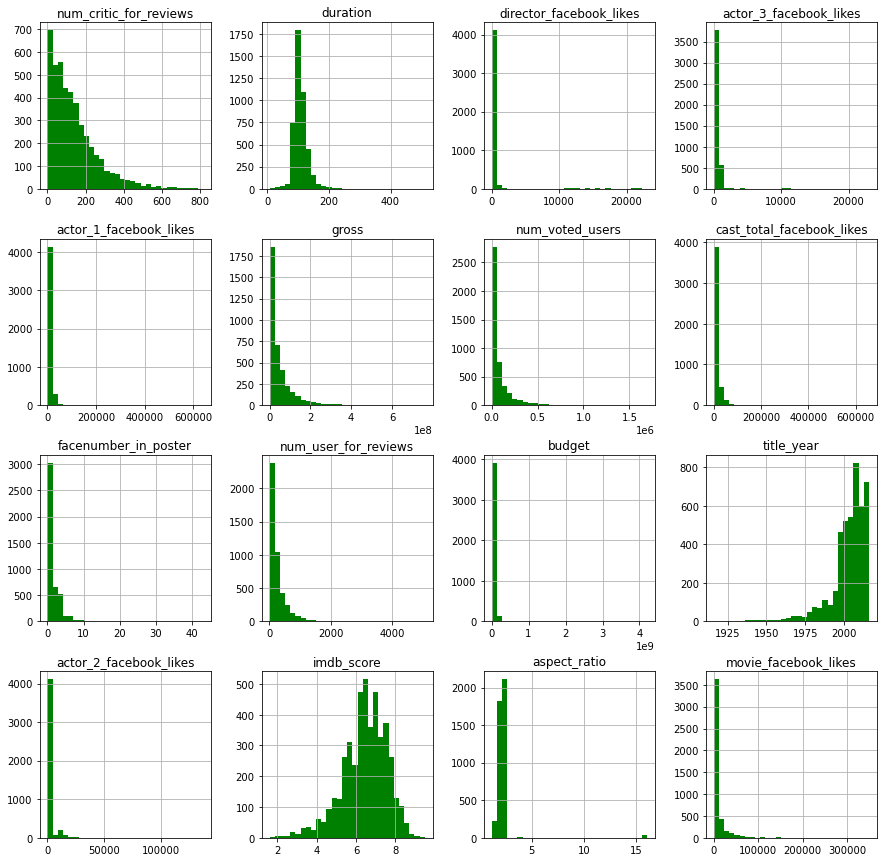
['id', 'color', 'director\_name', 'num\_critic\_for\_reviews', 'duration', 'director\_facebook\_likes', 'actor\_3\_facebook\_likes’,'actor\_2\_name', 'actor\_1\_facebook\_likes', 'gross', 'genres', 'actor\_1\_name', 'movie\_title', 'num\_voted\_users’, 'cast\_total\_facebook\_likes', 'actor\_3\_name', 'facenumber\_in\_poster', 'plot\_keywords', 'movie\_imdb\_link', 'num\_user\_for\_reviews’, 'language', 'country', 'content\_rating', 'budget', 'title\_year','actor\_2\_facebook\_likes', 'imdb\_score', 'aspect\_ratio', 'movie\_facebook\_likes']

Nous allons maintenant voir les quantités de données manquantes.



Il manque de nombreuses données de façon non homogène. Il va donc falloir trouver des solutions afin de traiter ces variables afin d’obtenir un modèle performant.

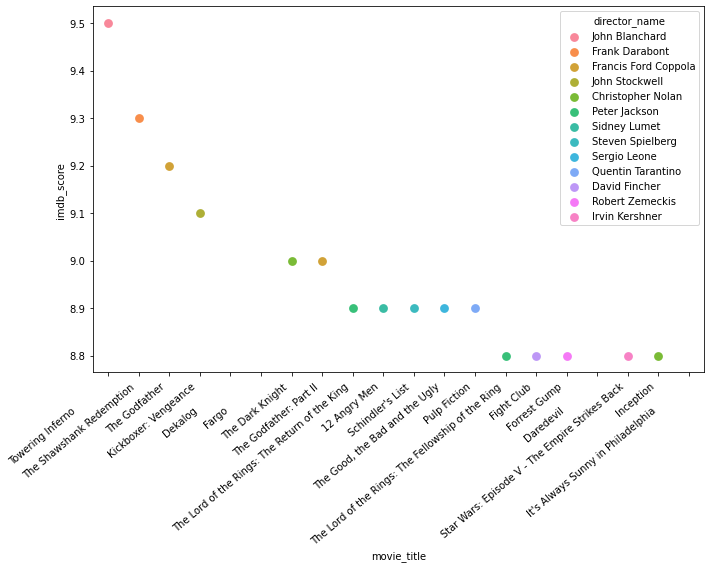
Les variables sont reparties de cet façon :



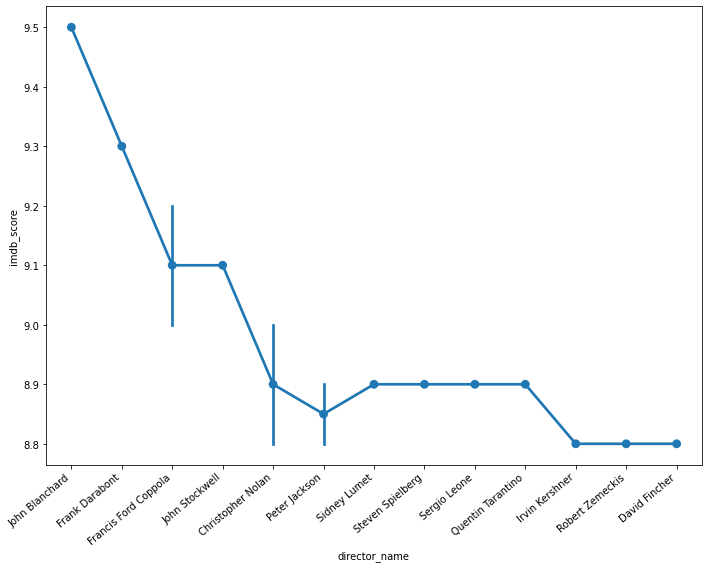
On remarque que les variables sont reparties plus ou moins homogènes et statistiques. Par exemple la variable cible, imdb\_score, suit une loi gaussienne. Les autres variables sont assez proches dans leurs répartitions.

## Visualisation

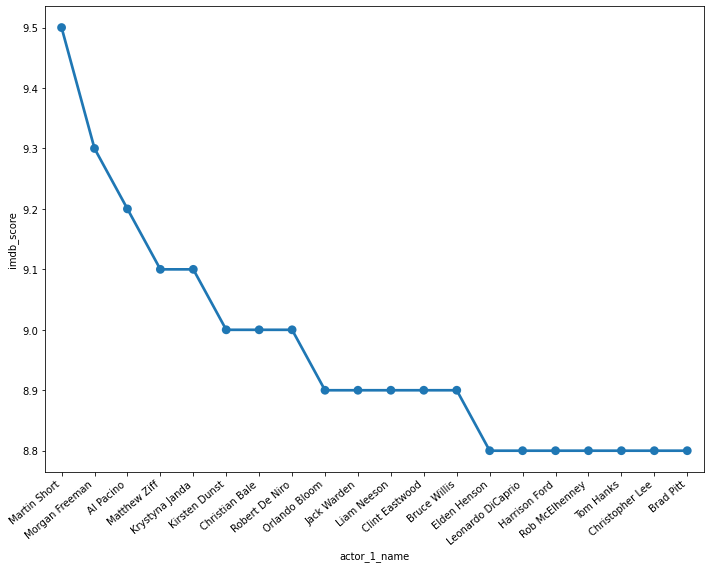
Les meilleurs films, avec leurs directeurs, sont :



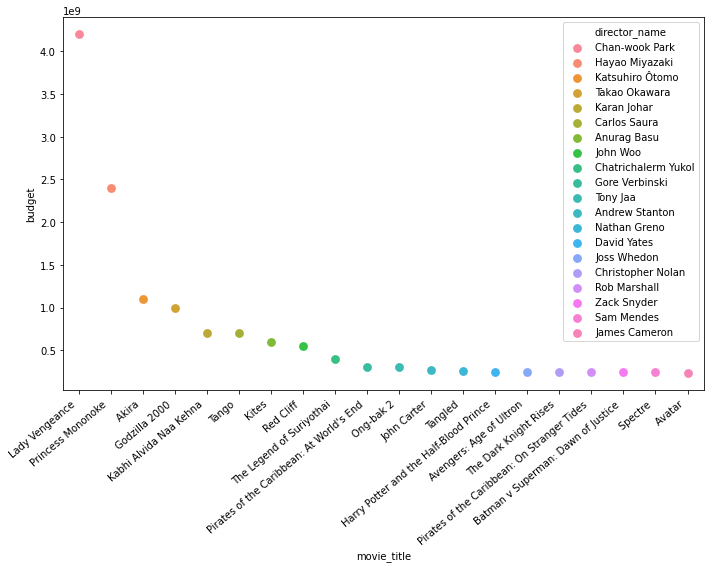
Les meilleurs directeurs avec la moyenne des notes de chacun de leurs films sont :



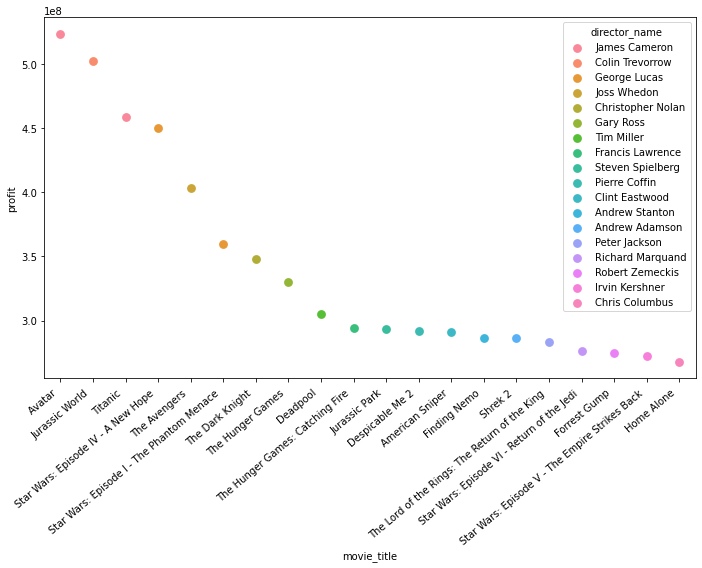
Les meilleurs acteurs, en faisant la moyenne de tous leurs films, sont :

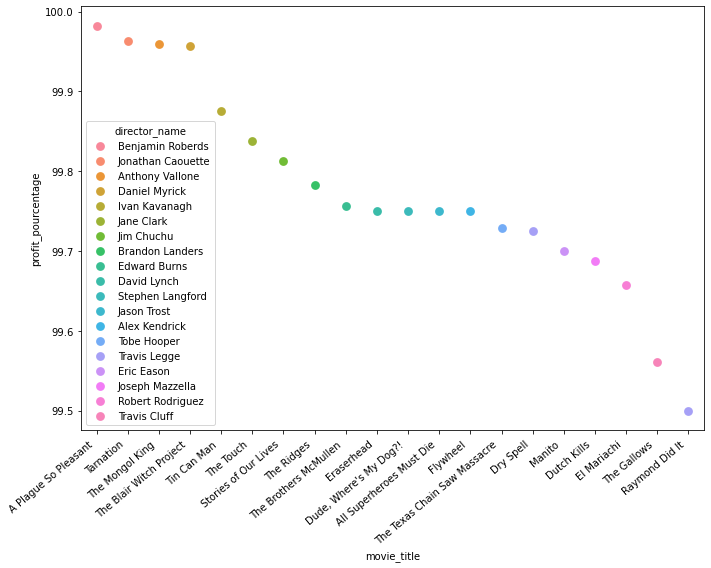


Les films ayant les plus gros budgets sont :



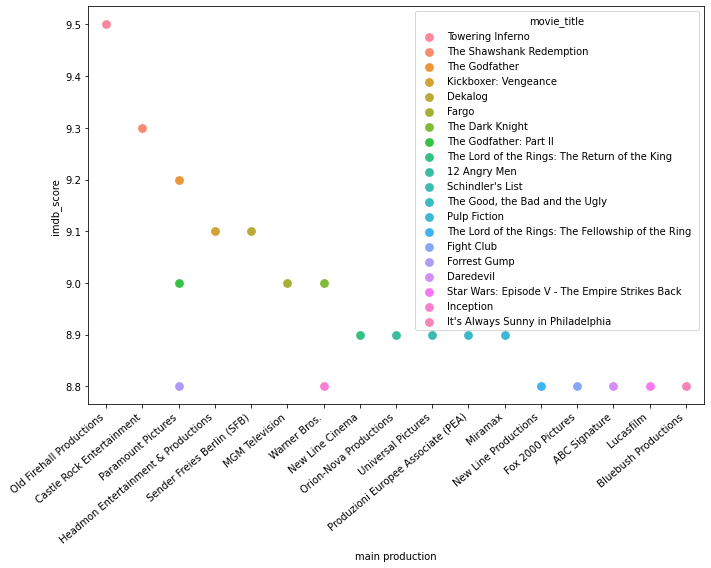
Nous avons ensuite créé deux nouvelles variables : profit et profit\_pourcentage correspondant aux profits nets et en pourcentage de chaque film.





Les films ayant les plus gros profits ne sont pas les films ayant les plus gros profits en termes de pourcentage.

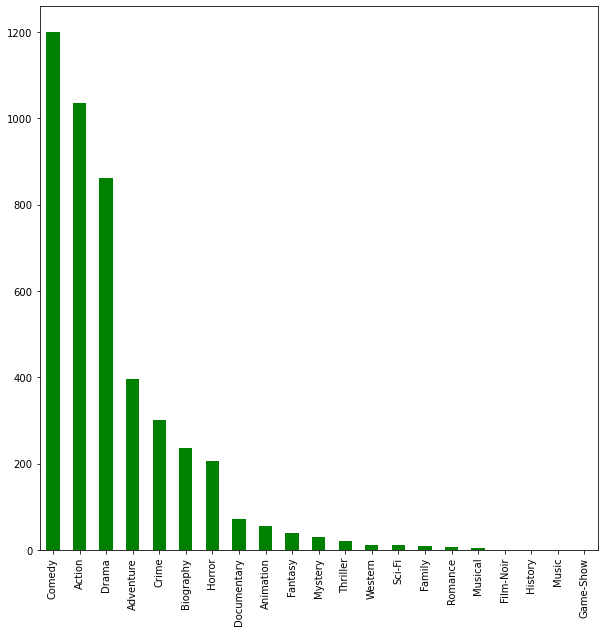
Nous avons ensuite scrapper sur le site imdb le metascore, les revenus mondiaux du film ainsi que la boite de production de chacun des films grâce à BeautifulSoup de façon très simple. En effet, le site est statique et facilement scrappable.



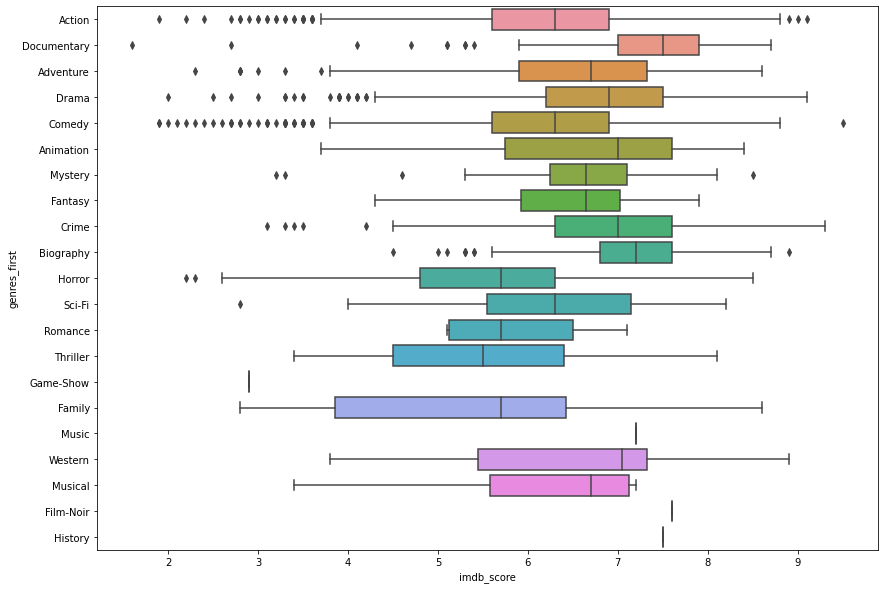
Certaines maisons de production semblent avoir des meilleures notes que d’autres, ce sont donc des variables explicatives.

Nous allons maintenant regarder les différentes répartitions des variables de façon plus précise.

* La première variable explicative que nous avons analysée est le genre des films. Chaque film possède un ou plusieurs genres.



On remarque que la plus grande majorité des films sont des comédies. Nous allons maintenant faire une analyse comparative entre les genres afin de voir si certains genres obtiennent de meilleures notes. Pour cela nous gardons seulement le premier genre de chaque film.



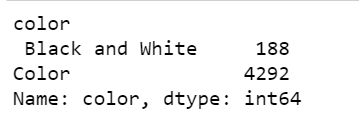
On peut facilement observer que les documentaires obtiennent de meilleures notes que les films familiaux.

# Traitement des données

L’objectif de notre projet est la prédiction de la note du film. Pour faire cela nous allons utiliser des algorithmes de machine learning. Ces algorithmes nécessitent des données numériques, des nombres, et aucune donnée textuelle. Nous avons donc besoin de transformé et traiter nos données. Nous allons également régler les données manquantes.

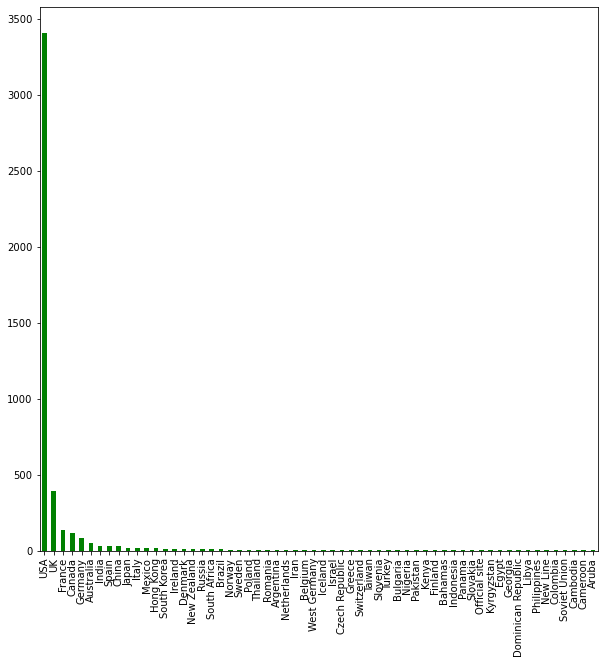
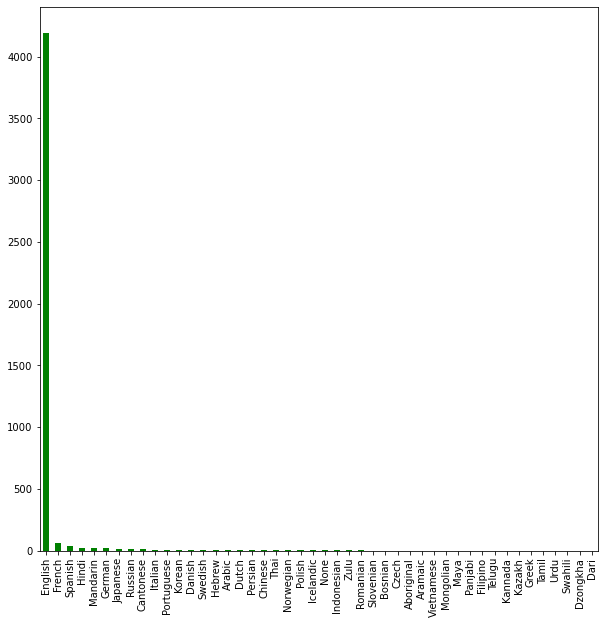
## Couleur des films

Les films sont repartis entre couleur et noirs et blanc.



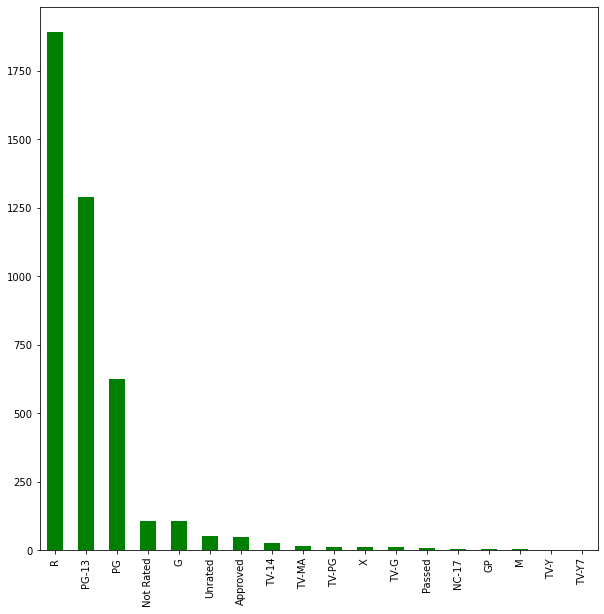
La plus grande majorité des films sont des films de couleur. Nous allons donc remplacer les données manquantes ainsi que les films en noir et blanc par 0 et les films de couleur par 1.

## Langue et pays



La très grande majorité des films sont américains ou anglais. De manière assez évidente, la langue des films est très majoritairement anglaise.

## Content rating (âge minimum)



On remarque facilement que les films ont en très grande majorité, 3 content rating principaux.

## La fonction most\_common

A partir de ces observations, nous avons choisit de créer une fonction permettant de traduire ces informations. La fonction compare chaque ligne aux mots les plus présents sur l’ensemble de la base de données. Par exemple nous appliquons la fonction à la variable ‘language’. 

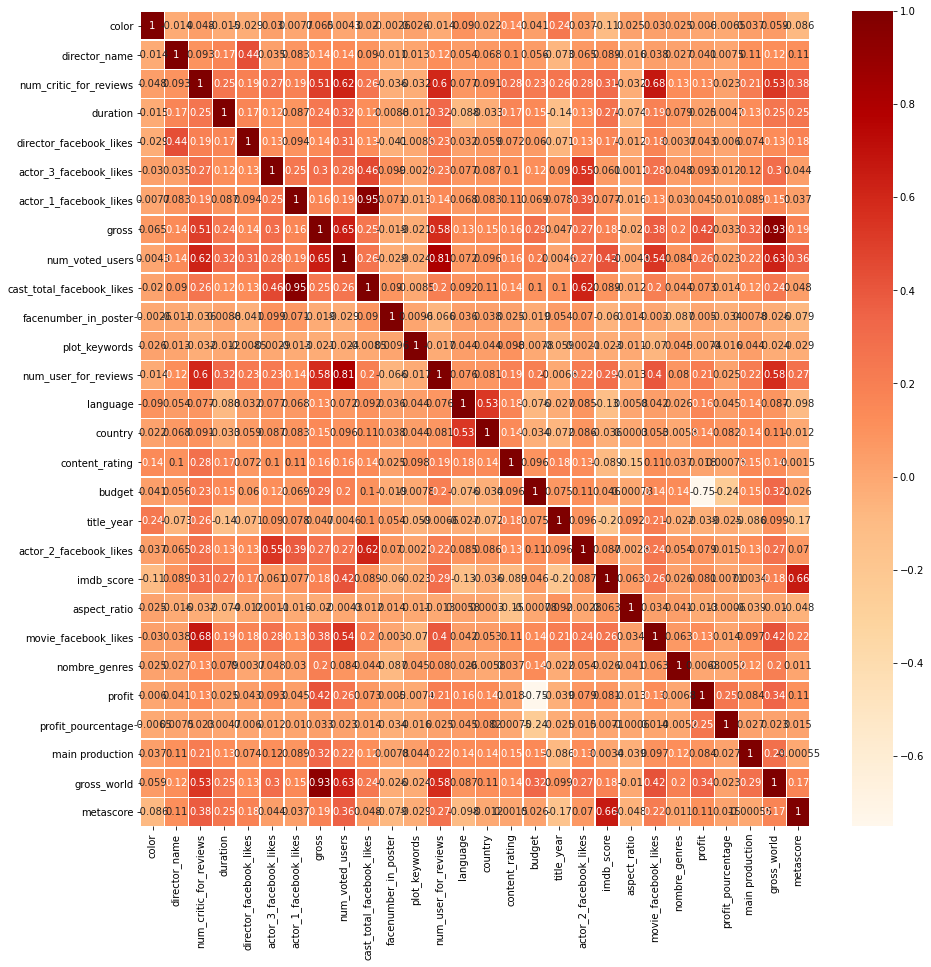
Nous choissons de comparer la taille de la données ‘language’ parmis les 2 plus présents. Ainsi, si le film est anglais ou francais, nous allons mettre 1 et 0 sinon. Nous appliquons cette fonction à ‘plot\_keyords’ qui sont les mots clés du résumé de chaque film, ‘content\_rating’, ‘country’ , ‘director\_name’ et ‘main production’.

## Autres données manquantes

Les autres données manquantes sont remplacées par les quartiles les plus optimaux sur chacune des autres données.



## La matrice de corrélation



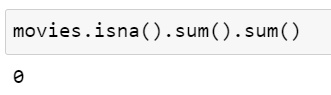
La matrice de corrélation permet d’observer les variables qui sont corrélés entre elles. Nous avons choisit de supprimer les variables ayant une corrélation supérieur à 0,6. Par exemple ‘num\_user\_for\_reviews’ et ‘num\_voted\_users’ sont très fortement corrélés, nous pouvons donc supprimer l’une des deux colonnes. De même pour le nombre de like sur les pages facebook des acteurs.

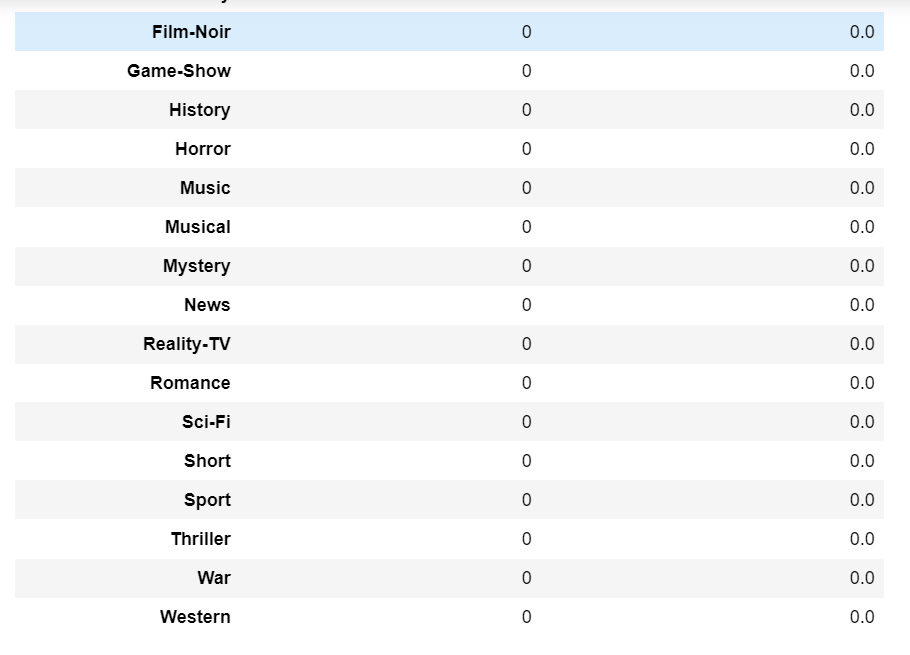
Nous pouvons également supprimer après vérification les variables peu ou pas explicatives.

## Les genres

Chaque film présente un ou plusieurs, nous avons donc décider de créer une colonne par genre et de mettre 1 si le film présente ce genre et 0 sinon. Cette étape est réalisée grâce à la fonction get\_dummies.

## Vérification





Il ne manque plus de données, toutes les données manquantes ont été traités. De plus, les données textuelles sont soit traité soit supprimés. Nous pouvons ensuite sauvegarder ces données dans un fichier csv afin de passer à la partie machine learning et prédiction du score.

# Algorithmes de machine learning

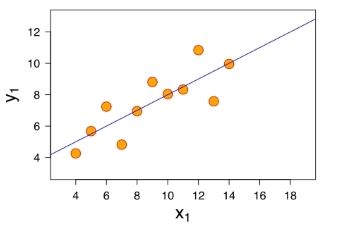
Nous avons à présent une base de données traité que nous pouvons utiliser pour nos algorithmes de prédiction des scores. Nous allons utiliser des algorithmes dis supervisé. En effet, nous avons déjà nos données avec des labels et nous allons nous en servir pour prédire de nouvelles données. Nous allons donc utiliser des algorithmes de Régression qui permettent de faire des prédictions.

Nous avons effectué plusieurs essais en testant différents algorithmes afin d’avoir une prédiction la plus précise possible. Nous avons comparé le score obtenu et ensuite choisit celui que nous allons utiliser. Nous avons dans un premier temps séparé nos données en données train et test (80% train et 20% test) afin de pouvoir calculer et vérifier la bonne prédiction de nos algorithmes.

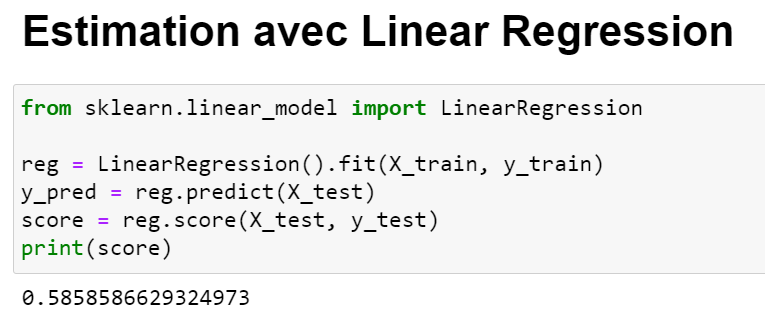
Nous allons a la fin comparer tous les scores et garder le meilleur.

## Linear Regression

Le premier algorithme de Machine Learning supervisé que nous avons testé est le Linear Regression. La régression linéaire est un algorithme qui va trouver une droite qui se rapproche le plus possible d’un ensemble de points. Les points représentent les données d’entraînement.

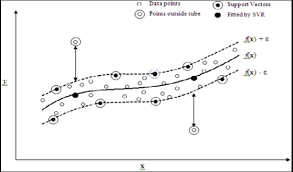


Nous avons donc lancer un algorithme de Linear Regression sur notre dataset ce qui nous a donner un score de 58,5% de bonne prédiction.

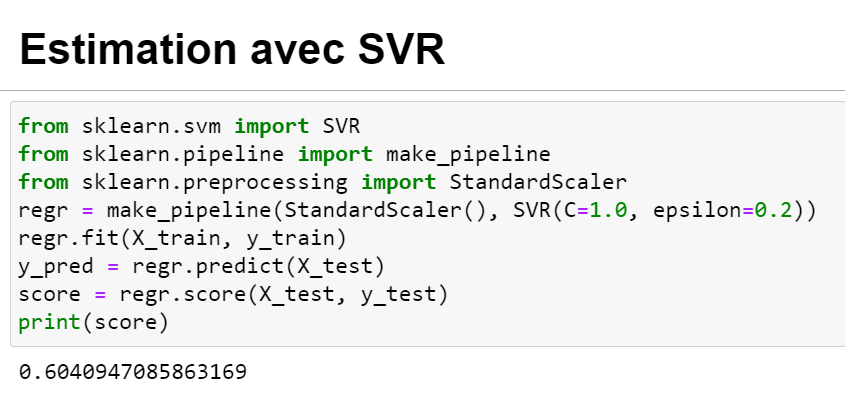


## SVR

Nous allons à présent en tester un autre pour voir si nous pouvons obtenir un meilleur score. Nous avons donc testé le modèle SVR (Support Vector Regression), le SVR est une régression vectorielle de type Epsilon-support. SVR peut prendre le paramètre optionnel epsilon qui spécifie le tube epsilon dans lequel aucune pénalité dans la fonction de perte de training n’est associée aux points prédits à une distance epsilon de la valeur réelle. Ce paramètre prend par défaut la valeur 0.1.



Pour ce modèle nous avons eu un résultat de 60,4% de prédiction exacte.



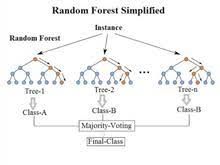
## Random Forest Regressor

Un random forest est constitué d'un ensemble d'arbres de décision indépendants.

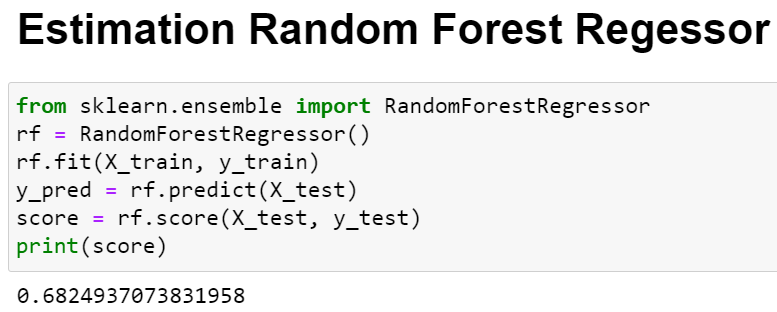
Chaque arbre dispose d'une vision parcellaire du problème du fait d'un double tirage aléatoire :

* Un tirage aléatoire avec remplacement sur les observations (les lignes de votre base de données). Ce processus s'appelle le tree bagging,
* Un tirage aléatoire sur les variables (les colonnes de votre base de données). Ce processus s'appelle le feature sampling.

A la fin, tous ces arbres de décisions indépendants sont assemblés. La prédiction faite par le random forest pour des données inconnues est alors la moyenne (ou le vote, dans le cas d'un problème de classification) de tous les arbres.

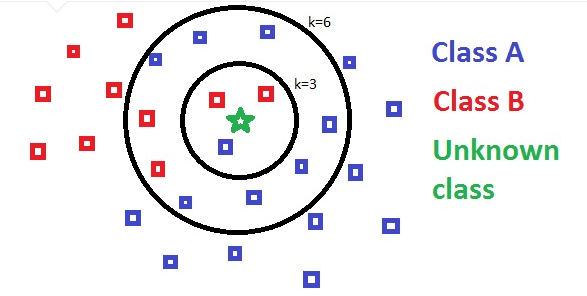


Nous avons obtenu un score de 68,2% de bonne prédiction grâce à ce modèle



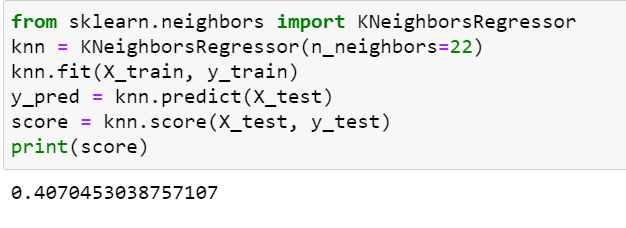
## Knn

Le premier algorithme de Machine Learning supervisé que nous avons testé est le k-nearest neighbors, dis knn. L’algorithme va chercher les K instances du jeu de données les plus proches de notre observation. L’idée est la suivante : à partir d’une base de données étiquetées, on peut estimer la classe d’une nouvelle donnée en regardant quelle est la classe majoritaire des k données voisines les plus proches (d’où le nom de l’algorithme). Le seul paramètre à fixer est k, le nombre de voisins à considérer.



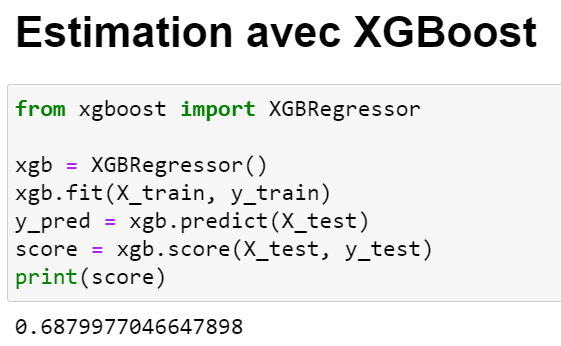
Nous obtenons un score de 10% car les données ne sont pas centrés et réduites, une fois cette étape réalisé nous obtenons 40%.





## XGBoost

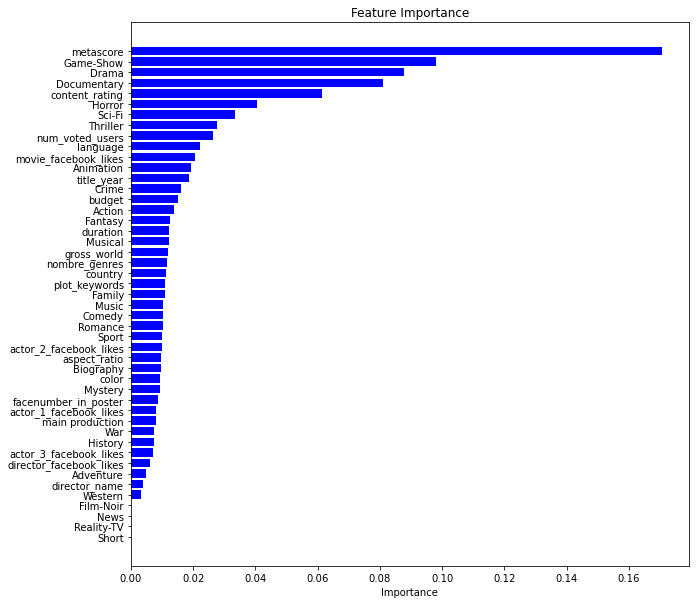
La dernière technique d’apprentissage supervisé que nous allons tester est XGBoost. Le Boosting de Gradient est un algorithme d’apprentissage supervisé dont le principe et de combiner les résultats d’un ensemble de modèles plus simple et plus faibles afin de fournir une meilleure prédiction. Pour ce modèle nous avons obtenu un score de 68,7%.



Nous pouvons en conclure que la meilleure technique d’apprentissage est le XGBoost. Nous utiliserons donc cette méthode pour la suite de notre projet.

## Sélection des features

Nous allons à présent voire l’influence de chacune des features sur le score de prédiction.

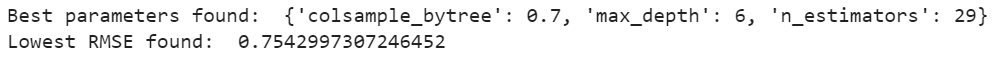


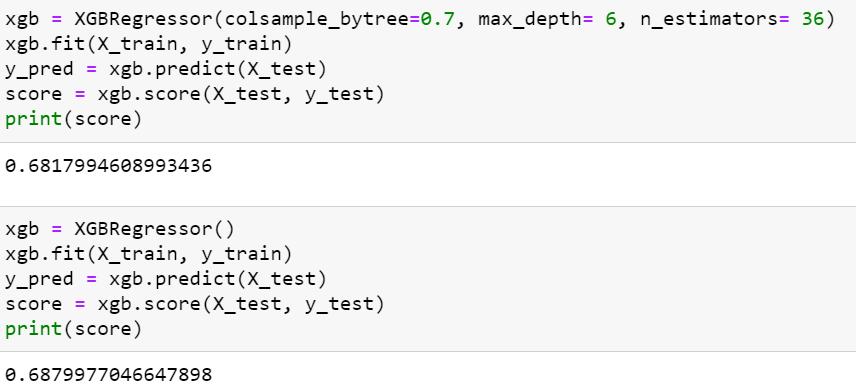
On observe facilement que certaines features se démarquent. On va donc voir si réduire le nombre de features permet d’améliorer le score de prédiction. On va donc garder seulement les 20 features ayant le plus d’impact (nous avons tester avec toutes les valeurs, de 1 à 47). Nous observons que le score n’augmente pas en réduisant le nombre de features. Nous allons donc garder l’ensemble des données.

## Recherche des paramètres optimaux

Nous pouvons ensuite chercher les paramètres optimaux afin d’améliorer le score de précision de notre prédiction. Nous allons utiliser la bibliothèque GridSearchCV et renter tous les paramètres à faire varier. Verbose=1 permet de suivre l’évolution de l’analyse et cv=4 permet de partionner les analyse par 4.



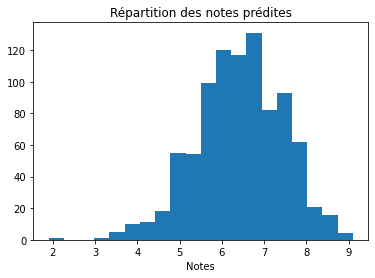




Une recherche des paramètres optimaux avec GridSearchCV donne des paramètres optimaux mais sont moins bon que sans paramètres. On va donc laisser le XGBoostRegressor comme tel.

## Analyse des résultats

Le XGBoostRegressor permet de prédire les notes des films sur les données test prises à partir des données train de façon aléatoire. Il faut dans un premier temps les arrondir afin qu’elles soient homogènes avec la notation du site. On obtient enfin cette répartition.



Cette répartition semble cohérente avec la répartition que nous avions de base dans les données train de notre dataset.

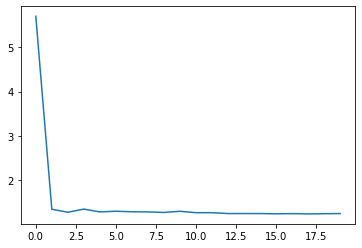
# Réseau de neurones

Les algorithmes de machine learning ont données des résultats mitigés, avec des scores ne dépassant pas 70%. Nous avons donc pensé à mettre en place des alternatives, en passant par les réseaux de neurones keras notamment.

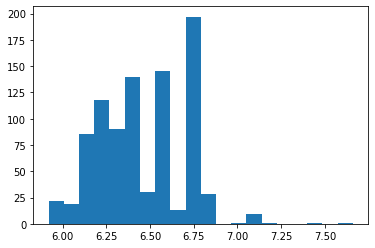
Nous avons donc créé un modèle séquentiel à 6 couches de la dimension de la base de données mesuarant l’erreur moyenne.



On applique ensuite ce modèle sur 20 epochs, on remarque en effet qu’au bout de la deuxième seulement l’erreur ne bouge plus.



On applique ensuite le modèle sur les données test que nous avons split comme lors de la partie machine learning. De même, on arrondi les résultats afin qu’ils soient homogènes.



On observe que la répartition des notes n’est pas homogène, elles sont toutes comprises en 6 et 7 ce qui n’est pas normal. On peut donc dire que notre système fait du surapprentissage et qu’il n’est pas performant. On ne va donc pas l’utiliser.

# Solution retenue et résultats

Notre réseau de neurones étant moins efficace que le XGBoost, nous avons donc choisi de garder l’algorithme de machine learning. Je pense qu’un meilleur réseau de neurones devrait cependant être plus efficace mais le notre est trop petit et fait du surapprentissage trop rapidement.

Nous avons donc appliqué le XGBoost sur les données test afin de prédire la note des films. Nous avons gardé l’id et la note dans un fichier csv que nous avons soumis au concours kaggle.

Ce projet nous a permis de réaliser et de mieux comprendre ce qu’est un projet de data science. Nous avons appris à traiter nos données de façon optimales afin d’améliorer les modèles. La partie réseaux de neurones nous a posé le plus de problèmes car nous avons eu du mal à comprendre sa mise en place et les différents types de couches et d’activations.

Nous obtenons, sur le concours kaggle un score de 65%, ce qui devrait être largement améliorable en continuant l’optimisation du traitement des données et en améliorant notre modèle de réseaux de neurones. Une base de données plus grande aurait également permis un meilleur modèle pour une meilleure prédiction.

Nous souhaitons remercier Mr Ala Bayoudh pour son aide et sa disponibilité tout au long de notre projet.