Rapport projet SY09 : Les épisodes et films de Scooby-Doo

https://github.com/rfordatascience/tidytuesday/blob/master/data/2021/2021-07-13/readme.md

Léonard Penverne et Baptiste Viera

8 juin 2022

1 Introduction

Ce rapport rend compte des travaux effectués dans le cadre du projet de SY09. Le but de ce projet est d'appliquer les techniques d'analyse de données vues pendant le cours de SY09 à un des jeux de données de tidy tuesday. Nous avons choisi le dataset regroupant les données sur les épisodes de la série "Scooby-Doo". Ce jeu de données, construit par un internaute et publié en premier lieu sur Kaggle regroupe la description des épisodes et films de Scooby-Doo regardés par l'auteur à travers 63 variables. Certaines de ces variables sont quantitatives tandis que d'autres sont catégorielles. Le jeu de données de base étant incomplet et peu utilisable en l'état, nous lui avons apporté de nombreuses modifications (détaillées dans la partie Préparation des données) avant d'appliquer les différentes méthodes d'apprentissage vues tout au long du semestre. Notre étude du jeu de données consiste en deux problèmes distincts : un problème de classification et un de régression. Pour le problème de classification nous nous sommes penchés sur la colonne "monster-real" et avons tentés de classifier les épisodes suivant si le ou les monstres rencontrés par le gang étaient seulement des humains déguisés (quand monster-real est à false) ou qu'il s'agissait d'authentiques créatures surnaturelles (quand monster-real est à true). Pour le problème de régression nous avons tenté de prédire la variable "IMDB" des données : cette variable quantitative décrit le score imdb (un site spécialisé dans le cinéma et la télévision) obtenu par le film ou l'épisode en question.

2 Présentation des données

Nos données sont les observations de 74 variables pour 603 épisodes ou films différents de Scooby-doo, le jeu de données contient aussi des épisodes de séries annexes où les personnages de Scooby-doo font des apparitions remarquées. Concernant les 74 variables, celles-ci sont forcément variées. Nous pouvons cependant les classer en plusieurs catégories. Premièrement les variables décrivant l'épisode de façon générale (durée, date de parution,chaîne de parution, saison, nombre de vues...). Deuxièmement, les données concernant la description des monstres de l'épisode (nombre, genre, espèce, sous-espèces, si le monstre est en fait un humain déguisé...) et des suspects et coupables (nombre, genre, motivations...). Troisièmement, les variables qui décrivent les actions du "gang" (quel membre attrape et démasque les monstres, quel membre se fait capturer, l'efficacité des pièges...). Dernièrement, des variables qui décrivent des aspects plus singuliers de l'épisode (présence de batman, de scrappy-doo, nom des doubleurs, quelles "catch phrases" ont été prononcées...). Une partie importante des variables (surtout celles qui ne sont pas comprises dans la description générale de l'épisode) sont booléennes ou catégorielles, ceci implique donc des choix de reformatage des données que nous détaillerons plus tard. Le nombre de variables étant important nous nous sommes concentrés, pour l'analyse exploratoire des données sur les variables que nous avons tenté de prédire et classifier par la suite, mais aussi, de manière subjective, sur des variables qui nous semblaient intéressantes à visualiser.

3 Analyse exploratoire des données

3.1 Evolution du score IMDB en fonction de différents paramètres

A travers ce graphique, nous avons voulu identifier l'évolution des scores IMDB ainsi que certains critères qui font la réussite d'un épisode de Scooby-Doo.

Finalement, il semble que Scooby-Doo reste intemporel : en effet la qualité des épisodes qui chutait doucement mais sûrement depuis 1970 a connu un rebond dans les années 2010. De plus, il semble que le public préfère que le monstre ne soit pas réel on peut le voir avec les épisodes de la fin des années 2000.

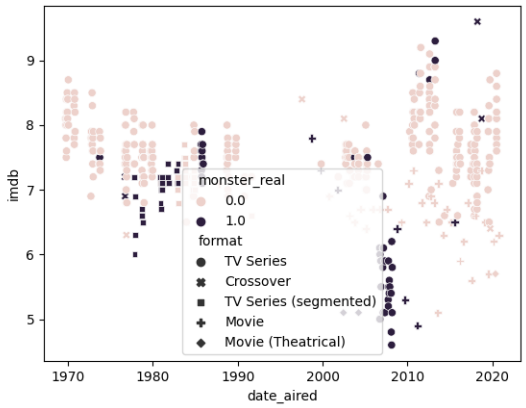

FIGURE 1 – Evolution du IMDB en fonction des années

3.2 L'action des protagonistes

Nous avons souhaité identifier parmi l'ensemble des protagonistes qui étaient ceux qui attrapaient et démasquaient le plus de monstres ainsi que ceux qui étaient le plus capturés.

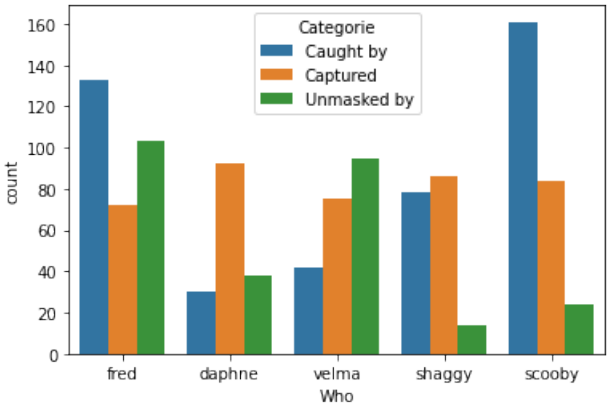

FIGURE 2 – Proportion des protagonistes dans chaque catégorie

3.3 Motivations des monstres

Nous avons souhaité déterminer les motivations principales des monstres/criminels sur l'ensemble des épisodes diffusés depuis 1969 puis par décennie.

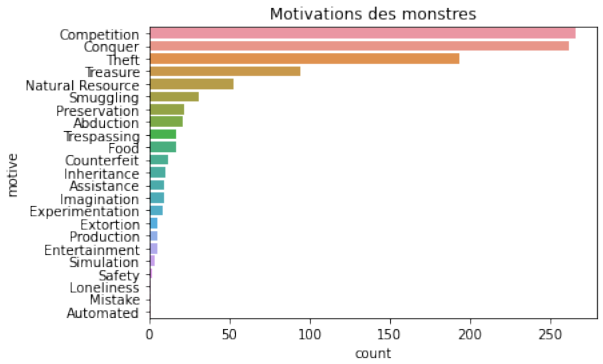

FIGURE 3 – Motivations des monstres

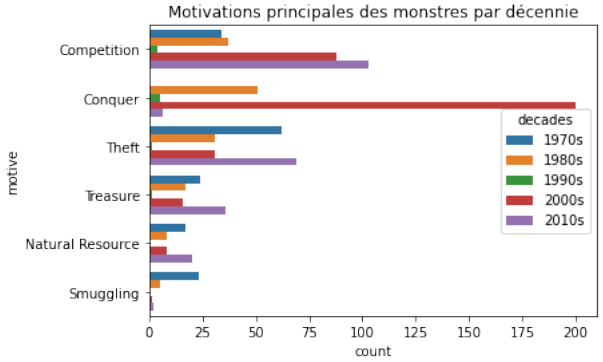

FIGURE 4 – Motivations des monstres par décennie (top 6)

4 Méthodes non paramétriques

4.1 Préparation des données

Comme mentionné précédemment nos données ont dû être nettoyées avant de pouvoir être exploitées. Nous avons donc résolu deux problèmes distincts pour avoir des données utilisables : les données manquantes et les données catégorielles.

Pour ce qui est des données manquantes nous ne pouvions pas simplement effacer les lignes avec des variables non déclarées. En effet, en suivant cette méthode nous n'aurions conservé qu'un tiers de nos données d'origine, modifiant donc sensiblement le jeu de données. Nous avons donc choisi de remplacer les données manquantes par la moyenne des données de la colonne en fonction de la chaîne de parution et de la saison de l'épisode. Cette méthode marche pour la variable quantitative mais pose problème pour les variables booléennes ou catégorielles. Dans le cas des variables catégorielles elles seront malheureusement exclues de l'analyse si non renseignées. Pour les variables booléennes nous avons fait le choix de

coder une observation manquante par la valeur la plus présente dans sa saison, c'est une hypothèse importante de notre analyse, mais elle nous permet d'exploiter un maximum de données.

En plus de ce remplacement, certaines colonnes ont été ignorées pour l'analyse du dataset et ce pour plusieurs raisons différentes. Premièrement, les colonnes contenant des informations différentes pour chacun des épisodes : le nom des monstres, le titre de l'épisode, le nom du coupable et les répliques prononcées par le coupable une fois arrêté ("if it wasn't for..." et "and that..."). Deuxièmement, les variables contenant trop de valeurs manquantes et ne pouvant être remplacées par la méthode des moyennes : les doubleurs de chaque personnage. Enfin nous avons décidé de ne pas conserver la variable indiquant le genre du monstre, car elle était peu renseignée et hautement corrélée avec l'existence réelle du monstre. Pour appliquer les méthodes vues en cours, il a fallu nettoyer nos données.

Enfin, pour pouvoir utiliser les variables catégorielles dans leur ensemble, nous avons utilisé le one-hot encoding. Comme il n'existait pas de relations d'ordre dans les données catégorielles, il nous a semblé judicieux de faire ce choix. Ce choix aura deux conséquences majeures qu'on retrouvera par la suite : d'un côté la création de nombreuses nouvelles colonnes et de l'autre la création de variables colinéaires.

Finalement nous nous retrouvons avec un jeu de données de 487 individus et 786 variables, que nous avons centré et réduit pour toutes les variables non booléennes. Ce jeu de données sera modifié par la suite comme expliqué pour chaque problème.

4.2 Matrice de corrélation

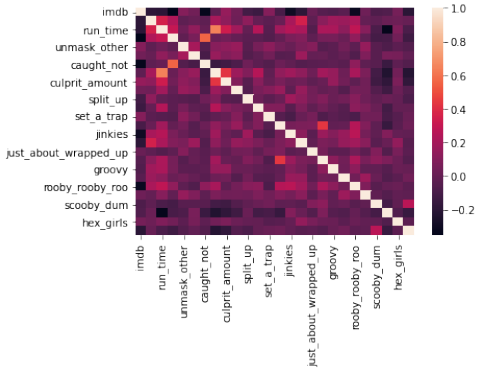

FIGURE 5 – Matrice de corrélation de notre jeu de données

On voit que dans notre dataset peu de variables sont corrélées entre elles. Malheureusement, cette matrice ne comporte pas les données booléennes du dataset ce qui réduit son utilité.

4.3 ACP

Effectuer une ACP sur nos données ne semble pas évident : en effet des données booléennes ne peuvent être utilisées dans une ACP. Hors, du fait du choix de one-hot encoding et de la nature booléenne de plusieurs variables, nos données sont donc inutilisables pour une ACP. Cela aura notamment des conséquences lors de la visualisation de nos régions de décisions, nous ne pourrons pas projeter nos données sur les axes principaux comme vu en TD.

Malgré cela nous avons voulu voir si une ACP n'utilisant que les variables quantitatives était intéressante. Cette ACP a été effectuée avec des données centrées et réduites

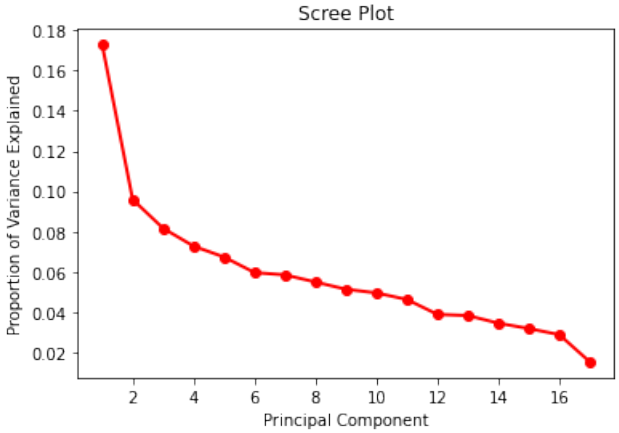

FIGURE 6 – PCA sur les données quantitatives

Malheureusement, une fois encore le résultat n'est pas exploitable. Pour notre analyse, une ACP aurait été très intéressante si elle permettait de réduire sensiblement la dimensionnalité de nos données. Hors ce n'est pas le cas ici : pour conserver 90 % de la variance expliquée nous devons conserver 14 axes. Idem si nous voulions utiliser la règle du coude : si l'on ne garde que les 11 premiers axes, on perd 20 % de la variance expliquée. Au vu de ces résultats (et des variables utilisées lors de la construction des arbres que nous détaillerons plus tard) nous avons donc pris la décision d'utiliser le jeu de données d'origine et non celui transformé.

5 Problème de classification

Nous avons donc choisi de tester les différentes méthodes de classification vues en cours sur la colonne

"monster real". Dans un premier temps nous avons effectué notre analyse avec toutes les variables. Puis nous avons décidé de choisir les variables les plus utilisées par le Bagging pour retester les méthodes vues en cours. La deuxième option présentant de meilleurs résultats, ce sont ceux que nous présenterons et commenterons dans ce rapport. Après le one-hot encoding les 16 variables sélectionnées grâce au Bagging sont codées sur 203 et 504 lignes.

TABLE 1 – Description des variables utilisées pour la classification

Nom	Type	Description
monster-real	bool	Monstre réel ou non
engagement	float	Nombre d'avis sur IMDB
set-a-trap	float	Nombre de pièges fait par le gang
run-time	float	Durée de l'épisode
jeepers	float	Nombre de fois où 'jeepers' a été dit
split-up	float	Nombre de fois où le gang se sépare
imdb	float	Score imdb de l'épisode
series-name	string	Nom de la saison
setting-terrain	string	Paysage de l'épisode
network	string	Chaîne diffusant l'épisode
monster-gender	bool	Genre du monstre
arrested	bool	Coupable arrêté ou non
monster-type	string	Type de monstre
motive	string	Motivation du méchant
number-of-snacks	string	Nombre de scooby-snacks mangés
format	string	Episode ou film
caught-shaggy	bool	Si shaggy attrape le monstre

Notre classification se fait donc sur une colonne légèrement déséquilibrée. Cependant, ce déséquilibre ne nous semble pas suffisamment important, nous utiliserons les méthodes vues en cours sans prétraitement pour rééquilibrer les poids des classes.

Pour la suite du problème de classification, on considérera, sauf si l'inverse est mentionné, que la validation des résultats s'est faite grâce à une 10-fold cross validation avec les fonctions (parfois modifiée) vues en TD

TABLE 2 – Proportion de monstres réels ou non dans le dataset nettoyé

Valeur de la colonne	Nombre d'observations
True	401
False	103

et à l'aide de la bibliothèque sklearn.La k-fold cross-validation nous permet de tester notre jeu de données sur 10 ensembles distincts et donc de réduire les risques de biais et d'overfitting. De plus, en raison du nombre réduit de données à notre disposition nous ne pensons pas qu'introduire un k supérieur à 10 soit souhaitable : nos classes étant déjà légèrement déséquilibrées nous ne voulons pas courir le risque d'avoir un modèle se testant sur un ensemble trop réduit.

5.1 K plus proches voisins

La méthode des K-plus proches voisins nous servira principalement d'élément de comparaison, une sorte de "benchmark" pour nos résultats. L'application de cette méthode doit être pondérée par plusieurs facteurs propres à nos données qui diminuent sa précision. Premièrement, la nature de nos données est encore une fois embarrassante pour l'application de la méthode. En effet, avec nos données à la fois booléennes et quantitatives, les distances de base proposées par sklearn ne sont pas adaptées. Nous avons donc choisi de comparer la méthode KNN en utilisant 2 métriques différentes : la distance de Manhattan et la distance de Hamming :

$$\mathrm{HD}_{\mathrm{raw}} = \frac{\|(\mathrm{codeA} \otimes \mathrm{codeB}) \cap \mathrm{maskA} \cap \mathrm{maskB}\|}{\|\mathrm{maskA} \cap \mathrm{maskB}\|}$$

Deuxièmement, le fléau de la dimensionnalité est très présent dans nos données : avec 207 colonnes on peut se permettre d'émettre des doutes sur la proximité entre les différents voisins lors du calcul des distances.

Pour cet algorithme, nous avons premièrement cherché le nombre de voisins optimal à utiliser, pour cela nous avons testé l'algorithme avec les 2 mesures de distances de 1 à 100 voisins, puis nous avons choisi le nombre de voisin maximisant l'accuracy sur une moyenne de 10 itérations avec le même nombre de voisins. Finalement, nous avons trouvé que le nombre optimal de voisins était de 5 pour la distance Euclidienne et de 3 pour la distance de Hamming.

Finalement, il semble que les 2 distances produisent des résultats similaires, cela est sûrement expliqué par nos données mixtes (à la fois catégorielles et quantitatives). Ce qui ne permet pas à une des deux dis-

TABLE 3 – Résultats des classifieurs KNN

Distance utilisée	Moyenne des scores
Hamming	0.920
Euclide	0.916

tances d'émerger comme significativement meilleure que l'autre.

Nous avons essayé de visualiser nos données, malheureusement, comme mentionné précédemment la visualisation est délicate. Comme nous n'avons pas pu projeter nos données sur 2 axes de variance expliquée et que nous avons plus de 2 variables, nous avons dû faire un choix. Nous avons donc utilisé les axes 'imdb' et 'durée de l'épisode' pour projeter nos variables et les valeurs qui ont été prédites par le classifieur. Cette visualisation n'est pas une agrégation des classes moyennes prédites lors de la 10-fold cross validation mais une simple indication sur une itération de l'algorithme des 3 plus proches voisins avec le jeu de données séparé en 70/30.

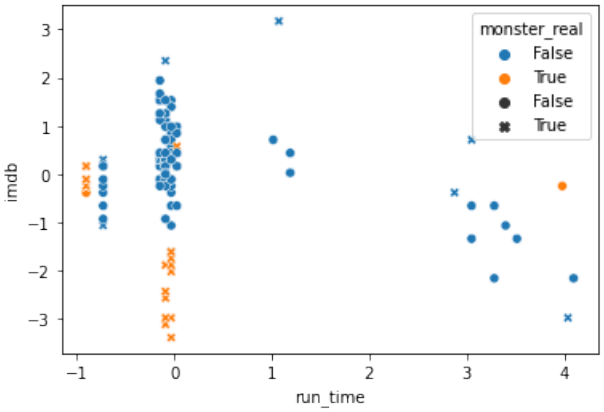

FIGURE 7 – Résultats de la classification KNN.

Sur le graphique la couleur correspond à la valeur prédite et la forme correspond à la valeur réelle. Les points bleus sont donc des monstres non-réels prédits comme tel, les croix jaunes des monstres réels prédits comme tels, les croix bleues des monstres réels prédits comme faux et les points jaunes des monstres iréels prédits comme réels.

5.2 LDA, QDA et Bayes Naïf

De la même façon, que pour les plus proches voisins nous avons utilisé la 10-fold cross validation pour cette partie. Après avoir établi un élément des comparaisons avec les méthodes KNN, nous avons voulu voir lesquels des classifieurs vus en cours était le plus performant pour notre jeu de données. Nous nous sommes donc penchés sur les méthodes d'analyse discriminante. Les résultats des 3 méthodes sont visibles dans le tableau 2.

TABLE 4 – Résultats des classifieurs d'analyse discriminante

Méthode utilisée	Moyenne des scores
LDA	0.920
QDA	0.662
NB	0.638

On remarque donc une grosse différence entre les méthodes et notamment une sous-performance de la QDA et du classifieur Bayésien naïf. Nous avons formulé plusieurs hypothèses vis-à-vis de ce résultat.

La sous performance de ces classifieurs peut selon nous être expliquée par 2 aspects. Premièrement, l'hypothèse forte de normalité des données est fortement violée par nos données booléennes. Deuxièmement, à cause de nos données catégorielles l'estimation des matrices de variances est peu précise : il n'y a pas assez d'individus pour estimer correctement les paramètres. Le fait de supposer la matrice de variance diagonale ne résout pas ce problème et le classifieur bayésien est aussi sous-performant. On pouvait s'attendre à ce résultat, comme nous l'avions expliqué précédemment les variables sont très peu corrélées et la matrice de covariance estimée pour la QDA doit donc être de base presque diagonale. La performance de la LDA nous a cependant interrogé, avec une dimension importante nous nous attendions à ce que ce classifieur souffre mais ce n'est visiblement pas le cas, il semble aussi ne pas être affecté par la non normalité des données.

Vous trouverez en annexe la visualisation de prédiction des classifieurs pour un même train-test, split. Malheureusement nous n'avons pas pu y ajouter les différentes frontières de décision en raison des problèmes déjà évoqués pour les KNN. Ces représentations sont adaptées du code suivant.

5.3 Regression Logistique

La régression logistique est pour l'instant la méthode la plus apte à prédire nos données : après une 10-fold cross validation on se retrouve avec une précision estimée à 0.942.

5.4 Arbres

Avant de nous pencher sur les méthodes arborescentes nous avons voulu explorer notre dataset au moyen

d'arbres. Premièrement, nous avons voulu, en vue de l'application des forêts aléatoires et du bootstrap aggregating, quelle mesure d'impureté des nœuds était la plus adaptée à notre jeu de données. Pour ce faire nous avons exploré la performance d'arbres avec une profondeur de 1 à 50 selon le critère de décision. La figure est disponible en Annexe.

Cette figure nous a fait comprendre que le critère de décision utilisé ainsi que la profondeur de l'arbre à utiliser ne sont malheureusement pas pré-réglables. De plus, nous n'avons pas pu mettre en place un algorithme satisfaisant pour le pruning post-croissance de nos arbres. Nous nous sommes donc orientés vers un élagage post-croissance de l'arbre. Finalement nous avons tiré de cette étape nos paramètres pour les méthodes arborescentes : nous utiliserons le pré-élagage avec comme critère l'indice de Gini.

Vous trouverez en annexe un exemple d'arbre construit avec l'indice de gini et un maximum de 10 feuilles pour plus de lisibilité.

5.5 Méthodes arborescentes

Les dernières méthodes que nous avons mises en place pour la classification sont les randoms forests et le bagging. Comme précisé précédemment, nous avons utilisé l'indice de Gini comme critère de décision.

Table 5 – Résultats des classifieurs arborescents

Méthode	Moyenne des scores
Bagging	0.924
Random Forest	0.926

Les résultats de ces classifieurs sont comparables à ceux obtenus jusqu'à présent. Cette fois-ci, il semble que nos données soient adaptées à ces méthodes. Cela n'a rien de surprenant, à chaque nœud la décision peut facilement être prise, ayant des variables booléennes pour la plupart il suffit de régler le seuil de décision à 0,5 pour avoir une partition facile des données restantes.

Comme mentionné précédemment nous avons utilisé ces méthodes avec tout le dataset de base pour choisir les variables que nous allions utiliser dans notre analyse. Pour ce faire nous avons utilisé les variables qui menaient à la meilleure accuracy lors d'une procédure de Bagging. L'algorithme de l'arbre étant glouton ceci a un intérêt particulier car notre variable à prédire est booléenne et nous n'avons pas à effectuer des tests ANOVA sur chaque colonne pour estimer les dépendances.

5.6 Conclusion sur la classification

Finalement, après avoir exploré plusieurs méthodes de classification, il ressort, selon nous, que la principale explication des différences de performance reste la forme particulière de nos données. Comme nous l'avons dit, le one-hot encoding utilisé pour exploiter nos variables catégorielles a créé une multitude de colonnes. Cette grande dimensionnalité et le fait que nos données soient majoritairement binaires a avantagé certaines méthodes (méthodes arborescentes) et désavantagé d'autres (méthodes d'analyse discriminante). En regardant de plus près nos résultats, nous restons aussi surpris par la performance des classifieurs KNN. En effet, au vu de la grande dimensionnalité de nos données nous ne nous attendions pas à une performance de plus de 90 %, encore une fois on peut se demander à quel point les 3 plus proches voisins sont réellement proches du point dont la classe est à déterminer.

Finalement au vu des performances similaires de nos classifieurs nous avons voulu estimer comment le déséquilibre de la classe à prédire impactait les différents classifieurs.

		Predicted	
		True	False
Actual	True	87	16
	False	13	388

Table 6 – Matrice de Confusion de la régression logistique

		Predicted	
		True	False
Actual	True	78	25
	False	11	390

Table 7 – Matrice de confusion de la random forest

Là aussi, la régression logistique se détache des autres. Si les autres méthodes produisent une matrice de confusion semblable à celle de la random forest, la régression logistique semble être la méthode la plus performante pour classer les True de la colonne. Ainsi c'est la seule méthode à avoir un true positive rate de près de 85% tandis que les autres méthodes tournent autour de 75%. Donc, en plus d'être le classifieur le plus performant, la régression logistique est celui qui est le moins impacté par le déséquilibre de la classe. Ce qui en fait le meilleur classifieur pour notre problème.

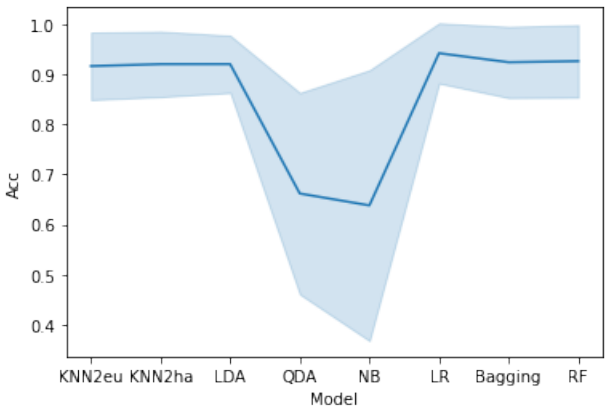

FIGURE 8 – Performance des différents méthodes de classification

6 Problème de régression

Dans un premier temps, après avoir basé notre prédiction du score IMDB sur le jeu de données avec traitement, nous obtenons les scores suivants :

TABLE 8 – Résultats des méthodes de régression

Méthode utilisée	Moyenne des scores
Linear Regression	-0.15
Random Forest	0.50

Dans Sckit-learn, les scores de régression linéaire utilisent le score R2. Un R2 négatif signifie que le modèle s'est mal adapté à nos données. Puisque R2 compare l'ajustement du modèle avec celui de l'hypothèse nulle c'est-à-dire une ligne droite horizontale, R2 est négatif dès lors que le modèle s'ajuste moins bien qu'une ligne horizontale. En d'autres termes, un R2 négatif signifie que la moyenne de notre échantillon de test serait une prédiction plus précise que celle donnée par notre modèle. Une des raisons est en partie liée au fléau de la dimensionnalité (nous avons plus de variables que d'individus). Plusieurs méthodes s'offrent ainsi à nous. Nous avons utilisé un modèle pénalisé (lasso) qui permet de résoudre le problème de la multicolinéarité entre les variables. En normalisant nos données et en appliquant la cross-validation, nous obtenons avec la méthode Lasso un score de 0.44 avec un MAE et MSE respectivement de 0.34 et 0.24.

Concernant le modèle de Random Forest, nous avons détecté du surapprentissage. En effet, le score obtenu à partir de notre jeu d'entraînement est de 0.93 alors que celui de notre jeu de test est de 0.50. Une des solutions pour y palier est de trouver les hyperparamètres optimaux à l'aide de GridSearchCV. Bien que cela réduise le surapprentissage, le score obtenu de 0.51 n'est pas convaincant. Nous avons ensuite procédé à une sélection des variables à l'aide de la méthode RFECV (recursive feature elimination cross validation) qui nous permet d'obtenir un score de 0.52 avec un MPAE de 4.2% avec 215 variables sélectionnées. Une autre explication plausible pourrait être liée au nettoyage des données que nous avons fait, engendrant du bruit à l'origine de mauvaises prédictions sur le score IMDB.

Après avoir essayé d'autres prétraitements qui n'étaient pas concluants, bien que discutable, nous avons donc décidé de travailler avec le jeu de données de base. De peur de créer du bruit, nous n'avons pas ainsi remplacé les valeurs manquantes, nous les avons supprimées. Nous avons au préalable fait une sélection des variables manuellement. Nous avons créé des visualisations de nos données afin de déterminer les variables qui ont une influence sur le score IMDB. Pour les variables quantitatives, nous avons étudié les corrélations puis utilisé un test de significativité afin de déterminer la dépendance des variables. Concernant les variables qualitatives, nous avons visualisé les distributions puis effectué l'ANOVA afin de déterminer une éventuelle dépendance de nos variables en fonction du score IMDB. Cette méthode est bien évidemment très discutable puisque, entre autres, les hypothèses émises sont fortes et le lien entre les variables explicatives n'est pas observé. Nous obtenons alors un nombre de 44 variables qui, transformées en one-hot, correspondent à 114 variables pour un total de 366 lignes. A noter que parmi ces variables, des variables non obtenues avec les tests de dépendance ont été ajoutées afin de garder des liens existants entre les variables explicatives.

TABLE 9 – Résultats des modèles de régression avec sélection de variables

Méthode utilisée	Moyenne des scores
Linear Regression	0.38
LassoCV	0.67
Random Forest	0.72

Suivant les modèles, après avoir appliqué la cross-validation et la sélection de variables afin d'améliorer les performances et de réduire au maximum le surapprentissage, nous obtenons les scores ci-dessus.

Concernant la recherche d'hyperparamètres pour le Random Forest, le nombre d'estimateurs optimal est alors fixé à 100, le critère utilisé est l'erreur absolue et la profondeur maximale de l'arbre est égale à 5. Voir figure 22 en annexe.

TABLE 10 – Erreurs des modèles de régression avec sélection de variables

Méthode utilisée	MAE	MPAE	MSE	RMSE
Linear Regression	0.36	0.053	0.34	0.59
LassoCV	0.31	0.044	0.18	0.43
Random Forest	0.28	0.040	0.15	0.39

Ci-dessous le nuage de points (épisodes) des scores IMDB prédits avec un RF en fonction des scores réels :

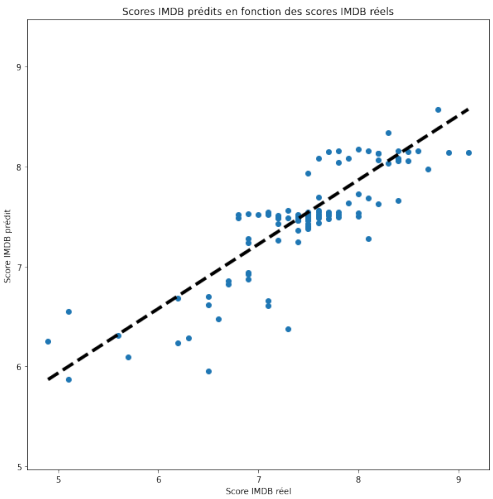

FIGURE 9 – IMDB prédit en fonction de l'IMDB réel

Enfin, il paraît ensuite intéressant de visualiser l'importance des variables dans notre modèle de random forest.

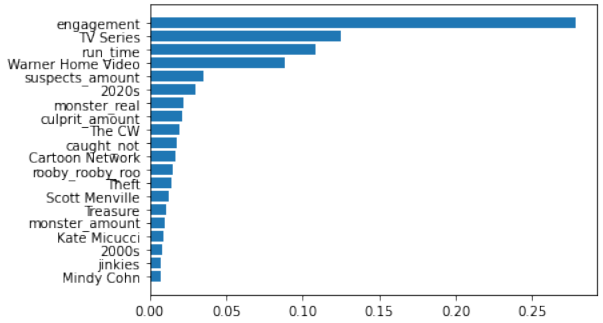

FIGURE 10 – Importance des variables

En analysant le signe des corrélations des variables explicatives avec la variable à expliquer, nous pouvons alors émettre, entre autres, les conclusions suivantes :

- Plus le nombre d'avis est faible, meilleur sera le score IMDB.
- Si le format est de type TV Series, meilleur sera le score IMDB.
- Plus le nombre de suspects sera faible, meilleur sera le score IMDB.
- Plus la durée est courte, meilleur sera le score IMDB.
- Si l'œuvre est diffusée par Warner Home video, le score IMDB aura tendance à être plus faible.
- Plus le nombre de coupables sera faible, meilleur sera le score IMDB
- La présence d'un faux monstre améliore le score IMDB
- Si l'œuvre est diffusée par Cartoon Network, le score IMDB aura tendance à être plus élevé.
- Plus le nombre de fois que sera dit "Rooby.rooby.roo" est faible, meilleur sera le score IMDB

6.1 Conclusion sur la régression

Nous avons donc observé que la régression linéaire multiple n'était pas adaptée à notre modèle du fait principalement des grandes dimensions et de la multicolinéarité entre les variables. Un modèle pénalisé comme le Lasso permet de pallier ce problème. Les méthodes arborescentes comme le Random Forest sont un choix judicieux. Grâce à cette dernière méthode, nous avons pu également déterminer quelles étaient les variables qui jouaient un rôle important dans le score IMDB. Il paraît nécessaire de mentionner que de potentielles améliorations pourraient être apportées. D'une part, nous aurions pu explorer d'autres méthodes pour remplacer les valeurs manquantes comme par l'ajustement d'un modèle de régression ou de classification. D'autre part, nous aurions pu essayer de tester d'autres modèles pénalisés comme l'Elastic Net ou encore d'autres méthodes d'arborescence comme le boosting de Gradient même si nous nous serions un peu trop éloignés des notions vues en SY09.

7 Annexes .

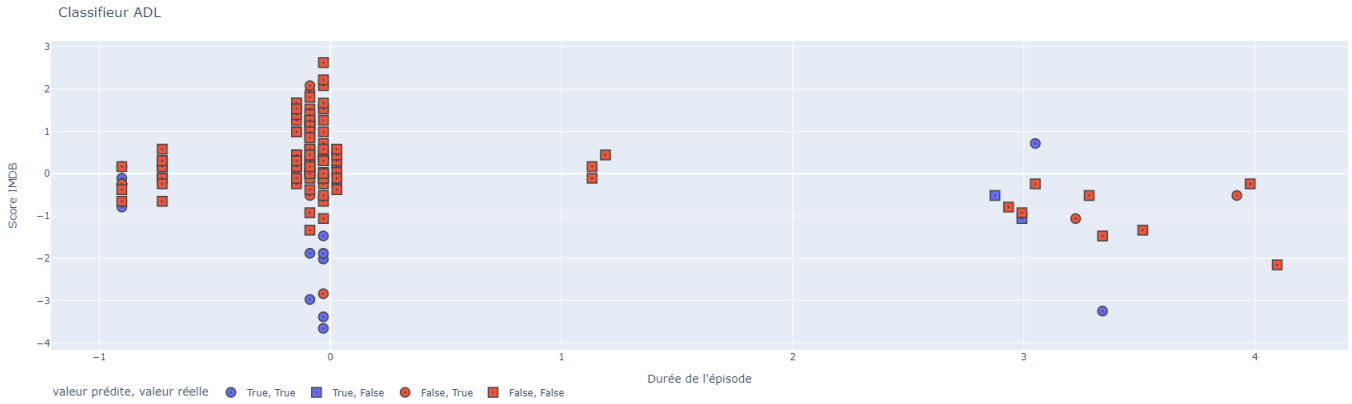

FIGURE 11 – Classifieur ADL

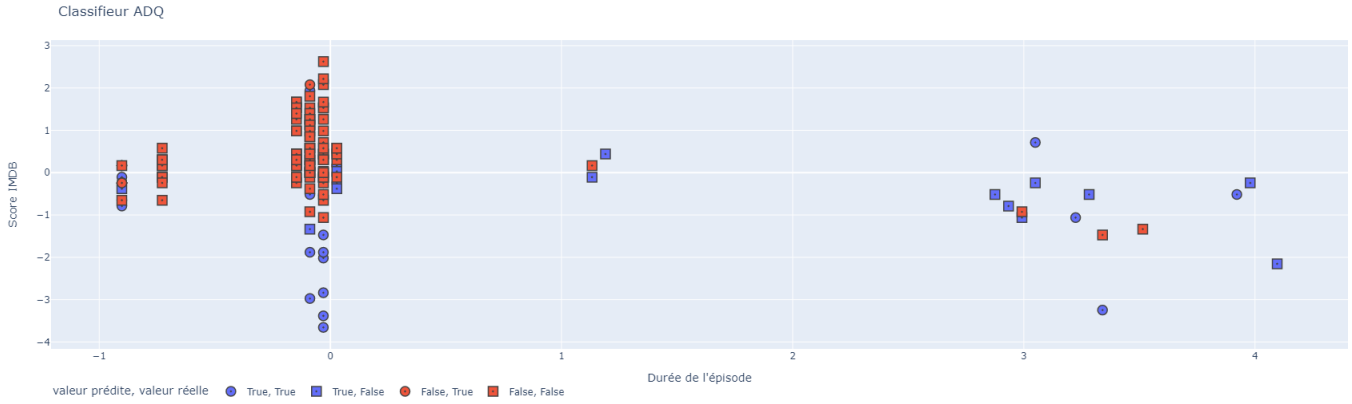

FIGURE 12 – Classifieur ADQ

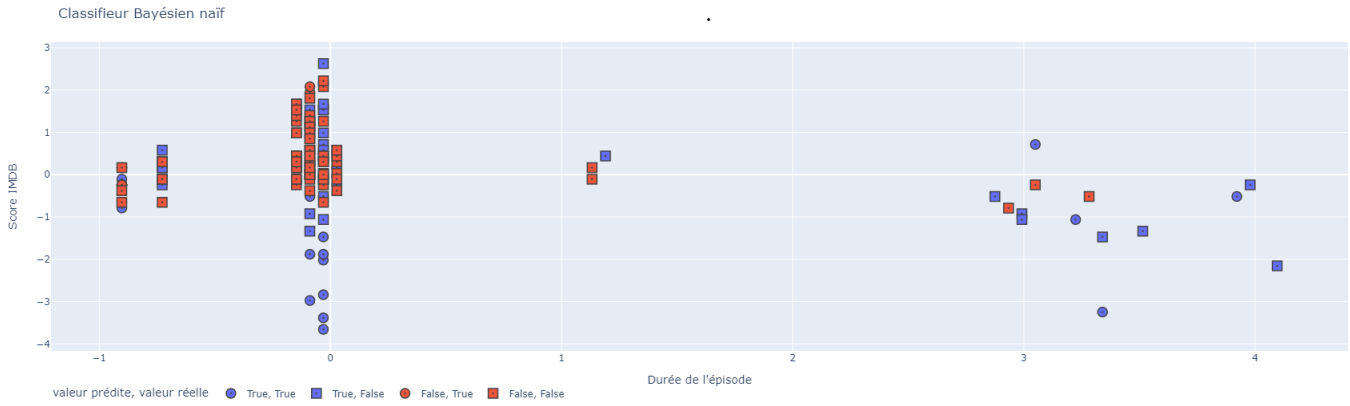

FIGURE 13 – Classifieur Bayésien naïf

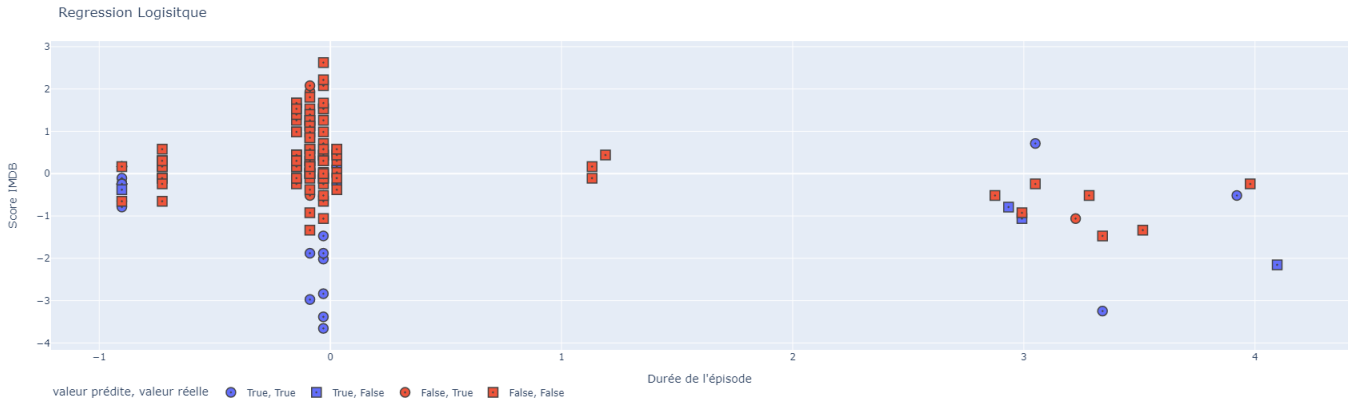

FIGURE 14 – Régression Logistique

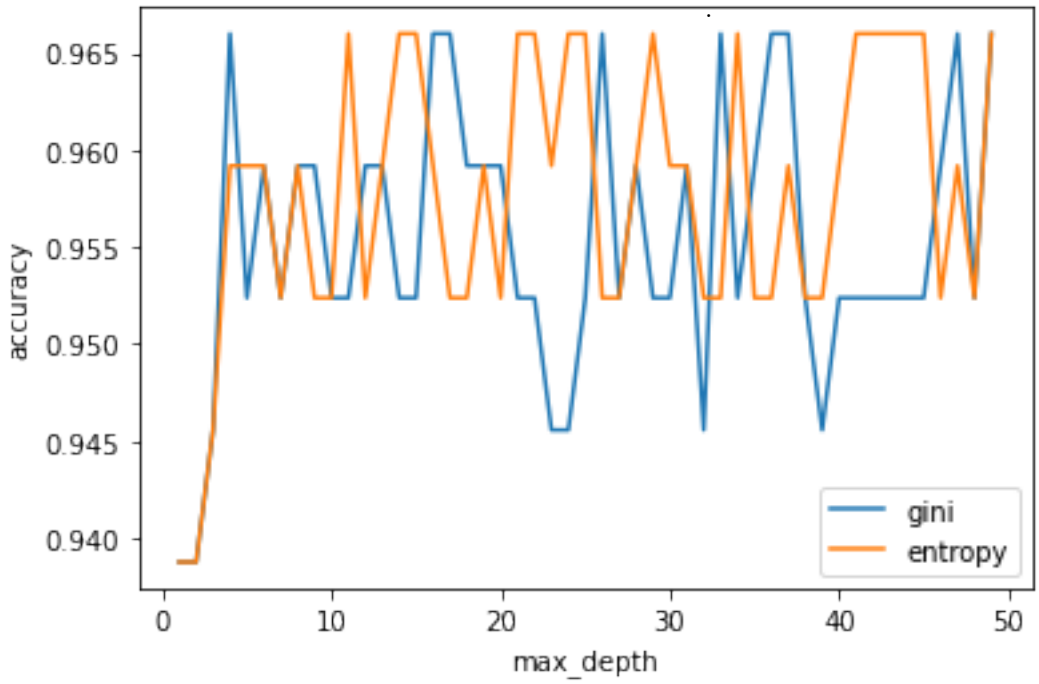

FIGURE 15 – Comparaison Indice de Gini vs. Entropie

.

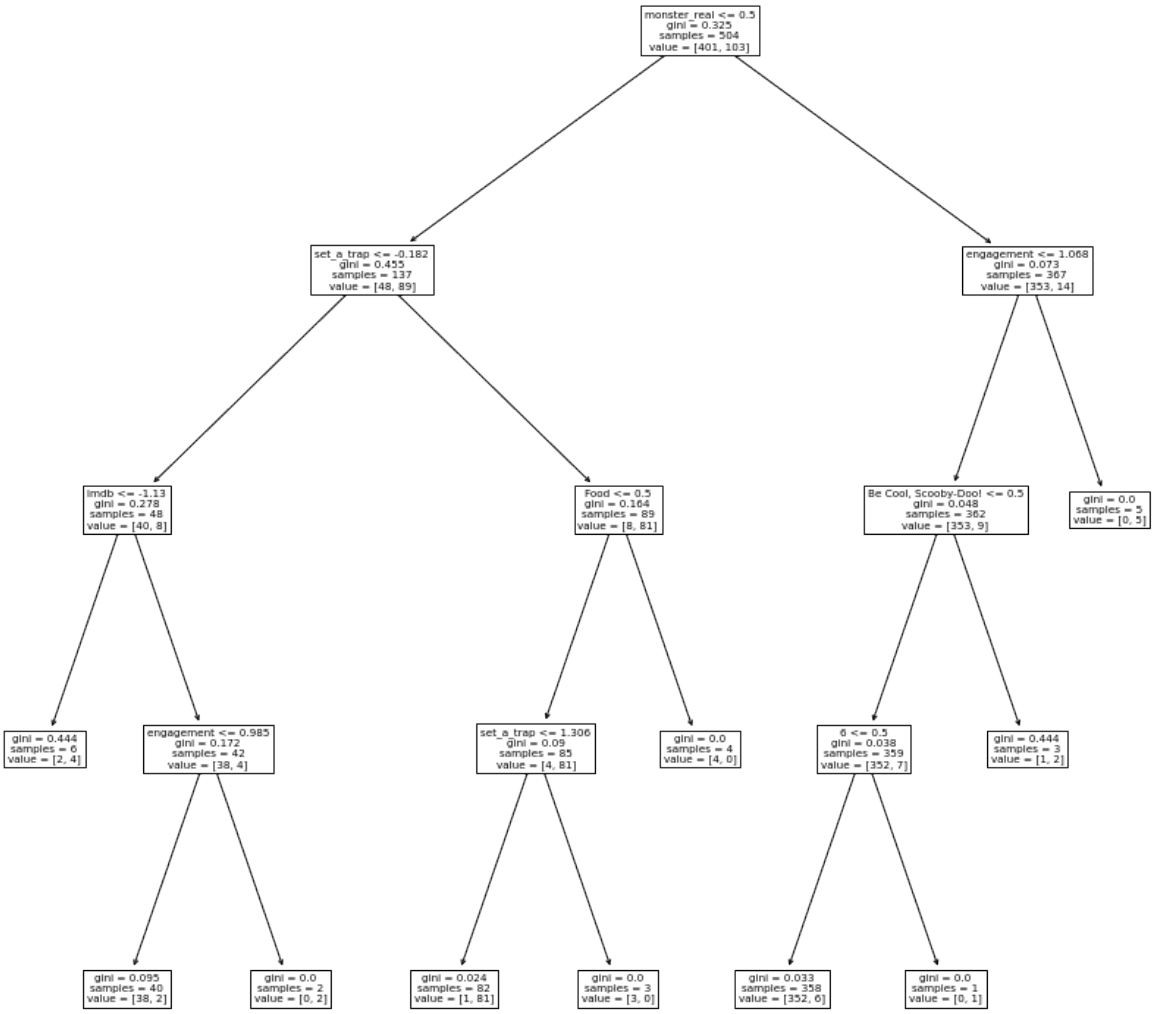

FIGURE 16 – Exemple d'arbre élagué pour le problème de classification (Existence du monstre)

.

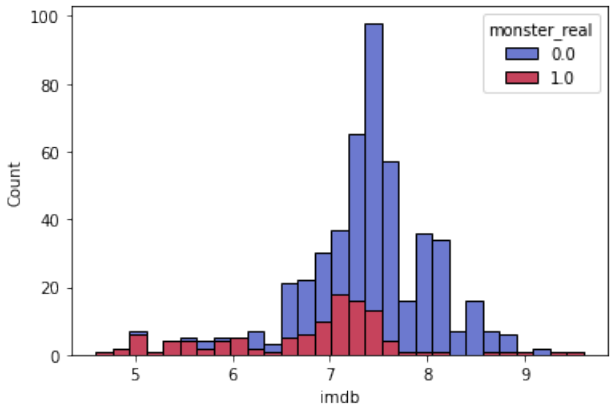

FIGURE 17 – Proportion d'existence de monstres en fonction du score IMDB

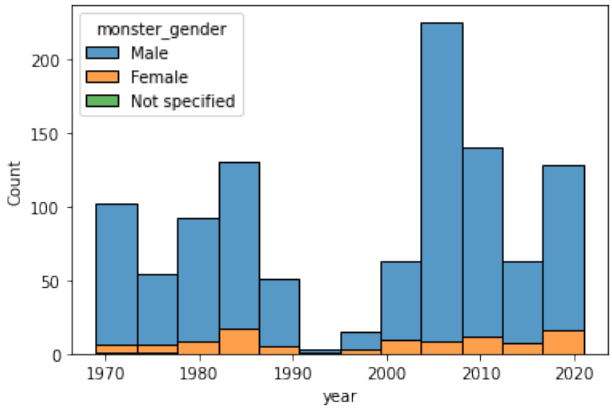

FIGURE 18 – Proportion du genre des monstres en fonction des années

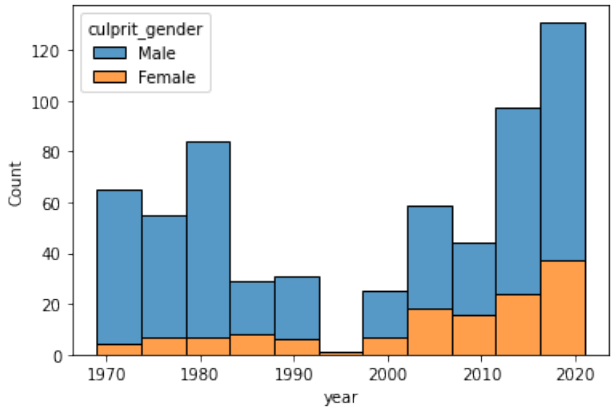

FIGURE 19 – Proportion du genre des coupables en fonction des années

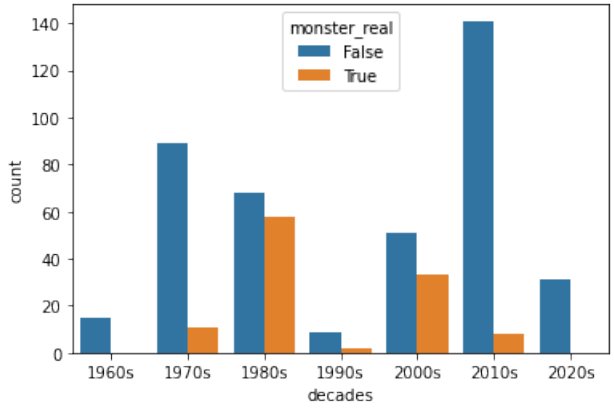

FIGURE 20 – Proportion du nombre de monstres par décennie

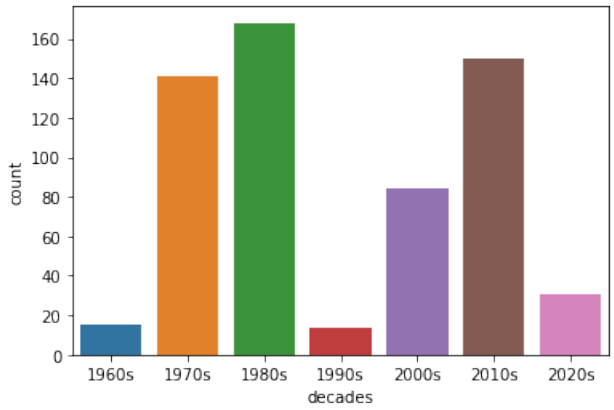

FIGURE 21 – Nombre d'épisodes par décennie

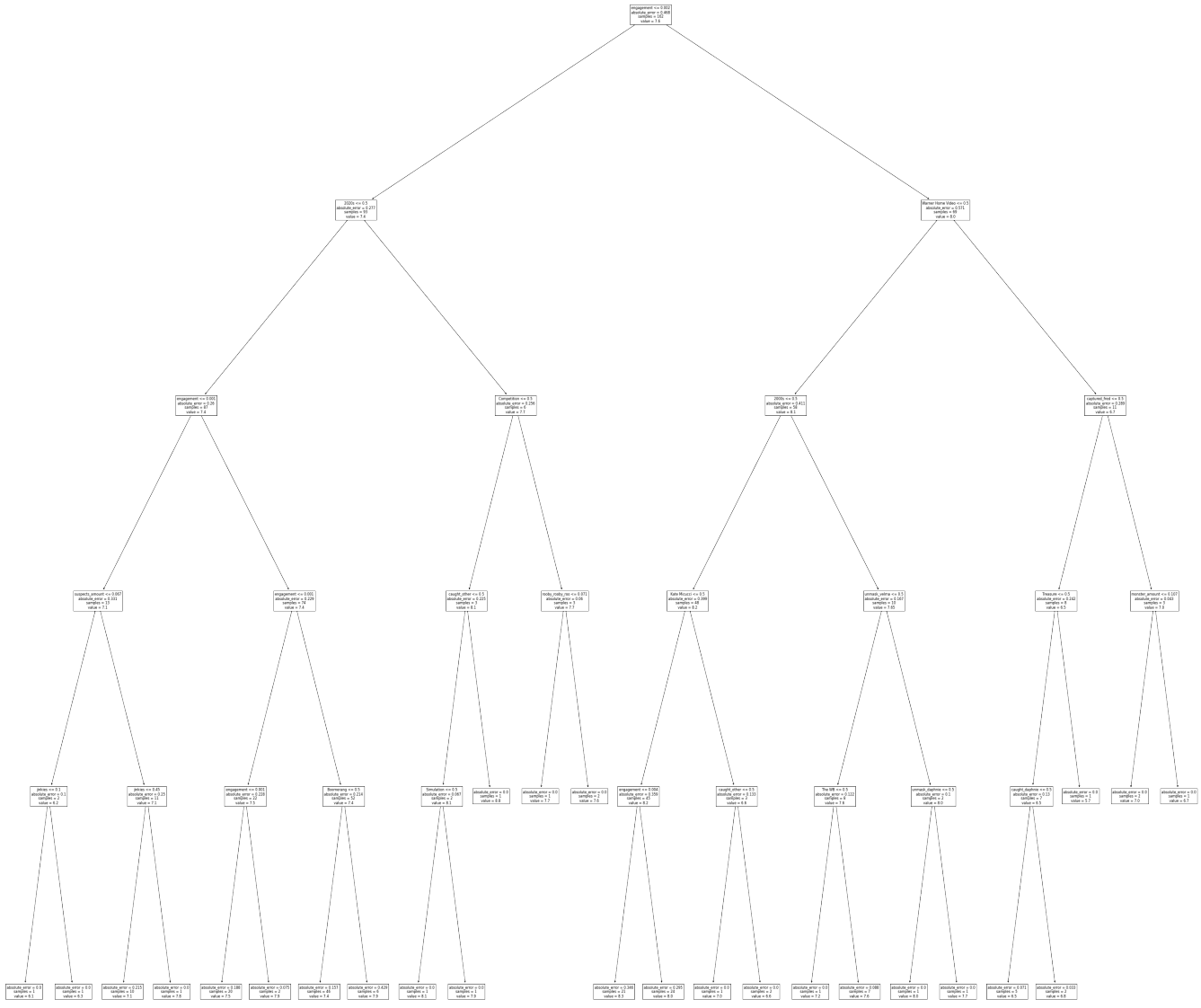

FIGURE 22 – Exemple d'arbre élagué pour le problème de régression (IMDB)