# SY09 Project Report: Scooby-Doo episodes and movies

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

Léonard Penverne and Baptiste

Viera June 8, 2022

## 1 Introduction

This report reports on the work done in the SY09 project. The goal of this project is to apply the data analysis techniques seen during the SY09 course to one of the tidy tues- day datasets. We have chosen the dataset gathering the data on the episodes of the series "Scooby-Doo". This dataset, built by an Internet user and first published on Kaggle, gathers the description of the Scooby-Doo episodes and movies watched by the author through 63 variables. Some of these variables are quantitative while others are categorical. As the basic dataset is incomplete and not very usable as it is, we have made many modifications to it (detailed in the Data Preparation section) before applying the different learning methods seen throughout the semester. Our study of the dataset consists of two distinct problems: a classification problem and a regression problem. For the classification problem we looked at the "monster-real" column and tried to classify the episodes according to whether the monster(s) encountered by the gang were only humans in disguise (when monster-real is set to false) or whether they were actual supernatural creatures (when monster-real is set to true). For the regression problem we tried to predict the variable "IMDB" from the data: this quantitative variable describes the imdb score (a site specialized in movies and television) obtained by the movie or episode in question.

## 2 Data presentation

Our data are the observations of 74 variables for 603 different Scooby-doo episodes or movies, the dataset also contains episodes of related series where Scooby-doo characters make prominent appearances. Concerning the 74 variables, they are necessarily varied. However, we can classify them into several categories. Firstly, the de-

Firstly, data about the episode in general (duration, release date, channel, season, number of views...). Secondly, data concerning the description of the monsters of the episode (number, gender, species, sub-species, if the monster is in fact a human in disguise...) and of the suspects and culprits (number, gender, motivations...). Third, variables that describe the actions of the "gang" (which member catches and unmasks the monsters, which member gets captured, the efficiency of the traps...). Lastly, variables that describe more singular aspects of the episode (presence of bat- man, scrappy-doo, names of the dubbers, which "catch phrases" were uttered...). An important part of the variables (especially those which are not included in the general description of the episode) are boolean or categorical, this implies choices of data reformatting which we will detail later. The number of variables being important, we concentrated, for the exploratory analysis of the data, on the variables that we tried to predict and classify later, but also, in a subjective way, on variables that seemed interesting to visualize.

## 3 Exploratory data analysis

### 3.1 Evolution of the IMDB score according to different parameters

Through this graph, we wanted to identify the evolution of IMDB scores as well as some criteria that make the success of a Scooby-Doo episode.

Finally, it seems that Scooby-Doo remains timeless: indeed the quality of the episodes which was falling slowly but surely since 1970 has experienced a rebound in the 2010s. Moreover, it seems that the public prefers that the monster is not real, as can be seen with the episodes of the late 2000s.

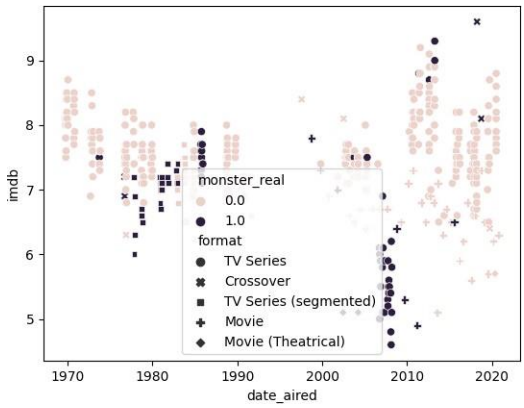


Figure 1 - Evolution of IMDB over the years

## 3.2 The action of the protagonists

We wanted to find out which of the protagonists were catching and unmasking the most monsters and which were being captured the most.

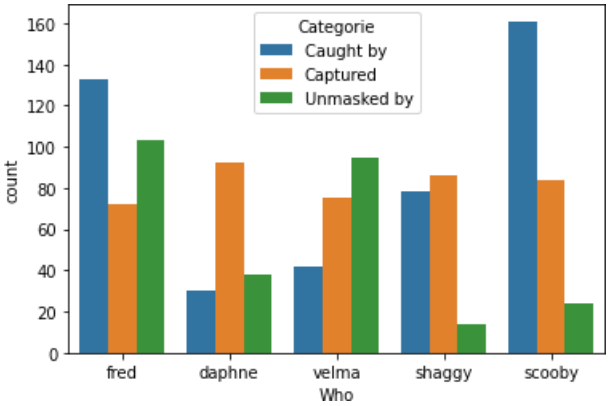


Figure 2 - Proportion of protagonists in each category

## 3.3 Motivations of the monsters

We wished to determine the main motivations of the monsters/criminals on all the episodes broadcast since 1969 and then by decade.

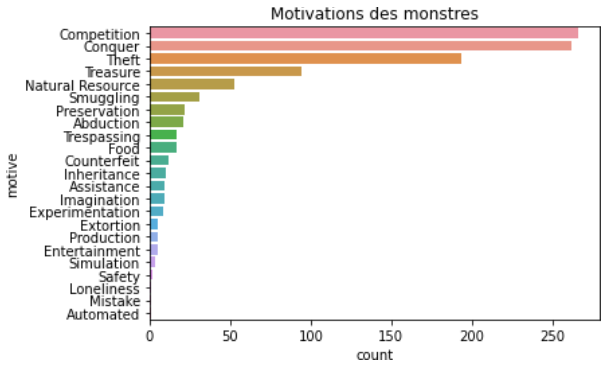


Figure 3 - Motivations of the monsters

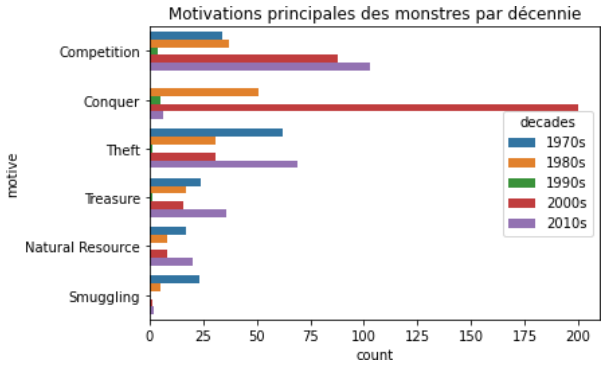


Figure 4 - Motivations of monsters by decade (top 6)

# 4 Non-parametric methods

## 4.1 Data preparation

As mentioned earlier, our data had to be cleaned before it could be used. We therefore solved two separate problems to have usable data: missing data and categorical data.

As for the missing data, we could not simply delete the rows with undeclared variables. Indeed, by following this method we would have kept only one third of our original data, thus significantly modifying the dataset. We therefore chose to replace the missing data by the average of the data in the column according to the channel of publication and the season of the episode. This method works for the quantitative variable but is problematic for Boolean or categorical variables. In the case of categorical variables, they will be excluded from the analysis if they are not filled in. For the Boolean variables we have chosen to

Coding a missing observation by the value most present in its season is an important assumption of our analysis, but it allows us to exploit a maximum of data.

In addition to this replacement, some columns were ignored for the dataset analysis for several different reasons. First, columns containing different information for each episode: the name of the monsters, the title of the episode, the name of the culprit and the lines spoken by the culprit once arrested ("if it wasn't for..." and "and that..."). Secondly, the variables containing too many missing values and that could not be replaced by the mean method: the doubles of each character. Finally, we decided not to keep the variable indicating the gender of the monster, because it was poorly informed and highly correlated with the real existence of the monster. In order to apply the methods seen in class, we had to clean our data.

Finally, in order to use the categorical variables as a whole, we used one-hot encoding. As there were no order relations in the categorical data, we thought it wise to make this choice. This choice will have two major consequences that we will see later: on the one hand, the creation of many new columns and, on the other, the creation of collinear variables.

Finally we end up with a dataset of 487 individuals and 786 variables, which we have centered and reduced for all non-boolean variables. This dataset will be modified later as explained for each problem.

## 4.2 Correlation matrix

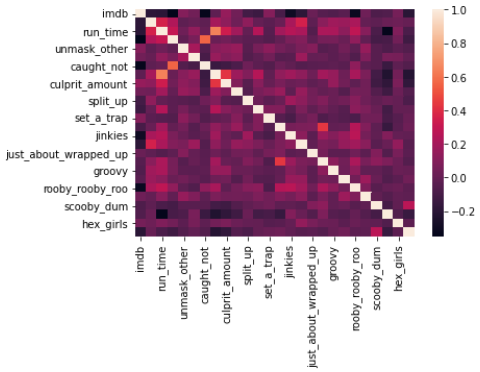


Figure 5 - Correlation matrix of our dataset

We see that in our dataset few variables are correlated between them. Unfortunately, this matrix does not include the Boolean data of the dataset which reduces its usefulness.

## 4.3 ACP

Performing a PCA on our data does not seem obvious: indeed Boolean data cannot be used in a PCA. However, due to the choice of one-hot encoding and the Boolean nature of several variables, our data are not usable for a PCA. This will have consequences when visualizing our decision regions, we will not be able to project our data on the main axes as seen in the tutorial.

Nevertheless, we wanted to see if a PCA using only quantitative variables was interesting. This PCA was performed with centered and reduced data

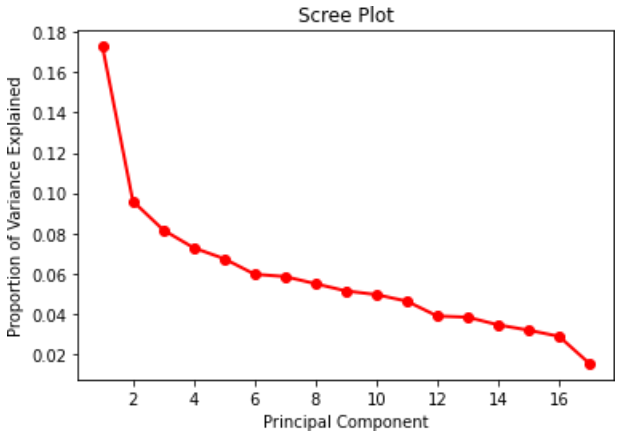


Figure 6 - PCA on quantitative data Unfortunately, once again the result is not exploitable. For our analysis, a PCA would have been very interesting if it would have allowed us to significantly reduce the dimensionality of our data. However, this is not the case here: to keep 90% of the explained variance, we must keep 14 axes. The same is true if we wanted to use the elbow rule: if we keep only the first 11 axes, we lose 20% of the explained variance. In view of these results (and of the variables used in the construction of the trees, which we will detail later) we decided to use the original data set and not the transformed one.

# 5 Classification problem

We have therefore chosen to test the different classification methods seen in class on the column

"monster real". First we performed our analysis with all the variables. Then we
decided to choose the variables most used by
Bagging to retest the methods seen in class. The second option presents better results, which we will present and comment in this report. After the one-hot encoding the 16 variables selected thanks to the Bagging are encoded on 203 and 504 lines.

Table 1 - Description of variables used for classification

| Name | Type | Description |
|---|---|---|
| monster-real | bool | Monster real or not |
| commitment | float | Number o f reviews on IMDB |
| set-a-trap | float | Number of traps made by the gang |
| run-time | float | Episode length |
| jeepers | float | Number of times jeepers' was said |
| split-up | float | Number of times the gang splits up |
| imdb | float | Imdb score of the epi sode |
| series-name | string | Name of the season |
| setting-terrain | string | Landscape of t h e epi- sode |
| network | string | Channel broadcasti ng the episode |
| monster-gender | bool | Type of monster |
| arrested | bool | Guilty arrested or no |
| monster-type | string | Type of monster |
| motivates | string | Motivation of the me- singing |
| number-of-snacks | string | Number of scooby- snacks eaten |
| format | string | Episode or movie |
| caught-shaggy | bool | If shaggy catches the monster |

Our classification is therefore based on a slightly unbalanced column. However, this imbalance does not seem to us to be sufficiently important, we will use the methods seen in class without preprocessing to rebalance the weights of the classes.

For the rest of the classification problem, we will consider, except if the opposite is mentioned, that the validation of the results was done thanks to a 10-fold cross vali- dation with the functions (sometimes modified) seen in TD

Table 2 - Proportion of real or not monsters in the cleaned dataset

| Value of the column | Number of observations |
|---|---|
| True | 401 |
| False | 103 |

The k-fold cross-validation allows us to test our dataset on 10 distinct sets and thus to reduce the risks of bias and overfitting. Moreover, due to the small number of data at our disposal we do not think that introducing a k greater than 10 is desirable: our classes being already slightly unbalanced we do not want to run the risk of having a model testing on a set too small.

## 5.1 K nearest neighbors

The K-nearest neighbor method will serve mainly as a comparison, a kind of benchmark for our results. The application of this method must be balanced by several factors specific to our data that decrease its accuracy. First, the nature of our data is once again an embarrassment for the application of the method. Indeed, with our data both boolean and quantile, the basic distances proposed by sklearn are not adapted. We have therefore chosen to compare the KNN method using 2 different metrics: the Manhattan distance and the Hamming distance:

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

Secondly, the plague of dimensionality is very present in our data: with 207 columns we can afford to have doubts about the proximity between the different neighbors when calculating distances.

For this algorithm, we first looked for the optimal number of neighbors to use, for this we tested the algorithm with the 2 distance measures from 1 to 100 neighbors, then we chose the number of neighbors maximizing the accuracy on an average of 10 iterations with the same number of neighbors. Finally, we found that the optimal number of neighbors was 5 for the Euclidean distance and 3 for the Hamming distance.

Finally, it seems that the two distances produce similar results, which is surely explained by our mixed data (both categorical and quantile). This does not allow one of the two dis-

Table 3 - KNN classifier results

| Distance used | Average Hamming |
|---|---|
| score | 0.920 |
| Euclid | 0.916 |

to emerge as significantly better than the other.

We tried to visualize our data, unfortunately, as mentioned before, visualization is tricky. As we could not project our data on 2 axes of explained variance and as we have more than 2 variables, we had to make a choice. We therefore used the 'imdb' and 'episode duration' axes to project our variables and the values that were predicted by the classifier. This visualization is not an aggregation of the average classes predicted during the 10-fold cross validation but a simple indication of an iteration of the 3 nearest neighbors algorithm with the 70/30 split data set.

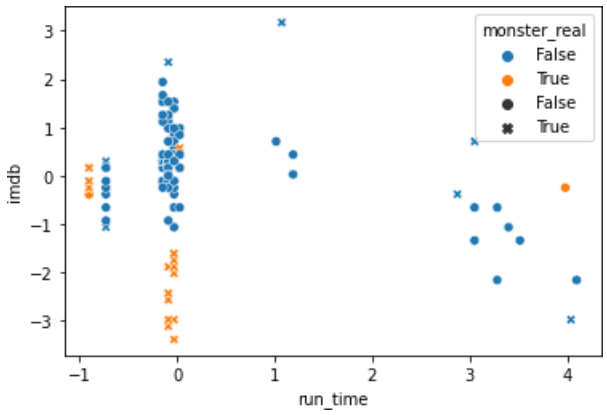


Figure 7 - Results of the KNN classification.

On the graph the color corresponds to the predicted value and the shape corresponds to the real value. The blue dots are therefore non-real monsters predicted as such, the yellow crosses are real monsters predicted as such, the blue crosses are real monsters predicted as false and the yellow dots are irreal monsters predicted as real.

## 5.2 LDA, QDA and Naive Bayes

In the same way as for the nearest neighbors we used the 10-fold cross validation for this part. After having established an element of comparison with the KNN methods, we wanted to see which of the classifiers seen in the course was the best performing for our data set. We therefore looked at discriminant analysis methods. The results of the 3 methods can be seen in Table 2.

Table 4 - Results of the discriminant analysis classifiers

| Method used | Average score |
|---|---|
| LDA | 0.920 |
| QDA | 0.662 |
| NB | 0.638 |

We thus notice a big difference between the methods and in particular an underperformance of the QDA and the naive Bayesian classifier. We have formulated several hypotheses regarding this result.

The underperformance of these classifiers can be explained by two aspects. First, the strong assumption of normality of the data is strongly violated by our Boolean data. Second, because of our categorical data, the estimation of the variance ma- terials is not very accurate: there are not enough individuals to estimate the parameters correctly. Assuming the variance matrix to be diagonal does not solve this problem and the Bayesian classifier also underperforms. This result was to be expected, as we explained earlier the variables are very poorly correlated and the co-variance matrix estimated for the QDA must therefore be nearly diagonal in basis. The performance of the LDA questioned us, with a large dimension we expected this classifier to suffer but this is obviously not the case, it also seems not to be affected by the non-normality of the data.

In the appendix, you will find the visualization of the prediction of the classifiers for the same test-train, split. Unfortunately we could not add the different decision boundaries because of the problems already mentioned for the KNNs. These representations are adapted from the following code.

## 5.3 Logistic Regression

Logistic regression is for the moment the most appropriate method to predict our data: after a 10-fold cross validation we find ourselves with an estimated accuracy of 0.942.

## 5.4 Trees

Before looking at the tree methods we wanted to explore our dataset by means of

of trees. First, we wanted to know, in view of the application of random forests and bootstrap aggregating, which measure of node impurity was most suitable for our dataset. To do this we explored the performance of trees with a depth of 1 to 50 depending on the decision criterion. The figure is available in the Appendix.

This figure made us understand that the decision criterion used as well as the depth of the tree to be used are unfortunately not presettable. Moreover, we could not set up a satisfactory algo- rithm for the post-growth pruning of our trees. We therefore turned to a post-growth elucidation of the tree. Finally, we derived from this step our parameters for the ar- borescent methods: we will use pre-pruning with the Gini index as criterion.

You will find in the appendix an example of a tree built with the gini index and a maximum of 10 leaves for more readability.

## 5.5 Tree methods

The last methods we implemented for classification are randoms forests and bag- ging. As previously mentioned, we used the Gini index as a decision criterion.

Table 5 - Results of the tree classifiers

| Method | Average score |
|---|---|
| Bagging | 0.924 |
| Random Forest | 0.926 |

The results of these classifiers are comparable to those obtained so far. This time, it seems that our data are suitable for these methods. This is not surprising, at each node the decision can easily be made, having Boolean variables for the most part it is enough to set the decision threshold to 0.5 to have an easy partition of the remaining data.

As mentioned before we used these methods with all the basic dataset to choose the variables we were going to use in our analysis. To do this we used the variables that gave us the best accuracy in a Bagging procedure. The tree algorithm being gluttonous, this has a particular interest because our variable to predict is boolean and we do not have to perform ANOVA tests on each column to estimate the dependencies.

## 5.6 Conclusion on classification

Finally, after exploring several classification methods, we believe that the main explanation for the differences in performance remains the particular shape of our data. As we said, the one-hot encoding used to exploit our categorical variables created a multitude of columns. This high dimensionality and the fact that our data were mostly binary favored some methods (tree methods) and disadvantaged others (discriminant analysis methods). Looking more closely at our results, we are also surprised by the performance of the KNN classifiers. Indeed, given the high dimensionality of our data, we did not expect a performance of more than 90%, and once again we wonder how close the 3 nearest neighbors really are to the point whose class is to be determined.

Finally, given the similar performance of our classifiers, we wanted to estimate how the disequilibrium of the class to be predicted affected the different classifiers.

| | | Predicted | |
|---|---|---|---|
| | | True | False |
| Actual | True | 87 | 16 |
| | False | 13 | 388 |

Table 6 - Confusion matrix of the logistic regression

| | | Predicted | |
|---|---|---|---|
| | | True | False |
| Actual | True | 78 | 25 |
| | False | 11 | 390 |

Table 7 - Confusion matrix of the random forest Here

again, the logistic regression stands out from the others. While the other methods produce a confu- sion matrix similar to that of the random forest, logistic regression seems to be the best performing method for classifying the True of the column. Thus it is the only method to have a true positive rate of nearly 85% while the other methods hover around 75%. Thus, in addition to being the best performing classifier, logistic regression is the one that is the least impacted by class imbalance. This makes it the best classifier for our problem.

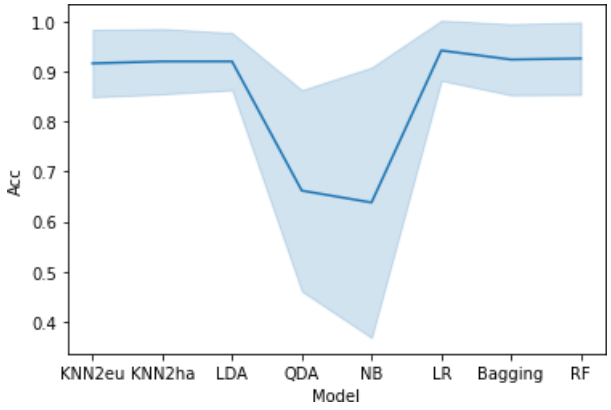


Figure 8 - Performance of the different classification methods

# 6 Regression problem

First, after basing our IMDB score prediction on the dataset with processing, we obtain the following scores:

Table 8 - Results of regression methods

| Method used | Average of the |
|---|---|
| scores Linear Regression | -0.15 |
| Random Forest | 0.50 |

In Sckit-learn, linear regression scores use the R2 score. A negative R2 means that the model did not fit our data well. Since R2 compares the fit of the model with the fit of the null hypothesis, i.e. a horizontal straight line, R2 is negative when the model fits worse than a straight line.
horizontal. In other words, a negative R2 means
that the mean of our test sample would be a more accurate prediction than that given by our model. Part of the reason for this is the scourge of dimensionality (we have more variables than we have subjects). Several methods are thus available to us. We have used a penalized model (lasso) which allows us to solve the problem of multicollinearity between the variables.
riable. By normalizing our data and applying cross-validation, we obtain with the Lasso method a score of 0.44 with a MAE and MSE respectively of 0.34 and 0.24.

Concerning the Random Forest model, we have detected overlearning. Indeed, the score obtained from our training set is 0.93 while that of our test set is 0.50. One of the solutions is to find the optimal hyperparameters. using GridSearchCV. Although this reduces overlearning, the score obtained of 0.51 is not convincing. We then proceeded to a selection of variables using the RFECV (recursive feature elimination cross validation) method which allows us to obtain a score of 0.52 with an MPAE of 4.2% with 215 selected variables. Another plausible explanation could be related to the data cleaning we did, generating noise that leads to bad predictions on the IMDB score.

After having tried other preprocessings which were inconclusive, although debatable, we decided to work with the basic data set. For fear of creating noise, we did not replace the missing values, we added them. We first made a manual selection of the variables. We created visualizations of our data in order to determine which variables have an influence on the IMDB score. For the quantitative variables, we studied the correlations and then used a significance test to determine the dependence of the variables. For the qualitative variables, we visualized the distributions and then performed ANOVA to determine a possible dependency of our variables on the IMDB score. This method is obviously very debatable since, among other things, the assumptions made are strong and the link between the explanatory variables is not observed. We then obtain a number of 44 variables which, transformed into one-hot, correspond to 114 variables for a total of 366 lines. Note that among these variables, variables not obtained with the dependency tests have been added in order to keep the existing links between the explanatory variables.

Table 9 - Results of regression models with variable selection

| Method used | Average score |
|---|---|
| Linear Regression | 0.38 |
| LassoCV | 0.67 |
| Random Forest | 0.72 |

Depending on the models, after applying cross-validation and variable selection to improve performance and minimize overlearning, we obtain the above scores.

Concerning the search for hyperparameters for the Random Forest, the number of optimal estimators is then fixed at 100, the criterion used is the absolute error and the maximum depth of the tree is equal to 5. See figure 22 in the appendix.

Table 10 - Errors of regression models with variable selection

| Method used | MAE | MPAE | IEM | 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 |

Below is the scatterplot (episodes) of predicted IMDB scores with an RF based on actual scores:

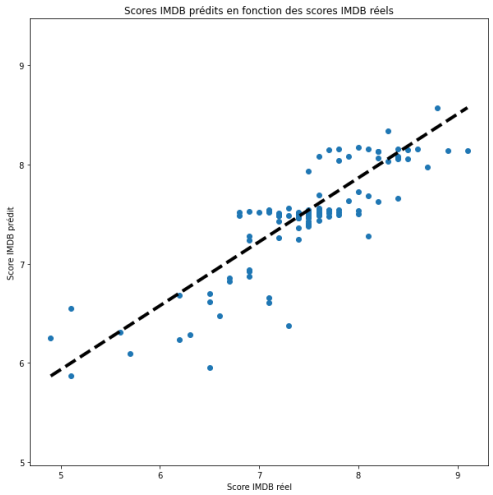


Figure 9 - Predicted IMDB vs. actual IMDB

Finally, it seems interesting to visualize the importance of the variables in our random forest model.

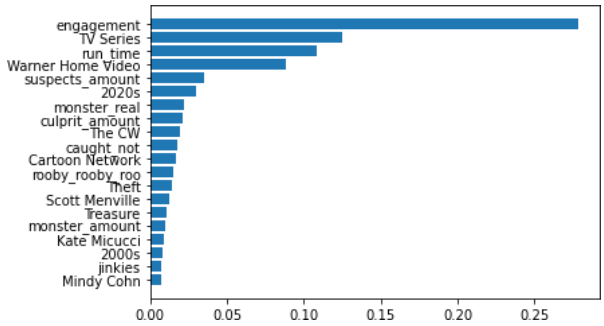


Figure 10 - Importance of variables

By analyzing the sign of the correlations of the explanatory variables with the variable to be explained, we can
then issue, among others, the following conclusions:

- The lower the number of reviews, the better the IMDB score.
- The better the TV Series format, the better the IMDB score.
- The lower the number of suspects, the better the IMDB score.
- The shorter the duration, the better the score IMDB.
- If the work is released by Warner Home video, the IMDB score will tend to be lower.
- The lower the number of culprits, the better the IMDB score
- The presence of a fake monster improves the IMDB score
- If the work is broadcast by Cartoon Network, the IMDB score will tend to be higher.
- The fewer times "Rooby.rooby.roo" is said, the better the IMDB score

## 6.1 Conclusion on regression

We therefore observed that multiple linear regression was not adapted to our model, mainly because of the large dimensions and the multicolumnarity between the variables. A penalized model such as the Lasso allows us to overcome this problem. Tree methods such as Random Forest are a good choice. With the latter method, we were also able to determine which variables played an important role in the IMDB score. It is worth mentioning that there are potential improvements that could be made. On the one hand, we could have explored other methods to replace missing values such as fitting a regression or classification model. On the other hand, we could have tried to test other penalized models such as the Elastic Net or other tree structure methods such as Gradient boosting even if we would have been too far away from the notions seen in SY09.

# 7 Appendices

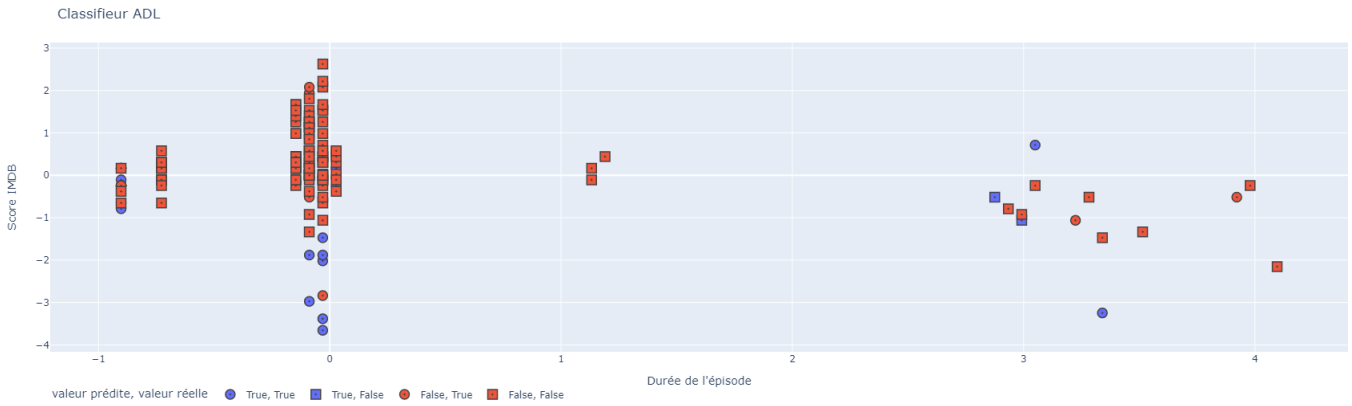


Figure 11 - ADL Classifier

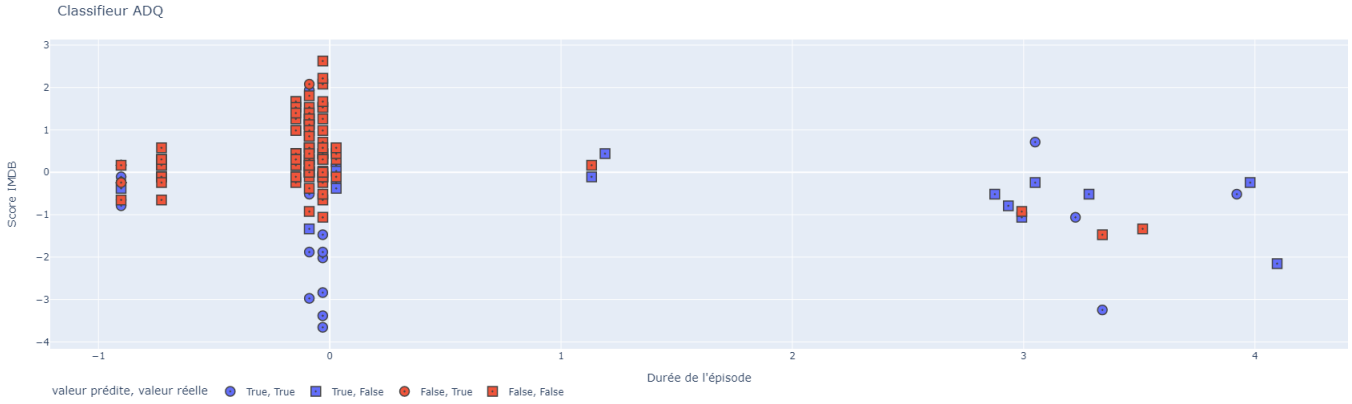


Figure 12 - ADQ Classifier

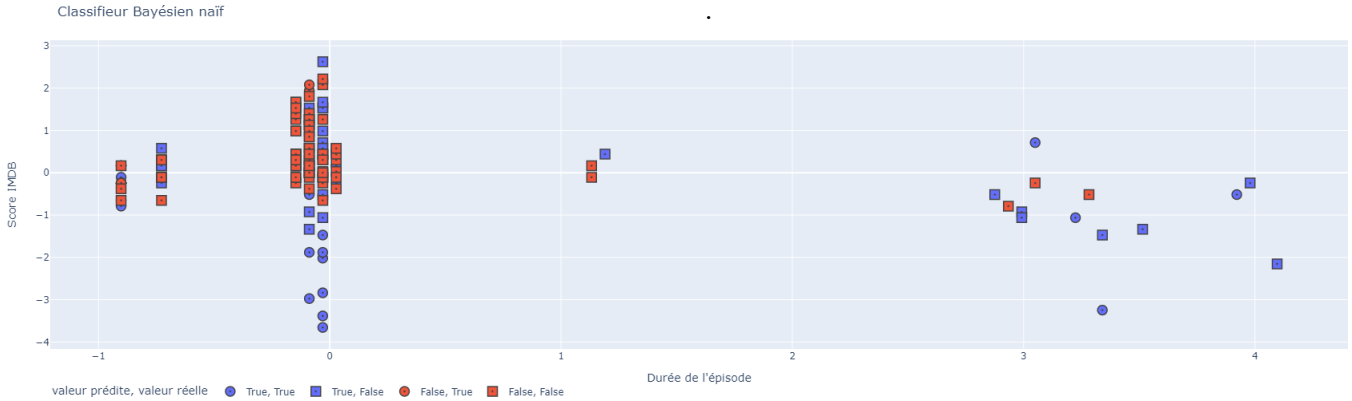


Figure 13 - Naive Bayesian Classifier

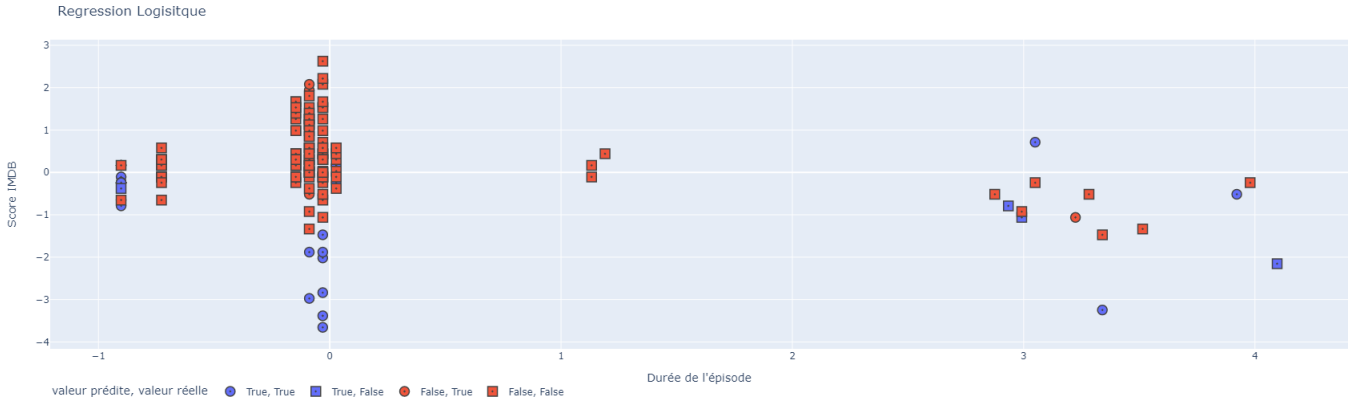


Figure 14 - Logistic regression

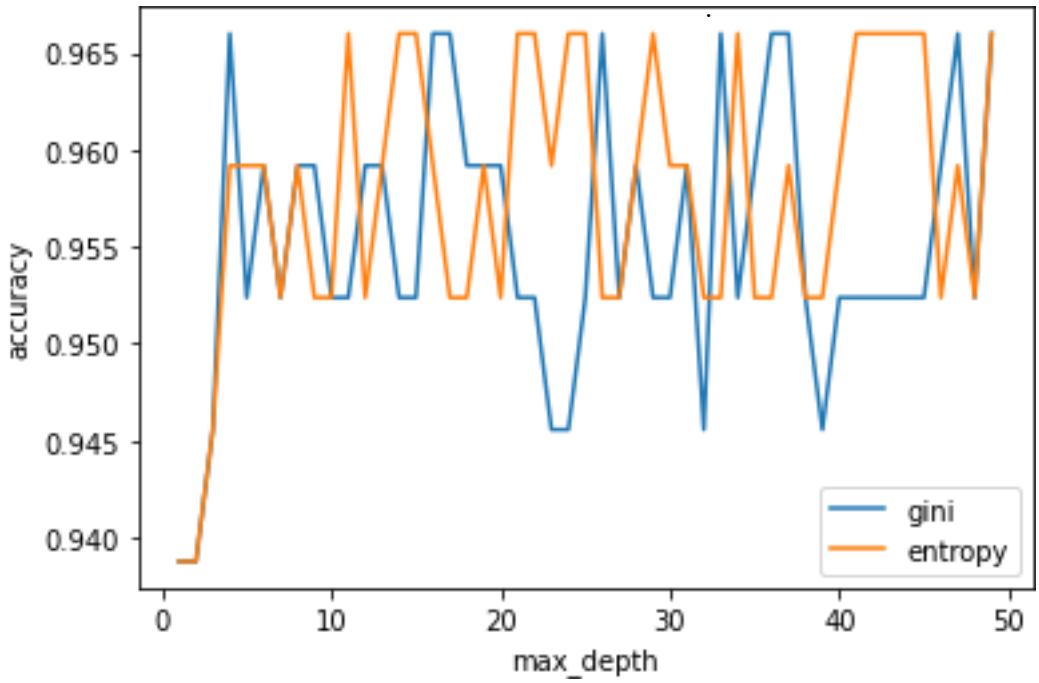


Figure 15 - Gini Index vs. Entropy Comparison

.

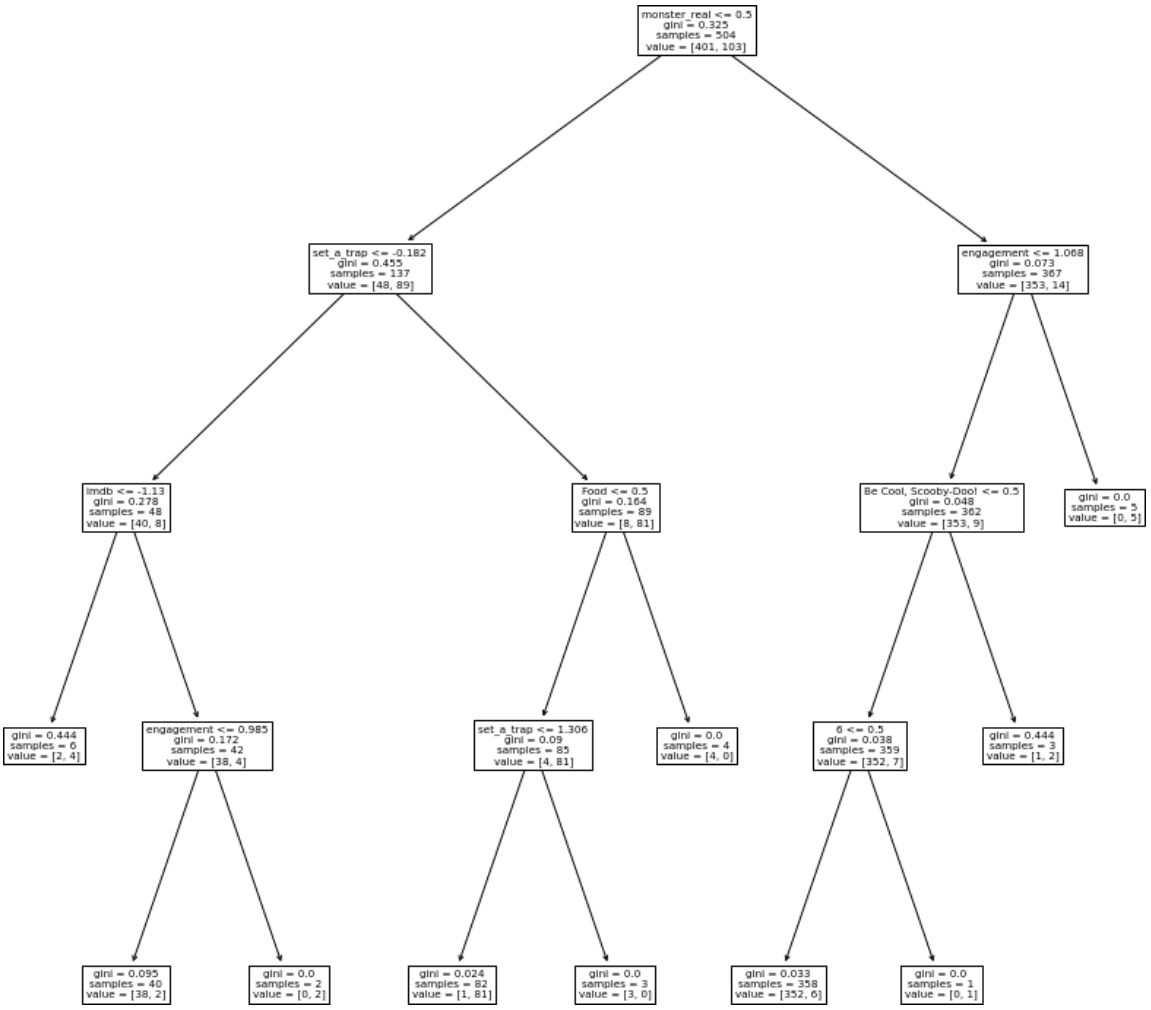


Figure 16 - Example of a pruned tree for the classification problem (Existence of the monster)

.

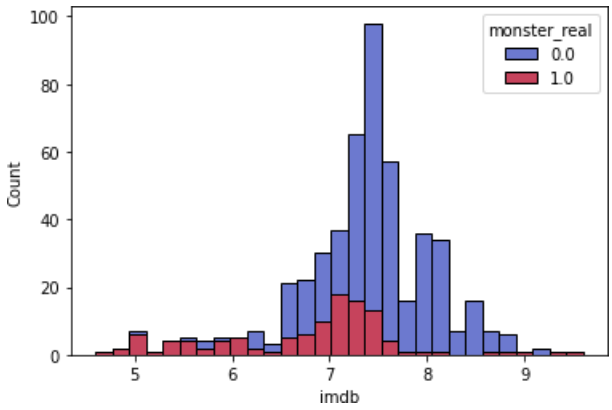


Figure 17 - Proportion of existence of monsters according to IMDB score

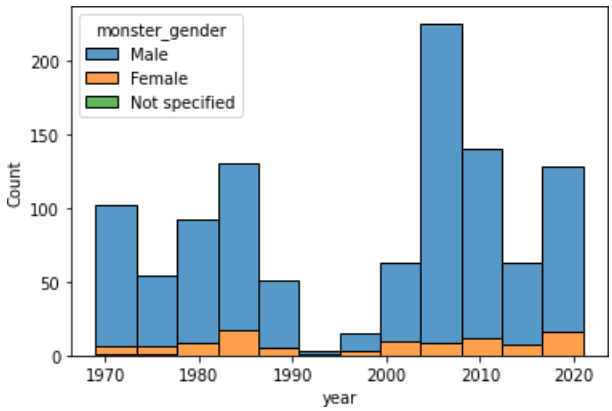


Figure 18 - Proportion of monster gender by year

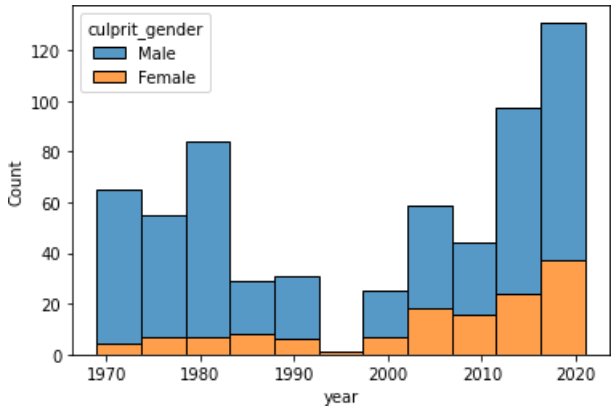


Figure 19 - Proportion of gender of perpetrators by year

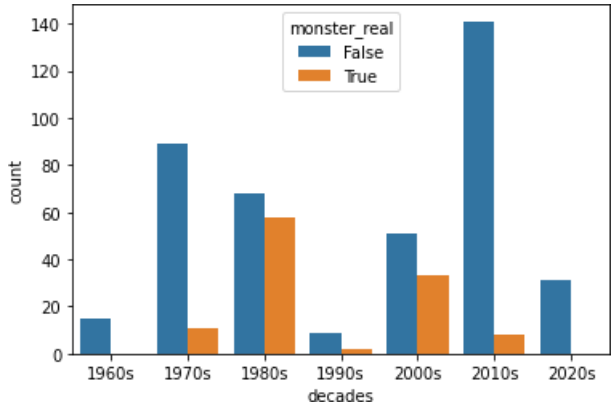


Figure 20 - Proportion of monsters per decade

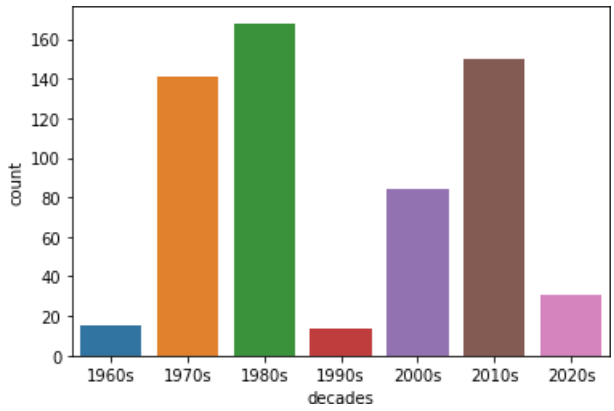


Figure 21 - Number of episodes per decade

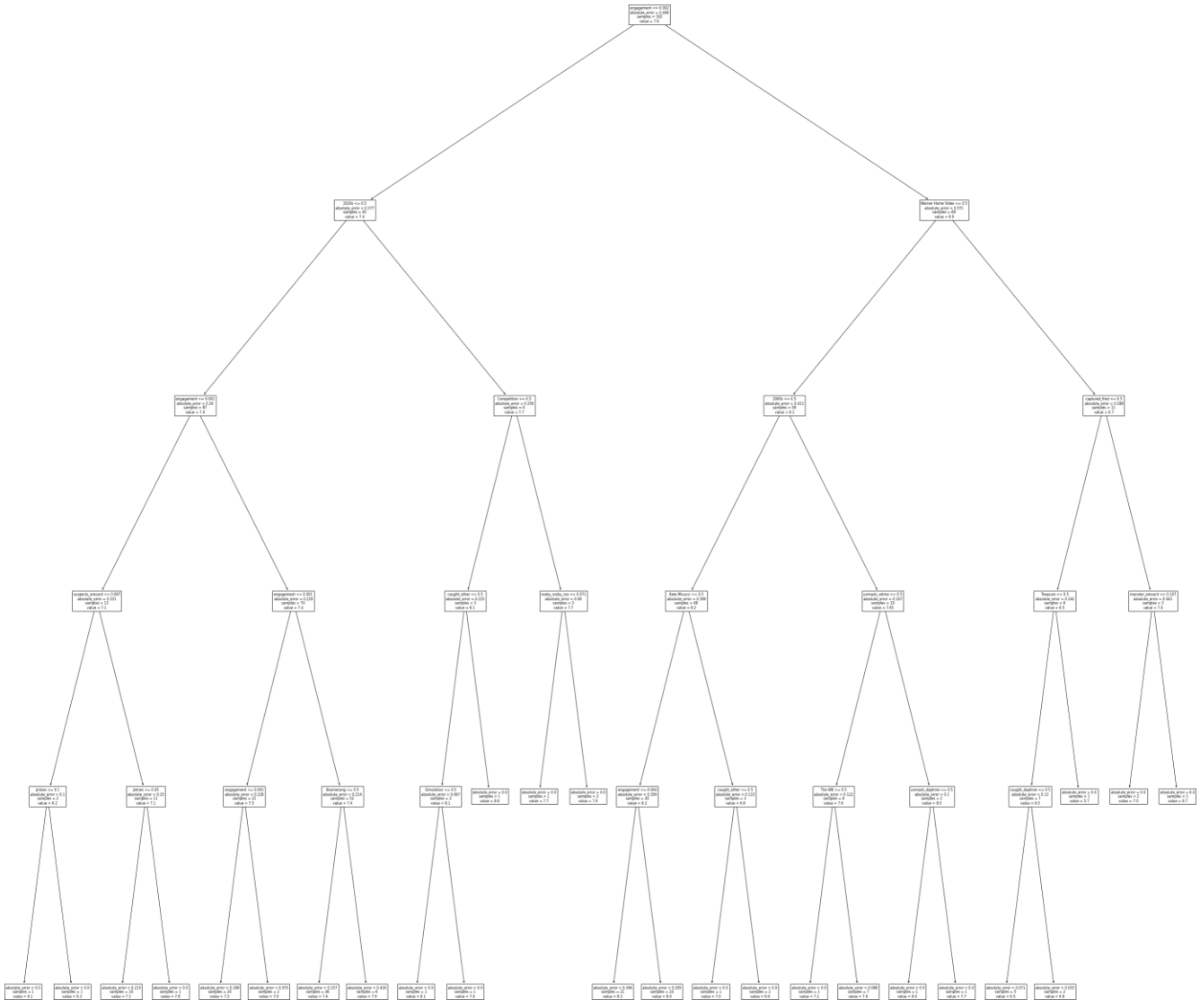

Figure 22 - Example of a pruned tree for the regression problem (IMDB)