POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Project Report

BoardGameGeek data analysis with Nonparametric Methods

Nonparametric Statistics

Authors: Davide Carrara and Francesco Romeo

Academic year: 2021-2022

---

1. Dataset Introduction

1.1. Introduction

The data for this analysis is an expanded version of the one provided by the weekly updated project *R TidyTuesday*: the complete dataset is available here.
The dataset corresponds to part of the database of the website *BoardGameGeek*, which is considered to be the most complete and recognized data source in the area of Board Games. *BoardGameGeek* is particularly relevant in the boardgame community since it provides a leader board of all the available games and its evaluation of game quality is usually trustworthy.
Specifically, the provided dataset, updated in January 2022, contains information regard 21630 games with a mixture of text data (mainly description of the games, user reviews, and general comments), numerical quantities (user ratings, information about players and more) and categorical variables.

1.2. Main Features

Here are reported some of the main features of the dataset, along with their meaning and possible uses.

- `id`, `name`: identification for each game
- `Image`: cover image of the game that could be used for classification purposes
- `Description`: text description of the game
- `Year`: the year of publication of the game
- `minplayers`, `maxplayers`, `suggested num players`: information about the number of players
- `minplaytime`, `maxplaytime`: information about the ideal playing time of the game
- `minage`: minimum required age for the game
- `category`, `mechanic`, `family`, `expansion`: categorical variables representing the topic of the game, its main playing mechanics and whether or not belongs to a particular family or expansion. Every game can (and usually does) have multiple categories and mechanics
- `designer`, `artist`, `publisher`: information about the people behind the game publication
- `average`, `numratings`: average rating of each game on *BoardGameGeek*, along with the number of ratings
- `averageweight`, `numweights`: complexity score assigned by users to the game, along with their numerosity
- `owned`, `wishing`, `trading`, `wanting`: information related to the internal marketplace of *BoardGameGeek*

A second dataset contains all the reviews of the website, that could be used for sentiment analysis.

2. Data Exploration - Feature Engineering

2.1. Removing useless features and cleaning

As a first approach, we removed some of the features that would not be useful for our model or that we would not have time to effectively exploit. We thus removed `family`, `publisher` and `artist`, since they presented very few common instances, and also chose to ignore `image` and the reviews dataset, as we would have needed to use a completely different approach to treat them.
We also deleted clearly flawed games, having `minage` and `playingtime` equal to 0 or negative. In the same way, we chose to not consider games published before 1900, as we imagine their dynamics to be very different from more recent ones, and they do not correspond to the contemporary idea of boardgames.

2.2. Feature Extraction

2.2.1 Category and Mechanic

We extracted the multiple mechanics and categories of each game into a dummy matrix, considering the fact that most games are characterized by more than one category (such as *Political, Fantasy, Children*) or mechanic (such as *Card drafting, Dice Rolling ...*). Only categories with more than 300 appearences and mechanics with more than 600 instances were kept, to improve robustness.

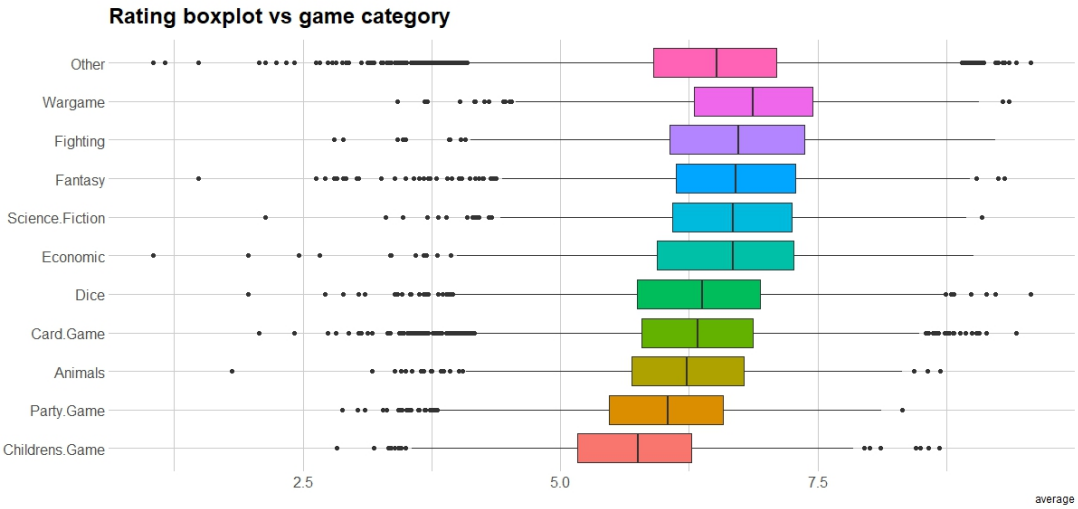


Figure 1: Boxplot of the 10 most common categories

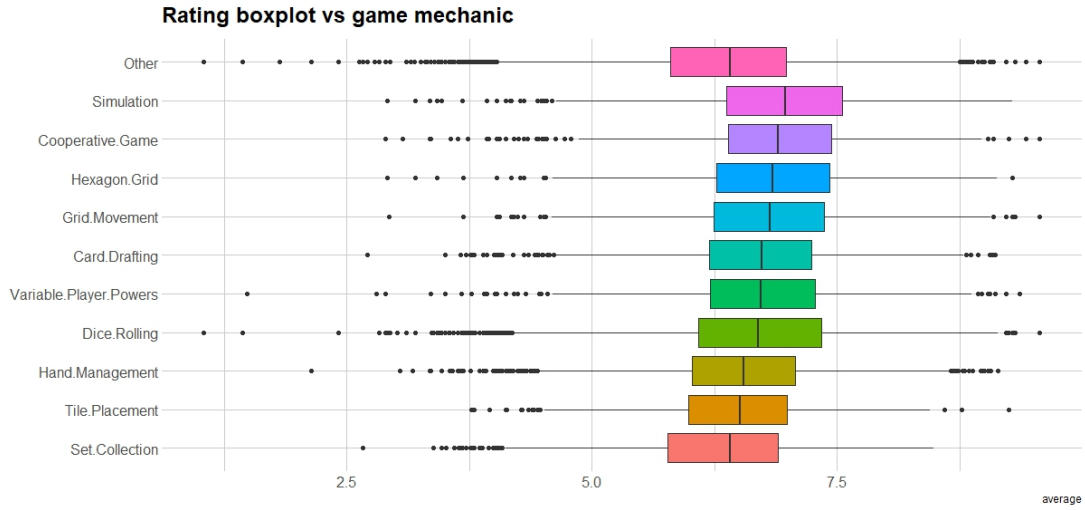


Figure 2: Boxplot of the 10 most common mechanics

As we can see from Figure 1 and 2, categories seem to have a significant effect on the average ratings. The effect of mechanics seems smaller but probably worth of some investigation. However, even after reducing the numerosity, we still have 39 possible values for `category` and 21 for `mechanic`. We will thus need another reduction technique.

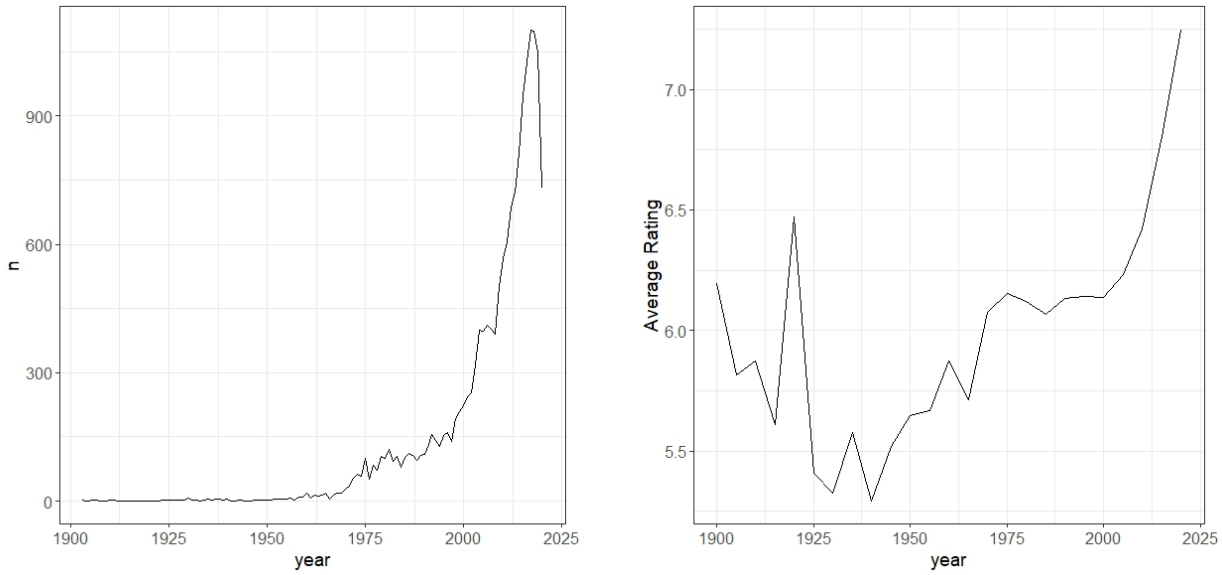
2.2.2 Suggested Number of Players

The `suggested number of players` column is expressed as a list of evaluation from users. For each possible number of players we have an indication of how many users considered it the best possible one, an acceptable one, or one which is not adapt for the game. We chose to build a single indicator keeping only the number of players with the most votes, considering the 'best' indication as a double vote.

2.3. Exploration

2.3.1 Evolution over time

As we can see from Figures 3a and 3b, time has a significant impact both on the game publication and their average rating. In particular, excluding the first part of the 20^{th} century, when the game production is quantitatively negligible, we can see a clear upward trend for the average and the number of games produced. The downward trend that can be seen in more recent years is probably due to data not being consolidated on the website and to a smaller game production during the pandemic period.

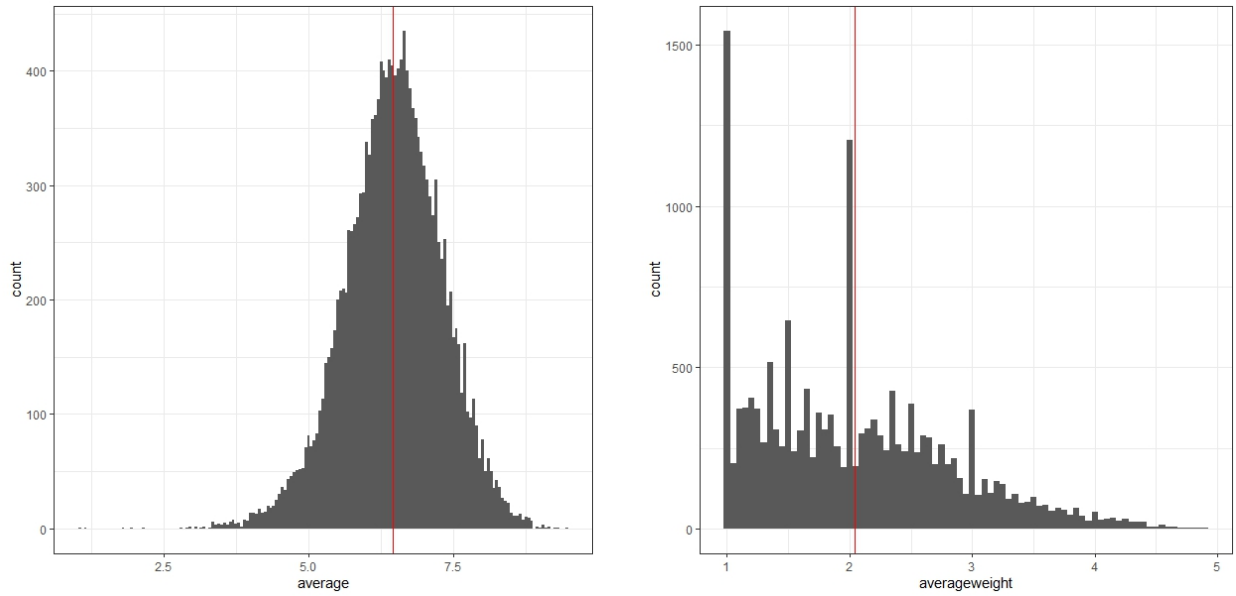


(a) Number of games over time (b) Average rating over time

Figure 3: Trend over time

2.3.2 Ratings and Complexity

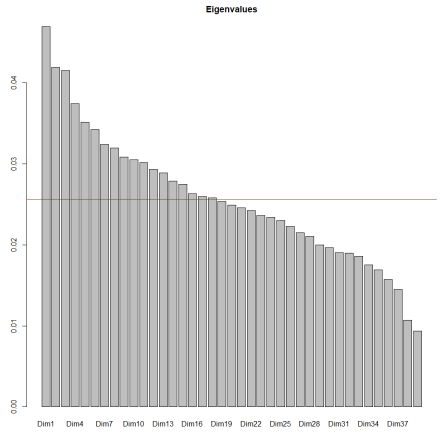
The following plots 4a and 4b represent the ratings given by the users and their evaluation of the game complexity. The distribution of the games ratings seems almost symmetric around a mean value of 6.46 on a 1-10 scale. On the other hand the complexity presents a much more skewed distribution, with mean value of 2.04 on a 1-5 scale.



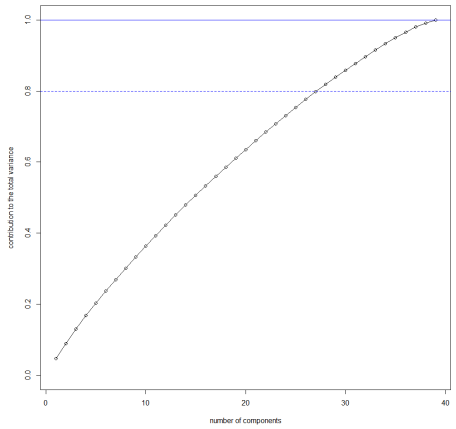
(a) Histogram of Users rating (b) Histogram of Games Complexity

2.3.3 Multiple Correspondence Analysis

A first step towards dimensionality reduction is Multiple Correspondence Analysis, that can be seen as the categorical correspondent of Principal Components Analysis, where the data matrix is substitued by the one of cooccurrences. Its implementation is described in [3] and is contained in the package `factoMineR`.
The result of MCA is comparable to the one of PCA in the interpretation, and the explained percentage of variance for each component is reported in the following figures.



(a) Percentage of explained variance for each direction, red line is the average



(b) Cumulative percentage of explained variance for each direction

A evident from Figure 5b, this approach is not particularly effective, as we would need to keep almost 80% of the dimensions in order to explain 80% of the variance, while losing interpretability. We will return to the problem of dimensionality reduction in the following section 3.

3. Robust Analysis

In order to have a more robust and effective analysis, we wanted to recognize outliers among our dataset. However, the different nature of our features requires different robust approaches, that will be explained in the following paragraphs.

3.1. Coupled Biased Random Walks

When considering fully categorical data, the literature regarding robust approaches is not as established as in the case of quantitative features. In particular, we would like to identify outliers based on uncommon associations of game categories, such as Childrens games and Warfare or Crime games.
In order to face this problem, we will use the techniques explained in [4] and implemented in the package `cbRw`.

3.1.1 Theoretical Aspects

Couple Biased Random Walk is an unsupervised outlier detection method, that will end up assigning an outlierness score to each data point by considering both intra-feature and inter-feature couplings. In particular, the model uses an iterative algorithm to propagate outlierness between feature values, basing on the assumption that outlying values will have strong couplings with other outlying values, while they will have weaker couplings with noisy values.

The categorical data is firstly mapped into a weighted graph, where each node represent a feature value. Each node is assigned a property value, which is based on the frequency of that node inside the feature and the frequency of the mode m for the same feature. In particular, the intra-feature outlierness of the value ν is given by the following formula:

$$\delta(\nu) = \frac{1}{2}(dev(\nu) + base(m))$$

where $dev(\nu) = \dfrac{p(m) - p(\nu)}{p(m)}$ and $base(m) = 1 - p(m)$. On the other hand the entry of the adjacent matrix for the graph $\mathbf{A}$(u, v) is equal to $p(u|v)$, and represent the weight of the edge. This can be interpreted as the outlierness propagation value: if u has an high outlierness and u,v are strongly coupled then this propagation value is high. Since $p(u|v) = 0$ when u, v belong to the same feature, we can consider A as the inter-feature outlierness matrix.
In order to build an unbiased random walk the transition matrix $\mathbf{W} = \mathbf{A}\mathbf{D}^{-1}$ would be used, where $\mathbf{D}$ corresponds to the diagonal matrix of A. However, a biased version is used with the following transition matrix:

$$\mathbf{W}_b(u, v) = \frac{\delta(\nu)\mathbf{A}(u, v)}{\sum_{v \in V} \delta(\nu)\mathbf{A}(u, v)}$$

The outlierness score of each value is then obtained as $\pi^*(v)$, representing the value of the stationary probability density in v.
The pseudocode algorithm is taken from [4] and reported in Figure 6 below.

Algorithm 1 *Estimate Value Outlierness (X, α)*

Input: X - data objects, α - damping factor
Output: $\boldsymbol{\pi}^*$ - the stationary probability distribution

for $i = 1$ to D **do**
  Compute $p(v)$ for each $v \in dom(f_i)$
  Find the mode of f_i
  Compute $\delta(v)$
  for $j = i + 1$ to D **do**
    Compute $p(u, v), \forall u \in dom(f_j)$
  end for
end for
Generate the matrix $\mathbf{W}_b$
Initialise $\boldsymbol{\pi}^*$ as a uniform distribution
repeat
  $\boldsymbol{\pi}^* \leftarrow (1-\alpha)\frac{1}{|V|}\mathbf{1} + \alpha \mathbf{W}_b^\intercal \boldsymbol{\pi}^*$
until Convergence, i.e., $||\boldsymbol{\pi}_t^* - \boldsymbol{\pi}_{t-1}^*||_\infty \leq 0.001$ or reach the maximum iteration $I_{max} = 100$
return $\boldsymbol{\pi}^*$

Figure 6: CBRW Algorithm

Some minor tweaks are then applied to assure convergence in case of a periodic or reducible graph.
Lastly, the relevance $rel(f) = \sum_{v \in dom(f)} value\_score(v)$ is computed giving higher relevance to features where it is possible to effectively distinguish the outliers.
In conclusion, an outlierness score is computed for each data point, with $score(x) = \sum_{f \in F} w_f \times value\_score(v)$, where $w_f = \frac{rel(f)}{\sum_{f \in F} rel(f)}$. Outlying objects will thus have higher outlierness score and can be treated as necessary.

3.1.2 Application

In the specific case of our dataset, we chose to apply CBRW to the matrix of our categorical features, representing whether a game presented a certain category or was played with a certain mechanic. The matrix is thus composed by 59 features (39 categories and 20 mechanics), with binary values.
We flagged as outliers games belonging to the top 0.5% quantile of the outlierness score distribution: we chose to use a conservative approach since we do not have a clear way of evaluating the overall result of the algorithm in terms of it finding uncommon categorical associations. In general, we notice that many of the games signaled as outliers are games adaptation of videogames or Tv Series (such as Lost: The Game or Warcraft: The Game). Among the others, the outlier present higher percentage of games of Fighting and Miniature categories and with Modular Board and Variable Player Power mechanic.

3.2. Quantitative Variables - MCD approach

We also decided to apply a Robust Approach to some of the quantitative variable of our dataset. In particular, we focused on `Average`, `Averageweight`, `Wanting` and `Owning`: these variables are mostly related to the quality of the game and its market value and can be interested as regression output.

3.2.1 Minimum Covariance Determinant

MCD, or Minimum Covariance Determinant, is one of the most popular joint location-scatter estimator, and is thoroughly explained in [2], with implementation in the R package `RobustBase`. The main logic behind MCD consists in defining a set of data points **H** of dimension h $<$ n such that the determinant of the covariance matrix is minimized.
The algorithm proceeds iteratively, starting from different possible **H** and computing a robust joint estimation of location and scale. At each iteration the furthest observations, considering Mahalanobis Distance from the

robust estimated parameters, are trimmed away and **H** is updated. After convergence (even if there is no theoretical guarantee to reach a global minimum when the FAST version of the algorithm is used) a reweighting step is usually performed, in order to increase efficiency of the algorithm. Outliers are once again identified using Mahalanobis distance with respect to the robust estimation of mean and covariance.

3.2.2 Application

We performed MCD on `Average`, `Averageweight`, `Wanting` and `Owning` with $\alpha = 0.95$.
In order to make this feature comparable we would need to scale them. However, one of our main objective is to estimate the mean and covariance matrix, thus we decided to simply divide each of them by its respective interquantile range.
The result can be observed in the following Figure 7

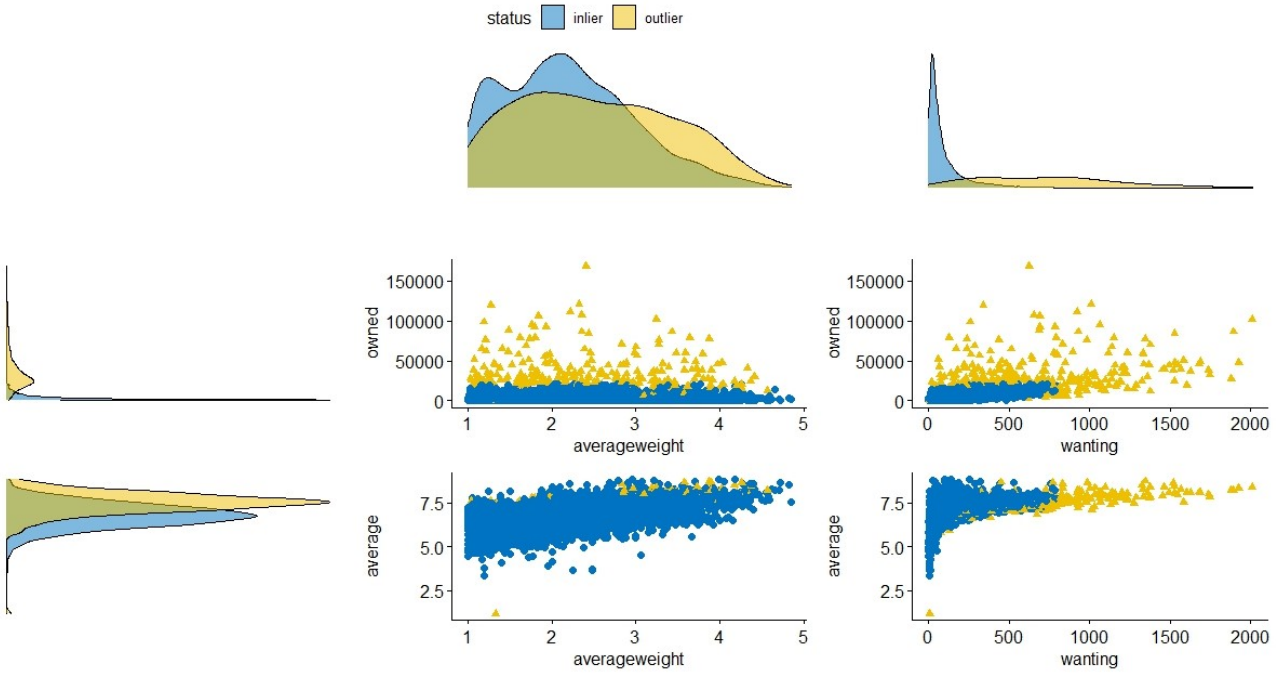


Figure 7: MCD Result

As we can see the result of MCD is mainly influenced by `owned` and `wanting`. This is sensible, since the dataset present some games with extreme values for this two features, while the data is more consistent for `average` and `averageweight`.
Comparing the marginal distributions for outliers and inliers we can observe that the effect on the market features is to remove some part of their long tails. On the other hand the distribution of `averageweight` seems to highlight a bimodal trend, while the distribution of the `average` is even more bell-shaped.

3.3. Clustering

In order to provide an effective and robust dimensionality reduction technique we moved to clustering. The main idea is that by defining a limited number of clusters, containing similar games in terms of category and mechanic, we can avoid using the same category and mechanic covariates in our models.

3.3.1 Rock : Robust Clustering

Our first attempt consisted in applying the technique described in [1] and implemented in the `rockCluster` package. The idea of the algorith, similar in certain terms to the cbRw one, is that using *links* between couples of values would be more effective than considering dissimilarities between data points when merging clusters. However, we could not manage to obtain sensible results, always ending up with extremely unbalanced and non representative clusters.

3.3.2 K-Medoids

For this region we resorted to K-Medoids algorithm. This clustering technique is similar to K-Means, in the sense that a certain number of centroids is randomly generated at the first iteration and is then updated step by step, minimizing a certain cost function. K-Medoids, however, is more robust, as it does not generally rely on euclidean distance but to a more general dissimilarity. Moreover, medoids need to be points belonging to the dataset, while the same is not true for K-means centroids.
When performing our clustering we chose to use the Gower distance, which is able to handle both quantitative and categorical data. Specifically Gower distance between two vectors with p features x_i, x_j can be computed as

$$S(x_i, x_j) = \frac{\sum_{k=1}^{p} s_{ijk}\delta_{ijk}}{\sum_{k=1}^{p} \delta_{ijk}}$$

where δ_{ijk} is just a coefficent saying whether the two variables can be compared on that specific feature, and is generally 1 or 0 in case of NA's.
In the case of full categorical data, we can define the coefficent $s_{ijk} = \mathbb{1}(x_{ik} = x_{jk})$. The Gower distance is then effectively computed as $d_{Gower} = \sqrt{1 - S_{Gower}}$

3.3.3 Application

We decided to apply separately K-medoids to category and mechanic features. At first we ran the clustering with a varying number of medoids form 2 to 8. In order to select the best model we observed the silhouette coefficent, which gives an indication on how much each point is similar to the other points in its cluster and different from the ones belonging to other clusters. As we can see from Figure 8, the correct number of clusters seems to be 6.

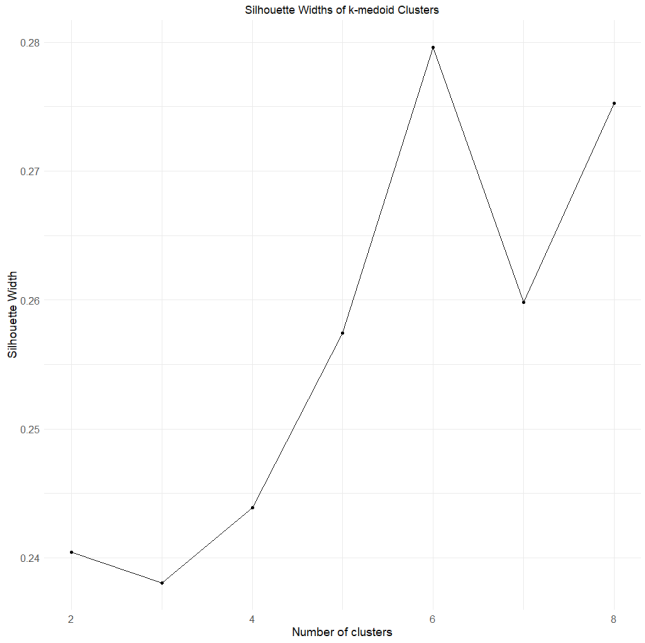


Figure 8: Silhouette coefficient

3.3.4 t-SNE

In order to visualize the effect of our clustering we need to resort to embedding and dimensionality reduction techniques. In particular, we chose to use t-distributed stochastic neighbor embedding, also known as t-SNE, described in [5] and implemented in the R package `Rtsne`. t-SNE is a non-linear dimensionality reduction technique that can be used to model points belonging to a high dimensional data space into a 2D or 3D system. This is obtained by an iterative minimization of the Kullback-Leiber divergence between the probability distribution over pairs of points in high and low dimensions.

In this kind of application and in presence of a significant clustering, we would expect a pattern linked to the clustering to appear in the lower dimensional representation of data.

Figure 9: t-SNE for category clustering

From Figure 9 we can observe that, although non perfect, the clustering structure that we obtained is preserved and observable in the low dimensionality space. A similar result is obtained when clustering on the mechanics.

4. Modelling

The cleaning of the data set through robust analysis and the creation of clusters for categories and mechanics are essential for the models we decided to build. Our point of view, as stated before, is the one of the boardgame producers: which dynamics or categories make a game more popular? How can we predict the rating of a game and its commercial success? We tried to address these and other related questions, whose answers can be important for the company in order to produce a very successful game.
In this section we will discuss the methodologies that we implemented in order to solve these questions, such as permutational anova tests and nonparametric regression techniques. In particular, we will focus on three quantities that can be interesting for the company: the average rating obtained by the game, the average complexity of the game as the users perceive it (column averageweight in the dataset) and the appeal of the game (columns wanting + owned in the dataset). Our aim is to explain these quantities using the characteristics of the game: the suggested number of players, the minimum age required to play the game, the playing time, and so on. These are indeed quantities that the producers can establish during the design of the game itself.
Moreover, it is also interesting to understand how the quantities of interest mentioned above are distributed among the clusters created on the categories and the mechanics. This can help us understand which are the most appreciated categories.

4.1. Permutational Anova Techniques

The first step in the design of a successful game is to understand which **type of game** are most appreciated by the users. We decided to obtain this information by means of permutational anova tests; in particular we investigated how the average rating, the complexity and the appeal of the game vary over the categories and the mechanics of the game.
The procedure that we implemented is the following: first perform a two-way anova using as grouping factors the clusters of the categories and the mechanics, then if we find a significant difference in one of the two groups for one of the three quantities of interest, we investigate this difference by performing an anova multiple ways, in which the i-th factor represents the membership to the i-th cluster. The two-way anova test can be formulated as follows: let X_{ijk} be the $k-th$ statistical unit belonging to $i-th$ cluster on categories and $j-th$ cluster on mechanics, the assumptions of the model are:

$$X_{ijk} = \mu_{ij} + \epsilon_{ijk} \quad \text{with} \quad \mu_{ij} = \mu + \tau_i + \beta_j + \gamma_{ij}$$

and the hypothesis of the test are:

- For the interaction term:

$$H_0 : \gamma_{ij} = 0 \ \ \forall i, j \qquad H_1 : \exists \ \bar{i}, \bar{j} \ \ s.t. \ \ \gamma_{\bar{i},\bar{j}} \neq 0$$

- For the factor 1:

$$H_0 : \tau_i = 0 \ \ \forall i \qquad H_1 : \exists \ \bar{i} \ \ s.t. \ \ \tau_{\bar{i}} \neq 0$$

- For the factor 2:

$$H_0 : \beta_j = 0 \ \ \forall j \qquad H_1 : \exists \ \bar{j} \ \ s.t. \ \ \beta_{\bar{j}} \neq 0$$

This formulation can be generalized also for the Anova multiple ways test that follows the two way anova, with the exception that, for the sake of simplicity and interpretability, we didn't consider the interaction terms.
The anova multiple ways helps us to understand which is the group that really affects the difference in the mean for the quantity of interest by testing the significance of each factor. In this case, since we will perform six tests, we decided to correct the level of significance of the test through a Bonferroni correction, that is, we considered as level of significance 0.1 divided by the number of clusters (6). This allows us to have an overall significance of 0.1 for the test related to the significance of the clusters.
We decided to use the procedure described above because there is no post-hoc analysis for permutational anova tests that could have helped us in understanding which are the clusters that lead to a difference in the mean.
It is worth to mention that in this section we used the Kmedoids clusters for our analysis. However, doing the same tests on the features clusters leads to the same results.

4.1.1 Permutational Anova for the average rating

We start by showing a boxplot of the quantity of interest over the clusters created for the categories of the game and for the mechanics:

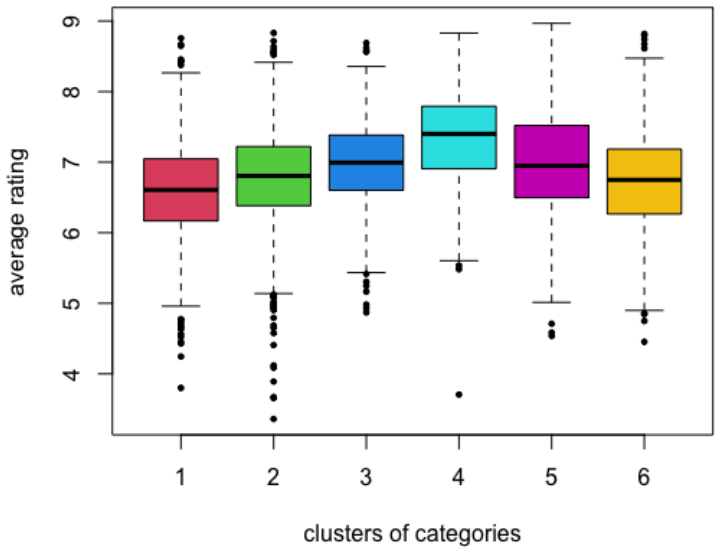


Figure 10: Boxplots of average rating over the clusters of categories

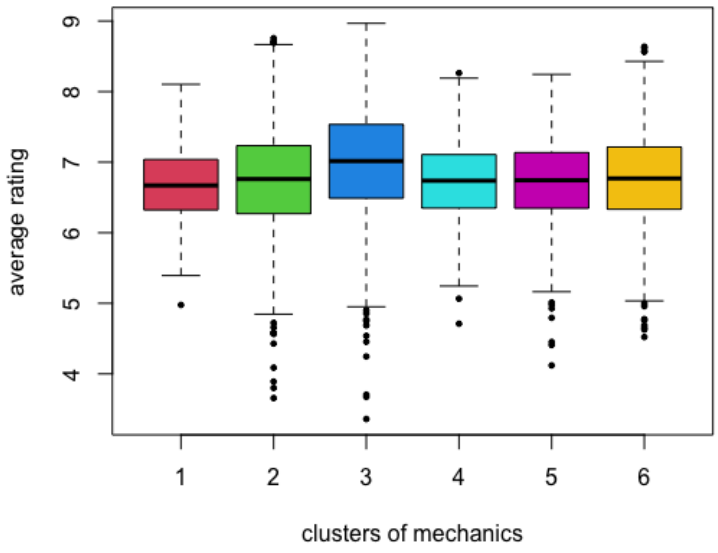


Figure 11: Boxplots of average rating over the clusters of mechanics

There seem to be a difference in the average rating among the clusters for the categories but not among those for the mechanics. Performing a permutational two-way anova we obtained that we have statistical significance to assess that the mean of the average rating differs only on the clusters for the categories. Indeed, for the interaction term we obtained a pvalue of 0.001, but for the term related to the clusters on mechanics the pvalue was equal to 0.82, while for the first factor, the clusters on categories, the pvalue was 0.009. Thus, we considered only the first factor as significant and we proceeded as described above performing an anova multiple ways in which each cluster is a grouping factor.
As a result, we obtained that belonging to fourth cluster increases the mean of the average rating of the game; we also obtained that the third cluster is not significant for the average rating, while the fifth has a coefficients that is close to zero, and the others have negative coefficients. Investigating over the fourth cluster, we noticed that the three most common categories are: War games, World War II, Ancient. These categories are not present in the other clusters, thus we can think that they overall increase the average rating significantly.
To confirm this qualitative result, we performed another anova test in which we distinguished between games that belongs to at least one of the three categories mentioned before. This test corresponds to an Anova one way, that can be described as follows: indicating with X_{ik} the $k-th$ statistical unit belonging to $i-th$ group, the assumptions of the model are:

$$X_{ik} = \mu_i + \epsilon_{ik} \quad \text{with} \quad \mu_i = \mu + \tau_i$$

and the hypothesis of the test are:

$$H_0 : \tau_i = 0 \ \ \forall i \qquad H_1 : \exists \ \bar{i} \ \ s.t. \ \ \tau_{\bar{i}} \neq 0$$

The boxplot related to this test is the following:

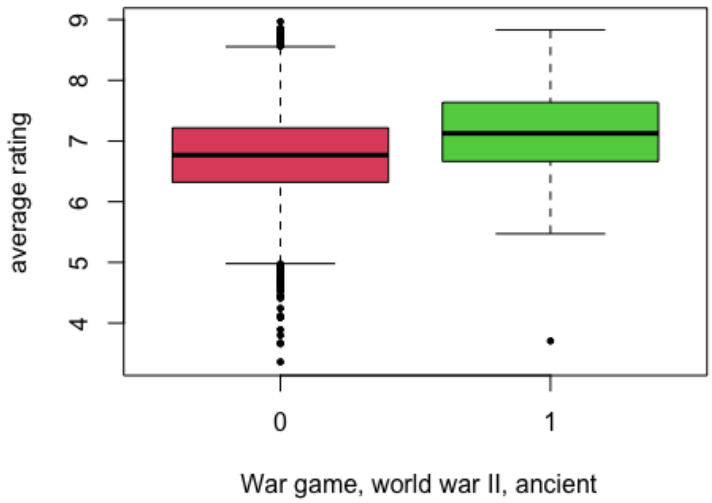


Figure 12: Boxplots of average rating for War games, World War II, Ancient vs other categories

and the permutational anova gives a pvalue equal to zero, thus we can conclude that there is statistical significance to reject H_0.
To conclude, we can say that the categories that leads to an higher average rating are War games, World War II, Ancient.

4.1.2 Permutational Anova for the average complexity

The average complexity of the game perceived by the user is difficult to evaluate for the producers, yet it can be very interesting. This is why we first wanted to see if some games are considered more complex than others only based on the mechanic of the game or on the type of game, and to do so we exploited a permutational anova test. Below we reported the boxplots of the average complexity as it is distributed among the clusters for categories and mechanics:

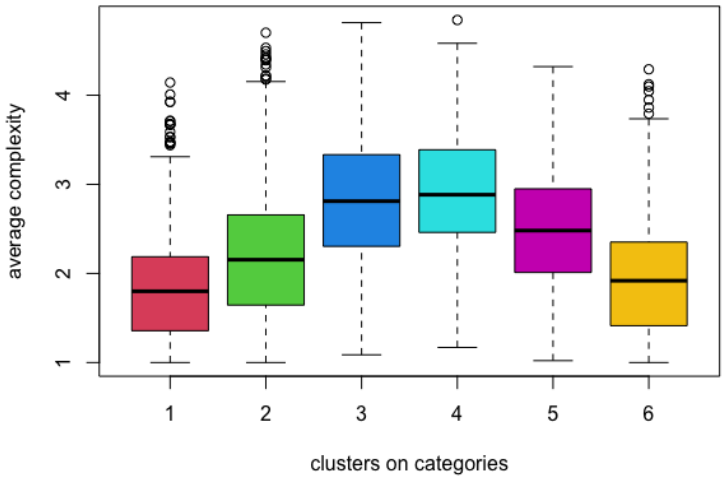


Figure 13: Boxplots of average rating over the clusters of categories

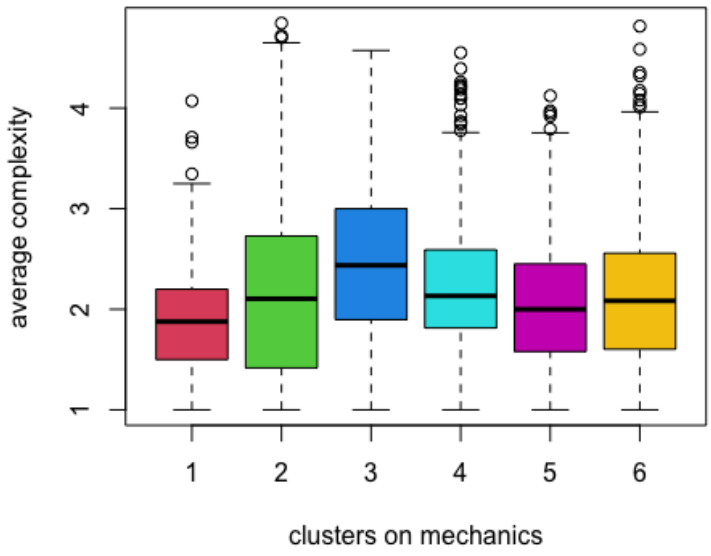


Figure 14: Boxplots of average rating over the clusters of mechanics

As before, performing a permutational two-way anova we obtained that we have statistical significance to assess that the mean of the average rating differs only on the clusters for the categories, which is consistent with the boxplots. In particular, the pvalues for the interaction term, the cluster on categories and the cluster on mechanics are respectively 0.036, 0.005 and 0.89. Thus we can focus on the factor related to the cluster on categories.
Performing an anova multiple ways in which each cluster is a grouping factor, we obtained that belonging to 3rd, 4th or 5th cluster increases the mean of the average complexity of the game. The most common categories in these clusters are:

- Cluster 3: Economic, Card.Game, City.Building
- Cluster 4: War games, World War II, Ancient
- Cluster 5: Fantasy, Fighting, Exploration

The 4th cluster indeed contains categories of games that can be considered complex games, and the same holds for the third cluster. But what is better: to have a more or a less complex game? We will answer later to this question, by building a model that tries to link the average complexity with the average rating.

4.1.3 Permutational Anova for the appeal

The appeal of the games is measured using the information contained in the columns *wanting* and *owned*, which express respectively the number of users of the website that want to buy the game and the number of users that already have it. This can be a significant indicator of how much the game has been successful, thus we would like to explain it in terms of category of the game and mechanic of the game through a permutational anova test.
Below we reported the boxplots of the appeal in the different clusters:

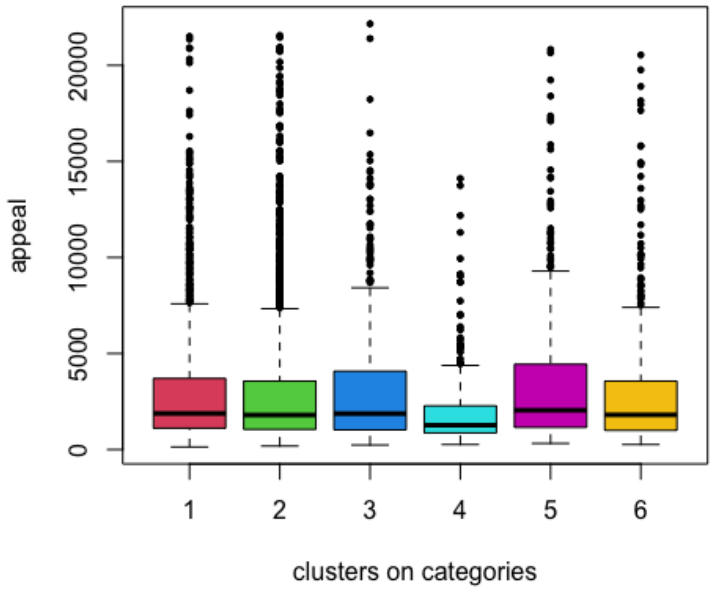


Figure 15: Boxplots of appeal over the clusters of categories

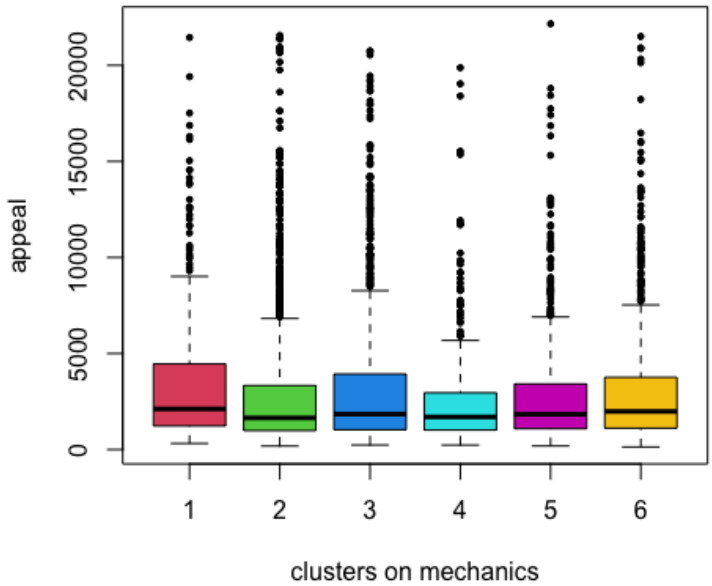


Figure 16: Boxplots of appeal over the clusters of mechanics

This time the clusters doesn't seem to help for distinguishing games with higher or lower appeal. Indeed, from the permutational two way anova test we don't have statistical significance to reject H0 neither for the interaction term nor for the single terms. In particular:

- The pvalue of the test for the significance of the interaction term is 0.20
- The pvalue of the test for the significance of the group corresponding to the clusters on categories is 0.80

- The pvalue of the test for the significance of the group corresponding to the clusters on mechanic is 0.90

So we can say that there is no statistical evidence to assess that the appeal differs on average among these clusters.

4.2. Regression Models

In this section, we will discuss the nonparametric regression techniques that we adopted in order to try to explain either the average rating of the game or the average complexity. The former is important because, if well explained, can help the producers to understand how much the game will be appreciated by the users. The latter is as well fundamental to be explained because it is somehow related to the former, and we will see that usually the more complex the game, the higher the average.
The models that we estimated are the following: weighted local averaging through Epanechnikov kernel, generalized additive models, xgboost. Before seeing the result it is worth to mention some theoretical notions related to these three models.

4.2.1 Introduction to the models

The three models we aim at implementing are completely different from each other. The first one is a nonparametric method for regression: given a focal point x_0, local averaging outputs as $\hat{f}(x_0)$ the weighted average of the points that are inside a neighborhood of the focal point itself. The weights of the points in the neighborhood are decided according to a kernel, and the width of the neighborhood is determined by the kernel and a scaling factor called *bandwith*.
We decided to use the Epanechnikov kernel, whose graphical representation is the following:

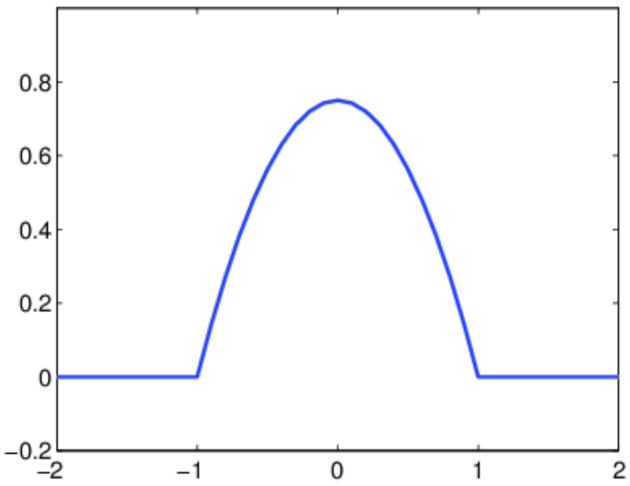


Figure 17: Example of Epanechnikov Kernel

In this plot the values on x axis are the distance of the statistical units from the focal point. As we can see from the plot, the closer the point to the focal point, the higher the weight. Moreover, we can see that the bandwith is not infinite, since this function has a finite support. This means that some points will have zero weight, thus they will not have any importance in the prediction in correspondence of the focal point. Thus, given the kernel function $K(\cdot)$ and the bandwith h, the estimate obtained in correspondence of a focal point x_0 is:

$$\hat{f}(x_0) = \frac{\sum_i^n w_i y_i}{\sum_i^n w_i} \qquad \text{where} \qquad w_i = K\left(\frac{x_i - x_0}{h}\right)$$

y_i are the values associated to the statistical units x_i in the dataframe. This formula holds in the univariate case, which is the framework in which we will use this model. However, we would like to use more complex models, and not only univariate ones.
Nonparametric models suffer from curse of dimensionality, thus it is better to avoid considering multivariate regression, but we can focus on functions of the following type: $f(\boldsymbol{x}) = f(x_1) + f(x_2) + ... + f(x_p)$ where p is the number of covariates we want to use.

This trick allows us to move to the second type of model we would like to use, which are the generalised additive models (GAM). This type of models focuses on functions of the shape indicated above, meaning that we can estimate each univariate model $f(x_i)$ independently and then combine them by summing them. In our case, we will use smoothing splines to estimate the model for a single covariate.

The last model we will estimate is called xgboost, which stands for eXtreme Gradient Boosting. Boosting is an ensemble method, meaning it's a way of combining predictions from several models into one. It does that by taking each predictor sequentially and modelling it based on its predecessor's error (giving more weight to predictors that perform better). The algorithm can be shortly described as follows:

- Fit a first model using the original data
- Fit a second model using the residuals of the first model
- Create a third model using the sum of models 1 and 2

Gradient boosting is a specific type of boosting which minimises the loss function using a gradient descent algorithm.
In particular, xgboost uses decision trees as weak predictors, and it is designed for optimal performance and speed.
For both xgboost models and GAM we tried to investigate how each covariate influences the predictions of the model. For GAM this is straightforward, given the fact that it is an additive model. On the other hand, xgboost is way more complex than GAM, thus to investigate how the covariates contribute for the prediction output we need the so-called SHAP values.

4.2.2 SHAP value

SHAP — which stands for SHapley Additive exPlanations — is the state of the art in Machine Learning explainability. SHAP values are derived from shapley values, which are used in game theory frameworks to quantify the importance of a player based on the contribution that he adds to coalitions that the players may form. In our context, the players are the covariates and the game is the prediction outcome of the model. As a consequence, the SHAP value quantifies the contribution of each feature to the prediction made by the model in correspondence of one observation.
Theoretically, the first step to perform in order to compute the SHAP values is to build the underlying model on any possible subset of features. Then, each of the estimated models should predict the output in correspondence of a statistical unit $\boldsymbol{x}_0$.
Now that the predictions are available, we can start computing the marginal contributions of each feature, which can be defined as follows: suppose our model has three covariates cov_1, cov_2 and cov_3, then the marginal contribution of cov_1 to the model with the other two covariates can be defined as

$$MC_{cov_1,[cov_1,cov_2,cov_3]}(\boldsymbol{x}_0) = \hat{f}_{cov_1,cov_2,cov_3}(\boldsymbol{x}_0) - \hat{f}_{cov_2,cov_3}(\boldsymbol{x}_0)$$

Where $\hat{f}_{cov_i,cov_j}(\boldsymbol{x}_0)$ is the prediction of the model with covariates cov_i, cov_j. In general, the marginal contribution of cov_1 to a specific model is the difference between the prediction that the model with cov_1 gives and the prediction of the model without cov_1, keeping the other covariates fixed. As a consequence, the overall effect of a covariate, that is its SHAP value, is the weighted sum of all the marginal contributions that can be computed for that covariate. For example, in a model with only three covariates, SHAP value of cov_1 is given by:

$$SHAP_{cov_1}(\boldsymbol{x}_0) = w_1 MC_{cov_1,[cov_1]}(\boldsymbol{x}_0) + w_2 MC_{cov_1,[cov_1,cov_2]}(\boldsymbol{x}_0) + w_3 MC_{cov_1,[cov_1,cov_3]}(\boldsymbol{x}_0) + w_4 MC_{cov_1,[cov_1,cov_2,cov_3]}(\boldsymbol{x}_0)$$

The weights are determined following two rules:

1. the sum of the weights of all the marginal contributions to 1-feature-models should equal the sum of the weights of all the marginal contributions to 2-feature-models and so on... In our example, this implies: $w_1 = w_2 + w_3 = w_4$
2. All the weights of marginal contributions to f-feature-models should equal to each other, with f being all the possible numbers of covariates. In our example, this means: $w_2 = w_3$

A formula to compute the weight of the marginal contribution related to the model with only f covariates is:

$$w_f = \left[f \binom{F}{f} \right]^{-1}$$

where F is the total number of covariates.

To wrap up, the SHAP value of a covariate in correspondence of a statistical units $\boldsymbol{x}$ is given by:

$$SHAP_{cov_i}(\boldsymbol{x}) = \sum_{set:cov_i \in set} [|set| \binom{F}{|set|}]^{-1} [Predict_{set}(\boldsymbol{x}) - Predict_{set \setminus cov_i}(\boldsymbol{x})]$$

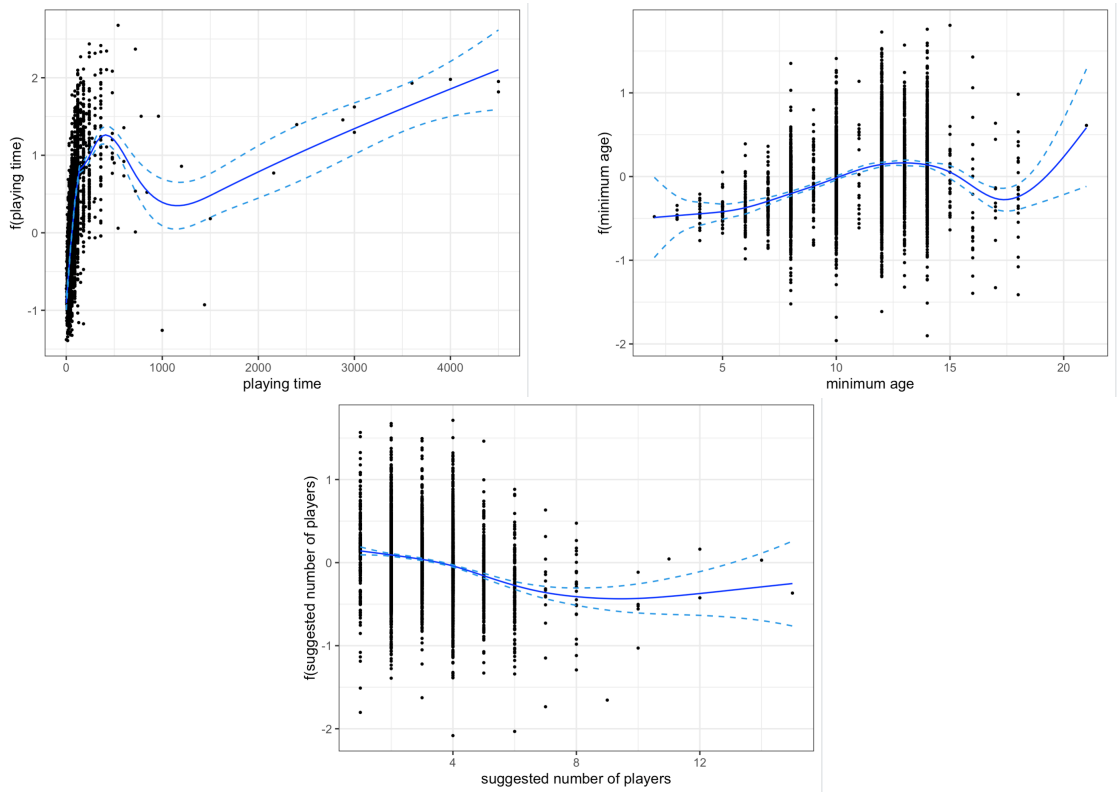
From a practical point of view this procedure is too expensive. We will use other techniques that compute the SHAP value approximately.

4.2.3 Regression models for the average complexity

The first regression model that we built to explain the average complexity uses as covariates the characteristic of the game:

$$y = f_1(suggested\ num\ players) + f_2(minimum\ age) + f_3(playing\ time)$$

where y is the average complexity. To build each univariate regression model f_i we used cubic B-splines. Below we reported a plot of the three smoothing regressors we built:



The resulting R^2_{adj} is equal to 70%, which is a fairly good amount considering that we only used three covariates. However, adding any other type of covariate we are not able to increase significantly the R^2_{adj}, so we decided to stick with this model.
Since we are dealing with a gam model, we can estimate the impact that each feature has on the output considering the value that the corrispondent univariate Bspline model has. We can average this values to quantify the importance of the three covariates and report the values in a bar plot:

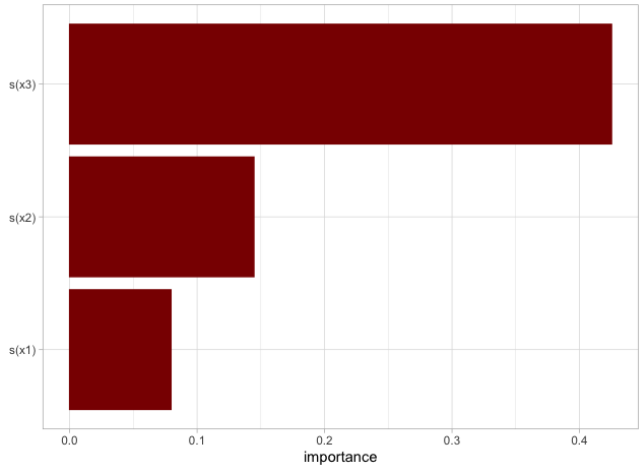


Figure 19: Importance of each feature in GAM

In particular, x_3 is the playing time, x_2 is the minimum age required and x_1 is the suggested number of players. This plot suggests that the playing time is the main feature to determine the complexity of the game. Games with a higher playing time are considered more complex by the users. The minimum required age is the second covariate in order of importance, while the suggested number of players is the least important.
The contributes that each covariate gives in the prediction of each statistical unit, that is the equivalent of the SHAP value for the xgboost algorithm, is displayed in the following plot:

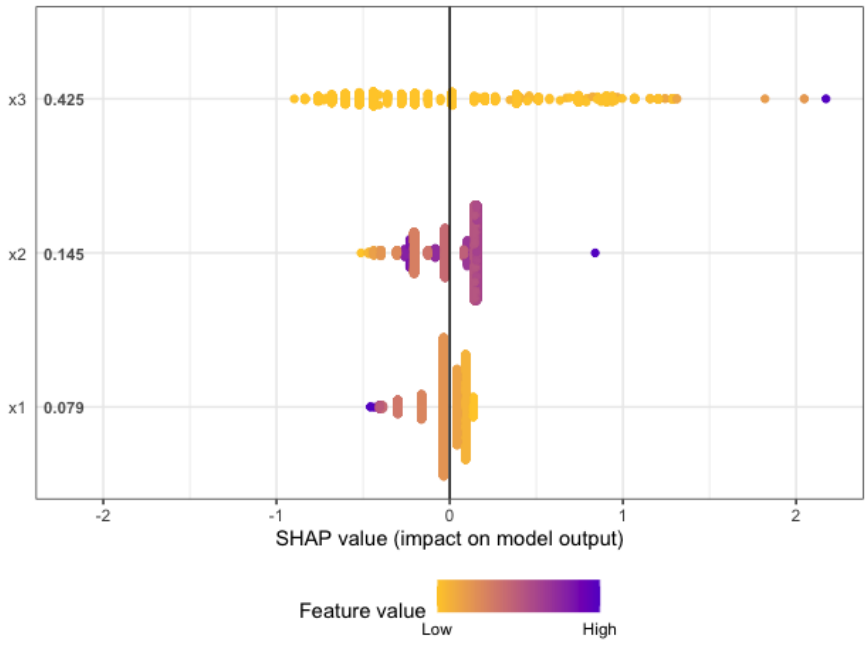


Figure 20: Importance of each feature in GAM

As we can see from this plot, low values of playing time are more likely to give a negative contribution for the complexity, which results in a negative SHAP values. The more the playing time increases, the more the complexity increases, the higher the SHAP value. A similar interpretation can be given for the minimum age required by the game: games with a higher minimum age (violet) usually correspond to larger SHAP value, which means that they are associated with more complex games.
GAM is a simple model that manages to obtain a good result in terms of R^2_{adj}. Anyway, we tried to build a more complex model in the attempt of explaining the average complexity better. Thus, we built an xgboost model using the same covariates. We trained the model using 10 estimators and a maximum depth equal to 6 for each decision tree. The RMSE of the model on the training set is equal to 0.397991. Given the complexity

of the model, we deployed SHAP values to recover interpretability. The results obtained are showed in the following plots:

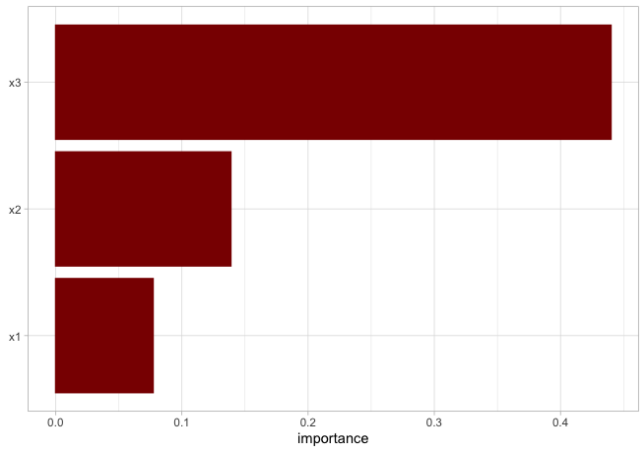


Figure 21: Importance of each feature in XGBOOST

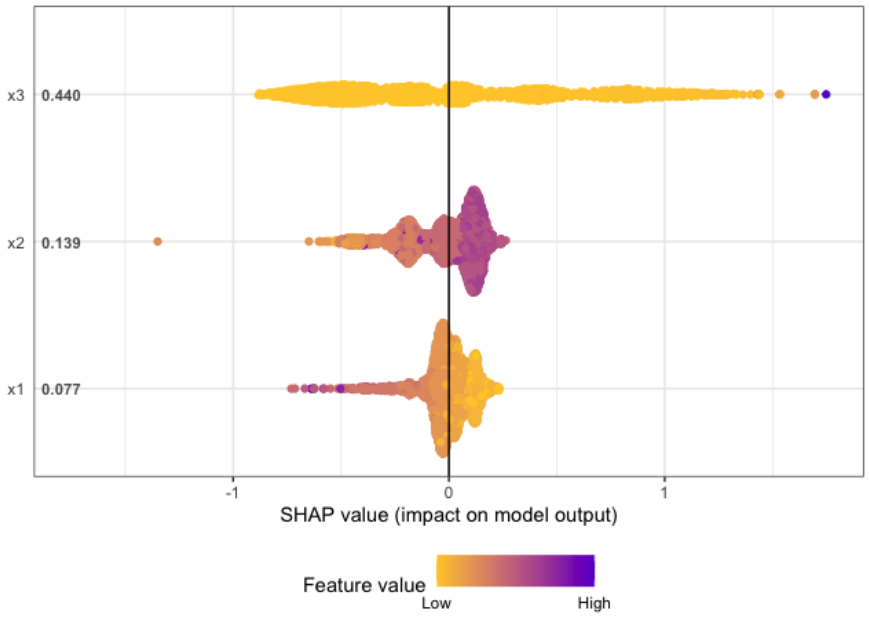


Figure 22: Importance of each feature in XGBOOST

The plots confirm the results seen in the GAM model. The playing time is the most important covariate for the complexity of the game.
Now that we have two models for the average complexity, we compare them by splitting randomly the data set in a training set and a test set and computing the RMSE on the test set. The results are basically the same: 0.41 for xgboost and 0.42 for GAM.
We can thus conclude that using GAM model can be better, since it is a simpler and more interpretable model which gives same result as xgboost, more complex and less interpretable.
As last attempt, we tried to insert in the GAM the clusters the games belongs to. This time, the model becomes:

$$y = x_1 + ... + x_6 + f_1(suggested\ num\ players) + f_2(minimum\ age) + f_3(playing\ time) + f_3(year)$$

where x_i represents a dummy variable indicating if the game belongs to the i-th cluster or not. As a consequence, the terms related to the categories are not smoothed using a B spline basis, but give a linear contribution to the output. Thus, we decided to introduce a penalization on these components, in order to obtain a sparse representation of the categories that most affect the average rating through Lasso feature selection. The results

in term of R^2_{adj} are better than the model of before, since we obtained an R^2_{adj} of 75% . Regarding the Lasso penalization, all the clusters are proved to be significant.
This last model is more complete, thus can provide better results. However, the results on the test set designed before are equal to GAM and xgboost (0.41), but this can be due to the test set chosen.

4.2.4 Regression models for the average rating

Regarding the average rating, we proceeded similarly to the average complexity, that is, we built a GAM model and an xgboost model. However, since we obtained a good model to estimate the average complexity of the game, we tried to use this quantity as covariate in an univariate local averaging regression model.
All these models have the average rating as response variable.
In the weighted local averaging regression model, we used the Epanechnikov kernel fixing the bandwidth to 0.5, which corresponds to a fair estimate of the variance of the average complexity. The qualitative results achieved with this model can be shown with the following plot:

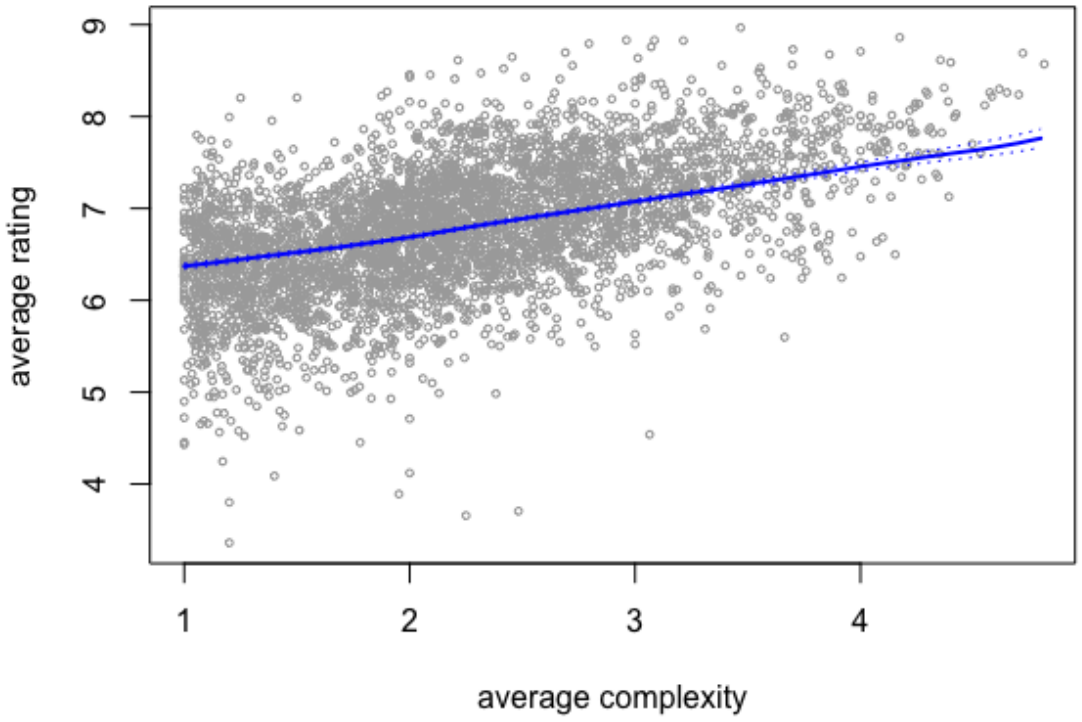


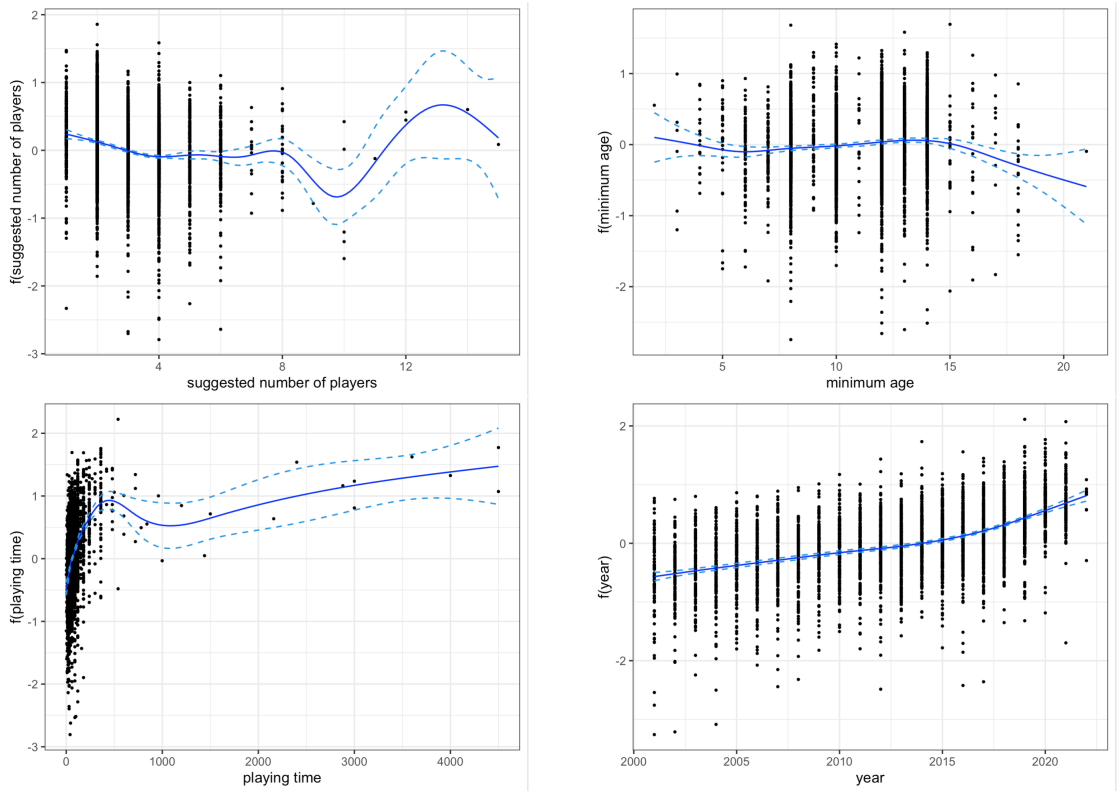
Figure 23: Local averaging with Epanechnikov Kernel

The behavior is more or less linear, with some small curvatures in correspondence of the integer values of the average complexity. The dashed lines are the prediction intervals, which are small given the high numerosity of the dataset. The model seems thus to have a small variance but a higher bias. The correlation between average rating and average complexity is clear from this plot. This justifies our attempts in modelling the average complexity in section 4.2.3.
The second model is a GAM that uses as covariates the playing time, the minimum age required, the suggested number of players and the year in which the game has been published.

$$y = f_1(suggested\ num\ players) + f_2(minimum\ age) + f_3(playing\ time) + f_3(year)$$

All the regressors are built using cubic Bsplines. The year is not so flexible from the point of view of the producers, but can be very useful for the platform *BoardGameGeek* to predict the average rating of games that

have been recently produced, hence our decision of inserting it as a covariate. Below we inserted the plots of the smoothing terms in correspondence of the covariates:



The R^2_{adj} obtained with the model is equal to 47%, which is not a great accomplishment. This is due to the fact that the average rating is more difficult to explain than the average complexity with the covariates that we have available, since the rules of the game such as the main topic of the game can play an important role for the average rating that the user will give.
We computed the marginal contributions of the covariates for this model too, obtaining the following plots:

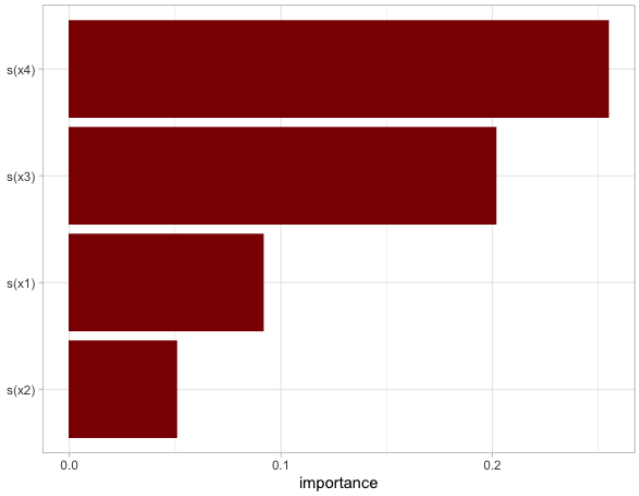


Figure 25: Importance of each feature in GAM

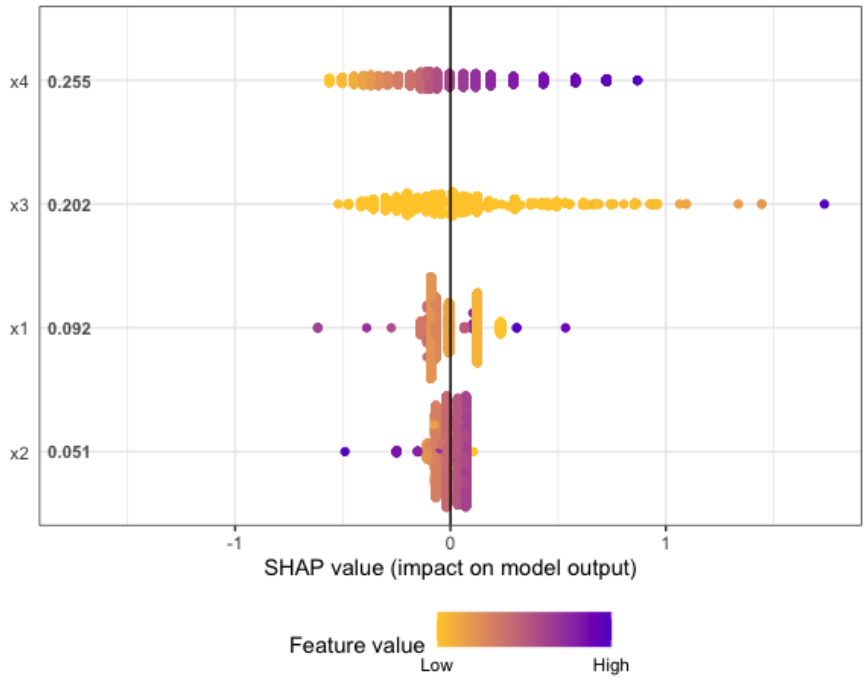


Figure 26: Importance of each feature in GAM

In each plot x_4 is the year, x_3 the playing time, x_2 the minimum age required and x_1 is the suggested number of players. The plots suggests that the year is the most important feature in the prediction of the average: increasing the year the SHAP value increases, meaning that more recent games have a higher average rating than older ones. This emphasizes that the board games have increased their appeal among the users and they became more successful in the last years.
Following the reasoning of before, we tried to build an xgboost model with the same covariates. We obtained an RMSE of 44%. The plots related to the SHAP values are the following:

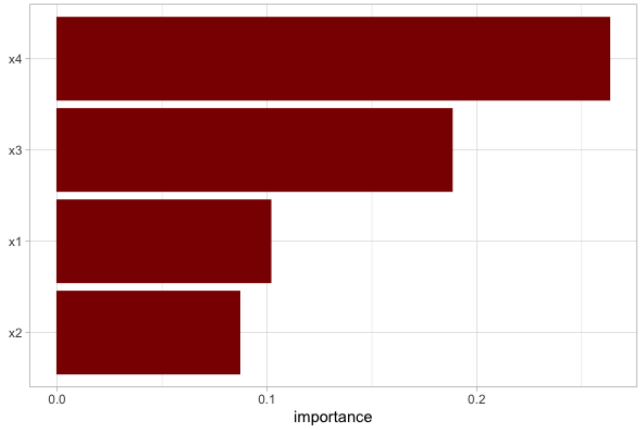


Figure 27: Importance of each feature in XGBOOST

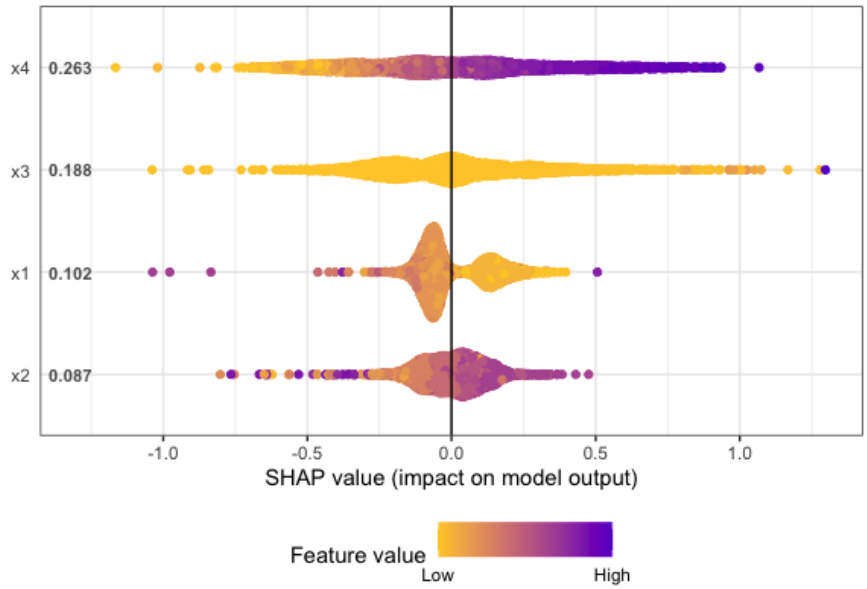


Figure 28: Importance of each feature in XGBOOST

Also for this model the year is the most relevant covariate.
To compare the three models we adopted the same procedure of before, splitting the data set randomly in training set and test set. The RMSE for the GAM and xgboost model are very similar (0.51), while the one of the local averaging model is slightly worse (0.59).
As last attempt, we tried to insert in the GAM the categories the games belongs to. This time, the model becomes:

$$y = x_1 + ... + x_n + f_1(suggested\ num\ players) + f_2(minimum\ age) + f_3(playing\ time) + f_3(year)$$

where x_i represents a dummy variable indicating if the game belongs to the i-th category or not. As a consequence, the terms related to the categories are not smoothed using a B spline basis, but gives a linear contribution to the output. Thus, we decided to introduce a penalization on these components, in order to obtain a sparse representation of the categories that most affect the average rating through Lasso feature selection. Unfortunately, the results in term of R^2_{adj} are similar to the model of before, we only gained few percentile points. Regarding the Lasso penalization, most of the categories are proved to be significant, and again War games have the highest coefficient in this linear model, which is consistent with the results obtained in section 4.1.1

5. Conclusions

Summing up, we can join the results obtained using the permutational anova tests and the linear models. From the point of view of the producers, it is important to notice that the average rating of a game increases on average if the game belongs to one of these three categories: War games, World War II and Ancient. The average complexity is also higher for these categories on average. Regarding the other characteristics, our GAM models allow the producers to predict both the average and the average complexity only using the characteristic of the game such as minimum age required, suggested number of players, playing time and year in which the game is published. The last covariate can be more useful for websites that want to predict how much the game will be appreciated from the users, and how much engagement will the game create, while for the company that produces the game it can be a fixed quantity, thus not so useful.

References

[1] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[2] Mia Hubert, Michiel Debruyne, and Peter J. Rousseeuw. Minimum covariance determinant and extensions. *WIREs Computational Statistics*, 10(3), dec 2017.

[3] Sébastien Lê, Julie Josse, and François Husson. Factominer: An r package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18, 2008.

[4] Guansong Pang, Longbing Cao, and Ling Chen. Outlier detection in complex categorical data by modelling the feature value couplings. IJCAI'16. AAAI Press, 2016.

[5] Laurens van der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15:3221–3245, 01 2015.

6. References

- The data can be found here
- An interesting article about SHAP values can be found here