Data Mining: Fundamentals

A.Y. 2024/2025

Group 12

Bruno Barbieri, Noemi Dalmasso, Gaia Federica Francesca Ferrara

Contents

1 Data Understanding and Preparation

The dataset *train.csv* contains 16431 titles of different forms of visual entertainment that have been rated on IMDb, an online database of information related to films, television series etc. Each record is described by 23 attributes, either discrete or continuous.

1.1 Discrete Attributes

Table 1.1 shows the discrete attributes of the dataset, their types and a brief description of each attribute.

Attribute	**Type**	**Description**
`originalTitle`	Categorical	Title in its original language
`rating`	Ordinal	IMDB title rating class, from `(0,1]` to `(9,10]`
`worstRating`	Ordinal	Worst title rating
`bestRating`	Ordinal	Best title rating
`titleType`	Categorical	The format of the title
`canHaveEpisodes`	Binary	Whether the title can have episodes: `True`/`False`
`isRatable`	Binary	Whether the title can be rated: `True`/`False`
`isAdult`	Binary	Whether the title is adult content: `0` (non-adult), `1` (adult)
`countryOfOrigin`	List	Countries where the title was produced
`genres`	List	Genre(s) associated with the title (up to 3)

Table 1.1: Description of discrete attributes

1.1.1 Merging and Removal of Discrete Attributes

The following discrete attributes were removed from the dataset:

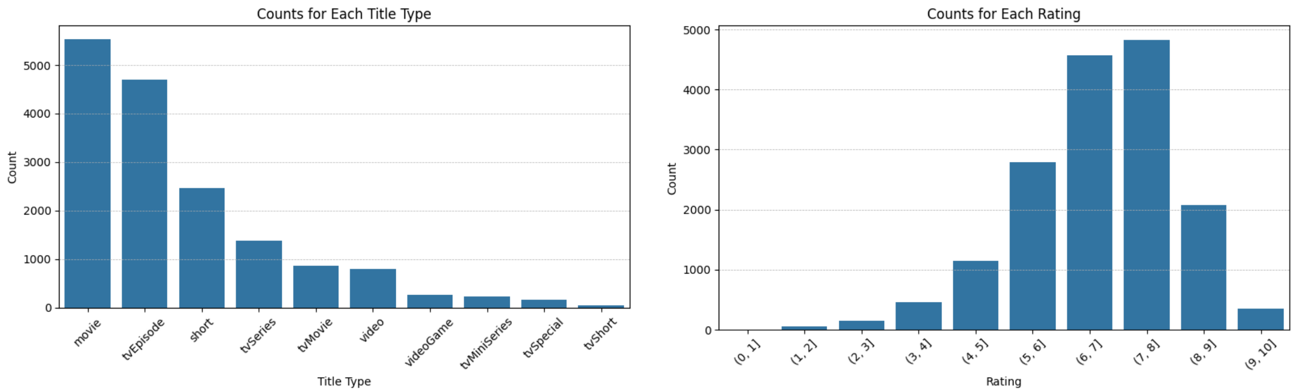
- `originalTitle` was removed because it is not relevant for the analysis;
- the `isRatable` variable was removed because all the titles in the dataset are ratable;
- `worstRating` and `bestRating` attributes were removed because they assume the same values for all records (1 and 10 respectively).

Additionally, the `isAdult` attribute is highly correlated with the presence or absence of *Adult* in `genre` (16 records differ in the train set, 1 in the test set), so the two were merged with a logical OR operation. This is not true for the *short* type in `titleType`, with 491 records having different values from the obtained feature. For this reason, the two were kept separate.

1.1.2 Discrete Attributes Analysis

This paragraph provides an overview of the discrete attributes in the dataset, focusing on their distributions and statistics. The following figures 1.1a and 1.1b show bar plots of `titleType` and `rating` attributes, respectively.
From figure 1.1a it is observed that the classes of the titleType attribute are unbalanced, with *movie* being the most frequent class (5535 records). It was observed that the class *tvShort* is the least frequent in the dataset, with only 40 records (around 0.24% of the dataset). Because of this, these rows were discarded from the dataset, as they were considered irrelevant for the analysis. The decision was not repeated for *tvSpecial* and *tvMiniSeries*, as they

cover slightly more than 1% of the dataset each (166, 1.01% and 224, 1.36%, respectively).



(a) Distribution of `titleType` (b) Distribution of `rating`

Figure 1.1: Distribution of the `titleType` and `rating` attributes

As shown in figure 1.1b, the `rating` attribute roughly follows a normal distribution, with a slightly asymmetric peak: a significant number of titles falls within the (6,7] and (7, 8] ranges (4565 and 4822 titles, respectively) while only a total amount of 67 titles falls within (0,1] and (1,2].

1.1.3 Encoding and Transformation of Categorical Attributes

The attribute `rating` was transformed by taking the upper bound of each rating interval's string representation. This approach was chosen because the minimum rating is 1, meaning the lowest interval corresponds only to ratings of 1. For consistency, the same transformation was applied to all other intervals. Multi-label one-hot encoding was applied to the `genres` column. Each unique genre was represented as a binary feature, allowing records that belong to multiple genres simultaneously to maintain this information; this generated 28 new features. Depending on the task, some were often discarded to avoid overfitting or to reduce the number of features. This will be discussed in the corresponding sections. Rows with no genres were assigned a vector of all zeros, indicating the absence of any genres.

The attribute `countryOfOrigin` was represented by grouping the countries by continent. The following variables have been created:

- `countryOfOrigin_AF` (Africa);
- `countryOfOrigin_AS` (Asia);
- `countryOfOrigin_EU` (Europe);
- `countryOfOrigin_NA` (North America);
- `countryOfOrigin_SA` (South America);
- `countryOfOrigin_OC` (Oceania);
- `countryOfOrigin_UNK` (Unknown country);
- `countryOfOrigin_freq_enc` (frequency encoding of the original list).

For each record, the first six features provide the number of countries for each continent. The `countryOfOrigin_UNK` variable counts the number of countries that are not recognized as belonging to a continent for that record. Additionally, `countryOfOrigin_freq_enc` provides the frequency encoding of the original list of countries as a whole, showing how frequently a specific combination of countries appears across the entire dataset. These transformations allow to keep most of the original information, while limiting the number of new features.

1.2 Continuous Attributes

Table 1.2 shows the continuous attributes of the dataset, their type and a brief description.

Attribute	Type	Description
`runtimeMinutes`	Integer	Runtime of the title expressed in minutes
`startYear`	Integer	Release/start year of a title
`endYear`	Integer	TV Series end year
`awardWins`	Integer	Number of awards the title won
`numVotes`	Integer	Number of votes the title has received
`totalImages`	Integer	Number of images on the IMDb title page
`totalVideos`	Integer	Number of videos on the IMDb title page
`totalCredits`	Integer	Number of credits for the title
`criticReviewsTotal`	Integer	Total number of critic reviews
`awardNominationsExcludeWins`	Integer	Number of award nominations excluding wins
`numRegions`	Integer	Number of regions for this version of the title
`userReviewsTotal`	Integer	Number of user reviews
`ratingCount`	Integer	Total number of user ratings for the title

Table 1.2: Description of continuous attributes

1.2.1 Removal and Merging of Continuous Attributes

The plot in figure 1.2 is a Pearson's correlation matrix that takes into account the continuous attributes of the dataset. The matrix shows that `ratingCount` and `numVotes` are perfectly correlated; for their redundancy, `ratingCount` was discarded.

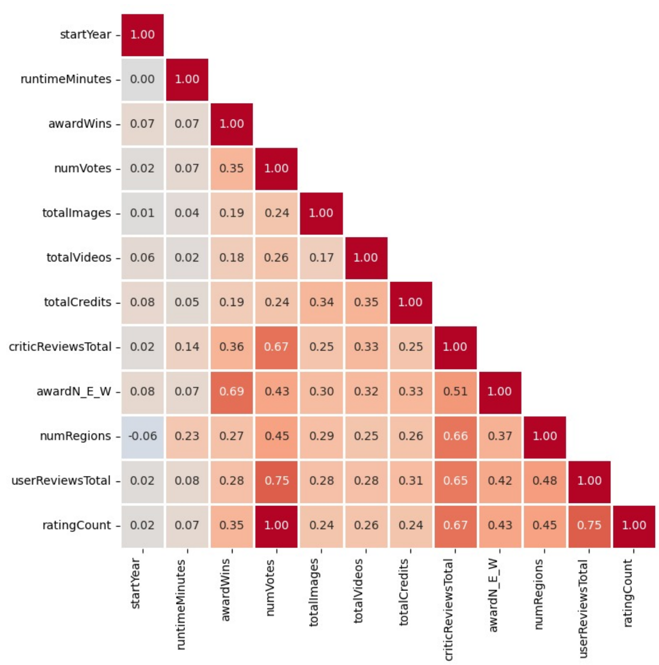
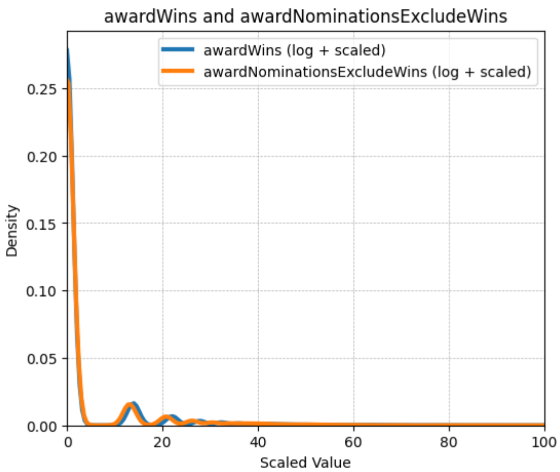
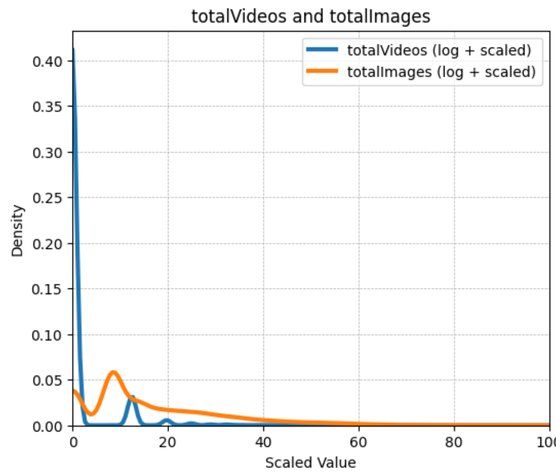


Figure 1.2: Correlation matrix

The attributes awardNominationsExcludeWins and awardWins were combined into totalNominations, due to their strong semantic similarity and high correlation (0.69). The new feature represents the sum of the two original attributes. This transformation also helps mitigate the impact of their heavy right skew (shown in figure 1.3a), resulting in a more meaningful and interpretable feature. Similarly, the totalVideos and totalImages attributes were combined into a single feature, i.e. totalMedia, representing the total number of media items associated with a title. Although the original attributes are not highly correlated, both exhibit skewed distributions (as in figure 1.3b), totalVideos in particular. Due to this, and to their similar semantic meaning, they were merged to form a more consolidated and interpretable feature.



(a) Kernel Density Estimation of awardWins and awardNominationsExcludeWins



(b) Kernel Density Estimation of totalVideos and totalImages

Figure 1.3: Distribution of the attributes that form the totalNominations and totalMedia features

Although criticReviewsTotal and userReviewsTotal also have a relatively high correlation (0.65), as well as a right-skewed distribution, it was decided that the two attributes should be kept separate because of their relevance in meaning. It is also worth noting that the two have high correlations with numVotes (0.67 and 0.75 respectively), but they were all kept because of the difference between votes and reviews.

1.3 Data Quality

Next, a proper evaluation of the observed data was conducted in preparation for the analysis. Once having checked that there are no duplicates and no incomplete rows in the dataset, attention was given at identifying missing values and outliers.

1.3.1 Missing Values

The following attributes were found to have missing values[1]:

- endYear: it is the feature with the highest number of NaN values (15617; about 95%). Although the feature is only relevant for *TVSeries* and *TVMiniSeries* titles, it still had approximately 50% missing values within those categories, limiting its usefulness even in the appropriate context. For this reason, the feature was discarded.

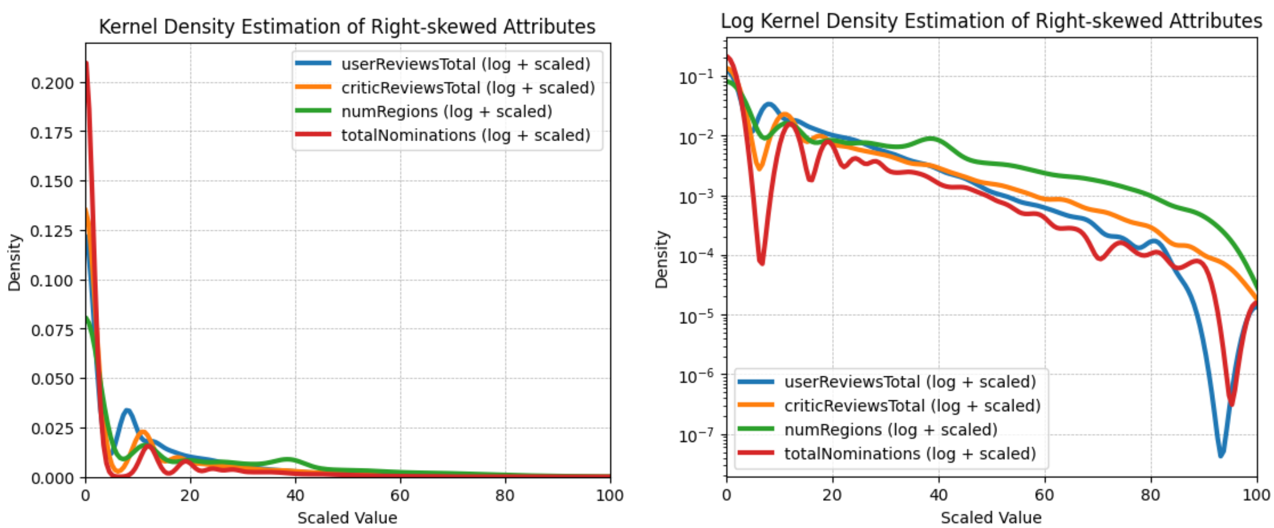
[1] missing values were marked as NaN only in awardWins; in the other listed columns "\N" was used, so it was then converted to NaN

- `runtimeMinutes`: this attribute has 4,852 missing values (29.5%). Two imputation strategies were employed, both based on random sampling within the interquartile range. One strategy used the `titleType` feature to define the range, while the other imputed values based off of the attribute's distribution alone. The choice of which of the two strategies to use depends on the specific task, and will be specified in the corresponding sections.
- `awardWins`: this feature has 2618 `NaN` values (about 16%). Since the mode associated with this variable is 0, it has been decided to substitute the missing values with 0.
- `genres`: it has 382 missing values (2.3%). Having dealt this variable with a multi-label one-hot encoding process (as has been described in the *Encoding and Transformation of categorical attributes* section), a vector of all zeros is assigned to record with missing genres values.

1.3.2 Semantic Inconsistencies, Feature Transformations and Outlier detection

While analyzing the dataset, it was observed that the *Videogame* type of the `titleType` attribute (259 records - around 1.58% of the dataset) was not consistent with the other values of the same feature, being *Videogame* a fundamentally different titleType. Other then this semantic inconsistency, these rows generated problems for some of the other attributes, such as `runtimeMinutes`, resulting in most values being missing and difficult to impute. Because of this, the samples were removed from the dataset.

Some features showed a heavy right-skewed distribution, with typical traits of Power-Law Distributions. Their Kernel Density Estimations are shown in figure 1.4.



(a) KDE of right-skewed attributes

(b) KDE of right-skewed attributes with log density

Figure 1.4: Kernel Density Estimation of the right-skewed attributes

The decay of these features is exponential in linear space (1.4a), while in logarithmic space there is a decline that can be approximated to a linear trend (1.4b). For this reason, when needed, a log-transformation was applied to these attributes to reduce the skewness and make them more suitable for some specific analysis. Because of right-skewness (without a power-law distribution), other attributes were also log-transformed:

- `numVotes`;

- `totalCredits`;
- `totalMedia`.

Regarding outliers, the feature that was found to be more problematic was `runtimeMinutes`. Similarly to missing values imputation (1.3.1), outlier detection was performed using two different strategies: the first approach computes outliers on each `titleType` separately, while the second is based on the distribution of the `runtimeMinutes` attribute alone. Figure 1.5 reports an analysis of the feature through the IQR method separately on each type.

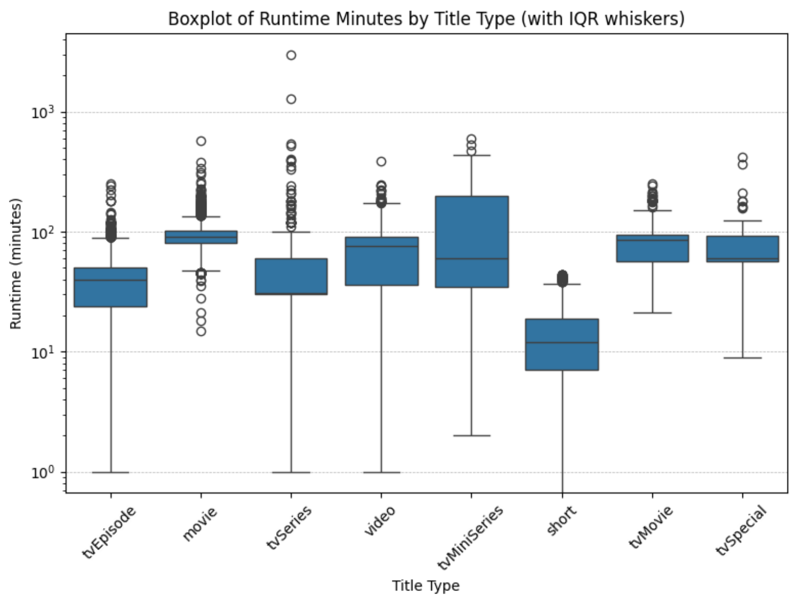


Figure 1.5: Boxplot of the `runtimeMinutes` attribute for each `titleType`

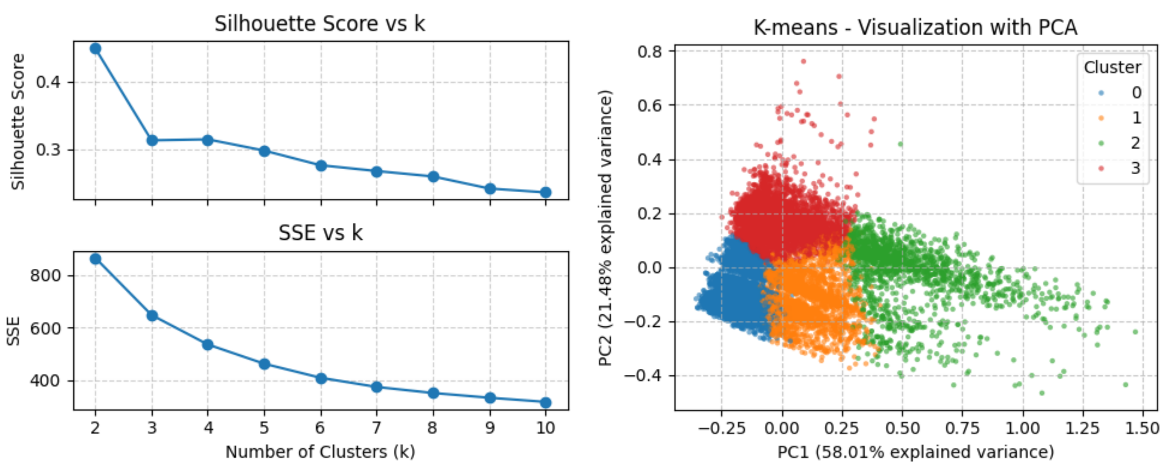
The boxplots show that there are samples that have been misreported, with runtimes of over 1000 minutes for *tvSeries*. This might be because of an inconsistency with the understanding of the meaning of the attribute, and in those cases it might be possible that the value refers to the total runtime of the series, rather than the runtime of a single episode. Another interesting observation regards the presence of a record with a runtime of 0 minutes for the *short* type; the record was removed from the dataset because it was regarded as an erroneous sample. Other than this sample, other outliers were not removed from the dataset by default. Instead, a case-by-case approach was adopted, testing each task and analysis both with and without the outliers. Notably, in every case, better results were obtained when outliers were excluded.

2 Clustering

This chapter of the report aims at illustrating the clustering analysis performed on the dataset at hand. The employed clustering techniques are K-means (Centroid-based), DBSCAN (density-based) and hierarchical clustering. The analysis conducted using these methods focused on a selection of the continuous attributes of the dataset, which were appropriately log-transformed (when needed - as mentioned in subsection 1.3.2), and normalized using `MinMaxScaler`.

2.1 K-means

Clustering analysis with K-means was performed using a carefully selected subset of features. This step was motivated by the algorithm's sensitivity to the curse of dimensionality, as including too many variables can negatively affect SSE and Silhouette scores. For this reason, although only `numVotes`, `totalCredits`, `userReviewsTotal`, `runtimeMinutes`, and `criticReviewsTotal`, were chosen for this task, they proved to be a solid choice due to their ability to represent meaningful aspects of the data. To identify the proper number of clusters, both SSE and Silhouette scores were evaluated. The objective was to find a configuration that reduces the SSE while maintaining a robust Silhouette score and a proper k. The plots in Figure X show that $k = 4$ provides a balance between the two values (obtaining SSE equal to 535.51 and 0.315 for the Silhouette score). Only for visualization purposes, Principal Component Analysis (PCA) was employed. The first two components account for 58.01% and 21.48% of the total variance, indicating that this projection provides a fairly informative view of the clustering structure. The cluster results are presented in figure 2.1b. The 4 distinct clusters appear well-separated, suggesting that K-means managed to capture meaningful groupings in the data.



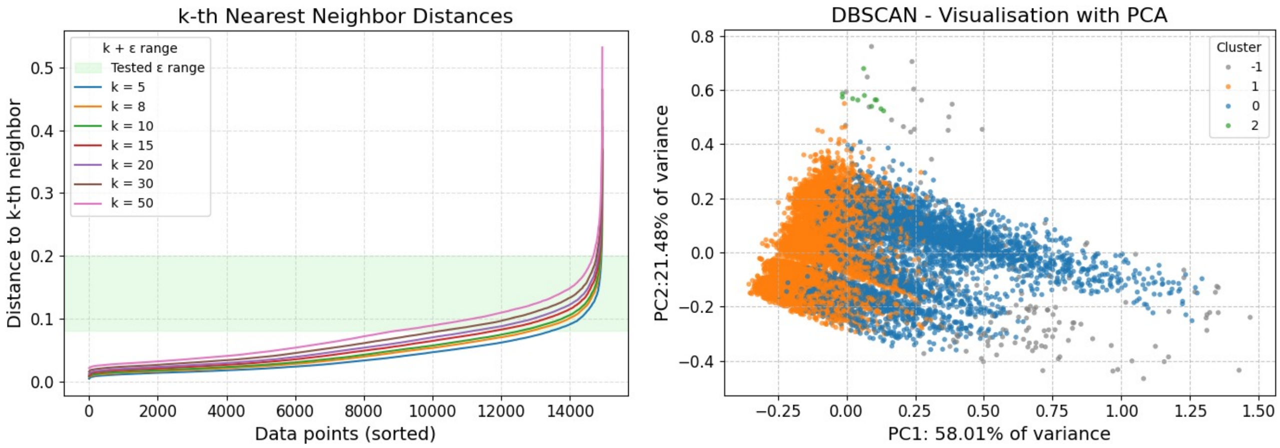
(a) SSE and Silhouette scores

(b) Clustering Visualization with PCA

Figure 2.1: K-means - Cluster analysis

2.2 DBSCAN

To determine suitable DBSCAN parameters, the k-th distance plot was used as starting point (Figure 2.2a). By varying k (*MinPts*) and observing the corresponding distance curves, the range between 0.08 and 0.2 was identified for a possible *Eps* value. The algorithm was applied to the following features: `numVotes`, `totalCredits`, `criticReviewsTotal`, `userReviewsTotal`, and `runtimeMinutes`. Although different combinations were tested, they all led to similar results. Effectively, initial experiments clearly revealed DBSCAN's sensitivity to highly sparse and imbalanced datasets; in most cases, the algorithm produced a highly populated single cluster corresponding to the dominant dense region, while leaving the sparser regions largely unstructured.



(a) K-th distance plot

(b) Visualization with PCA

Figure 2.2: DBSCAN - Cluster analysis

Initial experiments clearly revealed DBSCAN's sensibility towards high-sparsed and imbalanced dataset; in most cases the algorithm produced an highly populated single cluster in correspondence of the dominant dense region, leaving sparser regions largely unstructured. To address this issue and obtain more meaningful clustering, different combination of `min_eps` and `min_points` in the following ranges were tested:

- `eps_list` = [0.08, 0.09, 0.1, 0.12, 0.14, 0.16, 0.18, 0.2, 0.3, 0.4, 1.0];
- `min_points_list` = [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20].

Each combination was evaluated based on the number of clusters (with a minimum threshold of 2), size of the clusters, noise points, and silhouette score (filtered for values > 0.1), excluding the cases where only one cluster was created, or where the silhouette value was consisidered too low.
Even though the configurations did not produce fully satisfactory results, the most relevant ones are summarized in Table 2.1. The reported Silhouette scores are computed excluding noise points; therefore, no score is provided when only a single cluster is formed. It is important to note that, in this context, a high Silhouette score can be misleading, as it tends to favor cohesion of the main cluster, at the expense of structural granularity.

eps threshold	**Min points**	**Silhouette**	**Cluster composition**
$\varepsilon > 0.16$	for any tested value	---	never led to more than one cluster
$0.13 \leq \varepsilon \leq 0.16$	for any tested value	$[0.54,\ 0.70]$	led to one cluster and, occasionally, 1 or 2 minor ones
$\varepsilon < 0.13$	for any tested value	$[0.10,\ 0.31]$	the big cluster it's splitted in two different ones
$\varepsilon \leq 0.10$	lower values $[4,\ 5]$	$[0.10,\ 0.25]$	led to several, not meaningful, clusters of 5/6 points

Table 2.1: Relevant DBSCAN parameters configurations

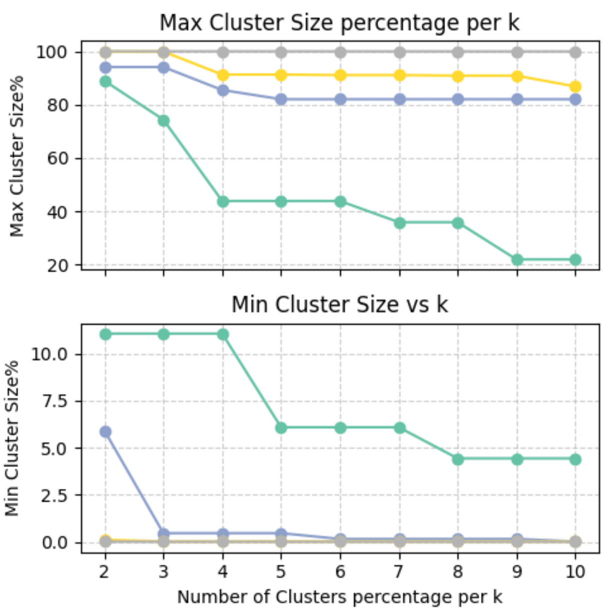
Exlcuding the combination with higher *Eps* value, that made the points collaps into one cluster, and the ones with low *Eps* and few `min_pts` ,that resulted in over-fragmentation, the analysis focused on *Eps* in the range $[0.11,\ 0.13)$. Despite the limitations of the dataset structure, in this range the algorithm was able to differentiate between regions of varying density,
One of the most interesting compromise (though still suboptimal in absolute terms) was found with $Eps = 0.11$, `min_points` = 7: As shown in (Figure 2.2b), this setting resulted in a 3 cluster structure: a compact cluster of 10378 points was extracted from the densest region, a less dense cluster with fewer points (4347) captured intermediate-density areas or mild outliers, and a very small one (only 11 points), but well-separated, cluster effectively captured

a potential relevant local pattern. Only 219 points were labeled as noise, and the Silhouette Score was 0.2699.

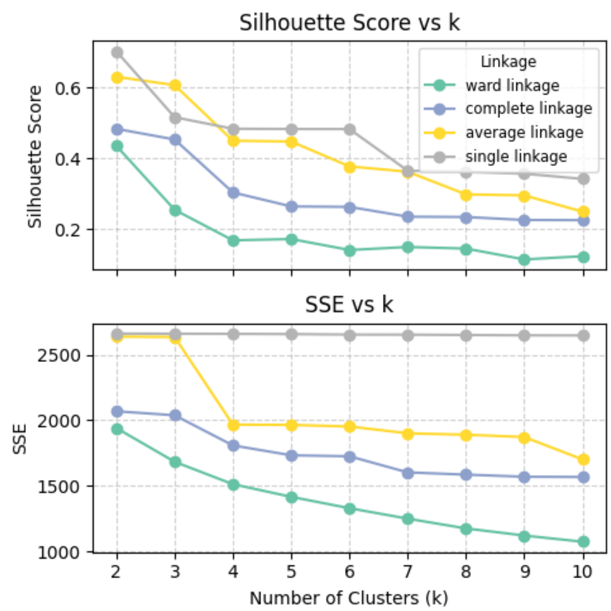
QUESTA PARTE VOLENDO PUò ANDARE NELLE CONSIDERAZIONI GENERALI
To conclude, these observations highlight DBSCAN's limitations when applied to datasets with skewed feature distributions and unbalanced local densities. Nonetheless, the exploration helped isolate parameter ranges where meaningful subclusters begin to emerge,beyond the dominant density mass.

2.3 Hierarchical clustering

Hierarchical clustering was performed using all linkages (Ward, Average, Complete, Single), with the Euclidean distance metric. After a careful analysis of multiple clusterings, it was decided to procede with all remaining numerical features, log transformed and normalized with `MinMaxScaler`. Figure 2.3 shows the results of the analysis, which includes the Silhouette and SSE scores, as well as the maximum and minimum percentage of points per cluster.



(a) Max/Min percentage of points per cluster



(b) Silhouette and SSE scores

Figure 2.3: Hierarchical clustering metrics for different numbers of clusters

From figure 2.3a, it can be observed that Single linkage produces a single cluster which contains basically all data points. This makes it unsuitable for this use case. Average and Complete linkages produce a cluster with a high maximum cluster size (above 90% of the dataset for most of the number of clusters tested for Average, and 80% for Complete). These results are likely due to two main causes:

- Usage of skewed features, which lead to areas with a very high density of points;
- High dimensionality of the dataset, which makes it more difficult to separate clusters effectively.

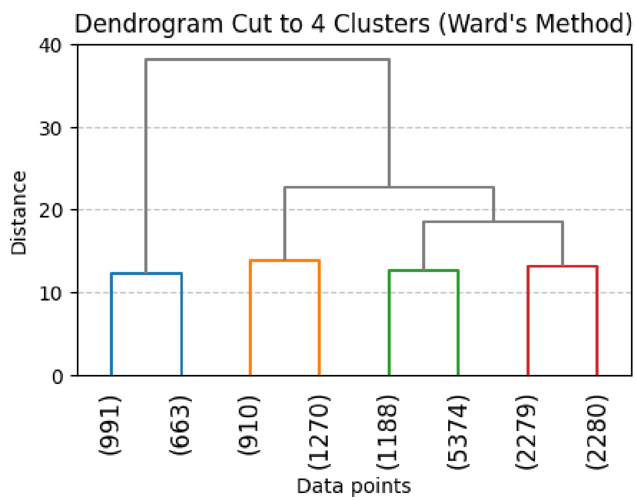
These issues are mitigated by Ward's method, which doesn't show a dominant biggest cluster, for all numbers of clusters tested. The minimum cluster size is also consistently more balanced across different numbers of clusters.

Since Ward's method is based on Squared Error, it's not surprising how its SSE is consistently lower than other linkages, as shown in the second graph of figure 2.3b. What's more interesting is that Ward's method has the lowest Silhouette scores for all numbers of clusters tested, which indicates that the clusters are not well separated.

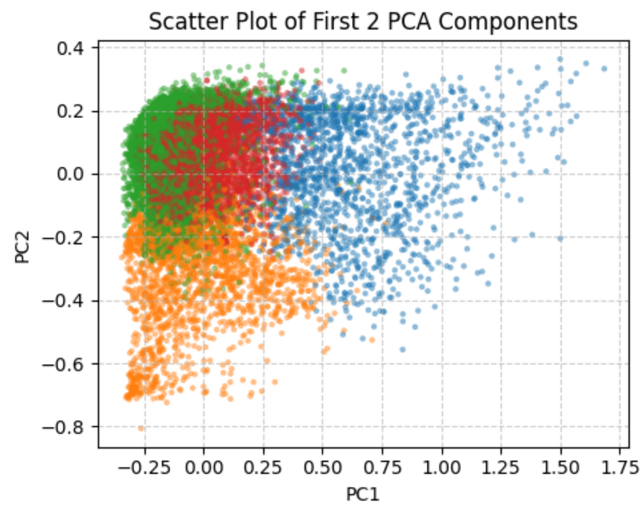
This is due to the fact that Ward's method groups the data which resides in the high-density area mentioned above into smaller clusters, leading to less well-defined boundaries between them. In the next two sections, the results of Ward's method and Complete Linkage will be analyzed; the other two will not be discussed further, as they provide limited insights on the dataset.

2.3.1 Ward's method

Figure 2.4 a dendrogram and a scatter plot for a clustering obtained through Ward's Method. Because of the parameters observed in the previous section, cleaner dendrograms, as well as consistency with other clustering methods, the hierarchical clustering is cut at 4 clusters.



(a) Ward's Method Dendrogram



(b) Ward's Method Scatter plot

Figure 2.4: Ward's method clustering

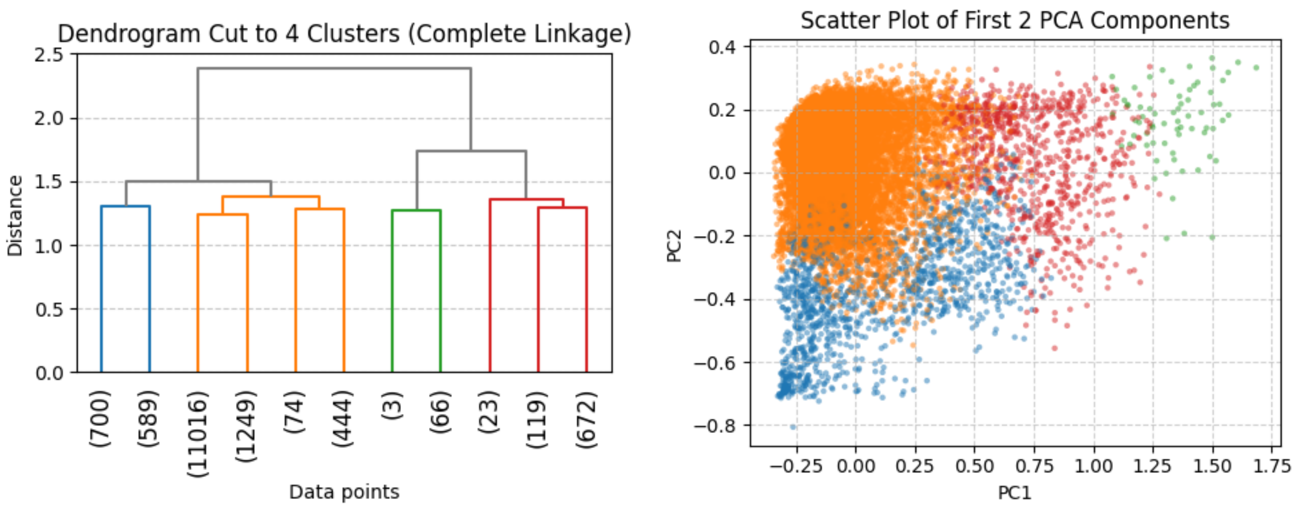
The dendrogram in Figure 2.4a reveals well-separated clusters, all of which merge at a similar linkage distance (approximately 12). This indicates that the increase in within-cluster sum of squares (SSE) is relatively consistent across all cluster merges, as expected from Ward's method. Notably, the smallest cluster is the last to be merged, which suggests it is more distinct from the others. As shown in Figure 2.4b, this cluster lies in a sparser region of the dataset. In contrast, the remaining three clusters originate from the denser region, and are therefore spatially closer to one another. Compared to K-Means, the clusters identified by Ward's method appear less clearly separated in the PCA projection, resulting in some overlap between adjacent groups.

2.3.2 Complete Linkage

Figure 2.5 shows the dendrogram and scatter plot for a clustering obtained through Complete Linkage. Since the tendency of this linkage is to merge into a single cluster, the clustering is cut at 4 clusters, which helps mitigating the issue. While selecting five clusters would have introduced an additional split within the largest cluster with similar SSE and Silhouette scores, the resulting group was found to be poorly separated and lacked meaningful distinction. As such, it was not considered a valuable contribution to the overall clustering structure.

As shown in the dendrogram in Figure 2.5a, the clusters merge at similar linkage distances (approximately between 1.2 and 1.4), indicating that the maximum within-cluster distances are comparable across clusters. With respect to the clustering obtained through Ward's method, the clusters have clearer boundaries along the PCA axes, as the denser area of the dataset is not split into multiple clusters.

It is also interesting to observe how the dendrogram structure differs from that of Ward's method. In the previous case, the smallest cluster was the last to be merged, reflecting its distinctiveness in terms of within-cluster variance. In contrast, the Complete Linkage dendrogram shows the two smaller clusters being merged before the root. This is observed with Single and Average Linkages as well, and is a product of the sparsity of these clusters' regions.



(a) Complete Linkage Dendrogram

(b) Complete Linkage Scatter plot

Figure 2.5: Dendrograms for hierarchical clustering with Complete Linkage

2.4 General considerations

PUNTI TRATTABILI (non in ordine):

- k-means raggiunge valori alti di silhouette ma perchè le variabili usate sono poche - questo può essere confrontato con ward's che usa invece più variabili (ma paragonabile per valore di silhouette? altrimenti fare confronto con metodo di hierarchical che raggiunge performance simili a kmeans- anche se meno paragonabile rispetto a ward's perchè lui + simile a kmeans dato che usa sse) −¿ vedere se dire in general considerations o se in k-means
- considerazioni generali su dataset, il clustering riesce a far emergere qualcosa riguardo alla struttura dei dati?
- k-means algoritmo più lineare mentre dbscan e hierarchical più complessi, che magari fanno emergere cose più interessanti
- confronto con titletype?
- se mettiamo hierarchical clustering come prima subsection, spiegare il perchè
- kmeans: - obiettivo → trovare cluster ben separati, con silhouette onesta e SSE bassa - risultato → 4 cluster ben separati, silhouette score di 0.315, SSE di 535.51 - confrontabile con metodi di hierarchical?
- DBSCAN: OBIETTIVO → trovare miglior compromesso tra struttura interpretabile (almeno 2 cluster), poco rumore e silhouette onesta. → e con cluster che anche se piccoli siano "effettivamente separati" problema →per caratteristiche del dataset stesso → DBSCAN tende a formare un cluster dominante nella zona densa che "cattura" la maggior parte dei dati (la classe maggioritaria) mentre dati + dispersi sono rappr come rumore o come micro-custer non significativi Spesso epsilon alta → porta a un comportamento più "aggressivo" nell'unire punti, risultando nel "cluster gigante" Tendenzialmente con epsilon più basso, l'algoritmo riesce a distinguere meglio le diverse densità,catturando sia la struttura principale che eventuali sottogruppi significativi. "To conclude, these observations highlight DBSCAN's limitations when applied to datasets with skewed feature distributions and un-

balanced local densities. Nonetheless, the exploration helped isolate parameter ranges where meaningful subclusters begin to emerge,beyond the dominant density mass."

3 Classification

Classification was performed on the available training set using three different algorithms: K-NN (*K-Nearest Neighbours*), Naïve Bayes and Decision Trees. The target variables chosen for this task are 2: `titleType` and `has_LowEngagement`. These will be discussed in more detail in the corresponding sections below.

3.1 Binary classification

The binary target variable used in this task, `has_LowEngagement`, was specifically defined for this purpose. It identifies records where the `numVotes` attribute is less than 100.

An analysis of semantically related features was run to decide whether to discard any. `userReviewsTotal` has a 75% correlation with `numVotes`, while `criticReviewsTotal` has 67% correlation. Due to its semantic similarity with `numVotes`, `userReviewsTotal` was discarded for this task. In contrast, `criticReviewsTotal` was retained as it was deemed to provide distinct and complementary information. Moreover, its correlation with other features was not considered sufficiently high to make the problem trivial.

An important aspect of the chosen binary classification task is the class imbalance, with 10287 records classified as *Low Engagement* and 4668 as *High Engagement* in the training set. This imbalance was taken into account during model training and evaluation, with a focus on macro-averaged F1-score to mitigate its impact on the results.

3.1.1 K-NN - Binary Classification

The features used for the K-NN algorithm were normalized, as the model is sensitive to unscaled values: log-transformation (when needed) and `StandardScaler` were applied to data. Then, to perform this task, care was taken in selecting the appropriate type and amount of features to avoid unnecessary increased dimensionality. For example, to provide the model with information about the origin of the titles, without introducing all 7 `countryOfOrigin_[continent code]` attributes, only `countryOfOrigin_freq_enc` was kept. The remaining variables of the dataset were maintained as they offered diverse and non-redundant insights.

For this algorithm, a portion of the training set was held out as a validation set to enable external validation after internal cross-validation. A randomized hyperparameter search was indeed performed on the reduced training set, with each configuration evaluated using stratified 5-fold cross-validation. This process aimed to identify the optimal value of k and explore additional algorithm parameters. The best configuration found is: `weights: 'uniform', n_neighbors: 9, metric: 'cityblock'`, which achieved the highest accuracy on the selected feature set.

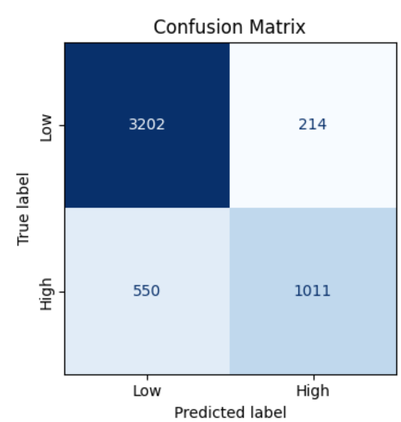


Figure 3.1: K-NN binary classification

The resulting model shows solid performance, achieving a test accuracy of 0.85. As expected, it performs better on the *Low Engagement* class, with a high recall (0.94) and F1-score (0.89). In contrast, the *High Engagement* class is more challenging for the model; while precision is reasonable (0.83), recall (0.65) indicates that a significant number of instances are misclassified as *Low Engagement*. This analysis is reflected in the confusion matrix in Figure 3.1, where 550 out of 1561 *High Engagement* samples were incorrectly labeled as *Low Engagement*.

To summarize, the macro average F1-score of 0.81 is suggesting that the model maintains a balanced performance across both classes.

3.1.2 Naïve Bayes - Binary Classification

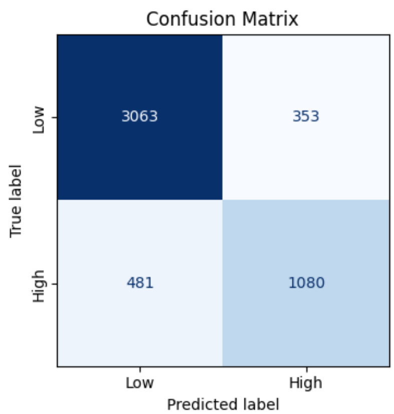


Figure 3.2: NB binary classification

To use Categorical Naïve Bayes, specific preprocessing steps were required: continuous attributes were discretized using semantically meaningful binning and **Country_orig** feature was binarized into a (0 / > 0) format; however, only the from_America_bin and from_Europe_bin were used for this particular task. All binning strategies applied here are consistent with those described in Subsection X of pattern mining section. Finally, the resulting categorical features were numerically encoded using OrdinalEncoder, as required by the CategoricalNB.
The results show good classification performance with an overall accuracy of 0.83, a solid outcome given the simplicity of the NB model. However, differences between classes emerged: the low engagement class performed better, with precision 0.87, recall 0.89, and F1 score 0.88, while the high engagement class had lower results (precision of 0.74, recall of 0.70 and F1 score of 0.72), Actually, the model struggles more to identify high engagement examples, as confirmed by the low recall (0.70) and by the 481 false negatives in the confusion matrix. Conversely, the low engagement class is recognized with greater reliability, with 3063 correct predictions out of 3416. Finally, the macro F1 average of 0.80 indicates a good balance overall, but some difficulty remains in handling the minority class.

3.1.3 Decision Trees - Binary Classification

For explainability purposes, features were not normalized nor transformed for the Decision Tree model, as it does not require such preprocessing because it's not based on distance measures, but rather on decision thresholds.
To identify the optimal hyperparameters, a Randomized Search was performed using Repeated Stratified 5-Fold Cross-Validation with 10 repeats on the training set, optimized for the macro-averaged F1-score. The best configuration found used Gini index as the splitting criterion, a maximum tree depth of 26, and a minimum of 3 samples per leaf. The obtained decision tree is shown in figure 3.3.

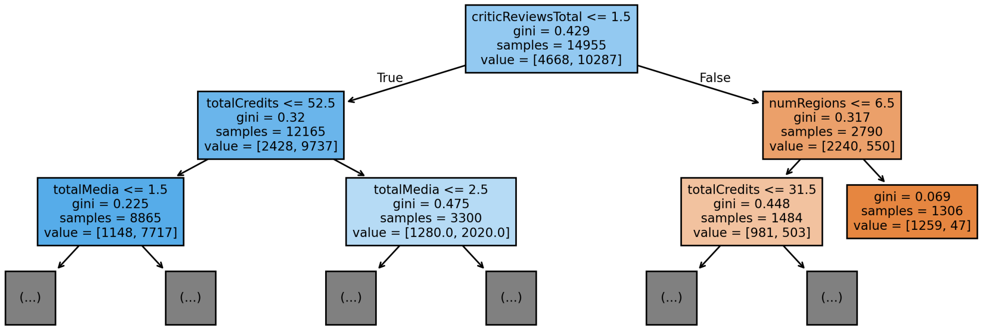


Figure 3.3: Decision Tree for binary classification

Unsurprisingly, the most important feature for the model was `criticReviewsTotal`, which amounted to 0.6 in the feature importance ranking. The following other 3 more important features were `totalCredits` (0.15), `totalMedia` (0.10) and `numRegions` (0.09). These four features take up around 93% of the total feature importance, and are all present in the first two splits shown in the Decision Tree.

Train performance was overall similar to the test performance; in particular, the respective accuracies were of 0.83 and 0.84, and macro-F1 scores were 0.82 and 0.80. In any case, post-pruning was tested, but did not yield any performance improvement, and was therefore not applied. The *High Engagement* class showed low Recall values (0.71 on train set, 0.68 on test set). This might be a consequence of class imbalance, as well as poor separability of the two classes. This assumption is further supported by the Precision scores of the class (0.78 on train, 0.75 on test).

3.1.4 Model Comparison - Binary Classification

After having performed the task using the three algorithms, some interesting aspects emerged. As expected, all the models, to a different extent, struggle to identify the minority class, *High Engagement*, resuling in lower recall values - 0.65 for K-NN, 0.70 for Naïve Bayes, and 0.68 for Decision Trees. This is evident in the confusion matrices, where a significant number of *High Engagement* records are misclassified as *Low Engagement*. In particular, K-NN handled the minority class the best in terms of precision and F1-score, achieving respectively 0.83 and 0.73. However, the fact that Naïve Bayes performed better in terms of recall (0.70) makes it a more valid alternative in scenarios where preventing high-engagement titles from being misclassified as low-engagement ones is a priority. Decision Trees are located in the middle, showing similar performances to Naïve Bayes, but having a slightly lower recall. In conclusion, considering all these aspects, K-NN is still considered the most efficient model for this task, as it also performs best globally. In the table 3.1 its overall results are summarized.

Accuracy	**Precision Macro Avg**	**Recall Macro Avg**	**F1-score Macro Avg**
0.85	0.84	0.79	0.81

Table 3.1: Overall metrics - K-NN

3.2 Multiclass classification

Among the multiclass features in the training set, `titleType` was selected as the target variable for this task, due to its relevance within the dataset. Because of their strong correlation with `titleType`, the feature `canHaveEpisodes` and the genre *Short* were excluded from the feature set. Furthermore, since the primary imputation method for missing values in `runtimeMinutes` relied on information from the chosen target variable, these values were re-imputed to avoid data leakage. Specifically, missing entries were filled by sampling from the overall distribution of `runtimeMinutes`, without referencing `titleType` itself.

One final point to note is the imbalance in the target feature (previously shown in figure 1.1a), which was explicitly taken into account during the design of the models. As for the binary classification task, macro-averaged F1-score was a key metric for model evaluation, as it provides a balance between each class's precision and recall.

3.2.1 K-NN - Multiclass Classification

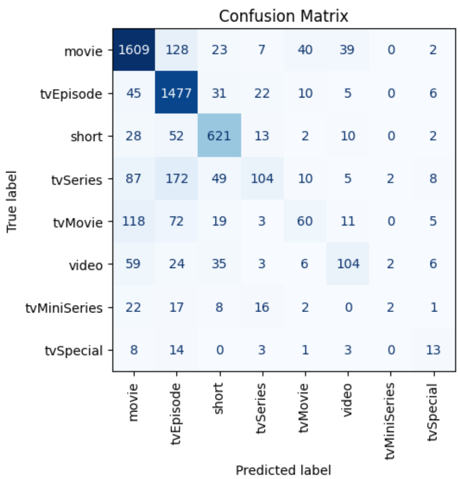


Figure 3.4: K-NN confusion matrix

For the multiclass classification task using the K-NN algorithm, the inclusion of a broad set of features is necessary and justified, as it was found to help the model to better distinguish between underrepresented classes - even if the results in their classification are not optimal. Even though the choice of keeping those classes lowers the overall performance (accuracy) of the model, it is a trade-off in favor of larger coverage and sensitivity across all classes. The hyperparameter tuning strategy that was employed is the same as in binary classification with K-NN. The configuration that achieved the best performances is: `weights:'uniform', n_neighbors:12, metric:'cityblock'`.

As expected, the model shows mixed performances across the different classes. The overall test accuracy reaches 0.76, which is a respectable result for a multiclass problem. However, since accuracy is a global metric, more attention should be paid to the performances of the single classes. Classes with a larger support, like *tvEpisode* (4690 records in the training set) and *movie* (5442), are handled quite well: *tvEpisode* reaches an F1-score of 0.83 and *movie* 0.84, both supported by high recall values (respectively 0.93 and 0.87). This is also clearly visible in the confusion matrix of Figure 3.4, where these classes dominate with a high count of correct predictions. On the other hand, the model struggles with 2 classes in particular: *tvMiniSeries* and *tvSpecial*, i.e. the classes with the least support (respectively 186 and 158). In particular, *tvMiniSeries* shows a very low recall (0.03), meaning that it is rarely correctly identified. Similarly, *tvMovie* is mostly misclassified, showing a low F1-score of 0.29.
To summarize, since the model achieves a macro average F1-score of 0.50, this reflects clearly this imbalance in performance: while the model performs well on the majority classes, it fails to generalize well especially the minority ones.

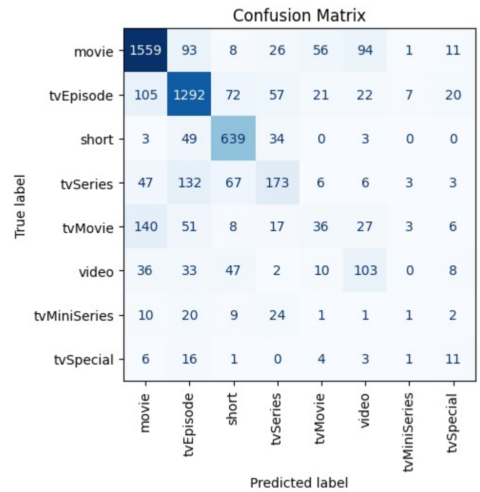
3.2.2 Naïve Bayes - Multiclass Classification

Also for the Multiclass Classification task the features has been preprocessed as explained in paragraph X, in order to use Categorical NB. For this task it has been interesting to note that the inclusion of genre features enhanced classification performance for all classes but especially for the ones with limited support: tvMiniSeries (186 records in the training set) was not classified at all and tvMovie (844) and tvSpecial (158) were poorly classified. This confirmed the choice of CategoricalNB over GaussianNB; being the first the most indicated to handle categorical features.

The model achieves an overall accuracy of 0.73, acceptable for a multiclass task but indicative of difficulties in handling all categories equally well. High F1-scores for the most represented classes—movie (0.83), tvEpisode (0.79), and short (0.81) — suggest effective recognition, likely due to their frequency in the dataset and the presence of more easily distinctive traits. The movie class also shows strong precision (0.82) and recall (0.84), confirming the model's reliability on this category.

As expected, performances drop significantly for underrepresented classes. tvMiniSeries (0.02), tvSpecial (0.21), and tvMovie (0.17) exhibit low F1-scores, indicating that the model has significant difficulties distinguishing correctly these examples. Other than their low support, this is probably linked to the presence of common features with more dominant classes, such as Movie or tvEpisode. This is supported by the confusion matrix, which shows, for example, that 140 tvMovie instances are misclassified as movie, highlighting a strong confusion between these two categories.

Figure 3.5: NB confusion matrix



The tvSeries class also presents issues, with moderate precision (0.52) but a rather low recall (0.40): indicating many missed instances To conclude, the very low F1 macro average score, actually reflects the inability of the model to perform in a balanced way on all the classes. On the other hand, the F1 weighted average results of 0.71, suggesting that the majority classes have a greater influence on overall performance than the minorities.

3.2.3 Decision Trees - Multiclass Classification

Like for the binary classification task, the Decision Tree model was trained without normalizing or transforming the features, making the model more interpretable. Feature selection was performed by studying feature importance, while hyperparameters were optimized with a Randomized Search, which used Repeated Stratified 5-Fold Cross-Validation with 5 repeats on the training set, optimized for the macro-averaged F1-score. The best configuration found used Entropy as splitting criterion, had a max depth of 14, a minimum of 5 samples per leaf, and a minimum of 8 samples in order to split an internal node.

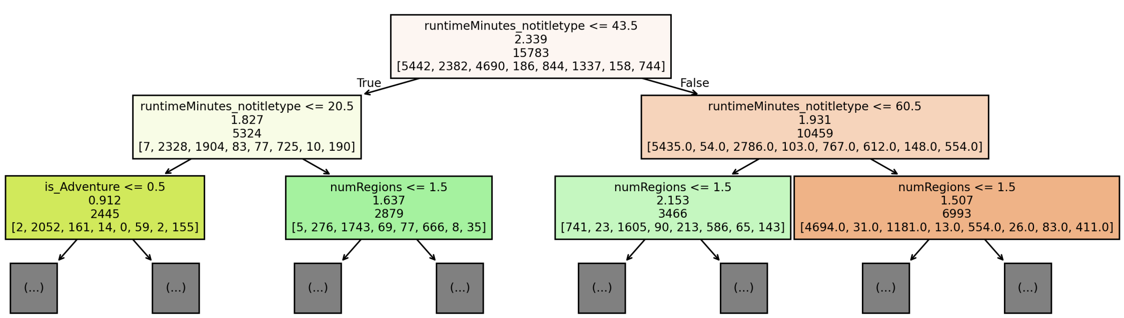
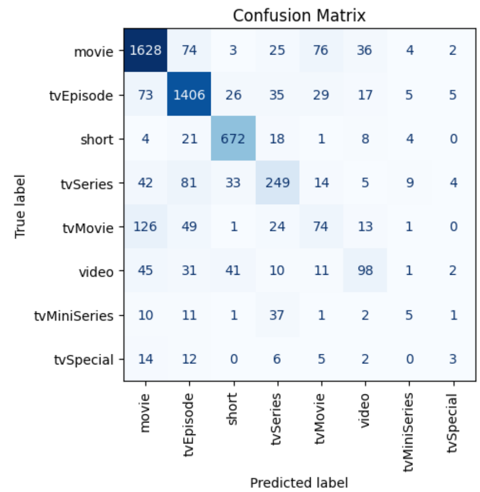


Figure 3.6: Decision Tree for multiclass classification

Figure 3.6 shows the Decision Tree obtained for the multiclass classification task. The most important feature for the model was `runtimeMinutes`, on which the first two levels of the tree are based, with a feature importance of 0.45. Three out of the four splits in the following level are based on `numRegions` being smaller than 2, giving the feature an importance of 0.13; the fourth is based on the *Adventure* genre, which has an importance of 0.01.

The model showed a general tendency towards overfitting, and many configurations were tested to prevent this. The overall accuracy shows a significant drop (from 0.86 on the train set, to 0.75 on the test set), as well the macro-averaged F1-score, going from 0.65 to 0.52. This was given from the *tvMiniSeries* and *tvSpecial* classes: these had f1-scores of 0.35, 0.22 respectively on the train set, and both had 0.10 on the test set. Since they were by far the least represented classes, they required a trade-off between low-represented classes classification and generalization. This can be seen in figure 3.7, which shows the fact that most predictions for these classes were misclassified.

Figure 3.7: DT Confusion matrix



In order to mitigate overfitting, post-pruning was tested, but did not give particular benefits. Higher values for the parameter α had minor positive effects on the generalization capabilities of the models, at the cost of losing predictions on the less represented classes.
Another aspect highlighted by the confusion matrix is the fact that *tvMovie* was often classified as *movie*, leading to a Recall value of 0.26 on the test set. This is likely due to the fact that the two classes are overlapping, since they are semantically related, and is further supported by the fact that the most common misclassification for *movie* was *tvMovie*, albeit with a low number of occurrences, likely due to it being the biggest class. A similar case can be made for *tvSeries* and *tvMiniSeries*, with 37 out of the total 68 of the *tvMiniSeries* records being misclassified as *tvSeries*.

3.2.4 Model Comparison - Multiclass Classification

The analysis of Decision Tree, Categorical Naïve Bayes, and K-NN models reveals several consistent patterns in their performance across this multiclass classification task. As expected, a common issue that was observed is the difficulty in correctly classifying underrepresented classes. Classes such as *tvMiniSeries* (186 records in the training set), *tvMovie* (844), and *tvSpecial* (158) consistently show low recall and F1-scores across all models. The poorest performance is observed for the first one, with recall values ranging from 0.01 (Naïve Bayes) to 0.07 (Decision Trees), and a maximum F1-score of 0.10 (Decision Trees). On the other hand, well represented classes, such as *movie* (5442 records in the training set) and *tvEpisode* (4690), are generally classified accurately, as previously discussed.
However, class imbalance alone does not fully explain the misclassifications. The confusion matrices showed systematic misclassification patterns, where minority classes were frequently absorbed into semantically related but more dominant categories. Across all models, but at different extent, *tvMovie* is frequently confused with *movie*; for instance, Naïve Bayes misclassifies 136 instances, resulting in a recall of 0.17. Similarly, *tvMiniSeries* is often predicted as *tvSeries*, e.g. resulting in 37 out of 68 instances misclassified by Decision Trees. K-NN also shows a tendency to predict *tvSeries* as *tvEpisode*. This is reflected in a very low recall (0.24) for *tvSeries* and in a quite high precision (0.76) for *tvEpisode*.
Putting these issues aside, Decision Trees achieved the best overall performance, with an accuracy of 0.79 and the highest value for macro average F1-score (0.52). Although all models performed well on the most represented classes, Decision Tree maintained greater stability even on the less represented classes, where the other models experienced a significant drop in performance. In the table 3.2 its overall results are summarized.

Accuracy	**Precision Macro Avg**	**Recall Macro Avg**	**F1-score Macro Avg**
0.79	0.55	0.51	0.52

Table 3.2: Overall metrics - Decision Tree

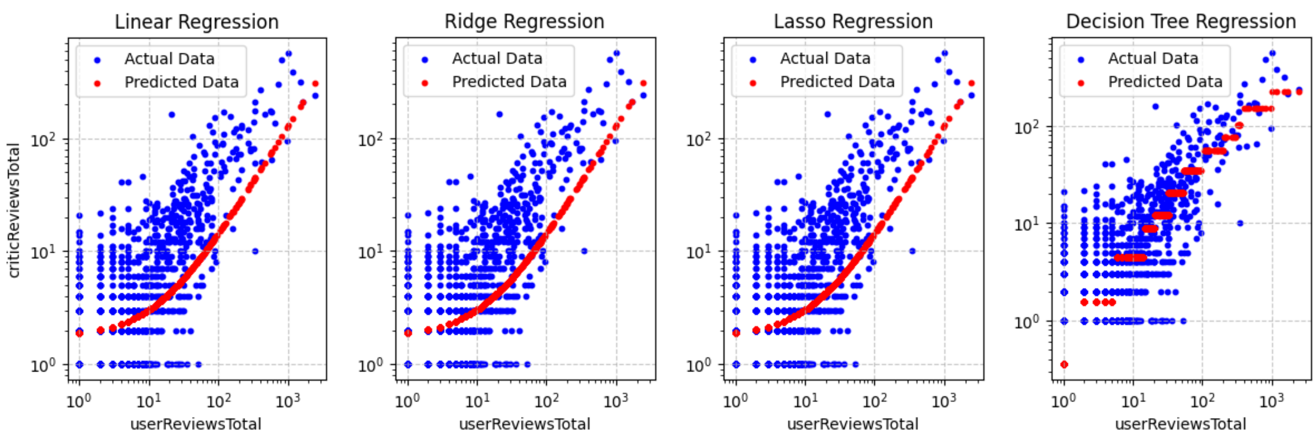
In summary, although these models struggle with minority classes, not discarding them represents a trade-off in favor of larger coverage and sensitivity across all classes.

4 Regression

Different regression techniques were applied to the dataset, testing on different combinations of attributes. The target variable chosen for Univariate and Multiple regression was `criticReviewsTotal`, while for Multivariate regression, the target variables were both `userReviewsTotal` and `criticReviewsTotal`. These were chosen because they offer important insights into the engagement that a product can generate, which also was the focus of the binary classification task in section 3.1.

4.1 Univariate and Multiple Regression

For Univariate Regression, the attribute `criticReviewsTotal` was chosen as the target variable. Aside from the semantic meaning, this choice was also made because it has a high correlation with the attribute `userReviewsTotal`, allowing univariate regression to be performed, while maintaining a clear separate semantic meaning. Figure 4.1 shows the results of the Univariate Regression task for all models but KNN, which doesn't provide an interesting visualization.



(a) Linear regressor (b) Ridge regressor (c) Lasso regressor (d) DT regressor

Figure 4.1: Univariate regression prediction results in Logarithmic space

	Intercept	Coefficient	MAE	MSE	R^2	test MAE	test MSE
Univariate							
Linear	1.76	0.13	0.007	0.25	0.36	0.007	0.33
Ridge	1.76	0.13	0.007	0.25	0.36	0.007	0.33
Lasso	1.76	0.13	0.007	0.25	0.36	0.007	0.33
DT	-	-	0.005	0.12	0.69	0.004	0.19
20-NN	-	-	0.005	0.14	0.66	0.004	0.22
Multiple							
Linear	-	-	0.000	0.000	1.000		
Ridge	-	-	0.000	0.000	1.000		
Lasso	-	-	0.000	0.000	1.000		
DT	-	-	0.000	0.000	1.000		
KNN	-	-	0.000	0.000	1.000		

Table 4.1: Classification report for binary classification

While Linear, Ridge and Lasso's predictions are similar in both
For Multiple Regression, the target variable was kept the same, in order to allow for a direct comparison of the results obtained with the two techniques.

4.2 Multivariate Regression

5 Pattern Mining

To perform this task, continuous attributes were discretized based on their distributions, aiming for bins that were both semantically meaningful and reasonably balanced in size. The selected numerical attributes (not normalized) for the pattern mining task, along with their binning, are:

Attribute	Binning
`runtimeMinutes`	VeryLowRTM (1-30), LowRTM (31-60), MediumRTM (61-90), HighRTM (91-220)
`rating`	VeryLowR (1-3), LowR (4-6), MediumR (7), HighR (8), VeryHighR (9-10)
`totalCredits`	VeryLowC (0-15), LowC (16-35), MediumC (36-65), HighC (66-15742)

Table 5.1: Binning of the continuous attributes

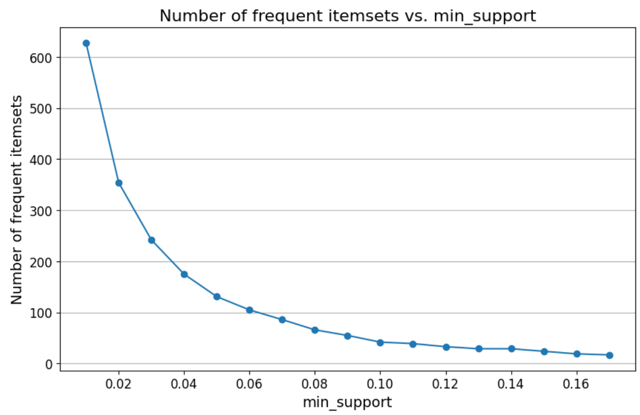
Regarding discrete attributes, `titleType` was considered for this task, kept in its original form. On the other hand, the values of each of the 7 attributes `countryOfOrigin_[continent code]`[1] were binarized into:

- `not_from_[continent code]`: when the value of the attribute is 0
- `is_from_[continent code]`: when the value of the attribute is ≥ 1

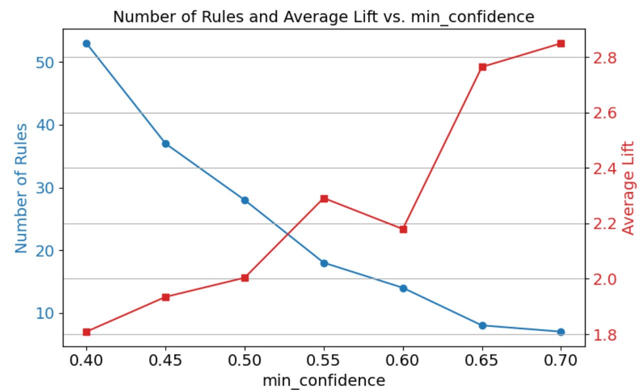
Binarization was preferred over creation of multiple bins, as it allows for a more linear interpretation of the results. An attempt was performed to include genres; however, this did not lead to more interesting results.

[1]see subsection 1.1.3 for the list of the features

5.1 Extraction and discussion of frequent patterns



(a) Plot of frequent itemsets with varying *support*



(b) Plot of lift and number of rules with varying *confidence*

Figure 5.1: Plots of minimum support and confidence - *Apriori* algorithm

Apriori was the algorithm chosen to extract frequent patterns. Figure 5.1a shows how the number of frequent itemsets changes with support values ranging from 0.01 to 0.18. The curve of the plot begins to flatten between 0.08 and 0.1, so a support value of 0.08 was selected, resulting in 66 frequent patterns.

It is interesting to observe the top frequent itemsets of size 1, 2, and 3, as shown in Table 5.2. From the itemset of size 1 it was noticed that approximately 48.5% of the objects in the dataset are from North America, highlighting the prevalence of this region.

The size-2 itemset reveals that 1/4 of the objects is both from North America and has a very low runtime. This pattern becomes more specific in the top size-3 itemset, where almost 10% of the data corresponds to TV episodes with both characteristics.

Size	**Support**	**Itemsets**
1	0.485	(is_from_NA)
2	0.204	(VeryLowRTM, is_from_NA)
3	0.094	(tvEpisode, VeryLowRTM, is_from_NA)

Table 5.2: Top itemsets of sizes 1, 2, and 3

5.2 Extraction of rules

After extracting the frequent patterns, association rules were generated. To find a value of *confidence* that balances the number of rules and their strength (measured with *lift*), the plot in Figure5.1b was analysed. A `min_confidence` of 0.55 was selected, guaranteeing an average *lift* of 2.3 and a significant number of rules, i.e. 18. The top 10 rules extracted (ranked by lift) are:

5.3 Exploiting rules for target prediction

One way to exploit the previously extracted rules is for target prediction. Firstly, rules with `VeryLowC` as target (rows 6 and 7 in the Table 5.3), show that short contents with very low runtime are highly likely to be associated with very low credits, probably reflecting the involvement of a limited production or cast. Both rules show a strong association, with a *lift* greater than 2.5, with the more specific one (VeryLowRTM and short) offering a better potential for targeted prediction.

Rule	Antecedents	Consequents	Ant. Sup.	Cons. Sup.	Sup.	Conf.	Lift
0	(short)	(VeryLowC, VeryLowRTM)	0.160	0.128	0.093	0.583	4.568
1	(VeryLowC, VeryLowRTM)	(short)	0.128	0.160	0.093	0.731	4.568
2	(HighRTM)	(movie)	0.177	0.319	0.154	0.866	2.719
3	(is_from_NA, short)	(VeryLowRTM)	0.082	0.375	0.082	0.995	2.654
4	(VeryLowC, short)	(VeryLowRTM)	0.094	0.375	0.093	0.994	2.650
5	(short)	(VeryLowRTM)	0.160	0.375	0.159	0.993	2.648
6	(VeryLowRTM, short)	(VeryLowC)	0.159	0.234	0.093	0.588	2.513
7	(short)	(VeryLowC)	0.160	0.234	0.094	0.587	2.511
8	(is_from_NA, LowRTM)	(tvEpisode)	0.121	0.303	0.090	0.742	2.447
9	(MediumRTM)	(movie)	0.229	0.319	0.165	0.720	2.260

Table 5.3: Top 10 rules extracted with *Apriori* (ranked by lift)

The target `is_from_NA` was then analysed to find the antecedents (as shown in Table 5.4), that increase the likelihood of an object of being from North America. Even though the *lift* value of those rules is below average, these rules were still considered, due to their meaningful interpretability. The analysis suggests that North American origin is associated with TV episodes, shorter durations, and high ratings or numerous production credits.

Rule	Antecedents	Consequents	Ant. Sup.	Cons. Sup.	Sup.	Conf.	Lift
12	(HighR, tvEpisode)	(is_from_NA)	0.144	0.485	0.092	0.639	1.317
13	(HighC)	(is_from_NA)	0.152	0.485	0.156	0.627	1.292
14	(tvEpisode, LowRTM)	(is_from_NA)	0.144	0.485	0.090	0.624	1.285
15	(tvEpisode)	(is_from_NA)	0.303	0.485	0.186	0.612	1.261
16	(tvEpisode, VeryLowRTM)	(is_from_NA)	0.153	0.485	0.093	0.611	1.259
17	(LowRTM)	(is_from_NA)	0.219	0.485	0.121	0.553	1.139

Table 5.4: `is_from_NA` as target