Data Mining 1 Project

Lorenzo Albani, Luigi Ascione, Tommaso Maitino

UNIVERSITÀ DI PISA

February 2, 2025

Contents

1 Introduction

This report documents the work completed for the Data Mining 1 project, the main objective of which is the analysis of an IMDb dataset that includes information on movies, TV shows, and other visual content. A structured analytical process has been undertaken to derive meaningful insights from the data. First, data preparation and understanding techniques have been applied to establish a robust analytical foundation. Subsequently, clustering methods have been employed to identify natural groupings within the dataset. Classification techniques have been implemented to categorize and predict outcomes based on established patterns. Regression analysis has been conducted to understand relationships between variables and make numerical predictions. Finally, pattern mining has been conducted to uncover significant relationships within the data. Through this systematic approach, substantial insights were extracted that could inform decision-making processes.

2 Data Understanding and Preparation

2.1 Data Semantics

The IMDb dataset compiles detailed information about movies, TV shows, and other visual entertainment formats, including community-generated ratings. Each entry contains essential details like the original title, release year, runtime, and the total number of user votes. The dataset also offers insights into significant factors such as awards, nominations, user reviews, and statistical ratings—ranging from top to lowest ratings, along with critic review counts. Additionally, it includes metadata such as the country of origin, genre, and the number of images and videos associated with each title. This dataset was last updated on September 1, 2024. The variables have been specified in table 1.

Name	**Description**	**Type**
originalTitle	Original title, in the original language.	object
runtimeMinutes	Primary runtime of the title, in minutes.	object
isAdult	Whether or not the title is for adults. 0: non-adult title; 1: adult title.	int64
startYear	Represents the release year of a title. In the case of TV Series, it is the series start year.	int64
endYear	TV Series end year.	object
numVotes	Number of votes the title has received.	int64
numRegions	The number of regions for this version of the title.	int64
worstRating	Worst title rating.	int64
bestRating	Best title rating.	int64
canHaveEpisodes	Whether or not the title can have episodes.	bool
isRatable	Whether or not the title can be rated by users.	bool
totalImages	Total number of images for the title on IMDb.	int64
totalVideos	Total number of videos for the title on IMDb.	int64
totalCredits	Total number of credits for the title.	int64
criticReviewsTotal	Total number of critic reviews.	int64
awardWins	Number of awards the title has won.	float64
awardNominationsExcludeWins	Number of award nominations excluding wins.	int64
titleType	The type/format of the title (e.g., movie, short, TV series, etc.).	object
rating	IMDb title rating class.	object
ratingCount	Total number of user ratings submitted for the title.	int64
countryOfOrigin	Country where the title was primarily produced.	object
genres	The genre(s) associated with the title (e.g., drama, comedy, action).	object
userReviewsTotal	Total number of user reviews.	int64

Table 1: IMDb Dataset Field Descriptions

2.2 Distribution of the variables and statistics

From the general statistics, it can be seen that the mean for **awardWins** is 0.49, while the median is 0. This highlights that at least 75% of units have never won an award, as indicated by the value of the third quartile. This attribute has a low standard deviation, meaning that the data are clustered near the mean. Figure 1 shows the average of award wins by title type, rating, year and country of origin. For **numVotes**, the average is much higher than both the median and the third quartile, suggesting that many titles have a low number of votes, while a few have a significantly high number of ratings. In this case, there is substantial variability, with the data distributed far from the mean.
Regarding **totalCredits**, the mean is somewhat higher than the median, with 75% of the titles having at least 65 credits. The high standard deviation value indicates that the data points are widely dispersed from the mean.
For the total count of critic and user reviews, it can be seen that, as before, the median is 0, while the mean is higher, meaning that at least 50% of samples have no reviews. On average, user reviews are more numerous than critic reviews. The statistics for **ratingCount** are very similar to those for **numVotes**, with only a few data points differing between the two variables.
Due to the high frequency and to the range of the data the distribution is not clear enough, the data needs to be transformed for a better analysis.

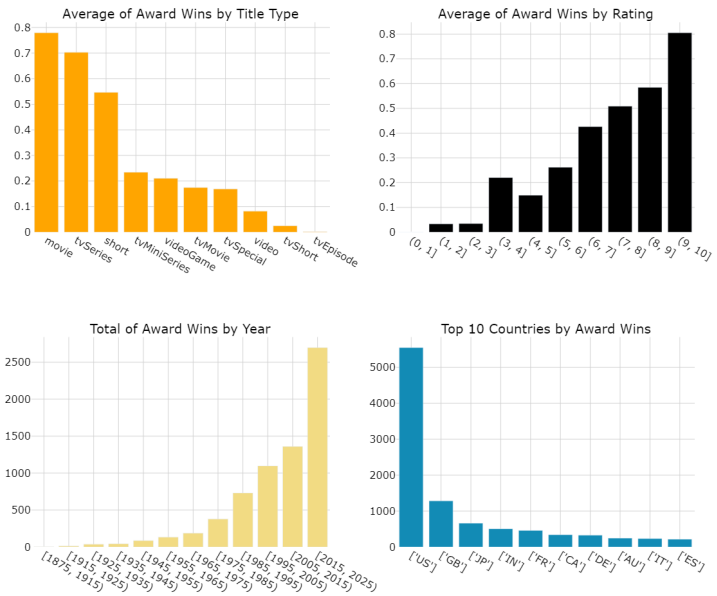
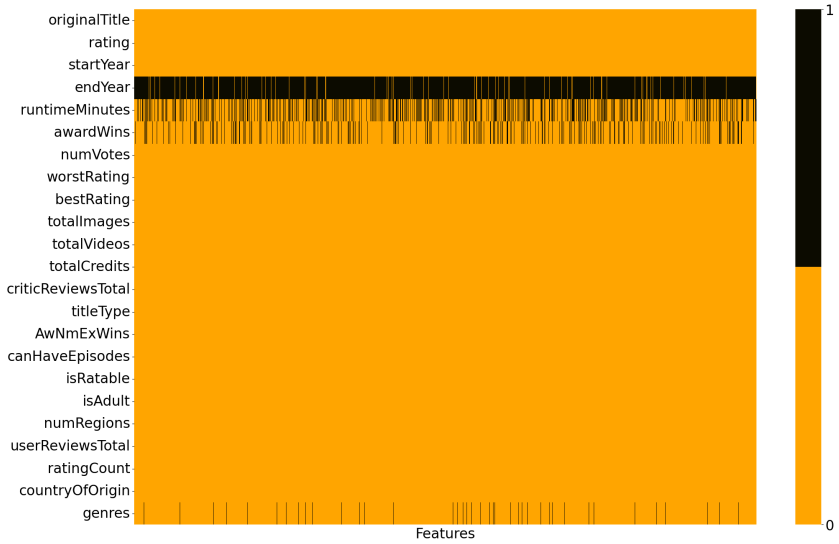


Figure 1: Distribution of **awardWins** related to **titleType**, **rating**, **startYear** and **countryOfOrigin**.

2.3 Assessing data quality

The attributes **endYear** and **awardWins**, originally of type object, are converted to float64. They are converted to float64 instead of int64 due to the presence of missing values, as these could not be converted to int64 values, which do not support NaN. The **isAdult** attribute, currently of type int64, is updated to boolean to better represent its binary nature. For the **runtimeMinutes** attribute, missing values are imputed with the average runtime, calculated based on the **titleType** grouping, except for the class Videogame. It was observed that 257 of 259 records belonging to this class lacked a value for **runtimeMinutes**. Replacing all missing values with the mean calculated from only two available samples would not have been methodologically appropriate, as these two samples would not have been representative of the entire class distribution. Therefore, a different approach was adopted: the missing values were replaced with 0, while preserving the original values of the two samples that contained actual **runtimeMinutes** data.

Attribute **awardNominationsExcludeWins** is renamed **AwNmExWins**.
Additionally, any $'/N'$ values in the dataset are replaced with NaN to standardize the handling of missing data.
The variable **endYear** has 15617 missing values, the 95% of the total. For this reason, it was decided to remove the corresponding column from the dataframe, as it may not contain useful information for the dataset. For **awardWins**, they were set to 0 because at least 75% of the records have won 0 awards. For **genre**, the rows where the value is null has been removed.
Figure 2 shows the distribution of missing values, summarized in table 2.



Variable	Missing Values
awardWins	2.618
runtimeMinutes	4.852
endYear	15.617
genres	382

Table 2: Resume of missing values.

Figure 2: The heatmap displays the data coded in boolean format to indicate missing values, where 1 represents missing data and 0 represents non-missing data.

Figure 3 shows the scatter matrix of the most interesting numeric variables for the research of outliers. On the diagonal, there are shown the univariate distributions of these variables.
Due to the data distributions within the variables, outliers are identified visually using scatterplots rather than calculated with the IQR method. This approach is preferred because the boxplots would be overly compressed and less informative.
The following outliers were eliminated:

- The two largest outliers from **runtimeMinutes**.
- The largest outlier from **AwNmExWins**.
- The two largest outliers from **userReviewsTotal**.
- The seven largest outliers from **totalCredits**.

Titles with 0 **totalCredits** were removed, as it is expected that every title should have at least one.

2.4 Variable transformations

The **ratings** are given as a range (ten equally spaced intervals from one to ten). They have therefore been replaced with the mean of the interval's endpoints. This change allows for more effective and feasible analysis.

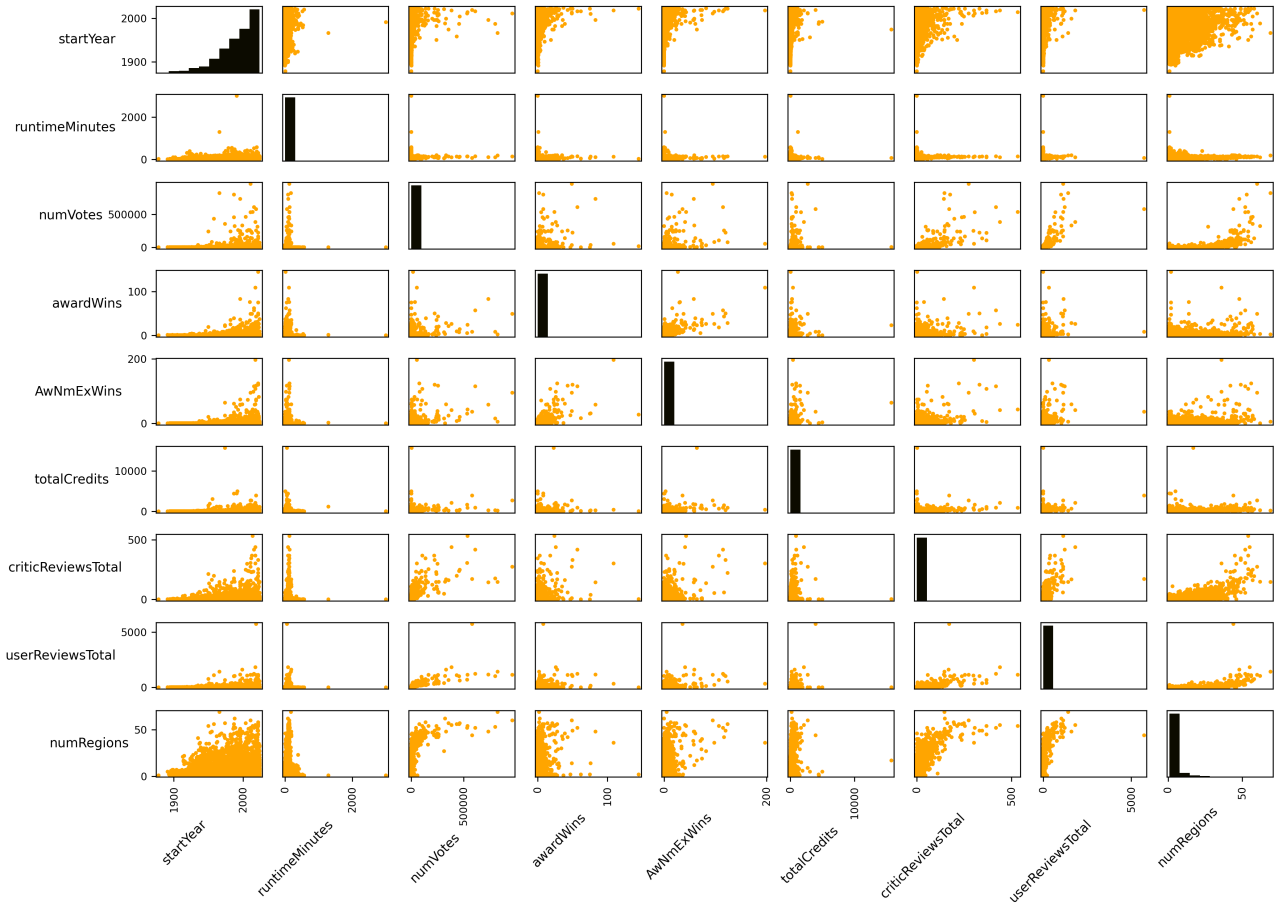


Figure 3: Scatter Matrix of the most interesting numeric variables for the research of outliers.

Attributes **userReviewsTotal** and **criticReviewsTotal** have been aggregated to create the attribute **ReviewsTotal**.
To make the distributions more homogeneous and reduce the impact of outliers without losing too much information, a logarithmic transformation is applied to some attributes. A small constant is added to all values to handle cases where the value is 0. The attributes transformed, for the above reasons, are respectively: **numVotes**, **awardWins**, **AwNmExWins**, **ReviewsTotal**, **totalImages**, **totalVideos**.
PCA was performed. The number of components was chosen based on the explained variance. According to Figure 4, two or three components were initially considered, but 5 components were selected as they represent 76% of the total variance.
From the estimation of the final communalities, all variables are well explained, except for **totalCredits**, which has a slightly lower value.
The PCA reveals the following insights:

- The first component is representative of several variables: **numVotes**, **totalImages**, **totalVideos**, **totalCredits**, **AwNmExWins**, **numRegions**, **ReviewsTotal**, and **popularityIndex**. This component can be interpreted as a general measure of popularity.
- The second component is strongly associated with **startYear** and **ratingMean**, making it a measure of recency and film rating.
- The third component has high loadings for **awardWins** and **AwNmExWins**, representing a dimension of the title's success.
- The fourth component distinguishes older and longer films with fewer videos and higher average ratings.
- The fifth component is a measure of modernity and duration.

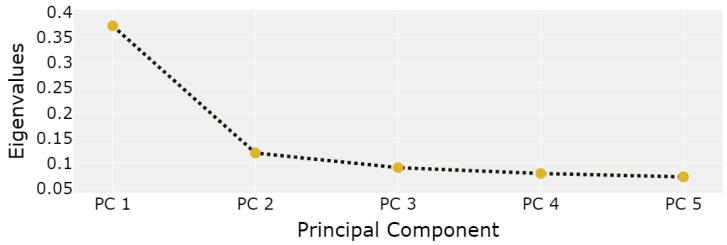


Figure 4: PCA Scree Plot

The PCA did not yield the expected results, and no further investigations were carried out.

2.5 Pairwise correlations and eventual elimination of variables

From the matrix shown in Figure 5, the correlation between each numeric variable in the dataset calculated using the Spearman correlation method can be observed.
It can be seen a high positive connection between **ReviewsTotal** and **numVotes**, as expected. So if the first variable increases the other one increases in average. **PopularityIndex** is high positively correlated with **totalVideos**. It has a slight connection with **numRegions**, so if the popularity of a title increases the other variables increases in average.
It was observed that the correlation between the variables **ratingCount** and **numVotes** is 1, leading to a more detailed analysis. By calculating the row-wise differences between the two columns, it was found that 88% of the rows are identical. For the remaining rows, the average difference is 0.1%. Due to these reasons the column **ratingCount** has been removed.
Other relevant positive correlation are between **awardWins** and **AwNmExWins**, between **numRegion** and **ReviewsTotal** and between **totalImages** and **numVotes**.
bestRating and **worstRating** were removed because the values in these columns are identical for almost all entries. Similarly, the attribute **isRatable** was eliminated for the same reason.

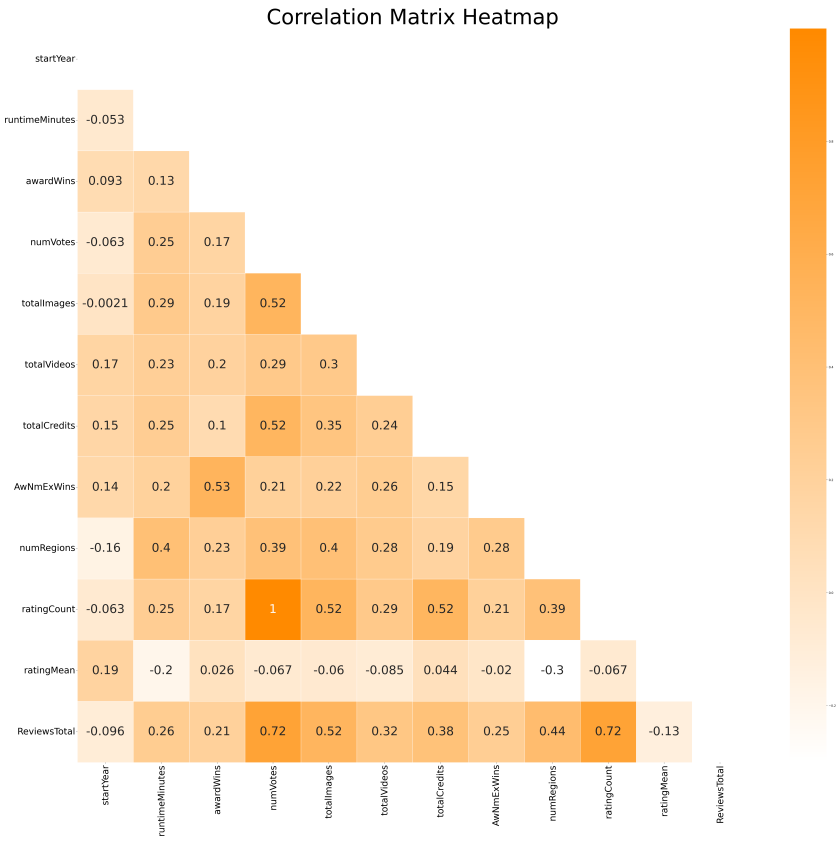


Figure 5: Correlation Matrix Heatmap, Spearman method.

3 Clustering

For clustering analysis the categorical variables have been eliminated. The remaining (numerical) variables from the dataset are respectively: **startYear**, **runtimeMinutes**, **awardWins**, **numVotes**, **totalImages**, **totalVideos**, **totalCredits**, **AwNmExWins**, **numRegions**, **ratingMean**, **ReviewsTotal**. In this phase, a comparative analysis was conducted between the dataset with log transformation and the dataset without it. Across all clustering algorithms, the log-transformed dataset demonstrated superior performances. Consequently, the presented graphs and results are based on the log-transformed data.

3.1 Analysis by centroid-based methods

In the analysis made by centroid-based methods it was chosen the min-max normalization, both for k-means and bisecting k-means, using the function `MinMaxScaler()` to transform the dataset values. For our purposes, this kind of normalization was preferred over the Z-score due to the non-uniform distribution of the values.

3.1.1 K-Means

In first place, the best number of clusters (K) was determined through the function `KneeLocator()`. Due to the poor quality of the results given by `KneeLocator()`, the choice of the number of clusters for the dataset have been made by looking at the SSE, Silhouette, average points for cluster and the cluster's points standard deviation graphs (Figure 6).

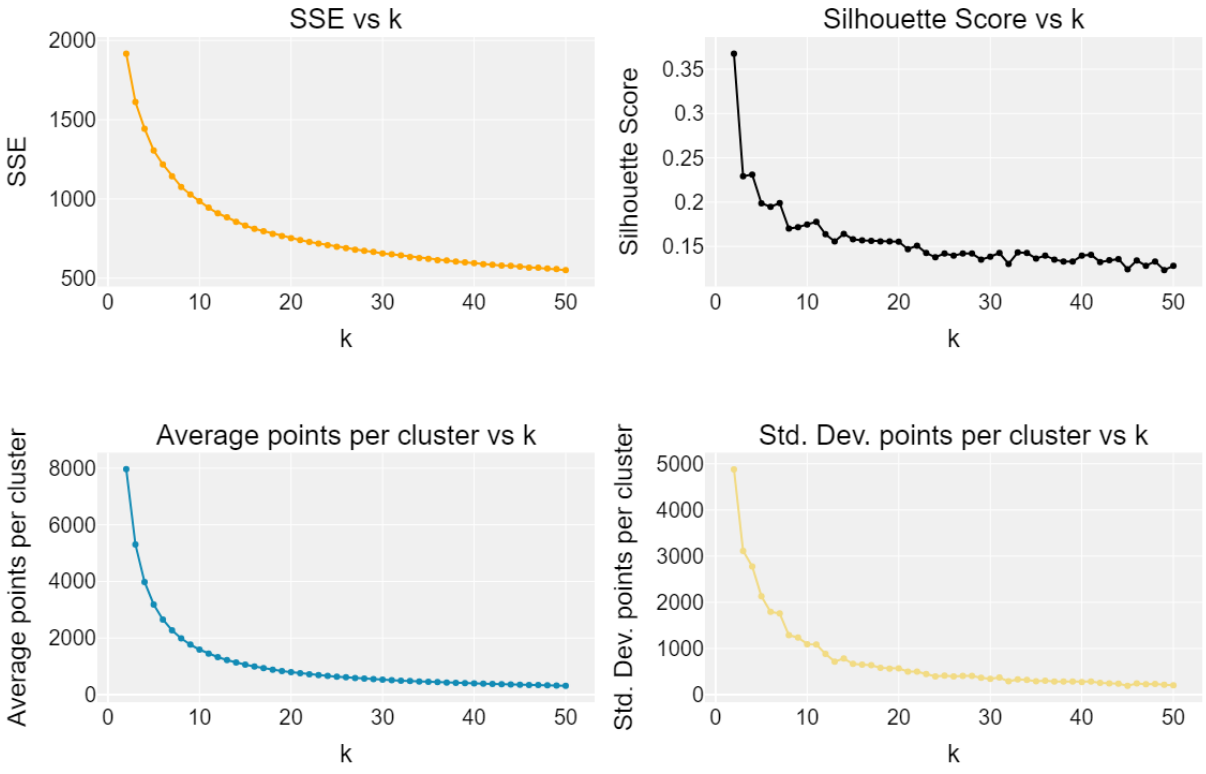


Figure 6: SSE, Silhouette Score, Average points per cluster and Std.Dev. points per cluster for every possible value of k.

The optimal number of clusters for k-means was chosen to be k=7, that leads to the following results:

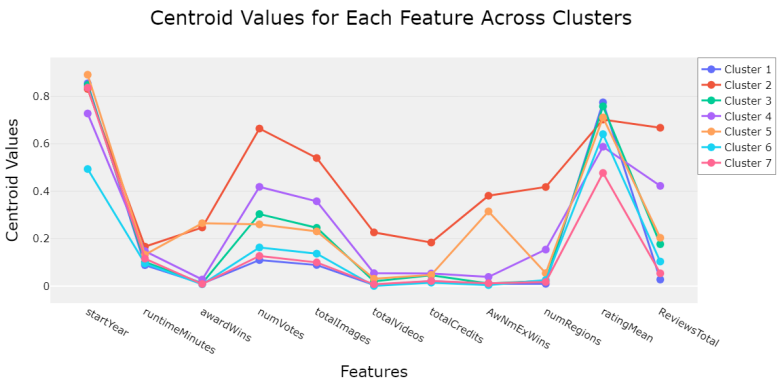


Figure 7: Centroid Values for each feature across clusters.

K = 7	Values
SSE	1141.29
Silhouette	0.198
Correlation S.I.M.	-0.436
Clusters Distribution (%)	[4.27, 16.47, 14.98, 2.19, 37.82, 16.05, 8.2]

Table 3: K=7 clustering results.

From Figure 7 can be seen that the most interesting features are respectively: **numVotes**, **totalImages**, **ratingMean** and **ReviewsTotal**. Graphically, the most interesting features can be identified based on the vertical distance of the centroids in the chart.
In sections 3.1.2 and 3.2, these variables were selected to best visualize the distribution of clusters within the graphs.
More detailed results can be found in the final discussion.

3.1.2 Bisecting K-Means

Another centroid-based clustering algorithm that has been used is Bisecting K-means. The split criterion chosen was to split the cluster with the highest SSE at each iteration. The algorithm was run with k=7 to be consistent with what was found for the K-means clustering in section 3.1.1. In order to confirm that 7 is a good choice of k, a silhouette plot was made for various values of k, using Silhouette Visualizer from the yellowbrick library. Figure 8 shows the silhouette plot for some choices of k and the scatter plot for the clustering with that k.. Again, k=7 seems to be a good choice, resulting in a good compromise between clusters balance and fewer number of points with negative silhouette.

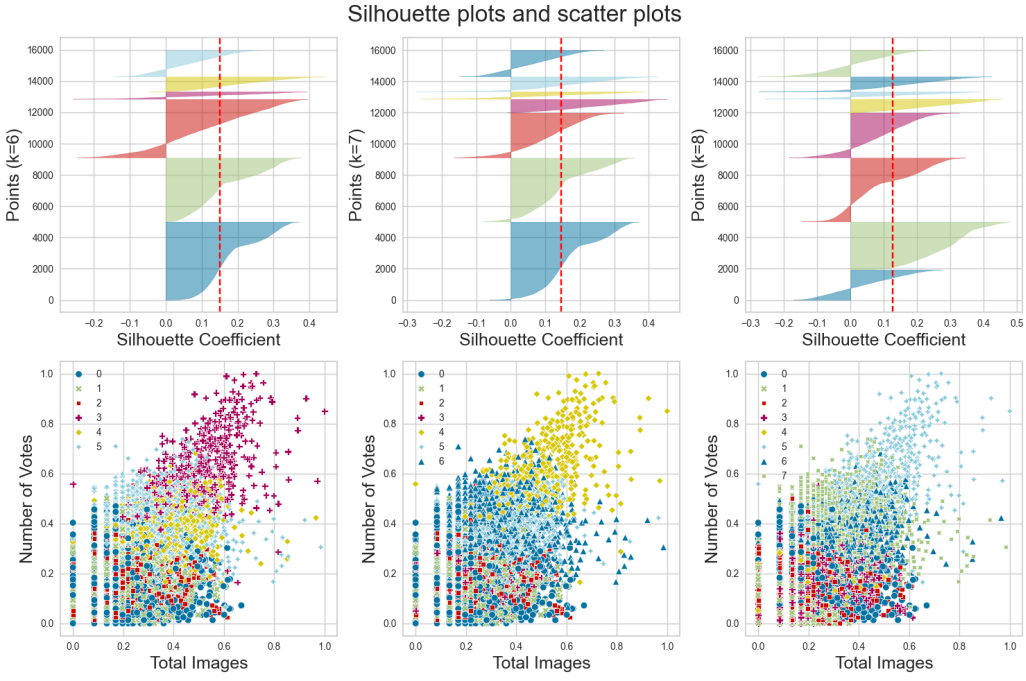


Figure 8: In the first row the silhouette plots for some choices of k for the Bisecting K-means algorithm. In the second row the scatter plots of the respective clusterings. The red dashed line represents the average silhouette.

The results of the clustering obtained with the Bisecting K-means algorithm, with k=7, are: $SSE = 1209$ and $sil = 0.15$.

3.2 DBSCAN

Density-based clustering algorithms like DBSCAN are highly versatile, as they can detect clusters of arbitrary shapes, unlike methods that rely on spherical or predefined structures. Additionally, it is robust against noise and outliers and eliminates the need to predefine the number of clusters, unlike in K-Means.
Also in this case it has been preferred min-max normalization over z-score to transform the dataset values.
The distances to the k-nearest neighbors for each point have been calculated using the `NearestNeighbors()` function. They were sorted and used for creating the k-distance graphs and to perform iteratively the `KneeLocator()` function to find the optimal ε for every k between 2 and 51.

	eps	min_samples	n_clusters	silhouette
0	0.10	2	425	-0.335382
1	0.10	4	97	-0.280003
2	0.10	8	26	-0.131305
3	0.10	16	15	0.004815
4	0.10	32	6	0.058086
...	...	...	...	...
95	0.19	64	1	-1.000000
96	0.19	128	1	-1.000000
97	0.19	256	1	-1.000000
98	0.19	512	1	-1.000000
99	0.19	1024	1	-1.000000

100 rows × 4 columns

Figure 9: Grid Search Output

Similar to K-Means, also in this case the `KneeLocator()` function produced poor quality results, so it was performed a grid search to find the optimal set of values for ε and $minPts$ (Figure 9).
There was used an exponential sequence of powers of 2 for $minPts$ values and a range of ε values from 0.10 to 0.20, chosen after identifying the optimal epsilon range through k-distance graphs. The grid search generated 100 results; for brevity, only a shortened output is presented in Figure 9.
The best result is: ε=0.18 $minPts$=4; $Silhouette = 0.20$.
In Figure 11 can be seen graphically the choice of epsilon.

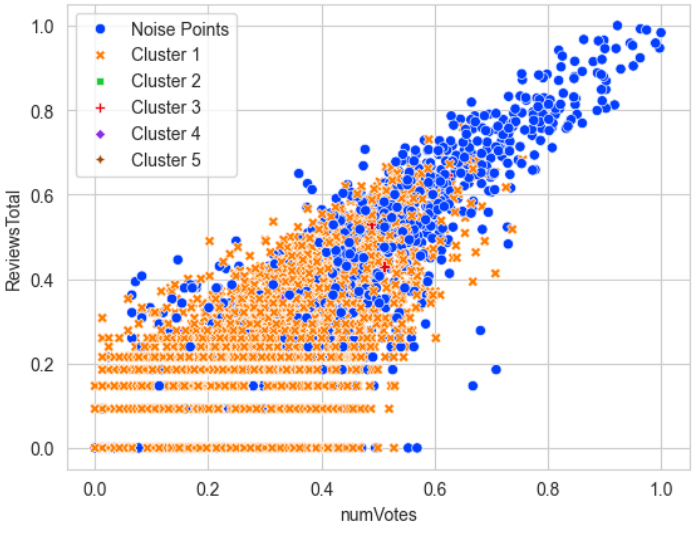


Figure 10: DBSCAN: ε=0.18, $minPts$=4, $Silhouette$=0.20.

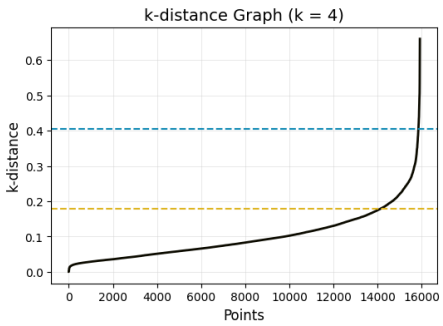


Figure 11: Distance from the 4th neighbor.
The blue dashed line is referred to `KneeLocator()` ε, the orange dashed line is referred to chosen ε.

In general, the grid search revealed that using DBSCAN consistently yields two possible outcomes: either a single large cluster with others very small clusters or more balanced clusters but with a high number of samples identified as noise points. Similarly, in this case (Figure 10), despite the creation of five clusters, the clustering remains highly unbalanced, with one very large cluster and the others significantly smaller.

3.3 Analysis by hierarchical clustering

For the hierarchical clustering analysis, the min-max transformation was applied, as it proved to be more suitable for this analysis.
Specifically, the metrics used were Euclidean and Manhattan. For both metrics, clustering was performed with the various linkage types: average, complete and single. Ward's method has only been used with the Euclidean metric as it does not support the Manhattan distance. Furthermore, the number of clusters was set to 7 to facilitate comparison with the k-means results. Figure 12 shows the dendrograms for the various types of linkage with the Euclidean distance.

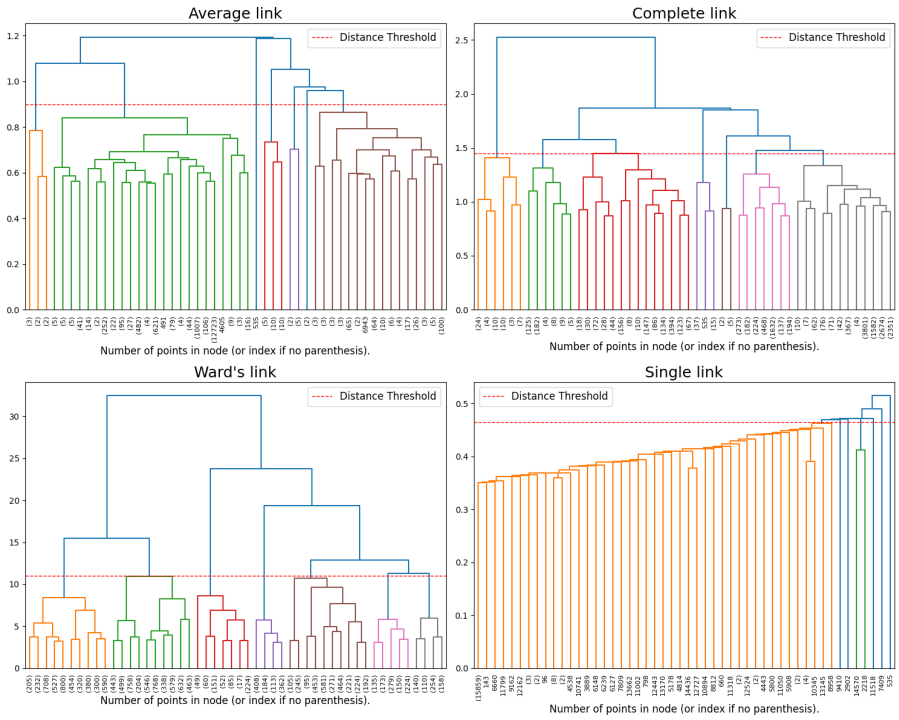


Figure 12: Dendrograms of hierarchical clustering performed with various types of linkage. For each of them the distance function used is Euclidean.

linkage	**metric**	**silhouette**	**clusters distribution (%)**
single	E	0.37	[99.94, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01]
	M	0.53	[99.94, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01]
complete	E	0.20	[8.27, 0.36, 69.36, 0.33, 0.04, 2.09, 19.52]
	M	0.28	[1.97, 0.57, 80.57, 16.32, 0.20, 0.23, 0.11]
average	E	0.52	[1.97, 97.75, 0.04, 0.04, 0.15, 0.01, 0.01]
	M	0.45	[98.24, 0.85, 0.68, 0.09, 0.08, 0.01, 0.03]
ward	E	0.12	[34.80, 30.05, 18.97, 1.11, 4.40, 4.24, 6.39]

Table 4: Silhouette score and clusters distribution for each type of linkage and metric (E=Euclidean, M=Manhattan).

The best results were achieved using Euclidean distance combined with Ward linkage.
Table 4 shows the silhouette scores and cluster distributions for each type of linkage.
It can be observed that, although the average linkage shows a higher silhouette score, this is primarily due to a highly imbalanced cluster distribution, with one large cluster and the remaining clusters being significantly smaller. In contrast, Ward linkage results in a lower silhouette score but provides a more

balanced cluster distribution. It can be seen that with single linkage it is not possible to obtain a good clustering, even changing the thershold distance to obtain a different number of clusters: this is probably due to the fact that single linkage is more sensitive to noise.

3.4 Final discussion

It can be concluded that DBSCAN does not provide optimal clustering, despite testing various combinations of epsilon and minPoints. In contrast, hierarchical clustering yields good results only with the ward linkage. However, the k-means algorithm is considered the most effective: although producing a slightly more unbalanced clusters distributions compared to hierarchical clustering, it achieves a higher silhouette score, indicating better-defined clusters. Even compared to bisecting k-means, the results for k=7 using k-means remain slightly better, with a lower SSE and a higher silhouette score. The clusters were subjected to an interpretative analysis:

- Cluster 1: stands out for containing the most recent titles (average start year: 2008) and having a relatively high average number of award wins (1.32).
- Cluster 2: stands out for having the second highest average runtime of movie (84.04 minutes) and a relatively high average number of votes (6.74).
- Cluster 3: stands out for containing the oldest titles (average start year: 1950) and having a relatively lowest average number of total credits (28.54). Therefore, they are old title from little productions.
- Cluster 4: it's the biggest cluster with 6014 record. It's the group with the shortest runtime minutes (mean 49.87) and it contain the "most unkown" title. It has the lowest average of **numVotes**, **numRegions** and **ReviewsTotal**.
- Cluster 5: stands out for having a relatively high average number of votes (5.39).
- Cluster 6: it's the smallest cluster with 350 units. Stands out for having the highest average runtime of movie (94.31 minutes), a very high average number of total credits (347.41) and it contains the "most known" movie with high production. It has the higest average of **numVotes, numRegions, ReviewsTotal** and **AwardNominationExcludeWins**.
- Cluster 7: this group has low value in all the characteristics but not the lowest, except for ratingMean with an average of 4.8. So, it contains the cluster with the worst movies in average.

Cluster	**Movie**		**Short**		**Special**		**TV Series**		**Video**		**Video Game**	
	abs %	**rel %**	**abs %**	**rel %**	**abs %**	**rel %**	**abs %**	**rel %**	**abs %**	**rel %**	**abs %**	**rel %**
1	6.84	61.76	5.59	20.29	5.41	1.18	1.60	14.56	1.18	1.32	2.74	0.88
2	20.31	47.54	31.54	29.70	3.38	0.19	9.57	22.57	0.00	0.00	0.00	0.00
3	18.89	48.60	14.70	15.21	14.19	0.88	9.59	24.84	26.54	8.50	21.46	1.97
4	4.36	76.57	0.00	0.00	0.00	0.00	1.18	20.86	0.26	0.57	3.20	2.00
5	21.86	22.28	42.15	17.28	59.46	1.46	48.83	50.12	57.78	7.34	41.55	1.51
6	11.07	26.60	4.25	4.11	14.86	0.86	26.02	62.95	9.80	2.93	29.68	2.54
7	16.66	78.33	1.78	3.37	2.70	0.31	3.20	15.16	4.44	2.60	1.37	0.23

Table 5: The table shows the relative and absolute percentage distribution of title types for each cluster created with the K-Means algorithm. For each title type, the relative percentage (*rel*) indicates the fraction of the cluster composed of that title type, while the absolute percentage (*abs*) represents the fraction of the total occurrences of that title type present in each cluster.

During the clustering the allocation of **titleTypes** within each cluster was analyzed. Table 5 shows the absolute percentage distribution of the types and the relative percentage distribution within each

cluster, emphasizing the internal structure of clusters. These insights provide a complementary view. Combining insights from Figure 7, Table 5 and the characteristics of groups, the following conclusions can be drawn:
Cluster 5 is composed primarily of TV series, also accounting for half of the total TV series. Concluding that it represents the most famous tv series. Cluster 2 consists of almost 80% movies and shorts. It can be assumed it represents primarily commercial movies and shorts. Since Cluster 1 is 80% composed of movies and shorts, we can infer that it primarily represents niche movie or short movie with a lot of award wins.

4 Classification

For classification, the chosen target variable is **titleType**. The variable has been remapped as follows: ['TV Series', 'Videogame', 'Movie', 'Shorts', 'TV Special', 'Video']. This remapping was deemed appropriate to better represent each group in the classification process.
originalTitle, **genres**, **countryOfOrigin**, **isAdult**. The following categorical variables were removed: **originalTitle**, **genres**, **countryOfOrigin**, **isAdult**. The variable **canHaveEpisodes** was converted into a binary format to make it suitable for the classification task.
Initially, a training set and a test set were provided. The training set was further split into a training set (60% of the samples) and a validation set (40% of the samples) to proceed with the parameters tuning. Stratification was applied during the data split to ensure that the class distribution is uniformly represented in both subsets. Before application on the test set, the built models were re-trained in the entire training set.
All models for each method within the classification were evaluated using the following metrics: accuracy, ROC and precision-recall curves, F1 score, and the confusion matrix.
All the results provided refer to the performance measure on the test set.

4.1 KNN

For the KNN algorithm, min-max normalization was applied using the `MinMaxScaler()` function.
A grid search was conducted to explore all KNN combinations, including: k values ranging from 5 to 120, weights set to either uniform or distance, and metrics set to either euclidean or cityblock. The `RepeatedStratifiedKFold()` function was used for cross-validation within the grid search to evaluate parameter combinations during the search process.
The best results were achieved with the combination of: n_neighbors = 10, weights = *distance* and metric = *manhattan*.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Movies	0.90	0.91	0.90	2092
TV Series	0.86	0.93	0.90	2084
Shorts	0.81	0.83	0.82	770
TV Special	0.00	0.00	0.00	46
Video	0.63	0.30	0.41	242
Videogame	0.92	0.28	0.43	83
Accuracy			0.86	5317
Macro Avg	0.68	0.54	0.58	5317
Weighted Avg	0.83	0.86	0.84	5317

Table 6: Classification report for KNN including precision, recall, F1-score, and support for each class.

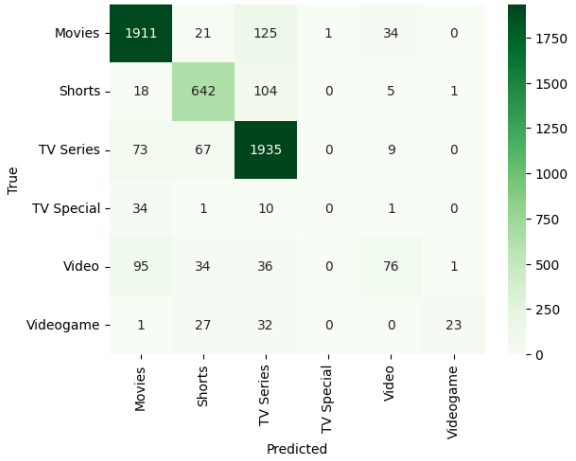


Figure 13: Confusion matrix of the KNN classifier.

In Table 6, the precision, recall, F1 score, and support for each class are reported. Additionally, the table includes the accuracy of the trained model, as well as the macro and weighted averages of each metric.
It can be observed from Table 13 that the best classification results are achieved for the classes **Movies**, **TV Series**, **Shorts**, and **Videogame**. Conversely, there are moderate results for the class **Video** and poor-quality results for the class **TV Special**.
To further analyze the model, the classes were binarized, and the following were created: a ROC Curve using the One-vs-Rest method to examine the false positive rate, and a Precision-Recall Curve, also using the One-vs-Rest method, to evaluate the false negative rate and gain a comprehensive perspective on the model's performance.

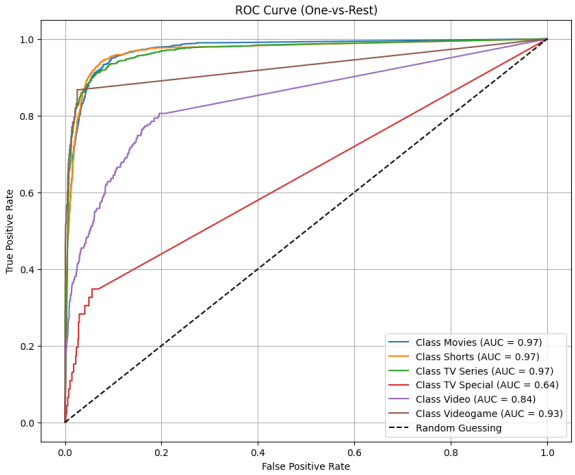


Figure 14: ROC Curve KNN, One-vs-Rest, AUC Score = 0.89.

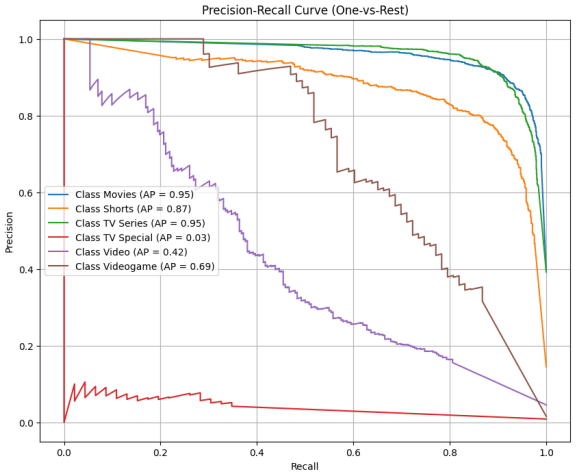


Figure 15: Precision-Recall Curve KNN, One-vs-Rest.

As it can be seen, both the ROC Curve (Figure 14) and the Precision-Recall Curve (Figure 15) confirm the excellent performance of the model in classifying the categories **Movies**, **TV Series**, **Shorts**, and **Videogame**, the moderate performance for **Video**, and the poor performance for **TV Special**, with the latter class approaching random guessing. This issue is particularly evident in the Precision-Recall Curve, where the average precision for **TV Special** is nearly 0.
These poor results for **TV Special** can be interpreted as a consequence of both the small number of records belonging to this **titleType** and the lack of features that distinguish it from the **Movies** and **TV Series** classes. In fact, in Table 13, it can be observed that records classified as **TV Special** are consistently misclassified as either **Movies** or **TV Series**, likely based on **runtimeMinutes** and the **canHaveEpisodes** variable.

4.2 Naïve Bayes

The Naive Bayes classifier is a classification algorithm based on Bayes' Theorem, with the assumption that features are conditionally independent given the class label.
In this study, we distinctly use two types of Naïve Bayes:

- **Gaussian Naive Bayes**: a variant applicable to continuous variables, where attributes are assumed to follow a Gaussian distribution. It predicts the output variable based on each parameter independently.
- **Categorical Naive Bayes**: suitable for discrete features that are categorically distributed.

Both models were trained with the CV cross validation.
With the Gaussian Naïve Bayes there's a moderate accuracy and, in general, a fair weighted avg for precision, recall and f1-support. **Movies** and **TV Series** have a high precision and a medium recall, **Shorts** has a good precision and high recall. For **Videogame** the model is very reliable and has a

very high recall. For **Video** and **TV Special**, the values are very low.

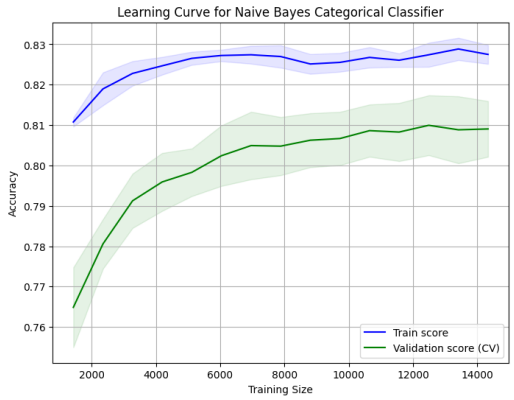


Figure 16: Learning Curve for Naive Bayes Categorical Classifier.

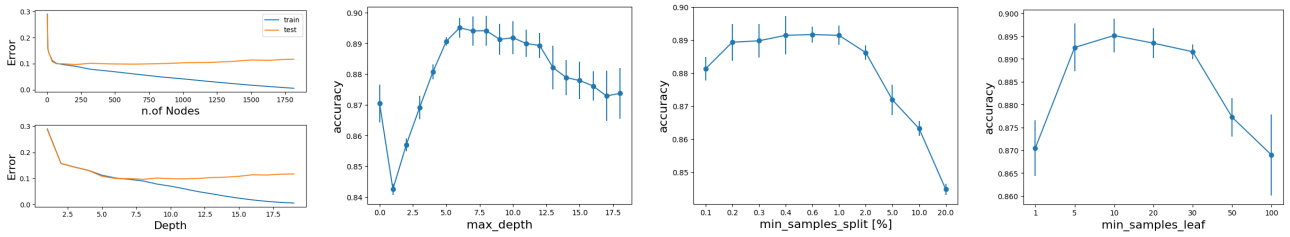
Class	Precision	Recall	F1-Score	Support
Movies	0.83	0.81	0.82	2092
TV Series	0.80	0.89	0.84	2084
Shorts	0.88	0.92	0.90	770
TV Special	0.00	0.00	0.00	46
Video	0.65	0.36	0.46	242
Videogame	0.71	0.06	0.11	83
Accuracy			0.82	5317
Macro Avg	0.65	0.51	0.52	5317
Weighted Avg	0.81	0.82	0.81	5317

Table 7: Classification report of Naive Bayes Categorical Classifier, including precision, recall, F1-score, and support for each class.

With the Categorical Naïve Bayes, the results of which are reported in Table 7, there's a high accuracy and high values in general in weighted precision, recall and F1-score. In **Movies**, **TV Series** and **Short** the three indicators have a high value. **Videogame** and **Video** have a good precision but a very low recall. The values for **TV Special** are zero so the model failed to correctly classify any examples of this class.
Looking at learning curve shown in Figure 16, the chosen split (60%/40%) ensures a balance between learning (training) and evaluation (validation). Validation initially shows higher variability (wider green area), but with a sufficient number of samples ($\sim 10,000$), the accuracy stabilizes.
In conclusion, categorical attributes produced a good model in terms of accuracy and weighted average of the three indicators. The Gaussian model, on the other hand, produced fair results but was worse than the categorical one.

4.3 Decision Trees

For the decision tree classifier, the variable **countryofOrigin** was also used, transforming it as follows: a binary column called **produced_in_USA** was created, which has a value of 1 if the sample was produced in the United States, 0 otherwise; this choice was made because the samples produced in the United States represent 44% of the whole dataset.
In first place, the base model was tested without setting any parameters. This model has a train accuracy of 1 and a test accuracy of 0.87, with a clear situation of overfitting. In order to avoid overfitting, a pre-pruning process has been applied, choosing the following parameters: the maximum depth of the tree (max_depth), the minimum number of samples required to split an internal node ($min\_samples\_split$) and the minimum number of samples required to be at a leaf node ($min\_samples\_leaf$). So, first the variation of training and validation errors with respect to the number of nodes and the depth of the tree has been observed. Then, the accuracy has been plotted against different values of the three parameters mentioned above, with the aim of exploring the range of the best values for each of them. For each of the three parameters, for the accuracy estimation, a 5-fold cross-validation was used. These graphs are shown in Figure 17.
Once the variation in accuracy across these parameters has been observed, a grid search was applied in order to identify the combination of parameters that produces the best model. All combinations were tested using both the **Gini index** and **entropy** as measure of impurity. Repeated stratified k-fold was used for the grid search, with $k = 5$ and 10 repetitions. The best parameters were found to be: $max\_depth = 9$, $min\_samples\_leaf = 9$, $min\_samples\_split = 0.3\%$, with *entropy* as criterion. The model with these parameters leads to an accuracy of 0.89, showing an improvement over the base model.
Subsequently, another model has been constructed using a post-pruning procedure, with the same aim



(a) (b) (c) (d)

Figure 17: (a) Plot of the training error and validation error vs number of nodes. (b) Accuracy variation w.r.t. *max_depth*. (c) Accuracy variation w.r.t. *min_samples_split*; *min_samples_split* is expressed as the percentage of the number of items of the dataset. (d) Accuracy variation w.r.t. *min_samples_leaf*. The error bars in all graphs represent the standard deviation.

of avoiding both overfitting and underfitting. To build this model, the tree was initially grown to its entirety, then pruned by the cost complexity pruning technique, parameterized by the cost complexity parameter *ccp_alpha*. Minimal cost complexity pruning recursively finds the node with the weakest link. As alpha increases, more of the tree is pruned, thus creating a decision tree that generalizes better, increasing the total impurity of its leaves. The training and validation sets accuracies were then plotted against *ccp_alpha* and the value of *ccp_alpha* that maximizes validation set accuracy was found to be $ccp\_alpha = 11 \cdot 10^{-4}$. Even in this case a better performance is obtained with entropy rather than with the Gini index. This model shows an accuracy of 0.90. All the values of the indicators used for the evaluation of the models are summarized in Table 8.

Model	**Train acc.**	**Test acc.**	**Precision**	**Recall**	**F-1 score**
Base	1	0.87	0.87	0.87	0.87
Pre-pruned	0.91	0.89	0.88	0.89	0.88
Post-pruned	0.91	0.90	0.88	0.90	0.88

Table 8: Performance metrics for different models of the decision tree classifier. All the values of *precision*, *recall* and *F-1 score* refers to the weighted averaged values.

In decision trees implemented in scikit-learn, it occasionally happens that a node split results in branches where all subsequent splits produce the same predicted label as the parent node. In these situations, such branches do not provide any additional predictive value. To address this, a function has been employed to streamline the tree by eliminating these redundant branches, retaining only the root node of the branch. This approach helps to reduce the tree's complexity, enhancing both its interpretability and its visualization.
For our problem, the model built with post-pruning is considered the best one, as it has slightly better accuracy and recall than the other model. To conduct a more in-depth analysis of the model performance, ROC curves, precision-recall curves and the confusion matrix were observed. They're shown respectively in Figures 18, 19 and 20.
The decision tree built with the chosen model has 163 nodes and it is shown in Figure 21, cut down to $max\_depth = 2$ in order to make it more understandable.
An analysis of the results has shown that Movies, Tv_series_related, and Shorts have been classified in an excellent way. However, Videos have often been misclassified as other **titleType**, resulting in a low recall (0.15), which is likely due to the low support for this class. The classifier fails to correctly classify the Specials, whereas Videogames have been perfectly classified.
It is also possible to see the importance of the features in the classification model: the most important one is **runtimeMinutes**, as we can also see from the first nodes of the tree in figure 21, which is clearly the most distinctive variable for title types; the second features in order of importance is **canhaveEpisodes**, as it helps to classify TV series. Based on the analysis of features importance, a decision tree classifier model has been constructed without the irrelevant features, as in case of a large number of irrelevant attributes, some of them may be accidentally chosen during the tree-growing process. So,

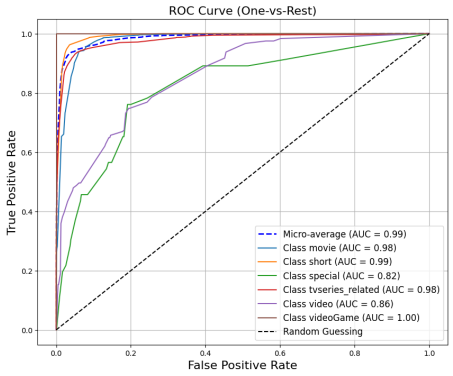


Figure 18: ROC curves for each class for the decision tree post-pruned with *ccp_alpha* and entropy as impurity measure.

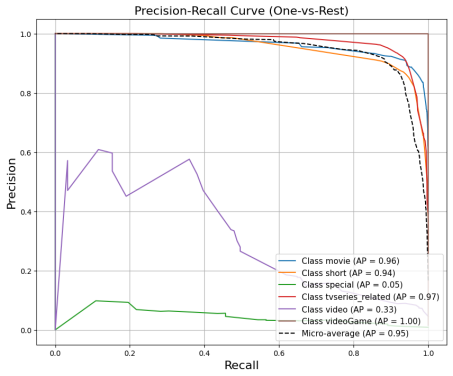


Figure 19: Precision-Recall curves for each class for the decision tree post-pruned with *ccp_alpha* and entropy as impurity measure.

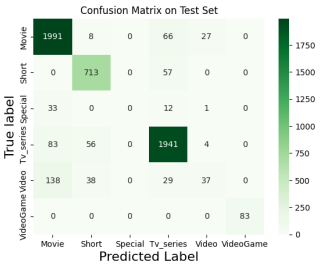


Figure 20: Confusion matrix of the decision tree post-pruned with *ccp_alpha* and entropy as impurity measure.

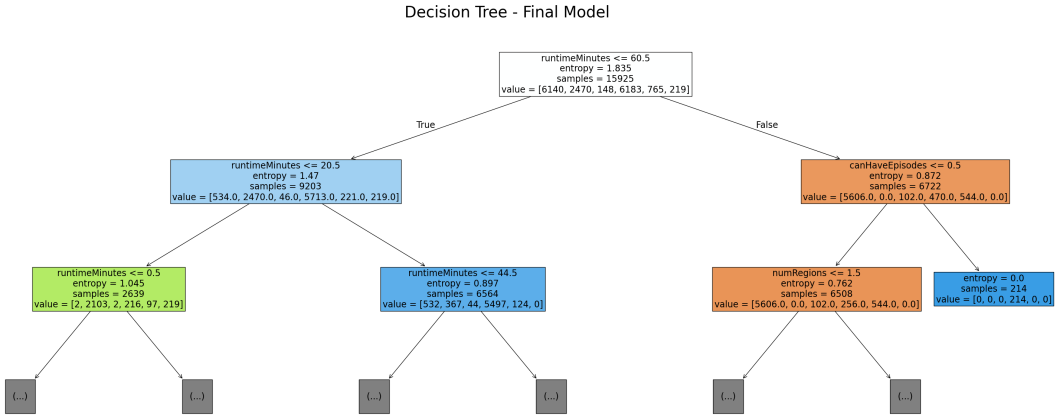


Figure 21: Start of the decision tree built with the post-pruning model. $ccp\_alpha = 11 \cdot 10^{-4}$, criterion: entropy.

to build this model, the features with an importance below 1% have been removed: **awardWins**, **numVotes**, **AwNmExWins**, **ReviewsTotal**, **totalVideos**, **ratingMean**. However, the results do not show an improvement.

4.4 Final discussion

The results obtained through KNN and Decision Tree are both excellent, although in terms of statistical metrics, the Decision Tree classifier proved to be the most performant among the three. The main difference between the two lies in the classification of the "Videogame" class, which is correctly classified in all cases examined by the Decision Tree classifier, whereas KNN achieves great but not flawless results. As for Naive Bayes, the Gaussian version produces unsatisfactory results, while the categorical version comes closer to the performance achieved by KNN. All three classifiers fails to predict Special, due to the low support of this class.

5 Regression

In this section, various types of regression are explored, including simple linear regression, multiple regression, ridge regression, Lasso regression, and advanced techniques such as decision trees and KNN models. For the models, k-fold cross validation and a grid search was applied to find the best parameters with respect to R^2 (an indicator of good adaptability), MSE (mean squared error), and MAE (mean

absolute error between predicted and true values). For ridge regression and lasso regression, the range of the alpha parameter was set between 0.0001 and 0.1. For the decision tree, a random search was conducted for max depth between 1 and 20, minimum split samples between 2 and 20, and minimum leaf samples between 1 and 20. For K-NN, the k-values ranged from 1 to 128. All data are z-standardized for the algorithms based on distances, aiming to improve the performances.

5.1 Simple Regression

For the simple regression two models have been built, one for numbers of votes with total reviews, and average rating of the movies with start year, the variables with the highest correlation. For the first model the two variables have a high correlation, in contrast, average rating has a very low correlation with all the variables of the dataset. The highest one is with **startYear** and it's about 0.1, thus our expectations are low.
The results are displayed in Table 9:

Models	**X**	**Y**	**Parameters**	**Coeff**	**Interc**	**R2**	**MSE**	**MAE**
Lin. Regression	ReviewsTotal	numVotes	-	1.206	3.017	0.647	1.052	0.79
	startYear	averageRating	-	0.011	-15.401	0.032	1.691	0.989
Ridge	ReviewsTotal	numVotes	alpha: 0.0001	1.206	3.017	0.647	1.052	0.79
	startYear	averageRating	alpha: 0.0001	0.011	-15.401	0.032	1.691	0.988
Lasso	ReviewsTotal	numVotes	alpha: 0.0001	1.206	3.017	0.647	1.052	0.79
	startYear	averageRating	alpha: 0.0001	0.011	-15.401	0.032	1.691	0.988
DecisionTree	ReviewsTotal	numVotes	max_depth: 5, min_samples_leaf: 3, min_samples_split: 13	-	-	0.648	1.049	0.79
	startYear	averageRating	max_depth: 4, min_samples_leaf: 14, min_samples_split: 19	-	-	0.041	1.676	0.982
KNN	ReviewsTotal	numVotes	n_neighbors: 99	-	-	0.64	1.079	0.79
	startYear	averageRating	n_neighbors: 127	-	-	0.034	1.687	0.988

Table 9: Simple Regression performances of Linear, Ridge, Lasso, Decision Tree and KNN models.

For **numVotes**, the model that performs slightly better is the Decision Tree. The model explains 64.8% of the variability in the data based on **totalReviews**. This means that there is a good relationship between the two variables, and the model can capture a substantial portion of the variation in the data. Therefore, the number of votes increases of 1.206 units for each increased unit of the number of reviews.
As expected, the second model has a very poor result. This means that there is no good relationship between **numVotes** and **startYear**. The model cannot capture a substantial portion of the variation in the data (3.4%). In other words, the start year of a movie does not seem to significantly influence the number of votes it receives.

5.2 Multiple Regression

In multiple regression, the target variables remain the same, but the models are trained on all other variables to improve results and predictions. During linear regression training, the presence of multicollinearity is verified by calculating the VIF (Variance Inflation Factor) score for the variables. None of them presents a value greater than 10, indicating that there is no multicollinearity problem that could affect the interpretation of the coefficients.
As expected, the results are better with multiple regression. For both target variables, the best model is KNN. The first algorithm predicts **numVotes** well, with an explained variability of 70.9%. The second one has low adaptability to the data, but compared to simple regression, we improved the R^2 value from 3.4% to 20%. In the linear regression, ridge and lasso, with the target **numVotes**,

Models	Y	Parameters	R2	MSE	MAE
Lin. Regression	numVotes	-	0.687	0.933	0.734
	averageRating	-	0.08	1.608	0.955
Ridge	numVotes	alpha: 0.0001	0.687	0.932	0.734
	averageRating	alpha: 0.0001	0.08	1.608	0.955
Lasso	numVotes	alpha: 0.0063	0.688	0.932	0.734
	averageRating	alpha: 0.0029	0.08	1.608	0.955
DecisionTree	numVotes	max_depth: 9, min_samples_leaf: 15, min_samples_split: 16	0.701	0.891	0.71
	averageRating	max_depth: 7, min_samples_leaf: 19, min_samples_split: 3	0.2	1.398	0.87
KNN	numVotes	n_neighbors: 32	0.709	0.867	0.701
	averageRating	n_neighbors: 66	0.209	1.382	0.865

Table 10: Performance metrics of Linear, Ridge, Lasso, Decision Tree, and KNN models.

we calculated the values of coefficients, and the higher value is for the instance **ReviewsTotal**. The coefficient value is about 1.173, suggesting that for each unit increase of the number of reviews, an increase of 1.173 units in **numVotes** is expected, holding all other variables constant. With the target **averageRating**, again, the most relevant feature is **ReviewsTotal** with a value of -0.250. It's not a so high value, but there is an inverse relationship between the two variables.

5.3 Multivariate Regression

In this phase, two variables have been chosen as targets: the number of votes and the number of regions where the titles are distributed. These variables as target are useful for analyzing the popularity, the distribution and the success of the title. As before, we checked the presence of multicollinearity calculating the VIF.

Models	Y	Parameters	R2	MSE	MAE
Lin. Regression	numVotes, numRegions	-	0.598	8.333	1.678
Ridge	numVotes, numRegions	alpha: 0.0001	0.598	8.333	1.678
Lasso	numVotes, numRegions	alpha: 0.0193	0.598	8.348	1.678
DecisionTree	numVotes, numRegions	max depth: 11, min samples leaf: 17, min samples split: 9	0.611	7.389	1.48
KNN	numVotes, numRegions	n neighbors : 13	0.686	5.669	1.35

Table 11: Performance metrics of Linear, Ridge, Lasso, Decision Tree, and KNN models.

Among the models tested, the KNN model performed the best, explaining 68.6% of the variance. KNN is effective in capturing the relationship between predictors and target variables. For the linear model, the highest coefficients are still for **ReviewsTotal**, suggesting that this variable has the most significant impact on the number of votes and regions.

5.4 Final discussion

In all these models, it is highlighted that the most significant variable, whether positively or negatively, is the total reviews (from users and critics). Therefore, we tried to build a model from another perspective. We constructed another Multiple Regression where the target is **ReviewsTotal** and we have obtained the best results so far for regression: the KNN algorithm captured about 74% of the information, with an MSE of 0.373 and an MAE of 0.461.
For all the regressions, non-linear models adapted better to the data. Specifically, for simple regression, the decision tree performed better, while for multiple and multivariate regression, KNN was the best.

6 Pattern miningn

For patterns and association rules mining the features **isAdult**, **awardWins**, **AwNmExWins** and **totalVideos** were removed, having respectively 97%, 89%, 88% and 90% of values equal to zero: the high frequency of these attribute-value pairs could generate redundant patterns, resulting in rules that do not carry new meaningful information. The attributes **countryOfOrigin** and **ReviewsTotal** were binarized. The former was transformed into a binary feature indicating whether the sample was produced in the USA (1) or not (0): this transformation was applied because 44% of the samples were produced in the USA and the remaining ones are originated from a wide variety of countries. The latter variable, **ReviewsTotal**, was binarized to indicate whether a film had at least one review (1) or no reviews (0): this was done because 48% of the samples has no reviews. For the categorical variables **rating**, **titleType** and **genres** it was decided to reduce the number of categories by remapping the possible values into four categories, since some values have a low frequency. The remapping was done by trying to obtain classes that are as balanced as possible, without losing meaning. Furthermore, for the attribute **genres**, a new binary column was created for each distinct possible category (*one-hot encoding*), in order to handle the cases in which an element is represented by more than one genre. The continuous attributes have been discretized each in four bins, with equal frequency binning, except for the feature **numRegions**, discretized in three bins.
Given its efficiency, FP-growth was selected as the preferred algorithm over Apriori, as it offers faster processing times.

6.1 Frequent pattern extraction

As a first step, an analysis of frequent, closed, and maximal itemsets has been performed, varying the parameters *min_sup* (support threshold) and *z_min* (minimum number of items in an itemset). Figure 22 shows the variation in the number of frequent, closed, and maximal itemsets as *min_sup* changes, with *z_min* fixed at 2.

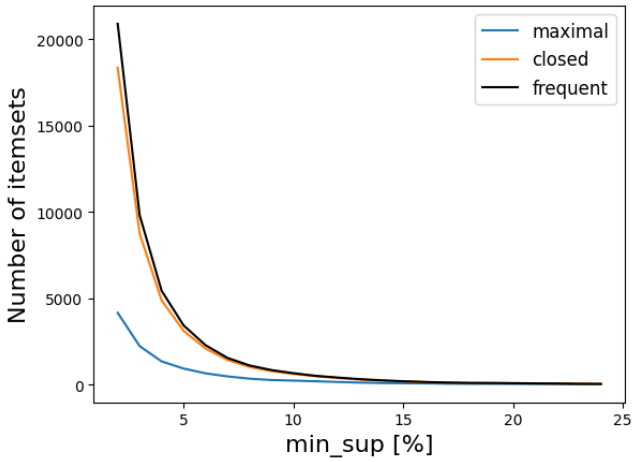


Figure 22: Number of frequent (black line), closed (orange line), and maximal itemsets (blu line) as the minimum support threshold (*min_sup*) changes. *z_min* = 2.

A range of *min_sup* values from 10 to 50, in steps of 10, was explored for all possible values of *z_min*. The most frequent itemsets, the only ones that exceed 50% support, which are also closed and maximal, are: ['1 regions', 'can't have episodes'], *supp* = 53.86%; ['genre Comedy/Family', 'no_episodes'], *supp* = 50.05%. Obviously, the most frequent ones are among those with two items, but they appear trivial and therefore not very interesting. More interesting itemsets are found with a larger number of items. Those with the highest cardinality found have five items. The three most frequent patterns found with five items, that are also closed and maximal, are:

- `[less than 30 minutes, reviews, genre Comedy/Family, 1 region, can't have episodes]`, supp=10.59%;

- `[Reviews, genre Comedy/Family, not produced in US, 1 region, can't have episodes]`, supp=10.5%;
- `[high number of votes, high number of regions, movie, No reviews, can't have episodes]`, supp=10.30%.

6.2 Association Rules

Using the frequent patterns identified in section 6.1, association rules were extracted. An initial investigation was conducted to examine how the number of rules and their lift are affected by varying the minimum confidence threshold *min_conf* in a range from 50% to 95%, in a step of 5%. The histograms in Figures 23 and 24 depict the distribution of confidence and lift values for the generated rules.

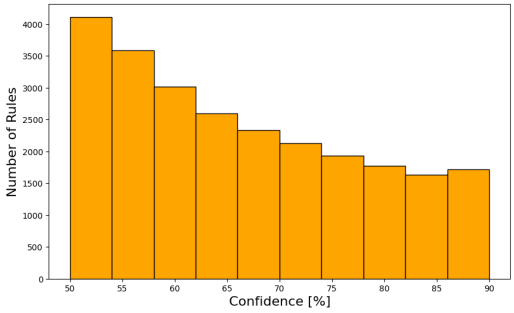


Figure 23: Number of rules generated varying the minimum confidence threshold *min_conf*.

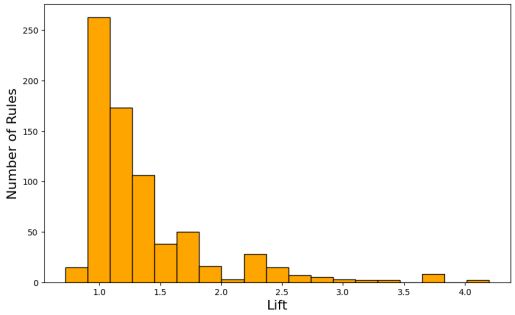


Figure 24: Number of rules per lift at a fixed confidence of $min\_conf = 50\%$.

The most useful extracted rules have been exploited in an attempt to predict a target variable, using the same test set employed for classification in section 4. For this purpose, focusing on rules where the consequent is a **titleType**, for each distinct values the rule with the higher lift is selected and used for the predictions. These predictions are evaluated using their respective confusion matrices, calculating precision, recall, accuracy, and F1-score. The results are summarized in table 12.

Consequent	**Antecedent**	**Support**	**Conf.**	**Lift**	**Acc.**	**Precision**	**Recall**
short	less than 30 min., genre Comedy/-Family, can't have episodes	14.68%	65%	4.10	0.90	0.63	0.92
movie	more than 90 min., many regions, can't have episodes	10.37%	98%	2.56	0.71	0.97	0.27
tv series related	(29.0, 53.0] minutes, genre Drama/Crime	10.89%	95%	2.44	0.71	0.96	0.27

Table 12: Association rules metrics and evaluation metrics of the predictions made with the rules with higher lift for each distinct **titleType** value.

The results shown in table 12 shows that in both cases with low recall, the specificity of the antecedent limits the rule's coverage. However, the high confidence and precision indicate that, when the rule is activated, it correctly predicts the consequent, even if it only applies to a small fraction of the dataset. In general, the high lift in all cases suggests that the rules are good at correctly predicting the consequent when applied, but their low recall implies that they fail to capture a large proportion of actual cases. The first rule, the one with the highest lift value, also has the highest recall: the rule captures the majority of actual cases of the consequent. This occurs because the support is sufficient for the rule to apply to a significant number of cases. However, the lower confidence is reflected in a

lower precision.
No rules were found with the consequent **other** due to the low support of this class.