



UNIVERSITÀ DI PISA



1st Project Report

Giuseppe Muschetta

Master's Degree Program in Computer Science
Specialization in Data Science and Business Informatics

University of Pisa, Academic Year 2024/2025

Contents

1	Data Understanding & Preparation	1
1.1	Introducing the variables with their meaning and characteristics	1
1.1.1	Numerical Variables	1
1.1.2	Categorical Variables	1
1.2	Distribution of Categorical Variables, Pairwise Correlations and Tranformations	1
1.2.1	Categorical Variables Correlation using Cramér's V	3
1.2.2	Analysis of Results	3
1.2.3	Categorical Variables Transformation	3
1.3	Distribution of Numerical Variables, Pairwise Correlations and Transformations	4
1.3.1	Correlations of Numerical Variables and Variable Elimination	4
1.3.2	Pearson and Spearman Matrix Correlation Analysis	4
1.3.3	Feature Elimination Summary	5
1.3.4	Numeric Variables Transformation	6
1.4	Handling Missing and Placeholder Values	6
1.5	Outliers Detection and Removal with LOF	7
2	Clustering Analysis	7
2.1	Clustering: K-Means	7
2.1.1	Feature Selection and Preparation	7
2.1.2	Determining the Optimal Number of Clusters (K)	8
2.1.3	Cluster Visualization and Profiles	8
2.1.4	Discussion of the Clusters	8
2.2	Clustering: DBSCAN Analysis	9
2.2.1	Feature Selection and Parameter Strategy	9
2.2.2	Advanced Parameter Search and Results	10
2.2.3	Deep Cluster Analysis	10
2.2.4	Cross-Analysis with K-Means and Conclusion	11
2.3	Agglomerative Clustering	11
2.3.1	Methodology and Linkage Method Comparison	11
2.3.2	Determining the Optimal Number of Clusters (k)	12
2.3.3	Cluster Profiles and Interpretation	12
2.4	Clustering: Final Discussion	13
3	Classification	14
3.1	Classification by Decision Tree	14
3.1.1	Target Variable Definition	14
3.1.2	Baseline Model Performance	14
3.1.3	Hyperparameter Tuning and Scoring Metric Selection	15
3.1.4	Tuned Model Evaluation	15
3.1.5	ROC Curve Analysis	15
3.1.6	Feature Importance Analysis	15
3.1.7	Conclusion	16
3.2	Classification by K-Nearest Neighbors (KNN)	16
3.2.1	Baseline Model Performance	16
3.2.2	Hyperparameter Tuning Strategy	16
3.2.3	Tuned Model Evaluation	17
3.2.4	Visual Performance Analysis	18
3.2.5	Conclusion	18
3.3	Classification by Naive Bayes	18
3.3.1	Gaussian Naive Bayes Performance	18
3.4	Classification: Final Discussion	19
4	Pattern Mining	19
4.1	Data Preprocessing for Transactional Analysis	19
4.2	Frequent Pattern Extraction and Analysis	20
4.3	Association Rule Mining for Prediction	20
4.4	A Custom Rule-Based Classifier	21

5	Regression	22
5.1	Univariate Regression Analysis	22
5.1.1	Target and Predictor Selection	22
5.1.2	Model Comparison and Results	22
5.2	Multivariate Regression Analysis	22
5.2.1	Predictor Selection	23
5.2.2	Model Comparison and Results	23

1 Data Understanding & Preparation

1.1 Introducing the variables with their meaning and characteristics

The dataset is presented in a tabular format, comprising **16,431 entries**, where each row corresponds to a unique media title (such as a movie or TV series), and **23 columns** representing its diverse attributes. The primary objective of this analysis is to dissect these features to uncover underlying patterns, assess data quality, and ultimately build predictive models.

A preliminary examination reveals a rich mix of data types. Among the most analytically interesting variables are:

- **Performance and Success Metrics:** Key indicators like `rating`, `numVotes`, and `awardWins` are central to evaluating a title's reception and success.
- **Core Descriptive Features:** Attributes such as `startYear`, `runtimeMinutes`, `genres`, and `countryOfOrigin` provide essential context for profiling and clustering.
- **Potential Target Variables:** Categorical features like `rating` and `titleType` are strong candidates for multi-classification tasks, while numerical fields could be used for regression or further analysis.

1.1.1 Numerical Variables

- `startYear`: An `int64` representing the release year of the title. The distribution spans from 1878 to 2024.
- `runtimeMinutes`: An `object` type that should be numerical (`int`), representing the duration of the title in minutes.
- `numVotes`: An `int64` indicating the total number of votes received from users.
- `awardWins`: A `float64` that should be an integer (`int`), counting the number of awards won. The float type is inappropriate as awards are discrete counts. It also contains 2618 null values.
- `awardNominationsExcludeWins`: An `int64` representing the count of nominations, excluding wins.
- `ratingCount`, `userReviewsTotal`, `criticReviewsTotal`: columns of type `int64` detailing engagement metrics from audience and critic.

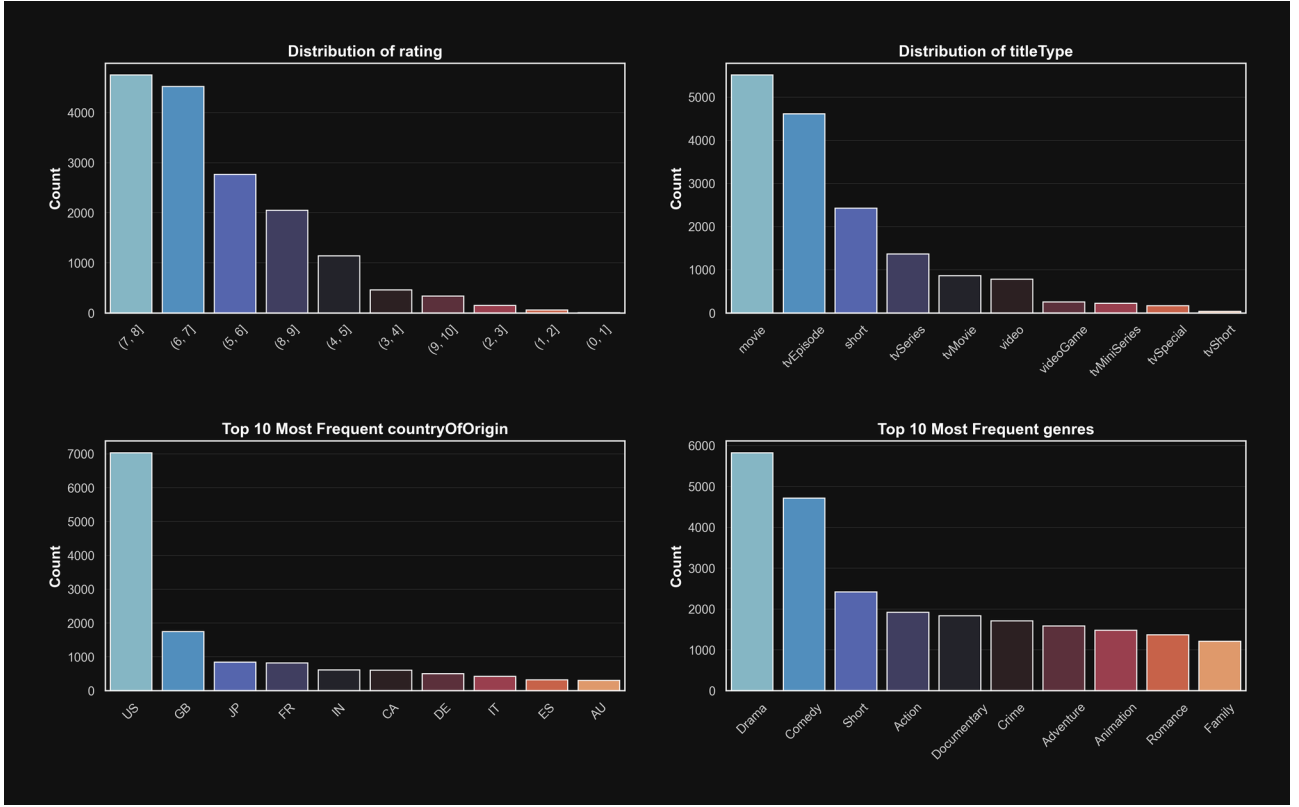
1.1.2 Categorical Variables

- `originalTitle`: An `object` containing the title's name. Due to its extremely high cardinality (nearly unique for each entry), it is unsuitable for direct use as a categorical feature in most models.
- `rating`: An `object` representing binned user ratings in interval notation (e.g., `(7.0, 8.0]`). It has 10 unique categories, making it an excellent candidate for classification tasks.
- `titleType`: An `object` with 10 unique categories (e.g., `'movie'`, `'tvSeries'`), also ideal for modeling.
- `countryOfOrigin` & `genres`: These are multi-label categorical features stored as `object` types. They exhibit high cardinality (860 and 858 unique combinations, respectively) due to their composite nature. Their formats are inconsistent:
 - `genres` uses comma-separated strings (e.g., `'Action,Adventure,Animation'`).
 - `countryOfOrigin` uses strings that mimic list notation (e.g., `'[FR', 'MK', 'SI']'`).

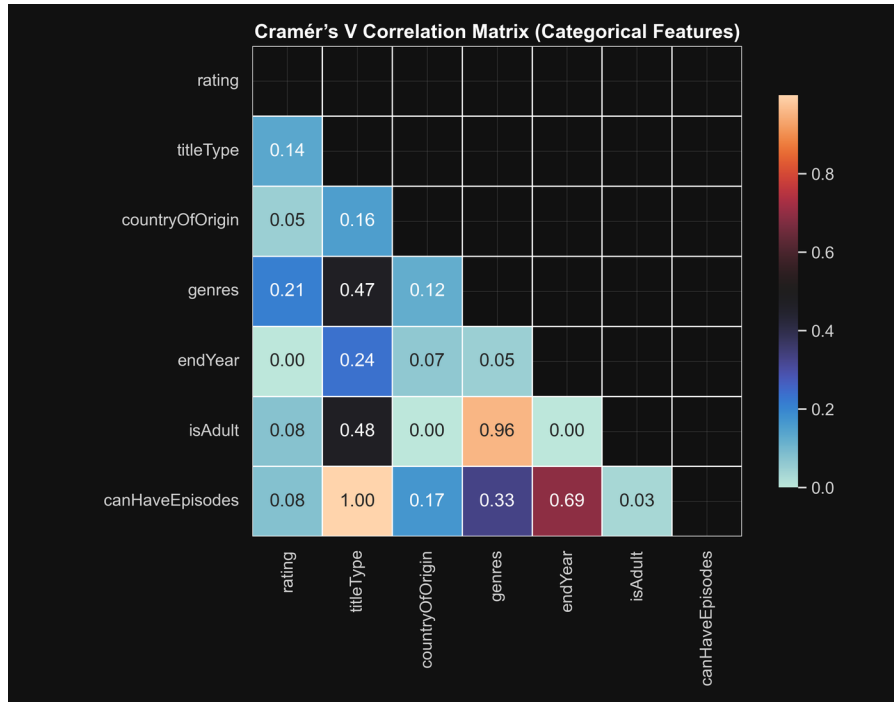
To handle these columns effectively, a two-step preprocessing approach will be implemented. **First**, the strings will be **parsed** into lists of individual labels. **Second**, a **Multi-Label Binarization** (similar to One-Hot Encoding) will be applied.

1.2 Distribution of Categorical Variables, Pairwise Correlations and Transformations

The categorical variables provide the core descriptive context for each title in the dataset. An analysis of their distributions, visualized in Figure 1, reveals several key characteristics and inherent biases of the data.



(a) Distributions of Individual Variables



(b) Cramér's V Correlation Matrix

Figure 1: Visual analysis of key categorical variables. **(a)** The distribution plots reveal the characteristics of each feature, such as the bias towards higher ratings and US-based productions. **(b)** The Cramér's V matrix quantifies the inter-variable association, highlighting high correlations (e.g., between **isAdult** and **genres**) that informed subsequent feature selection.

- **rating:** Ordinal variable with categories like (7, 8] and (6, 7]. Titles are mostly concentrated in higher rating bins, with very few in the lowest ones (e.g., (0, 1], (1, 2]). This reflects a strong bias toward positively rated content, which must be considered in modeling.
- **titleType:** Dominated by the **movie** category. Other types (e.g., **tvEpisode**, **short**) are present but less

frequent. The dataset mainly reflects feature films, influencing any derived conclusions.

- **genres (Top 10):** Drama and Comedy are the most common, consistent with global film trends. The frequent occurrence of `Short` aligns with the `titleType` distribution. The presence of varied genres like `Action`, `Documentary`, and `Crime` adds richness for genre-based analysis.
- **countryOfOrigin (Top 10):** Shows clear geographical bias. The US dominates by far, followed by the UK and Japan. Thus, models trained on this data will largely reflect American industry trends and may lack generalizability.

In summary, the *typical entry* is a well-rated American movie, typically in the drama or comedy genre. Recognizing these biases is crucial for valid model interpretation.

1.2.1 Categorical Variables Correlation using Cramér’s V

Cramér’s V is a measure of association based on the **chi-squared (χ^2) statistic**. It computes the chi-squared value for the contingency table of two categorical variables and then normalizes it to produce a value between **0 and 1**, inclusive.

- A value of **0** indicates no association between the two variables.
- A value of **1** indicates a perfect association, meaning one variable can be perfectly predicted from the other.

1.2.2 Analysis of Results

The Cramér’s V correlation matrix (Figure 1b) uncovered several important relationships among categorical variables.

Perfect Association ($V = 1.00$): The variable `canHaveEpisodes` was perfectly associated with `titleType`, meaning that knowing the type of a title fully determines whether it can have episodes. This confirmed the redundancy of `canHaveEpisodes`.

Very Strong Association ($V = 0.96$): A near-perfect correlation was found between `isAdult` and `genres`, showing that genre choices almost entirely capture whether a title is marked as adult. As such, `isAdult` was deemed unnecessary for modeling.

Moderate Association ($V \approx 0.47$): The link between `genres` and `titleType` was moderate, which makes sense, as certain genres tend to occur more often in specific title types (e.g., the `Short` genre in `short` films).

Weak Association: Most other variable pairs, especially those involving `rating` and `countryOfOrigin`, showed weak relationships. This confirms that they provide distinct information not captured by other features.

In summary, the Cramér’s V analysis helped guide feature selection by providing strong evidence to eliminate `canHaveEpisodes` and `isAdult`, reducing redundancy and multicollinearity without losing relevant information.

1.2.3 Categorical Variables Transformation

To prepare categorical features for modeling, each was numerically encoded using a method aligned with its structure (ordinal, nominal, multi-label). All encoders were fitted on the training set and then applied to both train and test splits to avoid data leakage.

1. **Ordinal Encoding (`rating`):** Since `rating` has ordered bins (e.g., (6, 7], (7, 8]), an `OrdinalEncoder` mapped each category to a corresponding integer, preserving the rank for the model.
2. **One-Hot Encoding (`titleType`):** Being a nominal variable, `titleType` was encoded using `OneHotEncoder`, creating a binary column per category (e.g., `movie`, `tvSeries`), while avoiding spurious ordering. The encoder was set to handle unseen categories in the test set.
3. **Multi-Label Binarization (`genres`, `countryOfOrigin`):** These features required a custom binarization. From the training set, the **top 10 most frequent** individual labels were extracted. For each, a binary indicator was created (1 if the label was present). This captures essential information while limiting dimensionality.

After encoding, the original categorical columns were dropped, and their numerical representations replaced them in both train and test datasets, ensuring a consistent and model-ready structure.

1.3 Distribution of Numerical Variables, Pairwise Correlations and Transformations

From Table 1, several insights emerge regarding the dataset’s numerical features: `worstRating` and `bestRating` have zero standard deviation and hold constant values (1 and 10, respectively). With no variance, they provide no predictive power and are removed.

The `count` row reveals missing values, especially in `awardWins`, which is addressed in Section 1.4 using `IterativeImputer`.

Many variables exhibit **right-skewed distributions**, where the mean significantly exceeds the median. For example, `numVotes` has a mean of 1,493 but a median of 36, highlighting the impact of outliers. Large standard deviations confirm wide variability and long-tailed distributions.

Additionally, variables span vastly different ranges: `numVotes` approaches one million, while `numRegions` peaks at 69. To address skewness and scale disparities, a transformation pipeline is applied: logarithmic scaling (`np.log1p`) and **standardization** before modeling.

Table 1: Descriptive statistics for the numerical variables of the **training set**

Variable	count	mean	std	min	25%	50%	75%	max
startYear	16431	1991.80	26.12	1878.0	1978.0	1997.0	2013.0	2024.0
awardWins	13813	0.49	2.97	0.0	0.0	0.0	0.0	143.0
numVotes	16431	1493.15	19780.76	5.0	15.0	36.0	148.0	964045.0
worstRating	16431	1.00	0.00	1.0	1.0	1.0	1.0	1.0
bestRating	16431	10.00	0.00	10.0	10.0	10.0	10.0	10.0
totalImages	16431	11.48	74.23	0.0	1.0	2.0	6.0	3504.0
totalVideos	16431	0.27	3.12	0.0	0.0	0.0	0.0	258.0
totalCredits	16431	61.34	174.12	0.0	16.0	35.0	65.0	15742.0
criticReviewsTotal	16431	2.78	15.41	0.0	0.0	0.0	1.0	567.0
awardNominations (excl. wins)	16431	0.55	3.91	0.0	0.0	0.0	0.0	197.0
isAdult	16431	0.03	0.16	0.0	0.0	0.0	0.0	1.0
numRegions	16431	3.54	5.85	1.0	1.0	1.0	3.0	69.0
userReviewsTotal	16431	7.33	66.50	0.0	0.0	0.0	2.0	5727.0
ratingCount	16431	1493.92	19784.78	5.0	15.0	36.0	149.0	964294.0

1.3.1 Correlations of Numerical Variables and Variable Elimination

To investigate the relationships between numerical features and identify potential multicollinearity, correlation matrices were computed using two distinct methods: **Pearson’s r** , which measures linear relationships, and **Spearman’s ρ** , which assesses monotonic relationships. The resulting heatmaps are displayed in Figure 2.

1.3.2 Pearson and Spearman Matrix Correlation Analysis

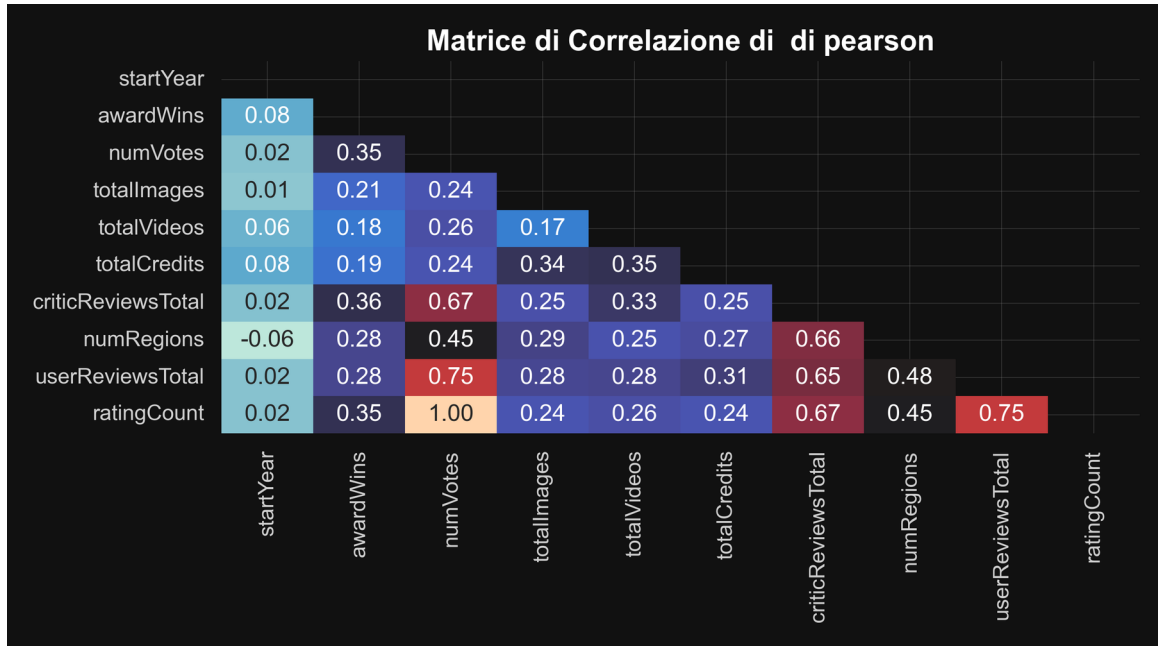
A detailed examination of the matrices reveals several key insights:

- **Perfect Correlation:** The most striking observation is a perfect correlation of **1.00** between `numVotes` and `ratingCount` in both matrices. This indicates that they are perfectly collinear and convey the exact same information. To avoid severe multicollinearity issues, which can destabilize many models, one of these variables must be removed. We will retain `numVotes` and drop `ratingCount`.
- **Strong Positive Correlations:** Several other variables exhibit strong positive correlations. Notably:
 - `userReviewsTotal` shows a strong correlation with `numVotes` (Pearson: 0.75, Spearman: 0.66).
 - `criticReviewsTotal` is also strongly correlated with `numVotes` (Pearson: 0.67) and moderately with `numRegions` (Pearson: 0.66).

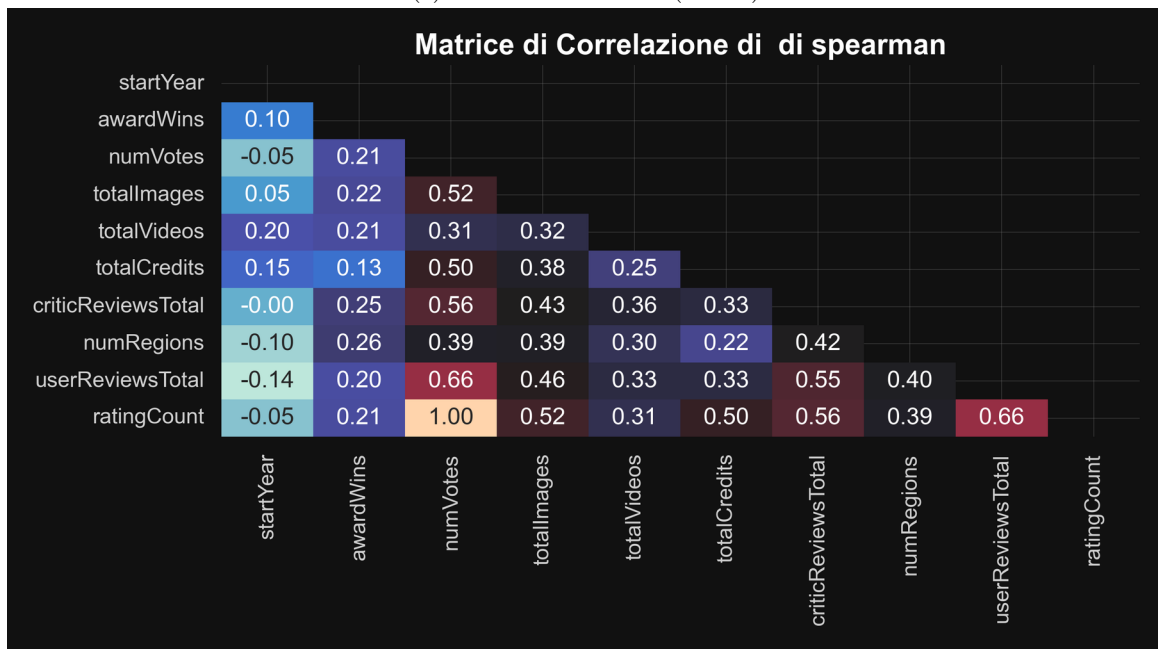
This suggests that as a title gathers more votes, it also tends to accumulate a higher number of user and critic reviews, which is an intuitive relationship.

- **Weak Correlations:** The `startYear` variable consistently shows very weak to negligible correlation with almost all other features, indicating that the release year, by itself, is not a strong linear or monotonic predictor of a title’s popularity or review volume.

- **Consistency Across Methods:** The patterns observed in the Spearman matrix are highly consistent with those in the Pearson matrix. This implies that the strong relationships identified are not just monotonic but are indeed primarily linear in nature.



(a) Pearson Correlation (Linear)



(b) Spearman Correlation Not Linear (Monotonic)

Figure 2: Comparison of Pearson (a) and Spearman (b) correlation matrices. Both methods highlight a **perfect correlation (1.00)** between **numVotes** and **ratingCount**, confirming their redundancy.

1.3.3 Feature Elimination Summary

- **worstRating** and **bestRating**: Removed due to having zero variance.
- **isRatable**, **isAdult**, **canHaveEpisodes**: Removed due to zero variance or perfect correlation with other categorical features.
- **ratingCount**: Removed due to its perfect correlation with **numVotes**.

This refined set of features will form the basis for the subsequent clustering and classification tasks.

1.3.4 Numeric Variables Transformation

As established in the initial analysis, the original numerical variables were characterized by severe **right-skew**. Most features exhibited distributions resembling an exponential decay, with the majority of data points clustered near zero and a long tail of sparse, high-magnitude values. Such distributions are suboptimal for many machine learning algorithms.

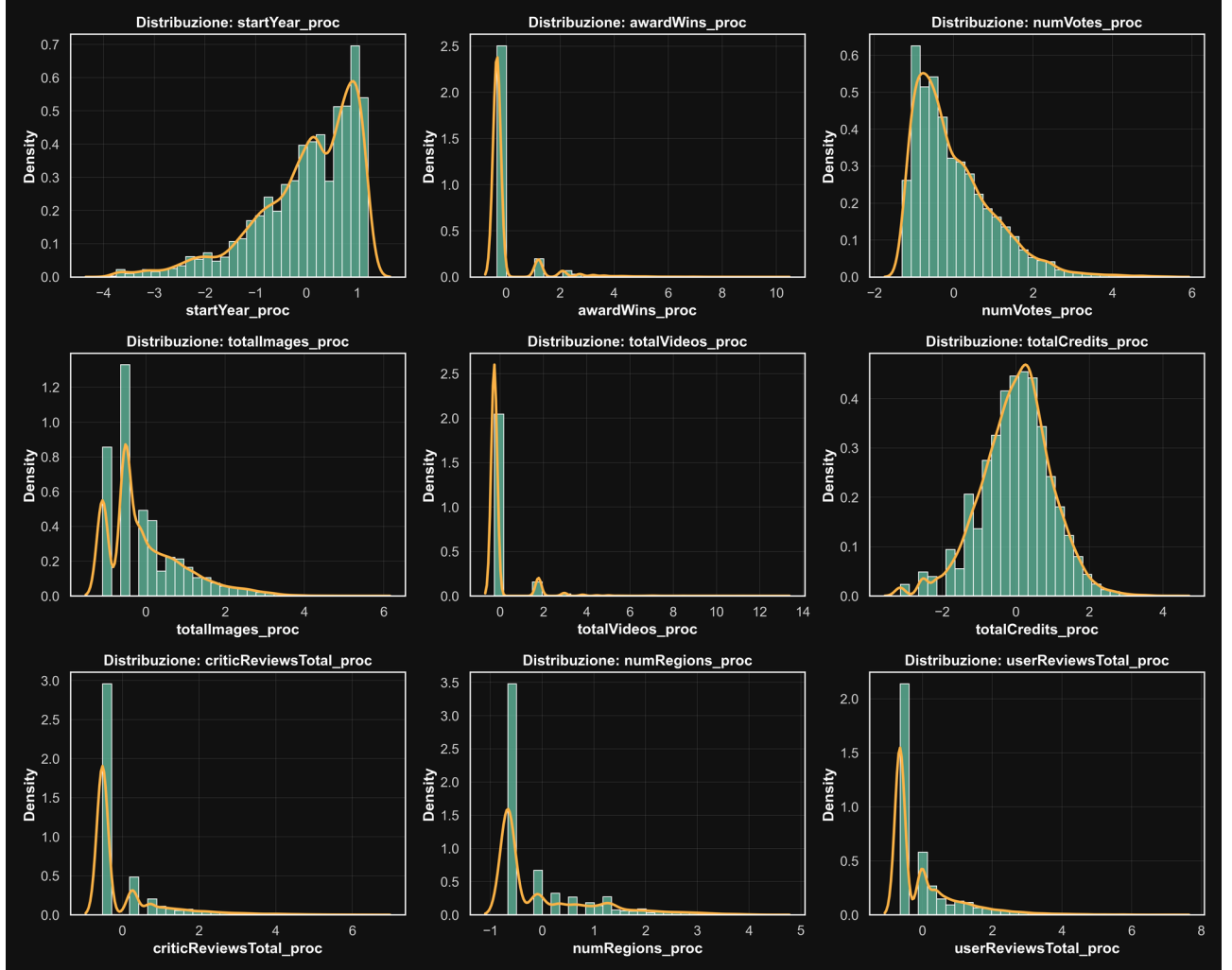


Figure 3: Histograms and density plots of the numerical variables after the full preprocessing pipeline: log-transformation and standardization. The distributions are now approximately more normal shaped.

To address this, a multi-stage preprocessing pipeline was applied to the numerical features (Figure 3):

1. **Logarithmic Transformation:** A $\log(1+x)$ transform was first applied to dramatically reduce the right-skew and compress the range of the data.
2. **Standardization:** Subsequently, a `StandardScaler` was used to rescale each feature to have a mean of 0 and a standard deviation of 1. This ensures all features are on a comparable scale, which is crucial for distance-based algorithms like clustering.

The result of this comprehensive process is shown in Figure 3: the distributions are now visibly more symmetric, the x-axes are now centered around zero and share a similar scale, confirming the effectiveness of the standardization.

1.4 Handling Missing and Placeholder Values

A multi-step approach ensured proper handling of missing data and avoided data leakage.

1. **Non-Standard Placeholder Conversion:** The placeholder string `\N`, found in columns like `runtimeMinutes`, was converted to `numpy.nan` for compatibility with imputation methods.

2. **Median Imputation for runtimeMinutes:** Missing values in `runtimeMinutes` were filled with the median (not mean, to resist outliers), computed only on the training data and applied to both splits.
3. **Model-Based Imputation for awardWins:** Over 2,600 missing values in `awardWins` were imputed using `IterativeImputer`, leveraging correlations with features like `numVotes`. The imputer was fitted on the training set and applied consistently across both sets.

1.5 Outliers Detection and Removal with LOF

Outliers were identified and removed using a rigorous LOF-based procedure:

1. **Application:** LOF was applied to log-transformed and standardized numerical features from the training set only.
2. **Visualization and Validation:** PCA was used to reduce dimensionality and visually confirm that LOF-flagged outliers were indeed isolated or peripheral points, as shown in Figure 4.
3. **Removal:** Confirmed outliers were excluded from the training data to enhance model robustness.

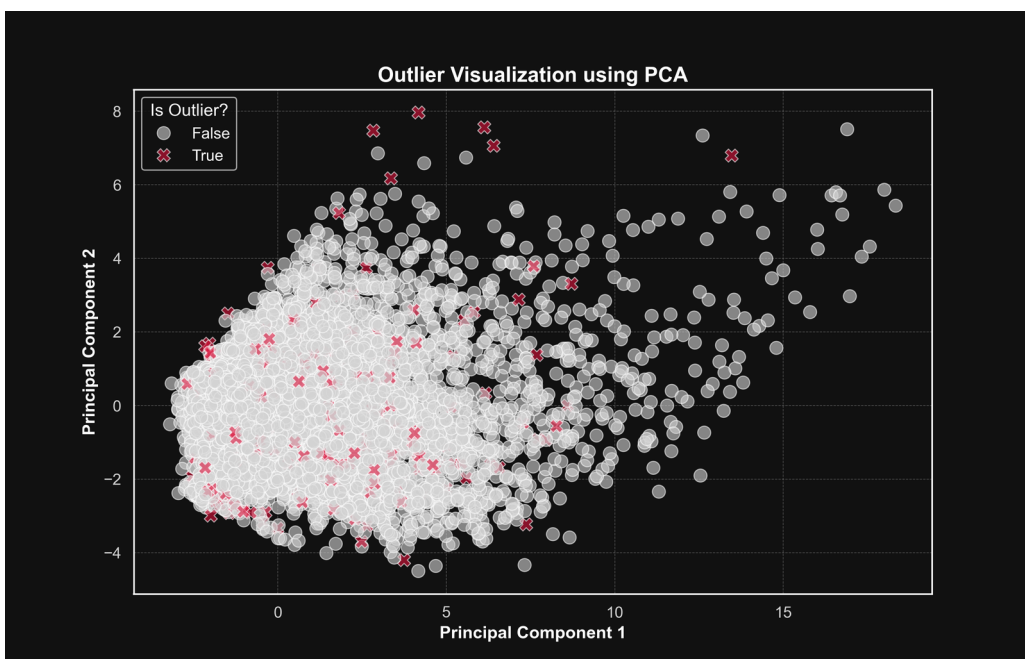


Figure 4: 2D PCA of processed training data. Outliers (peripheral or isolated points) validate LOF settings: `n_neighbors=20`, `contamination=0.05`, `metric='minkowski'`.

2 Clustering Analysis

2.1 Clustering: K-Means

In this section, we begin our exploration of the dataset's underlying structure using unsupervised learning. We employ K-Means, a centroid-based clustering algorithm, as our first method.

2.1.1 Feature Selection and Preparation

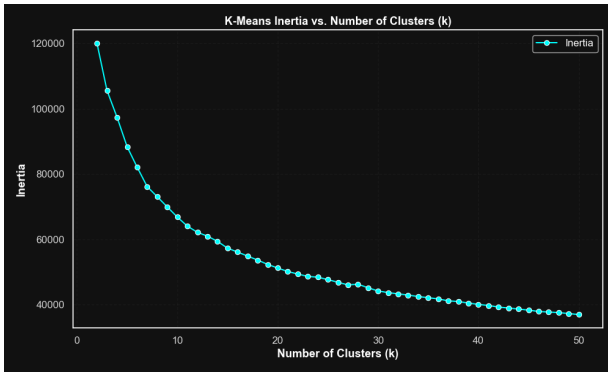
For the clustering task, we selected the 11 most important numerical features from our dataset already transformed and standardized. This choice is strategic for two primary reasons:

1. **Numerical Nature:** K-Means operates by calculating Euclidean distances in a multi-dimensional space, a process that requires all features to be numerical.
2. **Standardization:** Crucially, these features have already undergone log-transformation and standardization (mean of 0, standard deviation of 1). This scaling is a mandatory prerequisite for K-Means, as it prevents features with larger original scales (like `'numVotes'`) from disproportionately dominating the distance calculations over features with smaller scales (like `'awardWins'`).

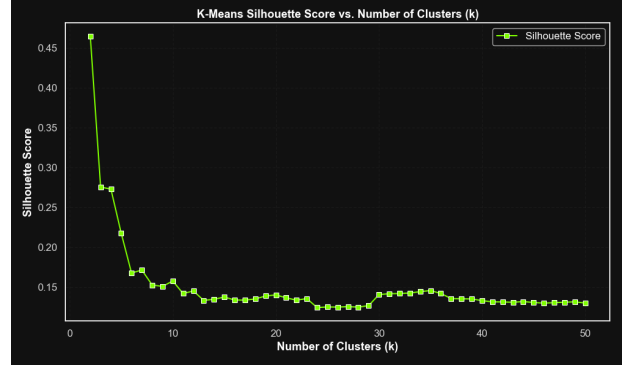
2.1.2 Determining the Optimal Number of Clusters (K)

The most critical hyperparameter for K-Means is 'K', the number of clusters. To identify the optimal K for our dataset, we employed a combination of two standard quantitative methods alongside qualitative analysis. We evaluated the model's performance for K values ranging from 2 to 50.

Figure 5: Elbow Inertia Method and Silhouette Method for selecting the optimal K.



The Elbow Method shows a less pronounced "elbow" around K=7.



The Silhouette Score best compromise is at 3 or 4 but fails to show a clear subsequent optimum, suggesting that K=7 is a good trade off

As shown in Figure 5, the quantitative metrics alone do not provide a single, unambiguous answer. A deeper analysis revealed a crucial insight: two very small, distinct clusters consistently appeared regardless of the chosen K (for $K \geq 4$). These *elite* clusters represent extreme outliers in terms of popularity and awards. Furthermore, our dataset is not a homogenous collection of movies, but a diverse ecosystem of content including feature films, documentaries, TV series, shorts etc. A small K would force these fundamentally different categories into the same group, obscuring the true structure of the data.

Therefore, we made an informed decision to select **K=7**. This value is the best compromise suggested by the Inertia curve and, more importantly, provides sufficient granularity to isolate the two elite, persistent clusters while allowing the algorithm to properly segment the remaining diverse content types into meaningful groups.

2.1.3 Cluster Visualization and Profiles

With K=7, we performed the final clustering. The resulting groups are highly imbalanced in size, as shown in Figure 6 (right), with a large general cluster and several smaller, specialized niches. To visualize the 11-dimensional clusters, we used Principal Component Analysis (PCA) to project the data onto a 2D plane. The resulting plot, shown in Figure 6 (left), demonstrates a reasonable degree of separation, confirming the existence of distinct groupings in the data.

To understand the identity of each cluster, we conducted a deep analysis of their quantitative and categorical characteristics. A summary of these profiles is presented in Table 2.

Table 2: Summary of K-Means Cluster Profiles (K=7)

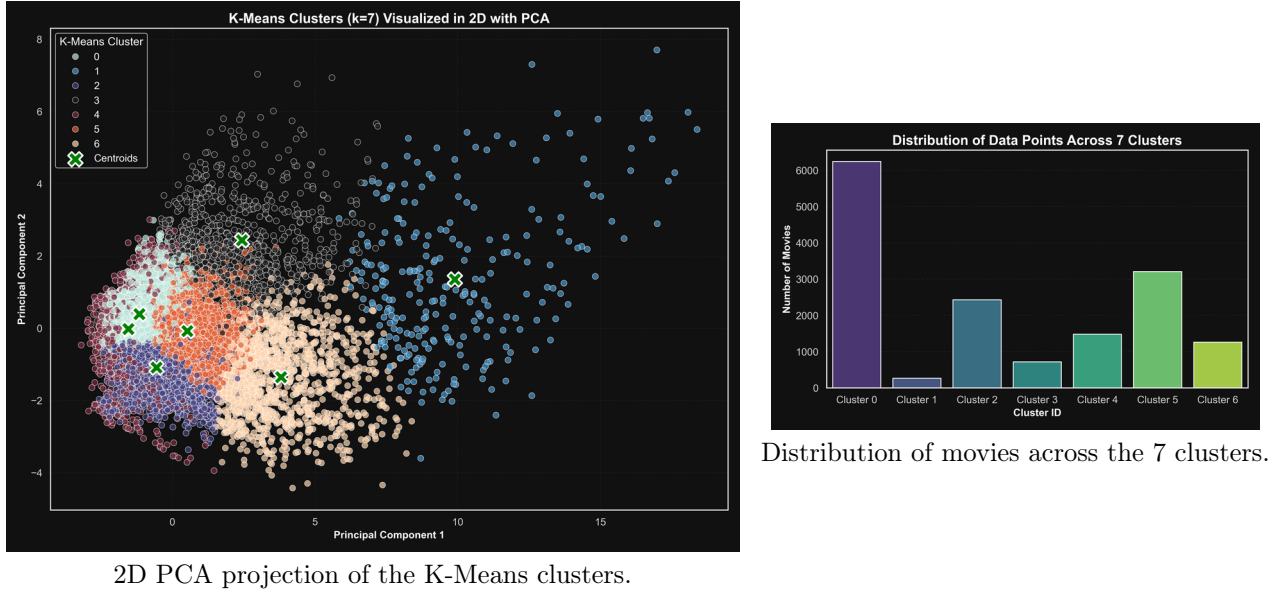
Cluster	Archetype	Size (%)	Avg. Votes	Avg. Awards	Avg. Runtime	Avg. Year	Top Title Type	Top Genre
0	The Sea of Obscurity	40.0%	20	0.0	56 min	2002	TV Episode	Drama
1	The Hollywood Elite	1.7%	19,232	3.9	87 min	2002	Movie	Drama
2	The Vintage Cinema Vault	15.6%	38	0.0	60 min	1955	Movie	Drama
3	The Festival Darlings	4.6%	130	3.0	62 min	2006	Movie	Drama
4	The Kingdom of Shorts	9.5%	23	0.1	9 min	1986	Short	Short
5	The World of Television	20.5%	170	0.1	55 min	2002	TV Episode	Drama
6	The Mid-Budget Successes	8.1%	967	0.3	77 min	1985	Movie	Drama

2.1.4 Discussion of the Clusters

The deep analysis reveals that the K-Means algorithm has successfully identified distinct and highly interpretable "ecosystems" within the IMDb dataset.

- **Cluster 1: The Hollywood Elite.** This is one of the two "elite" clusters we anticipated. It is a small but powerful group representing mainstream, commercially successful, and critically recognized feature films. It is defined by an extremely high average number of votes and awards.

Figure 6: Visualization and distribution of the 7 identified clusters.



- **Cluster 3: The Festival Darlings.** This is the second elite cluster. These are not blockbusters, but rather critically acclaimed independent films, shorts, and series that excel on the festival circuit. They are characterized by a high number of awards relative to their very low vote count.
- **Cluster 4: The Kingdom of Shorts.** Our choice of $K=7$ is vindicated by the emergence of this cluster, which has successfully isolated short films. It is defined by an exceptionally low average runtime (9 minutes) and is dominated by the ‘short’ title type and genre.
- **Cluster 2: The Vintage Cinema Vault.** This cluster groups together older productions, with an average release year of 1955. These are primarily feature films from a different era of cinema, characterized by low vote counts and a lack of digital presence.
- **Cluster 6: The Mid-Budget Successes.** This group represents the solid, well-regarded feature films that achieved a respectable level of popularity (avg. 967 votes) but did not reach the blockbuster status of Cluster 1.
- **Cluster 5: The World of Television.** This large cluster is the primary home for television content, composed mainly of TV series.
- **Cluster 0: The Sea of Obscurity.** As the largest cluster by far (40%), this group represents the vast baseline of content on IMDb: a mix of unknown TV episodes, obscure movies, and other low-impact titles with minimal votes, no awards, and very few reviews.

In conclusion, the K-Means algorithm has successfully partitioned the data not just by success, but by fundamental content type, creating a rich and interpretable map of the IMDb universe.

2.2 Clustering: DBSCAN Analysis

Following the centroid-based approach of K-Means, we now adopt a textbfdensity-based algorithm: **DBSCAN** (*Density-Based Spatial Clustering of Applications with Noise*). The goal is to examine the dataset from a novel perspective, identifying clusters based on regions of **high point density** rather than geometrically defined centers. This method excels at detecting clusters of *arbitrary shapes* and, crucially, at labeling outliers as **noise**.

2.2.1 Feature Selection and Parameter Strategy

To ensure consistency with the K-Means analysis, we employ the same set of **11 standardized numerical features**. DBSCAN’s behavior is governed by two main hyperparameters: **Eps** (the neighborhood radius) and **MinPts** (the minimum number of points required to form a dense cluster).

An essential step was the examination of the **k-distance plot** shown in Figure 7 (right). The graph displays a clear *elbow* around $Eps \approx 2.5$. However, empirical results indicated that this value alone failed to yield a valid

clustering solution—a typical outcome when dealing with data exhibiting **highly variable densities**. The plot’s real utility thus manifested in two core insights:

- It revealed the appropriate **order of magnitude** for Eps;
- It highlighted a **promising value range** between 1.0 and 2.5 for fine-tuned search.

This understanding laid the foundation for a more refined and targeted parameter exploration.

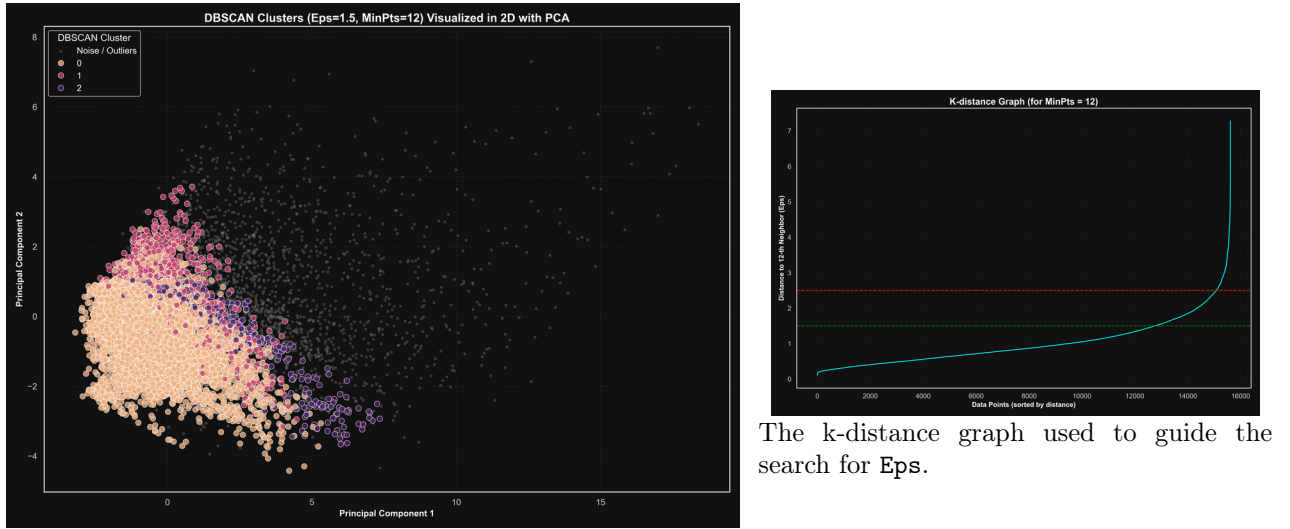
2.2.2 Advanced Parameter Search and Results

Acknowledging the difficulty of pinpointing a globally optimal configuration, we implemented a **criteria-based search strategy**. Our objective was to identify a configuration that avoided the trivial outcome of a single dominant cluster. After systematic experimentation, we selected a configuration that balanced expressiveness with parsimony:

- **MinPts:** 12 (following the $D + 1$ heuristic)
- **Eps:** 1.5 (determined via targeted exploration)

This setting yielded **4 meaningful clusters**, including the noise group (label -1). The cluster size distribution is reported in Table 3. The most evident pattern is the emergence of a **dominant core cluster** (Cluster 0), comprising nearly **80%** of the dataset—representing the dense center of IMDb content. Two other moderate clusters and a substantial number of noise points were also successfully identified. The left panel of Figure 7 illustrates the outcome in PCA-reduced 2D space.

Figure 7: Visual summary of the DBSCAN parameter selection process and final results.



Final clusters projected in 2D with PCA.

2.2.3 Deep Cluster Analysis

To characterize the discovered groups, we performed an in-depth analysis of both quantitative and categorical traits, treating the noise points as a cluster in their own right. The resulting profiles are summarized in Table 4.

Table 3: Cluster distribution for the final DBSCAN configuration (Eps=1.5, MinPts=12).

Cluster Label	Number of Movies	Percentage of Dataset
Cluster 0 (The Dense Core)	12,369	79.24%
Cluster 1	767	4.91%
Cluster 2	589	3.77%
Noise (-1)	1,884	12.07%

Table 4: Summary of DBSCAN Cluster Profiles

Cluster	Archetype	Size (%)	Avg. Votes	Avg. Awards	Avg. Runtime	Top Title Type	Top Genre
0	The Dense Core of Obscurity	79.2%	138	0.0	56 min	TV Episode	Drama
1	The Realm of Shorts & Niche Films	4.9%	132	1.3	65 min	Movie	Drama
2	The Populous Mid-Range	3.8%	888	0.0	72 min	Movie	Drama
-1	The Exile of the Elites (Noise)	12.1%	11,421	3.0	78 min	Movie	Drama

- **Cluster 0: The Dense Core of Obscurity.** This large cluster constitutes the heavily populated center of the IMDb universe. It includes low-visibility content such as TV episodes and shorts, with minimal engagement (votes, reviews, awards). DBSCAN recognizes this as one contiguous dense region.
- **Cluster 1 & 2: The Emerging Niches.** These mid-sized clusters correspond to segments with moderate density, separated from the core. Cluster 1 includes niche shorts and movies, while Cluster 2 comprises moderately successful titles—not quite outliers, yet distinctly more impactful than the core.
- **Noise Group (-1): The Exile of the Elites.** This is DBSCAN’s most remarkable discovery. Though labeled as noise, this group boasts the **highest average number of votes and awards**. Contrary to being irrelevant, this cluster contains the *elite* so distinct from the mass that they lack enough similar peers nearby to form a dense region.

2.2.4 Cross-Analysis with K-Means and Conclusion

To validate this insight, we conducted a cross-analysis with the K-Means output. Astonishingly, **89.78%** of the movies previously categorized as *elite* (Hollywood Elite and Festival Darlings) were tagged as **noise** by DBSCAN.

This is not a misclassification; it is a revelation. Through the strict lens of DBSCAN’s density-based definition, the most successful titles are not central nodes in any dense area, they are **anomalies**. They are so different, so rare, that they stand apart from the crowd.

In summary, K-Means provided an interpretable categorization based on geometric similarity, while DBSCAN exposed the **density structure** of the dataset. It isolated the dense heart of the IMDb landscape and, perhaps more importantly, redefined what it means to be "elite". In DBSCAN’s universe, the elite are not part of a cluster, they are considered *outliers*, brilliant and rare, orbiting far from the crowded galactic core. This analysis offers not a repetition, but a profound **complementary truth**. *The elite are not the many; they are the magnificent few.*

2.3 Agglomerative Clustering

Following the centroid-based (K-Means) and density-based (DBSCAN) approaches, we employ a third methodology: **Hierarchical Agglomerative Clustering**. This bottom-up technique does not require the a priori specification of the number of clusters. Instead, it builds a tree-like hierarchy of mergers, called a dendrogram, which provides a rich visualization of the data’s taxonomic structure. Our objective is to leverage this structure to identify natural groupings and compare the results with our previous findings.

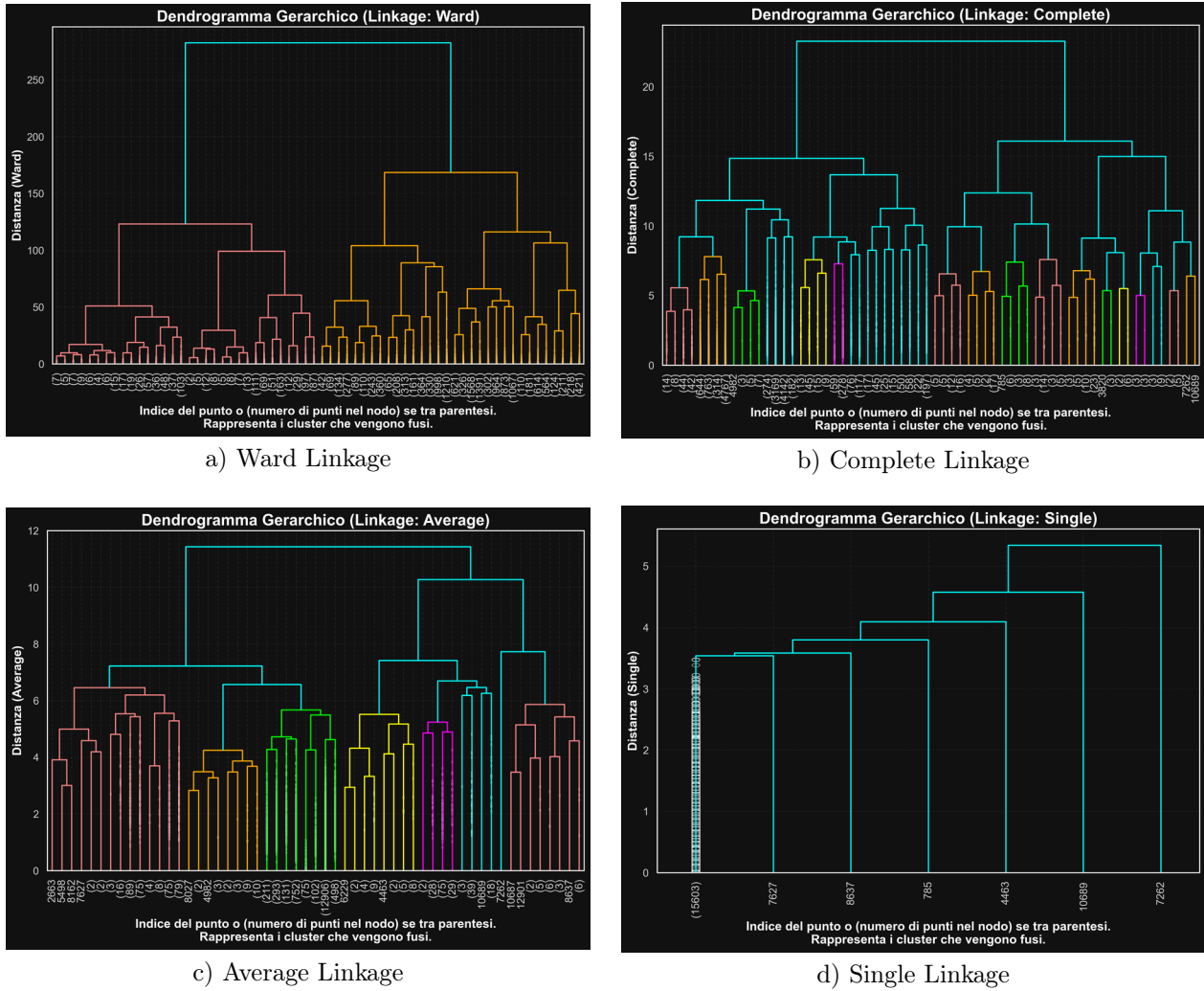
2.3.1 Methodology and Linkage Method Comparison

To ensure methodological consistency, the analysis was performed on the same set of 11 standardized numerical features used in the preceding clustering tasks. The core of hierarchical clustering lies in two choices: the distance metric, for which we used **Euclidean distance** as is standard for normalized spaces, and the **linkage criterion**, which defines how the distance between clusters is measured.

To select the most suitable linkage criterion for our dataset, we evaluated the four primary methods: *single*, *complete*, *average* and *ward*. The resulting dendrograms, shown in Figure 8, reveal stark differences in the structures they identify.

The **Single linkage** dendrogram (d) exhibits a severe "chaining effect," where individual points are sequentially merged into one massive cluster. This fails to produce distinct, interpretable groups and is thus unsuitable. Both **Complete** (b) and **Average** (c) linkage provide more reasonable and balanced structures. However, it is the **Ward linkage** method (a) that yields the most compelling result. Ward’s method, which seeks to minimize the total within-cluster variance at each merge, produces a remarkably clean and well-structured dendrogram with clear, large vertical gaps between major merges. This clarity facilitates a more objective choice for the final number of clusters. Based on this qualitative assessment, we selected **Ward’s method** as the definitive approach for our final analysis.

Figure 8: Comparative analysis of the four primary linkage criteria. The superiority of the Ward method, with its clean and balanced structure, is evident.



2.3.2 Determining the Optimal Number of Clusters (k)

Focusing on the Ward dendrogram, the final step is to "cut" the tree at an appropriate height to define the clusters. We identified that a cut-off at a linkage distance of **110** provides an excellent trade-off, capturing the five most significant and distinct macro-groups within the dataset. This decision results in an optimal number of **k=5 clusters**.

Figure 9 illustrates both the selected cut-off on the dendrogram and the resulting 2D PCA projection of the five clusters. The PCA plot visually validates our choice, showing a reasonable degree of separation between the identified groups, with distinct centroids marking their respective centers of mass.

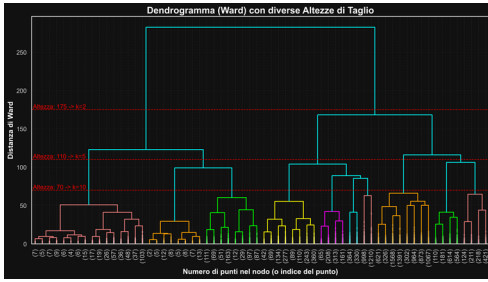
2.3.3 Cluster Profiles and Interpretation

A deep analysis was conducted on the five resulting clusters to understand their defining characteristics. By examining their quantitative averages (on the original data scale), dominant categorical features, and most representative exemplars, we derived a distinct profile for each group, summarized in Table 5.

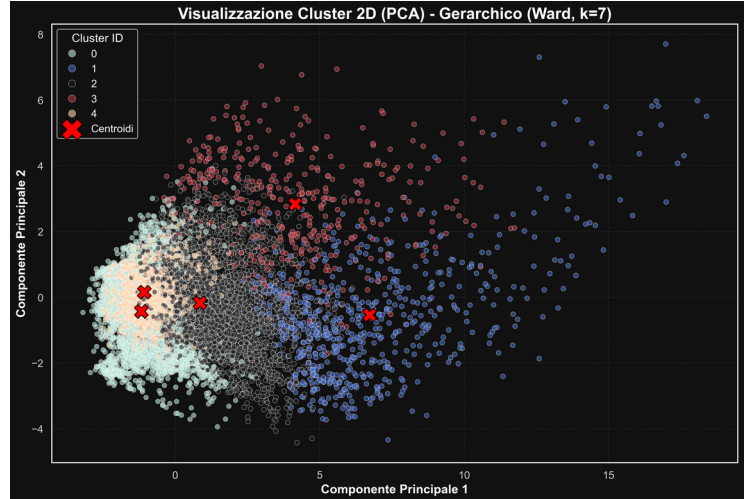
Table 5: Summary of Hierarchical Cluster Profiles (Ward, k=5)

Cluster	Archetype	Size (%)	Avg. Votes	Avg. Awards	Avg. Runtime	Top Title Type	Top Genre
0	The Acclaimed Elite	2.6%	1,215	9.8	85 min	Movie	Drama
1	The Popular Powerhouses	4.4%	20,130	1.5	92 min	Movie	Drama
2	The Vintage Cinema Vault	31.9%	45	0.0	60 min	Movie	Drama
3	The World of Television	15.7%	155	0.1	52 min	TV Episode	Drama
4	The Sea of Obscurity	45.6%	20	0.0	55 min	TV Episode	Drama

Figure 9: Final cluster determination ($k=5$) and visualization.



The Ward dendrogram with a cut-off at height 110, yielding 5 clusters.



2D PCA projection of the final 5 clusters.

- **Cluster 4: The Sea of Obscurity.** Constituting nearly half the dataset, this is the vast baseline of IMDb content. It is defined by extremely low engagement metrics (avg. 20 votes, 0 awards) and is dominated by individual TV episodes and other low-impact media, forming the dense, low-visibility core of the data distribution.
- **Cluster 2: The Vintage Cinema Vault.** This large cluster isolates older productions. While its engagement metrics are also low, its identity is cemented by its primary *titleType* being "Movie" and an average release year significantly earlier than the other clusters. It represents the extensive back-catalog of cinema history on the platform.
- **Cluster 3: The World of Television.** This group represents the ecosystem of established television series. Its members, mostly TV episodes, have a moderate vote count (avg. 155), distinguishing them from the obscurity of Cluster 4. They are the standard, serialized content that drives a significant portion of platform activity.
- **Cluster 1: The Popular Powerhouses.** This is the first of two *elite* clusters. Its defining characteristic is an exceptionally high average number of votes (over 20,000), indicating massive public engagement. These are the blockbusters, cult classics, and commercial hits that, while not necessarily winning the most awards, are central to popular culture conversations.
- **Cluster 0: The Acclaimed Elite.** This is the second, more prestigious elite cluster. In stark contrast to Cluster 1, its members are defined not by votes, but by an extraordinary average of nearly 10 award wins. This cluster represents the world of high-prestige cinema: festival darlings, Oscar winners, and critically-lauded films that are celebrated for their artistic merit over commercial success.

2.4 Clustering: Final Discussion

K-Means provided an interpretable, exhaustive segmentation into seven groups, capturing key archetypes like *Hollywood Elite* and *Vintage Cinema Vault*. Its strength lay in producing actionable clusters, though it was constrained by rigid assumptions (spherical shapes) and sensitivity to the choice of k , which required heuristic tuning.

DBSCAN, in contrast, excelled at detecting structural anomalies. It revealed a dense core of obscure content while casting elite titles as **noise**, reframing them as statistical outliers, an accurate portrayal of creative domain dynamics. However, it lacked full coverage, struggling with variable density and leaving much data unclassified.

Hierarchical Clustering with Ward's method validated K-Means partitions and added structural depth via the Dendrogram. The selection of $k=5$ was based on this clear visual hierarchy, leading to well-separated clusters like *Popular Powerhouses* and *Acclaimed Elite*.

In summary, **Hierarchical Clustering with Ward's method** emerged as the most effective, balancing K-Means' coverage with DBSCAN's structural sensitivity. It delivered both robust segmentation and a transparent roadmap, enabling a defensible interpretation of the data's intrinsic organization.

3 Classification

3.1 Classification by Decision Tree

In this section, we present the first of three classification approaches: the Decision Tree. Selected for its interpretability and ability to model non-linear patterns without heavy preprocessing, this model aims to predict a movie’s rating category using the preprocessed features.

Following careful feature engineering and selection, a final set of 41 predictors was used. The pipeline includes defining the target, training a baseline model, tuning hyperparameters to optimize performance, and evaluating the final configuration using a rigorous experimental protocol.

3.1.1 Target Variable Definition

The target variable for our classification, named `rating_binned`, was derived through a multi-step process designed to convert the original movie ratings into meaningful quality tiers. The process began with the original `rating` column, which was a categorical feature representing rating intervals (e.g., (5.0, 6.0], (6.0, 7.0]). During the data preparation phase, we applied an `OrdinalEncoder` to this feature. This first transformation converted the categorical intervals into a numerical, ordinal scale from 0 to 9, creating the intermediate variable `rating_encoded`.

Subsequently, for this classification task, we engineered the final `rating_binned` variable by grouping the values of `rating_encoded` into a three-tier class structure to categorize movies as 'Poor', 'Average', or 'Good'. The thresholds were chosen to create a reasonably balanced, yet meaningful, distribution of classes:

- **Class 0 (Poor):** Includes movies with an encoded rating less than or equal to 5 (corresponding to original IMDb ratings of 6.0 or lower).
- **Class 1 (Average):** Includes movies with an encoded rating of exactly 6 (corresponding to original IMDb ratings from 6.1 to 7.0).
- **Class 2 (Good):** Includes movies with an encoded rating greater than or equal to 7 (corresponding to original IMDb ratings of 7.1 or higher).

This partitioning resulted in the following class distribution in our training set: the 'Good' class constitutes the majority with 43.89% of the instances, followed by the 'Poor' class at 28.31% and the 'Average' class at 27.80%. While there is a moderate imbalance, it is not severe enough to necessitate complex resampling techniques like SMOTE at this stage, although the model’s performance on minority classes will be a key point of evaluation.

3.1.2 Baseline Model Performance

To establish a performance benchmark, we first trained a Decision Tree classifier using its default hyperparameters. The performance of this unconstrained model was modest, with detailed metrics presented in Table 6. The model achieved an overall accuracy of **50.15%** and **f1_macro** of **48%**. Notably, the F1-score for the 'Average' class was particularly low at 0.36. This behavior is not entirely unexpected; it can be hypothesized that the 'Average' category, by its very nature, represents a zone of higher subjective disagreement among viewers compared to the more clear-cut 'Poor' and 'Good' tiers, making it inherently harder to classify. The confusion matrix confirms this, showing a high number of misclassifications for 'Average' movies, underscoring the necessity of hyperparameter tuning.

Table 6: Performance Metrics for the Baseline Decision Tree Model

Classification Report					Confusion Matrix			
Class	Precision	Recall	F1-Score	Support				
Poor	0.48	0.49	0.49	1542				
Average	0.35	0.36	0.36	1522				
Good	0.61	0.59	0.60	2414				
Accuracy			0.50	5478				
Macro Avg	0.48	0.48	0.48	5478				
Weighted Avg	0.50	0.50	0.50	5478				
					True	Predicted		
					Label	Poor	Average	Good
					Poor	761	414	367
					Average	435	554	533
					Good	385	597	1432

3.1.3 Hyperparameter Tuning and Scoring Metric Selection

To mitigate overfitting and enhance generalization, we performed hyperparameter optimization using 5-fold cross-validation with `GridSearchCV`. Throughout our comprehensive analysis of all three models (Decision Tree, K-Nearest Neighbors, and Naive Bayes), we employed a dual evaluation approach, utilizing both **accuracy** and **f1_macro** as our primary metrics. While the class distribution (40%-30%-30%) is relatively balanced and would permit accuracy as a suitable optimization criterion, we selected **f1_macro** as the primary scoring metric for `GridSearchCV` to ensure balanced performance evaluation across all three classes. This choice provides a more robust assessment by giving equal weight to each class, preventing potential bias toward the majority class that could arise with accuracy optimization alone.

The comprehensive evaluation using both metrics allows for a more nuanced understanding of model performance, particularly in identifying how well each algorithm handles the challenging 'Average' class. Following exhaustive grid search, the optimal hyperparameters for this dataset were identified as: **criterion**: 'gini', **max_depth**: 15, **min_samples_split**: 12, **min_samples_leaf**: 1, **max_features**: None, **max_leaf_nodes**: 500

This configuration points towards a moderately constrained model compared to the baseline. A tree with moderate depth (**max_depth**=15) and controlled complexity through leaf node limitation (**max_leaf_nodes**=500) was found to provide the best trade-off between bias and variance, preventing the model from learning noise in the training data while maintaining sufficient expressiveness. The relatively high minimum samples per split (**min_samples_split**=12) further reinforces overfitting prevention, while allowing maximum granularity at the leaves (**min_samples_leaf**=1).

3.1.4 Tuned Model Evaluation

The Decision Tree model, retrained using the optimal parameters, demonstrated a marginal improvement in overall performance, with **accuracy** increasing to **55.66%** and **f1_score** increasing to **51%**. The detailed metrics in Table 7 reveal consistent behavior with slight performance enhancement across most metrics.

The recall for the 'Good' class (0.74) and 'Poor' class (0.56) remained strong, while the 'Average' class continues to present challenges with a recall of 0.27. This persistent difficulty with the middle category reinforces the inherent ambiguity in distinguishing neutral sentiments. The model maintains its tendency to be more decisive on clear-cut cases while struggling with the nuanced middle ground, pushing uncertain cases towards the more clearly defined 'Poor' or 'Good' categories.

Table 7: Performance Metrics for the Tuned Decision Tree Model

Classification Report					Confusion Matrix			
Class	Precision	Recall	F1-Score	Support	True Label	Predicted		
Poor	0.53	0.56	0.55	1542				
Average	0.40	0.27	0.32	1522				
Good	0.63	0.74	0.68	2414				
Accuracy			0.56	5478	Poor	864	294	384
Macro Avg	0.52	0.52	0.51	5478	Average	437	404	681
Weighted Avg	0.54	0.56	0.54	5478	Good	320	313	1781

3.1.5 ROC Curve Analysis

The Receiver Operating Characteristic (ROC) curves provide further insight into the classifier's discriminative ability. The Area Under the Curve (AUC) scores for the tuned model are shown in Figure 10

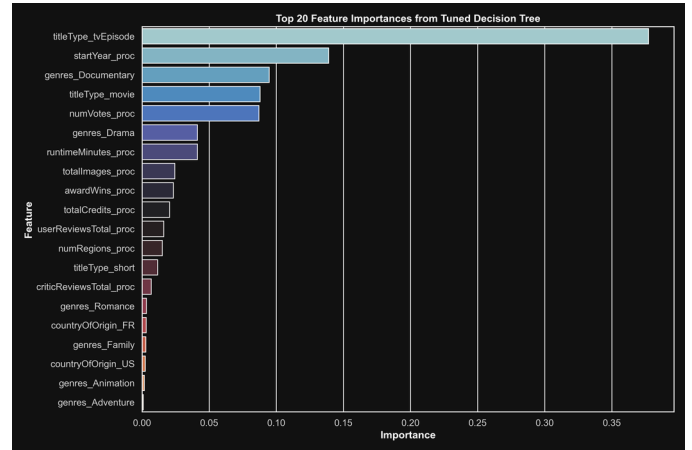
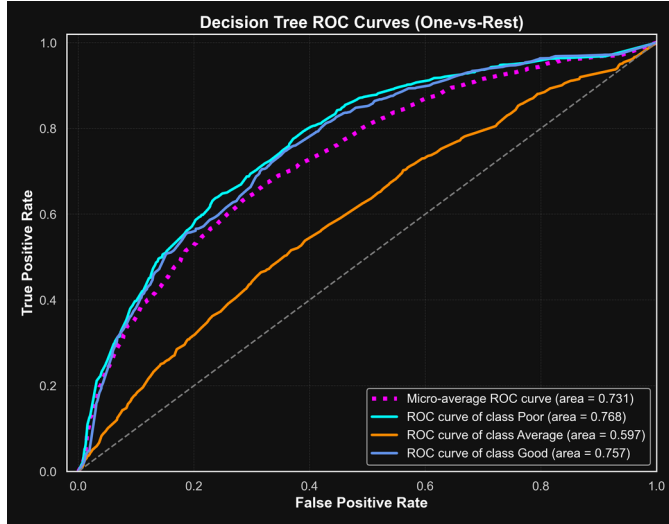
The overall micro-average AUC was **0.731**. The strong AUC scores for 'Poor' and 'Good' classes confirm the model's effectiveness in distinguishing them from the rest. Conversely, the AUC of 0.597 for the 'Average' class is only marginally better than a random guess, quantitatively confirming the model's profound difficulty in identifying this intermediate category, as anticipated.

3.1.6 Feature Importance Analysis

As shown in Figure 10:

1. **titleType_tvEpisode**: This binary feature is, by a significant margin, the most important predictor. This indicates that whether a title is a TV episode or not is the single most powerful piece of information for classifying its rating.

Figure 10: Visual evaluation of the tuned Decision Tree model.



ROC curves for the tuned Decision Tree. The AUC for the 'Average' class (0.607) highlights the model's difficulty with this category. Top 20 feature importances, showing the dominance of `titleType_tvEpisode` and `startYear_proc`.

2. **startYear_proc**: The processed release year is the second most important feature, confirming that temporal trends in filmmaking, audience tastes, or rating behaviors are critical predictors.
3. **Genre, Type, and Popularity**: Following the top two, a combination of other features are also highly predictive. Specifically, `genres_Documentary` and `titleType_movie` are highly ranked, suggesting that being a documentary or a standard movie provides a strong signal for its rating category. Immediately following is `numVotes_proc`, confirming that the movie's popularity remains a crucial factor in the model's logic.

3.1.7 Conclusion

The Decision Tree classifier achieved moderate success with hyperparameter tuning, improving overall accuracy and effectively identifying 'Poor' and 'Good' movies with high recall.

The model's critical weakness, the inability to classify 'Average' movies, reveals a **fundamental truth about human subjective perception**. Movies with clear consensus generate 'Good' or 'Poor' labels, while 'Average' represents a zone of high variance and disagreement. The model's tendency to push 'Average' cases toward extremes reflects real-world ambiguity where personal tastes vary significantly.

This difficulty exposes the inherent uncertainty of the 'Average' class rather than model failure. The available features effectively capture consensus but cannot model subtle personal preferences that define average experiences.

While providing valuable insights, the poor performance on 'Average' classification indicates the need for alternative algorithms. Future exploration of K-Nearest Neighbors and Naive Bayes will determine if different approaches can better navigate this inherent uncertainty that the Decision Tree has effectively highlighted.

3.2 Classification by K-Nearest Neighbors (KNN)

3.2.1 Baseline Model Performance

To establish an initial performance benchmark, we trained a KNN classifier with a standard, commonly used configuration of $K=5$ and the default Euclidean distance metric. The performance of this baseline model, detailed in Table 8, was already superior to the baseline Decision Tree, achieving an overall accuracy of **55.04%** and an $f1_macro$ of **53%**. However, consistent with our previous findings, the F1-score for the 'Average' class (0.36) remained significantly lower than for the 'Poor' (0.55) and 'Good' (0.67) classes, suggesting that even this fundamentally different model struggles to define the ambiguous middle ground.

3.2.2 Hyperparameter Tuning Strategy

Rather than relying solely on a brute-force grid search, we adopted a more sophisticated, two-phase strategy to identify the optimal hyperparameters for the KNN model.

Table 8: Performance Metrics for the Baseline KNN Model (K=5)

Classification Report					Confusion Matrix			
Class	Precision	Recall	F1-Score	Support	True Label	Predicted		
Poor	0.50	0.63	0.55	1542				
Average	0.39	0.33	0.36	1522				
Good	0.69	0.64	0.67	2414				
Accuracy			0.55	5478	Poor	964	349	229
Macro Avg	0.53	0.53	0.53	5478	Average	558	505	459
Weighted Avg	0.55	0.55	0.55	5478	Good	422	446	1546

Phase 1: Visual Identification of Optimal K. We first plotted the classification error rate against a wide range of K values (from 1 to 59). This "Elbow Method" for classification, shown in Figure 11 (left), revealed not a sharp elbow, but a "valley" of optimal performance, with the error rate reaching a stable minimum for K values between approximately 30 and 50. This initial analysis provided a crucial insight: it allowed us to narrow down the search space for K to the most promising region.

Phase 2: Targeted GridSearchCV. Armed with the knowledge from our visual analysis, we performed a targeted `GridSearchCV`. We focused the search for `n_neighbors` on the identified optimal range and simultaneously tested different `weights` and `metric` options. The final parameter grid was:

- `n_neighbors`: [29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53]
- `weights`: ['uniform', 'distance']
- `metric`: ['euclidean', 'manhattan', 'minkowski']

This intelligent, two-phase approach combines visual intuition with exhaustive searching, ensuring a robust and efficient tuning process. The search identified the following optimal configuration: `n_neighbors`: 49, `metric`: 'manhattan', `weights`: 'distance'

This result indicates that using the Manhattan distance and weighting neighbors by the inverse of their distance, with a large neighborhood of K=49, yields the best performance.

3.2.3 Tuned Model Evaluation

The optimized KNN model demonstrated a significant improvement over the baseline, achieving a final accuracy of **59.36%** and `f1_macro` of **54%**. This represents the best performance of any model thus far. The detailed metrics are presented in Table 9.

The tuned model shows a marked improvement in recall for both the 'Poor' (0.64) and 'Good' (0.80) classes. However, once again, this gain comes at the expense of the 'Average' category, whose recall drops to 0.23. This result powerfully reinforces our central hypothesis: even when using a distance-based algorithm, the model finds it optimal to resolve the ambiguity of the 'Average' class by assigning its instances to the more clearly defined 'Poor' and 'Good' categories. The model's logic, driven by proximity, still struggles to find a coherent, distinct "neighborhood" for average movies.

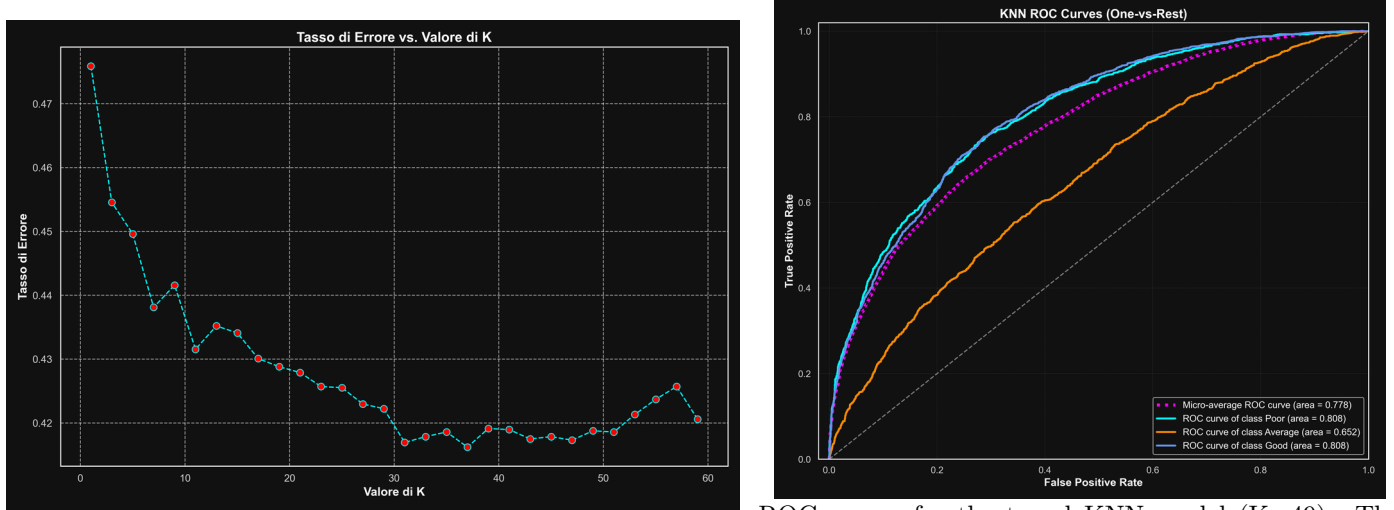
Table 9: Performance Metrics for the Tuned KNN Model

Classification Report					Confusion Matrix			
Class	Precision	Recall	F1-Score	Support	True Label	Predicted		
Poor	0.55	0.64	0.59	1542				
Average	0.48	0.23	0.31	1522				
Good	0.64	0.80	0.71	2414				
Accuracy			0.59	5478	Poor	981	177	384
Macro Avg	0.56	0.55	0.54	5478	Average	500	343	679
Weighted Avg	0.57	0.59	0.57	5478	Good	297	189	1928

3.2.4 Visual Performance Analysis

Figure 11 illustrates both the tuning process and the final performance. The error rate curve (left) visually confirms the "valley" of optimal performance for K that guided our targeted search. The final ROC curve analysis (right) reveals excellent discriminative ability for the 'Poor' (AUC = 0.808) and 'Good' (AUC = 0.808) classes. The AUC for the 'Average' class (0.652), while slightly improved over the Decision Tree, remains substantially lower, confirming its status as the most challenging category.

Figure 11: Visual analysis of KNN tuning and performance.



Error rate curve used to identify the optimal range for K , showing a performance valley between $K=30$ and $K=50$.

ROC curves for the tuned KNN model ($K=49$). The curves for 'Poor' and 'Good' classes show strong performance.

3.2.5 Conclusion

The K-Nearest Neighbors classifier achieved the highest performance with 59.36% accuracy and 54% $f1_macro$, excelling at identifying movies with strong consensus ('Poor' and 'Good' categories).

Critically, KNN corroborates the Decision Tree hypothesis: despite fundamentally different instance-based logic, it also struggles with the 'Average' class. This consistent behavior across disparate algorithms confirms that *the issue stems from data characteristics reflecting human nature, not algorithmic limitations*.

The 'Average' category represents a nebulous zone of subjective disagreement rather than a well-defined concept. KNN's inability to find coherent neighborhoods for these points mirrors real-world difficulty in defining mediocrity. Both models succeed in revealing a profound insight: the dataset accurately captures the inherent ambiguity of 'Average' ratings, making it predictably difficult for any classifier to master.

3.3 Classification by Naive Bayes

In this section, we evaluate our third and final classification algorithm, Naive Bayes. This model is fundamentally different from the previous two, as it is a probabilistic classifier based on Bayes' Theorem. Its core strength lies in its computational efficiency and simplicity, which stems from a **naive assumption**: that all features are conditionally independent of one another, given the class. Our analysis focuses on `GaussianNB`, the most theoretically appropriate variant for our mixed data type, but we also include an experiment with `BernoulliNB` to quantify the importance of correct model selection.

3.3.1 Gaussian Naive Bayes Performance

Given our feature set, which contains both standardized continuous values and binary flags, the `GaussianNB` classifier is the most suitable choice. This model assumes that the likelihood of continuous features for each class follows a Gaussian (normal) distribution. The primary hyperparameter to tune is `var_smoothing`, a stability parameter added to the variance of each feature to prevent issues with zero-variance features.

We used `GridSearchCV` to find the optimal `var_smoothing` value, searching over a wide range of logarithmically spaced candidates. The optimal value was found to be approximately **0.454**. The performance of the tuned `GaussianNB` model is detailed in Table 10.

The model achieved an overall accuracy of **54.31%** and `f1_macro` of **45%**. While this is a respectable result for such a simple model, it is lower than the performance of both the Decision Tree and KNN. Once again, the model’s performance on the ‘Average’ class is exceptionally poor, with an F1-score of only 0.14. This result provides further evidence that even a probabilistic approach struggles to model the ambiguous nature of the ‘Average’ category.

Table 10: Performance Metrics for the Tuned Gaussian Naive Bayes Model

Classification Report					Confusion Matrix			
Class	Precision	Recall	F1-Score	Support	True Label	Predicted		
Poor	0.50	0.57	0.53	1542				
Average	0.38	0.08	0.14	1522				
Good	0.58	0.82	0.68	2414				
Accuracy			0.54	5478	Poor	877	104	561
Macro Avg	0.49	0.49	0.45	5478	Average	540	128	854
Weighted Avg	0.50	0.54	0.49	5478	Good	337	107	1970

3.4 Classification: Final Discussion

After rigorous tuning, a clear performance hierarchy emerged, as summarized in Table 11.

K-Nearest Neighbors emerged as the best performer, achieving highest accuracy (59.36%) and micro-average AUC (0.778). Its instance-based nature with distance weighting effectively captured complex movie similarities, though at higher computational cost during prediction.

Decision Tree provided superior interpretability, revealing which features drive ratings (content type, release year). While solid in performance, it was surpassed by KNN’s predictive power.

Naive Bayes was the weakest performer due to its feature independence assumption poorly fitting correlated movie data. However, it demonstrated the importance of choosing appropriate model variants for the data structure.

The most significant insight transcends individual model performance: all three models consistently struggled with the ‘Average’ class. This reflects a fundamental truth about human perception—**excellence and failure generate consensus, while mediocrity breeds subjective disagreement**. Great or terrible films create strong, clear signals in data. Average films lack consensus, representing varied personal tastes that produce noisy, inconsistent signals inherently difficult to model.

While KNN achieved the best predictive performance, the true success lies in quantitatively confirming that subjective concepts like mediocrity remain resistant to simple classification, even with sophisticated algorithms.

Table 11: Final Performance Comparison of Tuned Models

Metric	Decision Tree	K-Nearest Neighbors	Gaussian Naive Bayes
Accuracy	0.5566	0.5936	0.5431
F1-Macro	0.51	0.54	0.45
F1-Score (Poor)	0.55	0.59	0.53
F1-Score (Average)	0.32	0.31	0.14
F1-Score (Good)	0.68	0.71	0.68
Micro-Avg AUC	0.749	0.778	0.697

4 Pattern Mining

Pattern mining is a data mining technique focused on discovering interesting, unexpected, and useful patterns in large datasets. Our approach was structured to not only identify these patterns but to make them actionable, culminating in a predictive classifier. This chapter is divided into four main phases: (1) Data Preprocessing, (2) Frequent Pattern Extraction, (3) Association Rule Mining, and (4) Classifier Construction and Evaluation.

4.1 Data Preprocessing for Transactional Analysis

The foundational step for any pattern mining task is to transform the data into a **transactional format**. Each data point must become a *transaction* and each of its characteristics an *item*.

Target Variable Creation The original `awardWins` numerical feature was converted into a binary target variable, `hasWon`, which is central to our predictive goal. A film is assigned `True` if it has one or more wins, and `False` otherwise. This transforms the prediction problem into a binary classification task.

Discretization of Numerical Features To incorporate numerical features into our transactional model, they must be converted into categorical items. We selected a comprehensive set of numerical columns for this purpose, employing **quantile-based binning** (specifically, `pandas.qcut`) as our primary **discretization** strategy. This method is particularly robust for features with skewed distributions (such as `numVotes`, where a few blockbusters have an extremely high number of votes) because it ensures that each resulting category contains roughly the same number of data points, for instance, `numVotes` was binned into five quintiles, labeled from `VeryLow` to `VeryHigh`.

The resulting DataFrame is a boolean matrix where each row represents a film and each column represents an item. A `True` value indicates the presence of that item in the transaction.

4.2 Frequent Pattern Extraction and Analysis

After converting the dataset into a transactional format, we employed the **FP-Growth** algorithm for frequent itemset extraction, due to its superior computational efficiency relative to Apriori.

Quantitative Analysis A critical parameter in frequent pattern mining is the minimum support threshold (`min_support`), which defines the minimum frequency an itemset must have to be considered *frequent*. To select an appropriate value, we conducted an experiment by running FP-Growth with a range of decreasing support thresholds. This analysis showed an exponential explosion in the number of patterns and computation time for thresholds below 4-5%. We selected a `min_support` of **0.03** as our working threshold, a value that strikes an optimal balance between computational feasibility and the discovery of a sufficiently rich set of patterns.

Qualitative Analysis and Key Discoveries The initial extraction yielded crucial insights into the dataset’s structure:

- **Uninformative Items:** We discovered two items, `_totalVideos_None` and `_nominations_None`, with a support of 100%. This indicated that they were present in every single transaction and thus provided no discriminatory information. We flagged these items to be filtered out before rule generation to reduce noise.
- **Class Imbalance:** The analysis starkly highlighted the dataset’s severe class imbalance. The item `target_hasWon_False` exhibited a support of approximately 87%, while `target_hasWon_True` had a support of only 13%. This is a realistic reflection of the film industry but poses a significant challenge for prediction, as any naive model would achieve high accuracy by simply always predicting the majority class.
- **Promising Patterns:** Despite the imbalance, we identified several frequent patterns associated with award-winning films. The most promising combinations consistently involved items like `titleType_movie`, `genres_Drama`, and `_numVotes_VeryHigh`.

4.3 Association Rule Mining for Prediction

The next logical step was to convert the discovered frequent itemsets into actionable association rules. Our goal was strictly predictive, so we focused exclusively on rules of the form $X \rightarrow Y$, where the consequent Y was our target variable.

Targeted Rule Generation We filtered the rules to keep only those where the consequent was either `{target_hasWon_True}` or `{target_hasWon_False}`. A significant challenge emerged immediately: due to the class imbalance, finding rules predicting the minority class (`True`) with high confidence was difficult. An initial `min_confidence` threshold of 60% yielded zero rules for predicting a win.

To overcome this, we lowered the `min_confidence` threshold to **25%**. This allowed us to capture rules that, while not certain, significantly increase the probability of a win compared to the 13% baseline. The quality of these rules was then assessed using the **Lift** metric, where $\text{Lift}(X \rightarrow Y) = \frac{\text{conf}(X \rightarrow Y)}{\text{supp}(Y)}$. A lift value greater than 1 indicates a positive correlation and also indicates that the presence of X increases the presence of Y .

The "Golden Rules" of Winning Our analysis uncovered a set of powerful predictive rules. The strongest rule discovered was:

$$\{\text{_numVotes_VeryHigh}, \text{_titleType_movie}, \text{_genres_Drama}\} \rightarrow \{\text{_target_hasWon_True}\}$$

This rule achieved a **confidence of 57.1%** and a remarkable **Lift of 4.36**. This means that a dramatic movie with a very high number of votes is over four times more likely to win an award than a randomly selected film. To further illustrate the profile of a successful film, we highlight two other high-impact rules:

$$\{\text{_titleType_movie}, \text{_criticReviewsTotal_Medium}, \text{_genres_Drama}\} \rightarrow \{\text{_target_hasWon_True}\}$$

This second rule, with a **confidence of 52.7%** and a **Lift of 4.02**, reinforces the importance of the Drama genre, but introduces a new element: a medium volume of critic reviews. It suggests that critical reception, even if not overwhelmingly high in quantity, is a key factor for dramatic films. Another powerful rule points towards production scale:

$$\{\text{_titleType_movie}, \text{_totalCredits_VeryHigh}, \text{_numVotes_VeryHigh}\} \rightarrow \{\text{_target_hasWon_True}\}$$

With a **confidence of 52.6%** and a **Lift of 4.01**, this rule highlights the synergy between high popularity (**_numVotes_VeryHigh**) and large production scale (**_totalCredits_VeryHigh**). It paints a picture of a blockbuster film with a large cast and crew that captures significant public attention. Collectively, these high-lift rules consistently reinforced the profile of a winning film as one that is a popular, large-scale production, and often of the Drama genre.

4.4 A Custom Rule-Based Classifier

The final and most ambitious phase of our project was to operationalize these rules by implementing from scratch a custom, interpretable classifier.

Model V1: The "Talent Scout" Our first implementation was a simple yet effective "talent scout" model. For a given film from the test set, it checks if any of our 75 *golden rules* (those predicting a win) apply. If at least one rule's antecedent is a subset of the film's items, the model predicts **True**; otherwise, it defaults to the majority class, **False**. The performance of this initial model is detailed in Tables 12 and 13.

Table 12: Classification Report (Model V1)

Class	Precision	Recall	F1-Score
hasWon=False	0.95	0.47	0.63
hasWon=True	0.18	0.81	0.29

Table 13: Confusion Matrix (Model V1)

	Pred. False	Pred. True
Real False	2262	2533
Real True	129	554

The evaluation of this model on the held-out test set was highly informative. We deliberately ignored accuracy as a misleading metric in this imbalanced context. The key results were:

- **Recall: 81.1%**. An outstanding result. The model successfully identified 554 out of 683 actual award-winning films in the test set. This demonstrates its strength as a high-recall screening tool.
- **Precision: 17.9%**. The cost of high recall. To capture so many true positives, the model generated a large number of false positives (2533). When it predicts a win, it is correct only 18% of the time.

This V1 model proved to be an excellent screening tool, capable of creating a high-quality candidate pool from a large dataset.

Model V2: The "Balanced Judge" To improve precision, we implemented a V2 of our classifier, introducing a **confidence_threshold** parameter. In this version, a rule only "fires" if it matches a film *and* its confidence is above this threshold. Based on our trade-off analysis shown in Figure ??(b), we selected a threshold of **0.50** to create a more balanced model. The performance of this "Balanced Judge" is shown in Tables 14 and 15.

By setting a higher bar for prediction, the V2 model achieves a much healthier balance. The **Precision** for the positive class more than doubled, rising from 18% to **40%**. This comes at the expected cost of **Recall**, which decreased to 40%. Crucially, the **F1-Score** for the positive class improved significantly from 0.29 to **0.40**, indicating a much better overall model from a practical standpoint.

Table 14: Classification Report (Model V2)

Class	Precision	Recall	F1-Score
hasWon=False	0.91	0.92	0.92
hasWon=True	0.40	0.40	0.40

Table 15: Confusion Matrix (Model V2)

	Pred. False	Pred. True
Real False	4396	399
Real True	413	270

Tuning and Final Selection via Trade-off Analysis The process of selecting the 0.50 threshold was guided by the Precision-Recall trade-off analysis shown in Figure ??(b). This curve visualizes the spectrum of possible models we can create simply by adjusting the confidence threshold.

We have not only built a predictive model but have also provided a mechanism to control its behavior, allowing it to be adapted for different use cases: from high-recall initial screening to high-precision final selection. The interpretability of the underlying rules remains a key advantage over more complex "black-box" models.

5 Regression

5.1 Univariate Regression Analysis

This section details the first phase of our regression analysis, focusing on the univariate case. The objective is to predict a chosen target variable using only a single, powerful predictor.

5.1.1 Target and Predictor Selection

The selection of the target and predictor variables was a data-driven process guided by a thorough analysis of the Pearson and Spearman correlation matrices computed during the data understanding phase. We chose the most strongly correlated variable in agreement with both r and ρ .

- **Target Variable (Y):** `userReviewsTotal_proc`. We selected the processed number of user reviews as our target. This variable represents a key measure of audience engagement and is indisputably continuous. We specifically use the processed version ('_proc'), which has undergone a **log-transformation** and **standardization**. This is a critical step to mitigate the original data's strong positive skew, thereby linearizing its relationships with other variables and better satisfying the core assumptions of linear regression models.
- **Predictor Variable (X):** `numVotes_proc`. The correlation analysis revealed that the number of votes a title receives has the strongest linear relationship with the number of user reviews (Pearson's $r > 0.75$). This is a logical relationship, as a higher volume of votes indicates greater viewership, which naturally leads to more written reviews. We use the **processed version** of this variable for the same reasons outlined for the target.

Our univariate task is therefore to predict the transformed number of user reviews based solely on the transformed number of votes.

5.1.2 Model Comparison and Results

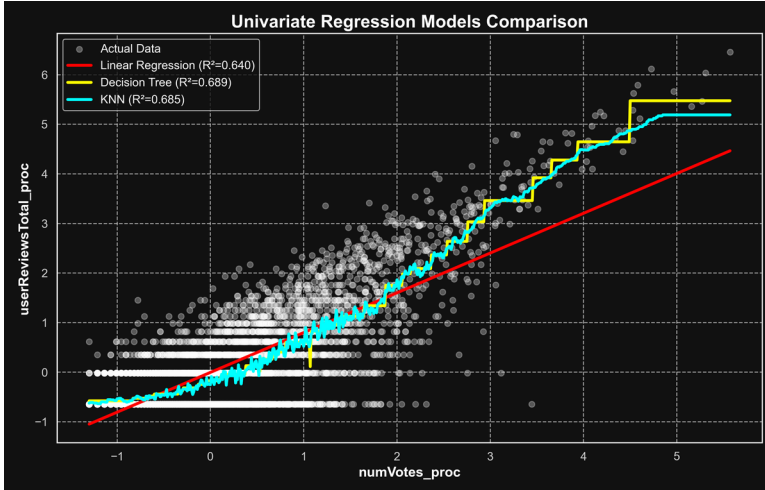
We trained and tuned three different regressors on this univariate task: Linear Regression, a Decision Tree Regressor, and a K-Neighbors Regressor. The performance metrics are summarized in Table 16, with a visual comparison of the models' fits provided in Figure 12.

The results are compelling. While the Linear Regression model provides a solid baseline with an R^2 of 0.640, the non-linear models demonstrate a superior ability to capture the underlying pattern in the data. The tuned **Decision Tree Regressor** emerges as the best performer in the univariate case, achieving the highest R^2 of **0.689**. The visual comparison clearly shows how its step-function fit follows the dense regions of the data more closely than the simple straight line of the linear model. The KNN regressor performs almost identically to the Decision Tree, confirming that a non-linear approach is more suitable for this specific relationship.

5.2 Multivariate Regression Analysis

Building upon the univariate baseline, this section details our multivariate regression analysis. The objective is to leverage a curated set of multiple predictors to improve upon the predictive accuracy achieved with a single feature. We will use the same target variable, `userReviewsTotal_proc`, and evaluate the same set of algorithms on this more complex task.

Table 16: Performance of Univariate Regression Models



Univariate Model Performance Metrics

Model	MAE	RMSE	R ²
Linear Reg.	0.4488	0.5982	0.6402
Decision Tree	0.4012	0.5558	0.6893
KNN	0.4037	0.5601	0.6846

Figure 12: Comparison of model fits for the univariate task.

5.2.1 Predictor Selection

For the multivariate task, we selected an elite squad of the top six numerical features most correlated with our target variable, as identified in our initial correlation analysis: `numVotes_proc`, `criticReviewsTotal_proc`, `numRegions_proc`, `awardNominationsExcludeWins_proc`, `totalCredits_proc`, `awardWins_proc`.

5.2.2 Model Comparison and Results

We trained and tuned the Linear Regression, Decision Tree, and KNN models using this set of six predictors. The performance of each model, summarized in Table 17, shows a dramatic improvement over the univariate case, confirming the value of a multivariate approach.

The results clearly indicate that the **K-Neighbors Regressor** is the top-performing model in the multivariate setting. It achieved the highest R² of **0.748** and the lowest error metrics (MAE and RMSE). The Decision Tree follows closely behind, while the Linear Regression model, though significantly improved, is the weakest of the three as expected.

The visual analysis in Figure 13 provides a clear intuition for these results. The "Actual vs. Predicted" plots show how the predictions of each model align with the **oracle** line of perfect prediction.

Table 17: Performance of Multivariate Regression Models

Metric	Linear Regression	Decision Tree	K-Neighbors Regressor
Mean Absolute Error (MAE)	0.3883	0.3636	0.3628
Root Mean Squared Error (RMSE)	0.5247	0.5049	0.5007
R-squared (R ²)	0.7231	0.7437	0.7479

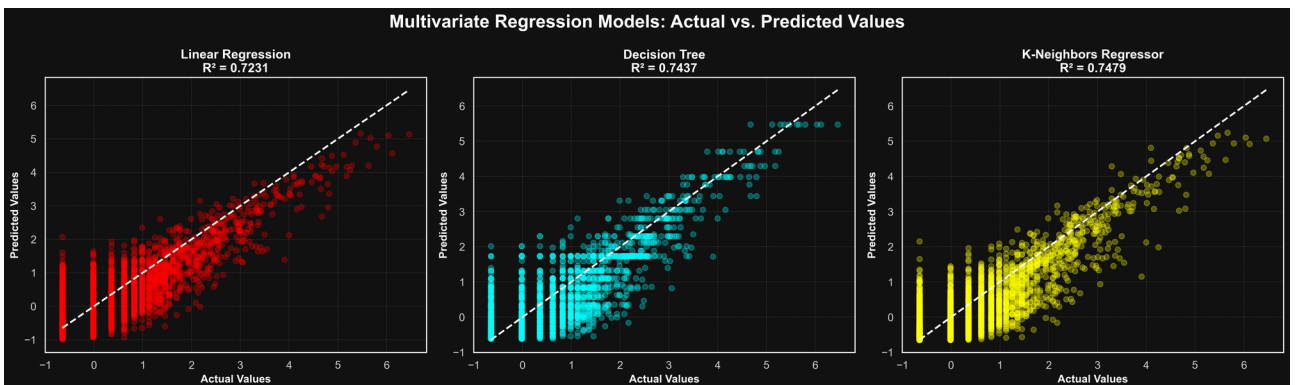


Figure 13: Comparison between actual and predicted classes across the three evaluated models.