Data Mining II Report

Elvis Lleshi

May 2025

Contents

1 Introduction **2**
- 1.1 Dataset Overview . . . 2
- 1.2 Data Preparation . . . 4
- 1.3 Normalization of Time Series . . . 7

2 Outlier Detection **7**

3 Imbalanced Learning **9**

4 Time Series **12**
- 4.1 Motifs, Discords, and Shapelets . . . 12
- 4.2 Time Series Clustering . . . 14
- 4.3 Multi-class Classification . . . 18

5 Advanced Classification **21**
- 5.1 Hyperparameter Tuning . . . 22
- 5.2 Model Performance Comparison . . . 25
- 5.3 ROC Curve Analysis . . . 26

6 Explainability **28**
- 6.1 Global explainability . . . 28
- 6.2 Local explainability . . . 29

7 Advanced Regression **30**

8 Conclusion **31**

1 Introduction

1.1 Dataset Overview

In this project, we work with two datasets:

- **imdb.csv**: A tabular dataset containing metadata about movies. It consists of **149,531 rows** and **32 columns**, including attributes such as title, genre, cast, production details, and user interaction data like ratings and reviews.
- **imdb_ts.csv**: A time series dataset representing the domestic daily gross income of movies. It includes **1,134 rows** and **104 columns**, where the first 100 columns represent the revenue for the first 100 days from the release date, and the remaining columns contain metadata such as genre and rating.

These datasets serve as the foundation for all subsequent preprocessing, modeling, and time series analysis conducted throughout the project.

Table 1: Feature Description of `imdb_ts.csv` Dataset

Attribute	**Description**	**Data Type**
genre	A list of genres associated with each movie (e.g., `['Drama', 'Romance']`).	object (list as string)
rating	The average user rating received by the movie.	float
rating_category	The categorical label assigned based on the rating range (e.g., low, medium, high).	object
0–99	Daily gross income values for each day after release (from day 0 to day 99). Each column corresponds to a single day.	float

Table 2: Feature Description of `imdb.csv` Dataset

Attribute	**Description**	**Data 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 adult. 0: non-adult title; 1: adult title.	int
startYear	Represents the release year of a title. In the case of TV Series, it is the series start year.	int
endYear	TV Series end year.	object
numVotes	Number of votes the title has received.	int
numRegions	The regions number for this version of the title.	int
worstRating	Worst title rating.	int
bestRating	Best title rating.	int
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 within the IMDb title page.	int
totalVideos	Total Number of Videos for the title within the IMDb title page.	int
totalCredits	Total Number of Credits for the title.	int
criticReviewsTotal	Total Number of Critic Reviews.	int
awardWins	Number of awards the title won.	int
awardNominationsExcludeWins	Number of award nominations excluding wins.	int
titleType	The type/format of the title (e.g. movie, short, tvseries, etc.).	object
rating	IMDB title rating class.	object
ratingCount	The total number of user ratings submitted for the title.	int
countryOfOrigin	The 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 Users Reviews.	int
castNumber	Total Number of Cast individuals present within the IMDb title page.	int
CompaniesNumber	Total Number of companies that worked for the title.	int
regions	The regions for this version of the title.	object
averageRating	Weighted average of all the individual user ratings.	float
externalLinks	Total Number of External Links the title has within the IMDb page.	int
quotesTotal	Total Number of quotes the title has within the IMDb page.	int
writerCredits	Total number of writer credits of the title.	int
directorCredits	Total number of director credits of the title.	int
soundMixes	Technical specification of the sound mixes available for the title.	object

1.2 Data Preparation

The preprocessing phase involved several important steps aimed at cleaning, transforming, and enriching the dataset to ensure its usability for downstream modeling.

Missing Values

The column `countryOfOrigin` was the only one containing missing values, with approximately 39,987 missing entries (˜26.74% of the dataset). All other columns were complete. Values such as `\N` were considered as missing and were converted to NaN for proper processing.

Duplicate Removal

Upon inspection, 10 rows were identified as duplicates and were removed to avoid data leakage and redundancy.

Feature Elimination

Columns such as `worstRating` and `bestRating` were discarded. These columns exhibited no meaningful variability, with fixed values of 1 and 10 respectively, and thus provided no discriminative power for modeling.

Feature Engineering

A new feature, `durationYears`, was created by calculating the difference between `endYear` and `startYear`. Since `endYear` included many invalid values (e.g., `\N`), the missing durations were filled using the most frequent valid duration in the dataset (mode).

Column-Specific Treatments

- `soundMixes`: Though not technically missing, many entries were empty (e.g., `[]`). The missing values were imputed using the most frequent value (mode) within groups defined by `titleType`, `genres`, and `startYear`.
- `runtimeMinutes`: This feature contained `\N` values (approximately 40,195 rows), which were converted to NaN. These were then imputed using the median `runtimeMinutes` value within each `titleType` category. Finally, the column was cast to integer type.
- `regions`: Missing entries marked as `\N` were replaced with the placeholder label `unknown`.

Correlation-Based Feature Reduction

A Pearson correlation matrix was computed to examine the relationships between numerical variables. Highly correlated features (correlation coefficient>0.8) such as `numVotes`, `castNumber`, `externalLinks`, and `userReviewsTotal` were identified and considered for removal to reduce multicollinearity and improve model generalization.

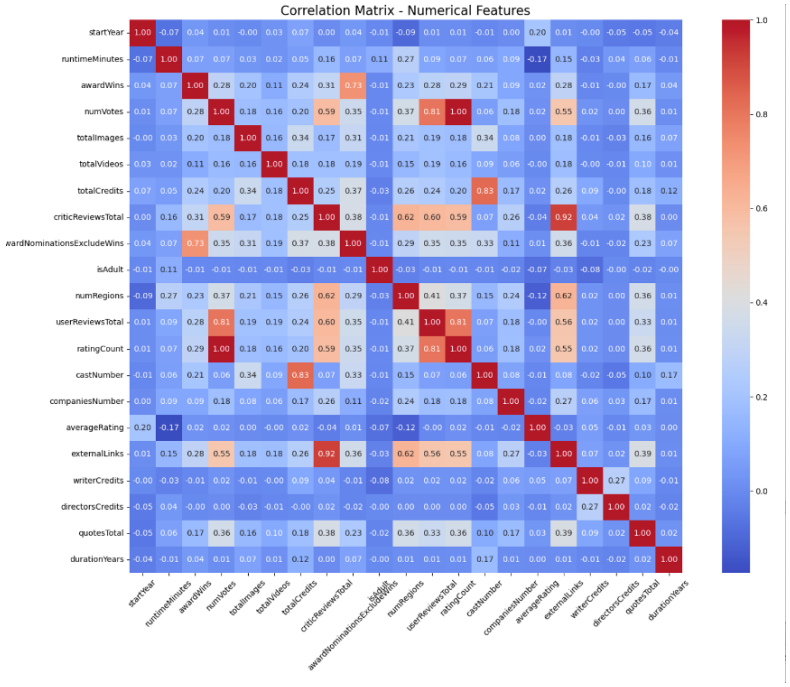


Figure 1: Correlation Matrix of Numerical Features

Categorical Transformations

The `rating` column, originally containing interval-like textual values (e.g., `(0,1]`), was converted into a categorical variable by retaining the upper bound of each interval (e.g., `1` from `(0,1]`).

Genre Processing

The `genres` column consists of multiple genres concatenated in a single string (e.g., `Documentary,Short`). After splitting them into lists, frequency analysis revealed the presence of 28 unique genre types. To simplify the feature space and mitigate the effects of sparsity, we grouped less frequent genres into a single category labeled `Other`.

This decision was guided by the need to improve the class distribution balance, especially for tasks involving classification where class imbalance can degrade model performance. Retaining only the most common genres while aggregating rare ones ensures a more stable learning process and reduces noise caused by underrepresented categories.

Finally, only the first genre in each list was selected as the `main_genre`, as it typically represents the dominant theme of the movie.

The `imdb_ts.csv` dataset consists of 1,134 rows, each representing a unique movie, and 104 columns. Among these, columns `0` to `99` correspond to the daily gross revenue for the first 100 days following a movie's release. The remaining columns contain metadata such as `genres`, `rating`, and `rating_category`.

Metadata Cleaning and Genre Simplification

The `genres` field was initially stored as a string representation of a list (e.g., `['Drama', 'Action']`) with up to three genres per movie. This led to a total of 184 unique genre combinations. To reduce complexity and ensure consistency in the metadata, the following steps were applied:

- The string-formatted lists were parsed into actual Python lists.
- The first genre from each list was extracted and stored as a new feature called `main_genre`, based on the assumption that it reflects the dominant thematic aspect of the film.
- A frequency analysis was performed across all genres to evaluate their distribution.
- Genres with very low frequency were grouped under the label `Other` to reduce noise and sparsity.

The bar charts below visualize the distribution of genres before and after this simplification step. As can be seen, the most important and representative genres (e.g., `Action`, `Comedy`, `Drama`) were preserved, allowing for a more balanced and interpretable feature space that is beneficial for downstream tasks such as classification and clustering.

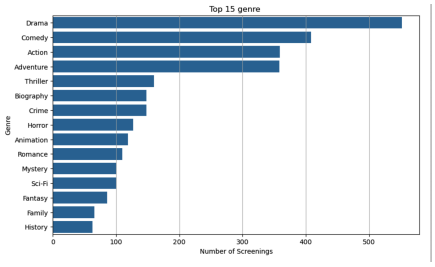


Figure 2: Top 15 genres before simplification

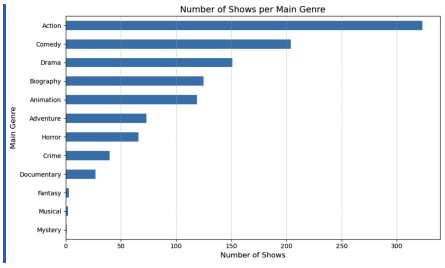


Figure 3: Main genre distribution after simplification

Rating Transformation

Two rating-related features were included in the dataset: a continuous score (`rating`) and a binned version (`rating_category`). The latter was specifically retained to support classification tasks, where ratings were grouped into discrete categories such as low, medium, and high.

1.3 Normalization of Time Series

To enable meaningful comparisons and consistent input for machine learning models, the 100-dimensional time series data was normalized using `MinMaxScaler`. This transformation scaled each revenue trajectory to a uniform range $[0, 1]$, thus preserving relative growth patterns while removing the influence of raw revenue magnitudes.

Note on Time Series Representation

Given that each time series in our dataset spans exactly 100 timestamps, the dimensionality is already manageable and well-suited for traditional analysis methods. Therefore, there was no need to apply dimensionality reduction techniques such as Symbolic Aggregate approXimation (SAX) or Piecewise Aggregate Approximation (PAA), which are typically used for very long time series. Using such techniques here could have led to unnecessary loss of temporal detail.

Final Dataset Structure

After the cleaning and transformation steps, each instance in the dataset contained:

- A normalized time series of daily revenues (length 100),
- A simplified and unified genre label (`main_genre`),
- Both a numerical and categorical representation of the movie rating.

This prepared dataset provided a clean and structured foundation for advanced time series analyses, including clustering, dimensionality reduction, and pattern discovery.

2 Outlier Detection

Outlier detection aims to identify anomalous observations that may distort statistical patterns or adversely affect model training. To address this, we employed three distinct methods from different algorithmic families: Isolation Forest, Local Outlier Factor (LOF), and KNN-based detection.

As a second method for outlier detection, we employed *Isolation Forest*, a tree-based algorithm that isolates anomalies rather than profiling normal points. This method is particularly efficient in high-dimensional datasets and works by

recursively partitioning data and measuring how easily individual points can be isolated. Shorter average path lengths indicate stronger anomalies.

Feature Selection and Preprocessing

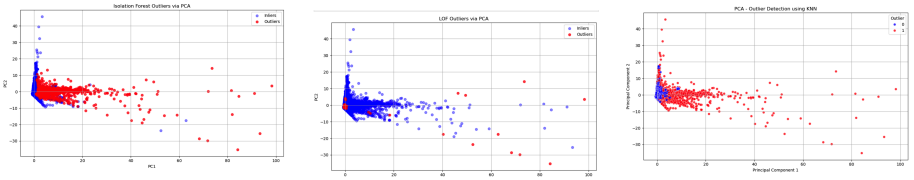
We used the same set of numerical features selected for the LOF and KNN models, including:

- `runtimeMinutes`, `averageRating`, `ratingCount`, `awardWins`,
- `companiesNumber`, `writerCredits`, `directorsCredits`,
- `totalImages`, `totalVideos`, `quotesTotal`, `durationYears`

The data was standardized using `StandardScaler`. The Isolation Forest model was configured with a contamination rate of 1% to detect the most anomalous entries. A total of **1,496 outliers** were identified.

Outlier Detection via PCA Projection

To visualize how different outlier detection algorithms behave, we projected the data into 2D using PCA and highlighted the detected outliers. Figure 4 compares Isolation Forest, LOF, and KNN-based methods side-by-side.



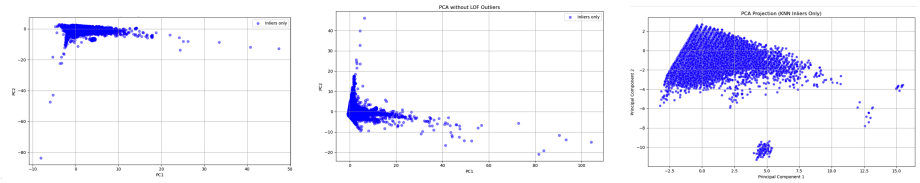
(a) Isolation Forest (b) Local Outlier Factor (LOF) (c) KNN-Based

Figure 4: PCA-based visualization of outliers detected by different algorithms.

Observation. Isolation Forest identifies globally sparse observations, LOF highlights locally deviating points near medium-density clusters, and KNN tends to mark edge points near dense regions. Each method captures different aspects of the data structure.

Outlier Removal and Common Anomalies

Removing outliers has a noticeable impact on the distribution of data. As shown in the PCA visualizations below, each method—Isolation Forest, LOF, and KNN—yields a visibly more compact and structured space after filtering. This improves cluster separation and model reliability in later stages such as classification or clustering.



Isolation Forest LOF KNN

Figure 5: PCA Projections after Outlier Removal by Method

After comparing the three methods, we identified 8 data points that were classified as outliers by all of them. These consistently anomalous observations likely represent highly divergent entries within the dataset.

Table 3: Outliers Detected by All Three Methods

ID	IF Score	LOF Score	KNN Score
22889	0.208	9.98	100.97
12910	0.166	7.69	60.30
10092	0.142	8.18	131.37
59537	0.137	8.07	149.31
24239	0.132	7.55	61.34
12902	0.130	9.46	156.41
96671	0.089	6.23	78.13
13598	0.049	12.30	117.73

Observation. These data points score high across all methods, suggesting they are structural anomalies. Their persistent detection across diverse techniques indicates they should be further investigated or removed prior to any modeling task.

3 Imbalanced Learning

The binary classification target selected in this project is `isAdult`, which specifies whether a title is intended for an adult audience (1) or not (0). An initial inspection of class distribution showed a strong imbalance:

- **Not Adult (0):** 146,798 titles (˜98%)
- **Adult (1):** 2,733 titles (˜2%)

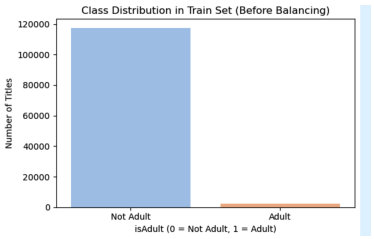


Figure 6: Class Distribution in the Training Set Before Balancing

This imbalance poses a risk of skewing learning algorithms toward the majority class, which can result in poor generalization and very low recall for the minority class. To address this, we tested several sampling techniques, including both oversampling and undersampling methods:

- **Random Oversampling:** Duplicates samples from the minority class at random to balance the dataset.
- **Random Undersampling:** Reduces the number of majority class samples by randomly removing instances.
- **SMOTE (Synthetic Minority Over-sampling Technique):** Generates synthetic samples of the minority class using feature-space interpolation.
- **KNN-based Oversampling:** Uses k-nearest neighbors to generate new samples based on local structure.
- **Cluster Centroids:** Performs undersampling by replacing clusters of majority samples with their centroids.
- **Tomek Links:** A data cleaning method that removes ambiguous samples near the class boundary, helping classifiers distinguish between classes more effectively. It does not directly balance class frequencies, but improves decision boundaries.

Baseline Performance Before Balancing

To assess model behavior on the original, imbalanced dataset, we trained both a Decision Tree and a KNN classifier. As expected, both models performed well on the majority class but struggled to identify minority class instances due to the severe imbalance.

Table 4: Baseline Performance Before Balancing (Class 1: Adult)

Classifier	**Accuracy**	**Recall (1)**	**F1-score (1)**
Decision Tree	0.99	0.82	0.84
KNN	0.98	0.62	0.64

These results confirm that high accuracy alone is misleading in imbalanced settings, as it does not reflect poor performance on the minority class. This highlighted the necessity of applying data balancing techniques to ensure more equitable and meaningful classification.

Classifier Performance with Oversampling Methods

To assess the effectiveness of different balancing methods, we trained and evaluated both Decision Tree and KNN classifiers on datasets balanced using various oversampling strategies. The table below summarizes the accuracy, precision, and F1-score for class 1 (`Adult`) across each configuration.

Table 5: Classification Performance by Balancing Method and Classifier (Class 1: Adult)

Balancing Method	**Classifier**	**Accuracy**	**Precision (1)**	**F1-score (1)**
Random Oversampling	Decision Tree	0.99	0.98	0.98
Random Oversampling	KNN	0.91	0.76	0.79
SMOTE	Decision Tree	0.98	0.90	0.90
SMOTE	KNN	0.93	0.81	0.83
KNN Oversampling	Decision Tree	0.93	0.80	0.82
KNN Oversampling	KNN	0.92	0.79	0.81

Classifier Performance with Undersampling Methods

The table below summarizes the classification performance of Decision Tree and KNN classifiers on datasets balanced using different undersampling techniques. The focus remains on class 1 (`Adult`) metrics.

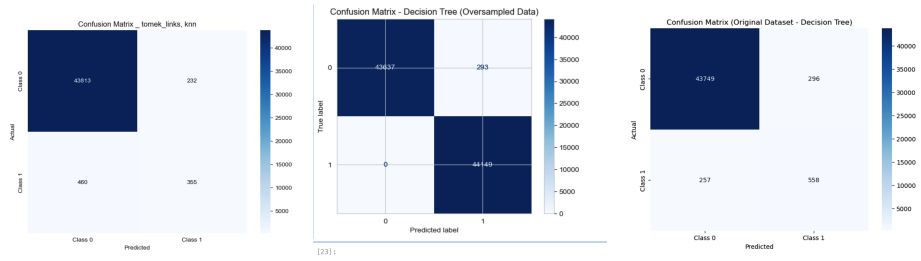
Table 6: Classification Performance by Undersampling Method and Classifier (Class 1: Adult)

Undersampling Method	**Classifier**	**Accuracy**	**Precision (1)**	**F1-score (1)**
Random Undersampling	Decision Tree	0.92	0.18	0.30
Random Undersampling	KNN	0.85	0.10	0.18
Tomek Links	Decision Tree	0.98	0.60	0.51
Tomek Links	KNN	0.99	0.65	0.67
Cluster Centroids	Decision Tree	0.63	0.05	0.09
Cluster Centroids	KNN	0.63	0.05	0.09

Comparison of Confusion Matrices

To visually assess model performance across different balancing strategies, we compared the confusion matrices of the best-performing configuration from each category:

- **Tomek Links + KNN** was the top performer among undersampling techniques, reaching an accuracy of 0.99. Its confusion matrix shows effective minority class detection with low false positives.
- **Random Oversampling + Decision Tree** yielded the highest overall performance across all tested methods, achieving an accuracy of 0.99 and an F1-score of 0.98 for the minority class . This configuration produced near-perfect classification for both classes.
- **Original Dataset + Decision Tree**, despite a high overall accuracy (0.99), exhibited lower recall and F1-score for class 1 , reflecting the negative impact of class imbalance.



Tomek Links + KNN | **Random Oversampling + Decision Tree** | **Original Dataset + Decision Tree**

Figure 7: Confusion Matrices for Top Configurations from Each Strategy

4 Time Series

4.1 Motifs, Discords, and Shapelets

Pattern mining in time series can provide valuable insights into both regular and anomalous behaviors. We focused on three key types of patterns:

- **Motifs**: subsequences that appear frequently within a time series;
- **Discords**: subsequences that are the most dissimilar to any other part of the series;
- **Shapelet-like patterns**: short, discriminative segments that highlight distinct temporal behavior.

Illustrative Example

To illustrate these concepts, we applied motif and discord detection to the 100-day normalized revenue trajectory of a representative movie (row 5 in the

dataset). As shown in Figure 8, two motifs were found in different regions of the series, and one clear discord—indicating a segment with unique dynamics.

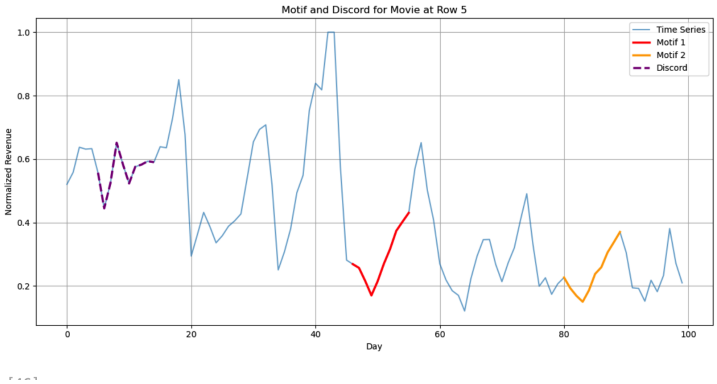


Figure 8: Motif and Discord Detection for Movie at Row 5

This initial analysis highlights how motifs and discords can reveal regular patterns and unusual deviations, respectively—serving as a foundation for behavioral interpretation.

Global Discord Analysis Across the Dataset

To generalize this analysis, we extended discord detection across the entire dataset. For each movie's revenue time series, we used the STUMPY library to compute the matrix profile (with subsequence length $m = 10$) and extracted the strongest discord score.

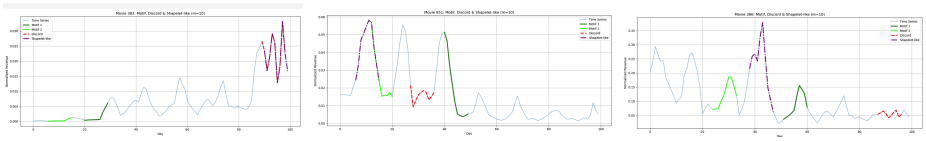
Movie Index	Discord Score
382	3.61
851	3.56
366	3.52
7	3.43
172	3.43

Figure 9: Top 5 Movies with the Highest Discord Scores ($m = 10$)

These discord subsequences deviate significantly from the remainder of each time series, offering clues to unique audience behavior or data irregularities.

Motif, Discord, and Shapelet-like Pattern Alignment

To explore the interaction between these patterns, we visualized motif, discord, and shapelet-like segments jointly for the most anomalous cases. Figure 10 displays the alignment of these patterns in the top 3 movies with highest discord scores.



Movie 382 **Movie 851** **Movie 366**

Figure 10: Motif, Discord, and Shapelet-like subsequences in the top 3 anomalous movies

- In several cases (e.g., Movie 382), discord and shapelet-like segments overlap—suggesting they reflect informative anomalies.
- Motifs typically occur earlier in the trajectory, revealing stable revenue patterns.
- Shapelets often align with sudden increases or drops, encoding behavioral shifts.

Conclusion. This combination of global and local pattern detection enriches the interpretability of temporal behavior, highlighting both structure and irregularity within movie performance dynamics.

4.2 Time Series Clustering

To explore temporal structure and segment the movies based on their revenue evolution over time, we applied unsupervised clustering techniques to the time series data. Specifically, we utilized two methods: **K-Means** and **Hierarchical Clustering**. These methods were chosen to allow comparison between partition-based and agglomerative approaches.

Clustering was performed using two different distance metrics:

- **Euclidean Distance**, which captures shape similarity point-by-point;
- **Dynamic Time Warping (DTW)**, which accounts for temporal distortions and flexible alignment between sequences.

For K-Means clustering, we determined the optimal number of clusters by evaluating the **Silhouette Score** for values of k ranging from 2 to 10. As shown in Figure 11, the highest silhouette score (0.757) was obtained at $k = 2$, suggesting two dominant behavioral clusters across the time series.

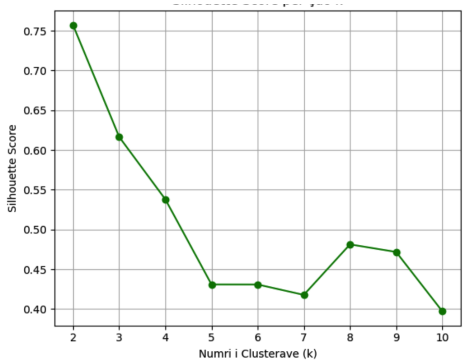


Figure 11: Silhouette scores for different values of k using Euclidean distance.

Cluster Visualization by Distance and Method

To better interpret the clustering results, we visualized the assignments using PCA and t-SNE for the two distance metrics—Euclidean and DTW—applied to both K-Means and Hierarchical clustering.

K-Means Clustering with PCA The following plots show the clusters obtained using K-Means with both Euclidean and DTW distances, visualized using PCA.

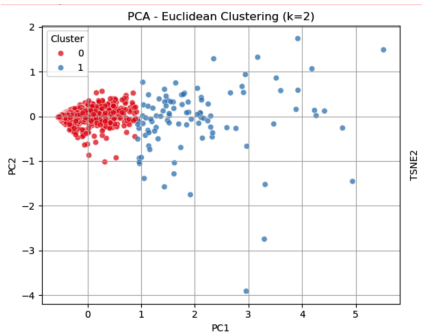


Figure 12: PCA - K-Means with Euclidean Distance

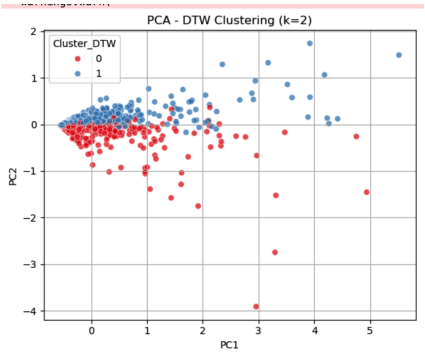


Figure 13: PCA - K-Means with DTW Distance

To evaluate how the clustering assignments changed with the distance metric, we compared the results of K-Means using Euclidean and DTW distances. A total of **677 movies** were assigned to different clusters when switching from Euclidean to DTW. Table 7 shows a sample of these cases.

Table 7: Sample of Movies with Changed Cluster Assignment (Euclidean vs DTW)

Index	Cluster (EUCL)	Cluster (DTW)	Genre	Rating Category
4	0	1	Adventure	High
7	0	1	Action	High
8	0	1	Action	Medium High
15	0	1	Drama	Medium High
16	0	1	Animation	Medium High

This comparison highlights that DTW tends to reassign movies whose temporal structure deviates slightly from the typical trajectory, emphasizing its sensitivity to sequence alignment.

Hierarchical Clustering with t-SNE For Hierarchical clustering, we used t-SNE to project the clusters. The following plots illustrate the result of using both Euclidean and DTW distances.

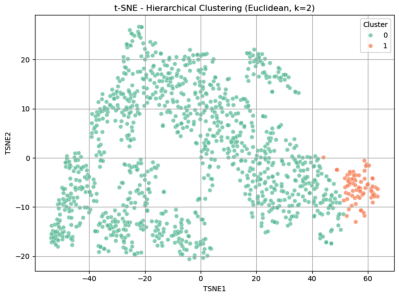


Figure 14: t-SNE - Hierarchical with Euclidean Distance

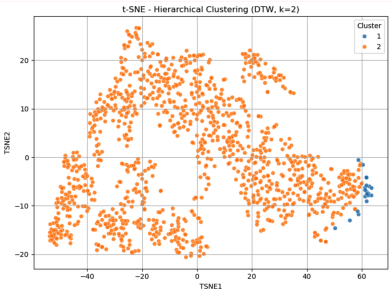


Figure 15: t-SNE - Hierarchical with DTW Distance

To evaluate the impact of the distance metric on Hierarchical clustering, we also examined the cluster assignments using Euclidean and DTW distances. Unlike K-Means, the hierarchical model exhibited more consistent assignments across distances. Out of the total dataset, only a small portion of movies changed their cluster membership. Table 8 presents a sample of such cases.

Table 8: Sample of Movies with Changed Cluster Assignment (Hierarchical: Euclidean vs DTW)

Index	Cluster (EUCL)	Cluster (DTW)	Genre	Rating Category
59	1	2	Action	High
103	1	2	Comedy	Medium
131	1	2	Animation	Medium High
215	1	2	Drama	Medium High
279	1	2	Adventure	Medium

These results suggest that Hierarchical clustering is less sensitive to the choice of distance metric compared to K-Means, producing relatively stable cluster structures even under dynamic temporal alignment such as DTW.

Cluster Metadata Insights

To better interpret the characteristics of the clusters, we analyzed their metadata distributions—specifically genres and rating categories—using the clustering results from both K-Means and Hierarchical algorithms, with Euclidean and DTW distances. This comparison helps illustrate how different distance metrics shape the segmentation based on underlying movie properties.

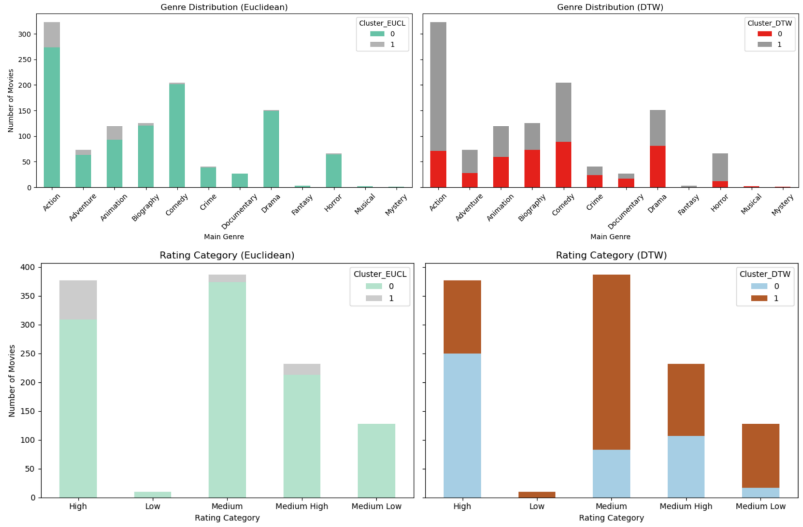


Figure 16: Genre and Rating Category Distribution per Cluster (K-Means: Euclidean vs DTW)

K-Means. As shown in Figure 16, the clusters obtained using Euclidean distance concentrate `Action`, `Comedy`, and `High`-rated movies in Cluster 0. Meanwhile, the DTW-based clustering offers a more even distribution, with Cluster

1 encompassing a larger variety of genres and more `Medium High` and `Medium Low` rated titles. This supports the idea that DTW captures temporal structure better than Euclidean distance, which focuses on magnitude similarity.

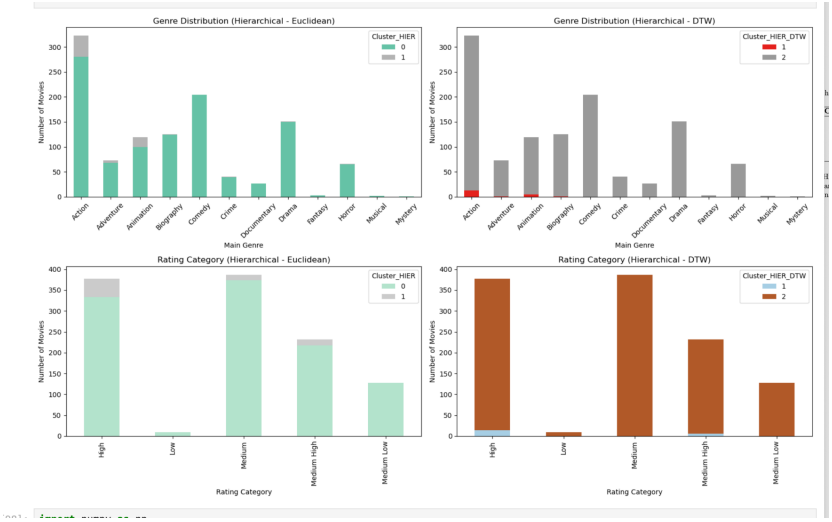


Figure 17: Genre and Rating Category Distribution per Cluster (Hierarchical: Euclidean vs DTW)

Hierarchical Clustering. In Figure 17, we observe that Euclidean-based clusters are less balanced—Cluster 0 dominates with popular genres and high ratings. In contrast, DTW again reveals richer heterogeneity: Cluster 2 shows a strong presence of `Documentary`, `Drama`, and `Medium` ratings. This confirms that DTW's sensitivity to sequence dynamics can yield semantically distinct clusters aligned with viewer engagement and content types.

4.3 Multi-class Classification

To evaluate the predictive power of temporal patterns in the revenue time series, we trained K-Nearest Neighbors (KNN) classifiers using two distance metrics: Euclidean and Dynamic Time Warping (DTW). We conducted separate experiments to predict both `main_genre` and `rating_category`. Classification performance was assessed through standard metrics such as accuracy, precision, recall, and F1-score.

Performance Overview

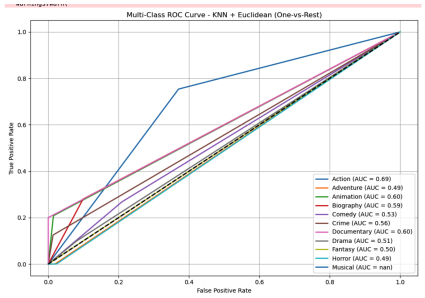
Table 9 summarizes the classification performance across both tasks and distance metrics.

Table 9: KNN Classification Metrics for Main Genre and Rating Category

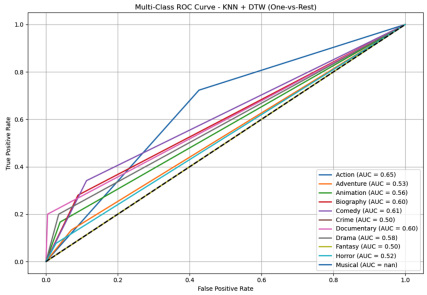
Target	Distance	Accuracy	Precision	Recall	F1-Score
Main Genre	Euclidean	0.34	0.31	0.34	0.30
	DTW	0.36	0.34	0.36	0.33
Rating Category	Euclidean	0.47	0.44	0.47	0.45
	DTW	0.48	0.47	0.48	0.46

ROC Curve Analysis

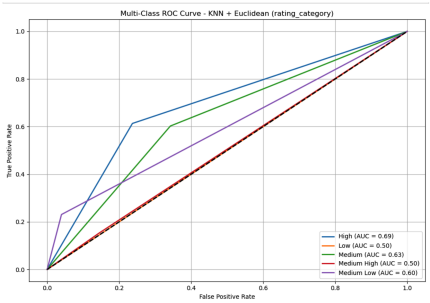
Figure 18 shows the ROC curves for each configuration. These plots give insight into class-wise performance using a one-vs-rest evaluation strategy.



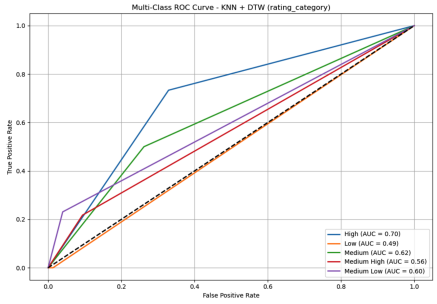
(a) Main Genre - KNN + Euclidean



(b) Main Genre - KNN + DTW



(c) Rating Category - KNN + Euclidean



(d) Rating Category - KNN + DTW

Figure 18: ROC curves for genre and rating classification using KNN and two distance metrics

(a) ROC për `main_genre` me distancë Euclidean tregon AUC më të lartë për klasën *Action*, ndërsa klasat e tjera mbeten më afër vijës diagonale. **(b)** Me DTW, përmirësimi është modest për disa klasa si *Animation* dhe *Biography*, por *Action* mbetet më e dallueshme. **(c)** Për `rating_category`, Euclidean identifikon më mirë klasat *High* dhe *Medium*, por ka vështirësi me kategoritë e

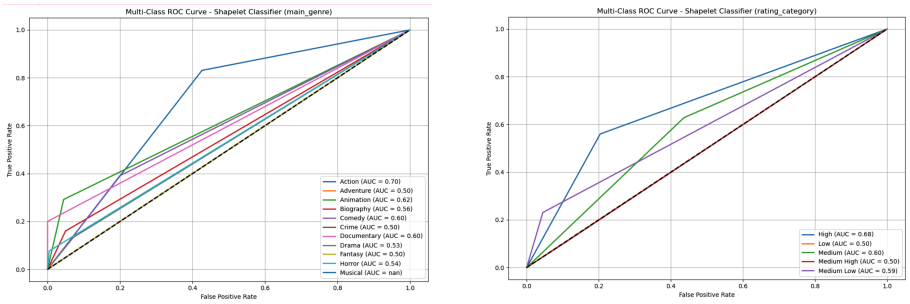
ndërmjetme. **(d)** DTW ndihmon në dallimin e kategorive si *Medium High*, duke ruajtur performancën e mirë për *High*.

Shapelet-based Classification Performance

The Shapelet Transform Classifier was applied to assess the predictive power of local shape patterns in forecasting both main genres and rating categories. The table below summarizes the classification results using accuracy, precision, and recall, while the ROC curves provide a visual comparison across classes.

Table 10: Classification Performance using Shapelet Transform Classifier

Task	**Accuracy**	**Precision**	**Recall**
Main Genre	0.38	0.34	0.38
Rating Category	0.45	0.42	0.45



(a) Main Genre - Shapelet Classifier

(b) Rating Category - Shapelet Classifier

Figure 19: ROC curves using Shapelet-based classifier for genre and rating prediction

The ROC curves highlight that the shapelet-based classifier performs best on frequent classes like `Action` and `High` ratings, with AUC scores up to 0.70. Less frequent classes, such as `Adventure` or `Low`, show near-random performance (AUC ≈ 0.50), reflecting the imbalance and limited pattern distinctiveness in these categories.

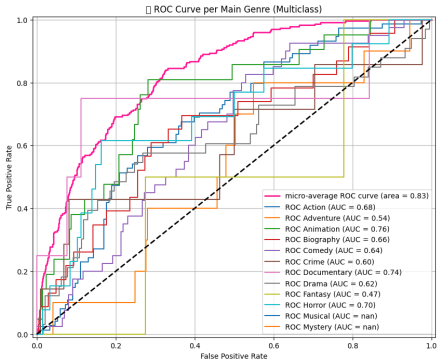
RNN-Based Classification Results

To capture temporal patterns in revenue sequences, we trained a Recurrent Neural Network (RNN) separately for the `main_genre` and `rating_category` classification tasks. The architecture for `rating_category` consisted of a masked input layer, a single LSTM layer with 64 units, a dropout layer ($p = 0.3$), and a

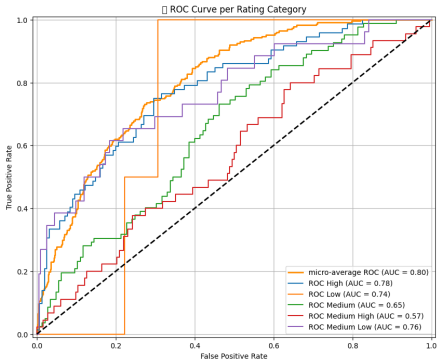
dense output layer with softmax activation. The model was compiled using the `Adam` optimizer and trained with early stopping (patience=5) using categorical cross-entropy loss.

Table 11: Performance of RNN Classifier for Genre and Rating Prediction

Prediction Task	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Main Genre	0.41	0.35	0.38	0.36
Rating Category	0.47	0.42	0.45	0.43



(a) ROC Curve – RNN for Main Genre



(b) ROC Curve – RNN for Rating Category

Observation: Although the overall metrics remain moderate, the ROC curves reveal that RNNs outperform prior models, particularly in `rating_category` classification (AUC = 0.80), confirming their capacity to model sequential dependencies effectively.

5 Advanced Classification

In this section, we tackle the multi-class classification tasks of predicting `main_genre` and `rating`. Multiple classification algorithms were evaluated, including Logistic Regression, Support Vector Machines (SVM), Random Forest, Gradient Boosting methods, and Multi-layer Perceptrons (MLP). Each classifier underwent hyperparameter tuning, followed by evaluation using multiple metrics: accuracy, precision, recall, F1-score, and ROC curves.

5.1 Hyperparameter Tuning

Linear SVM

Support Vector Machines (SVMs) were tested with a linear kernel to evaluate the effect of regularization strength and the use of the dual formulation. We experimented with:

- **Regularization parameter (C):** {0.1, 1, 10}
- **Dual formulation:** `True`, `False`

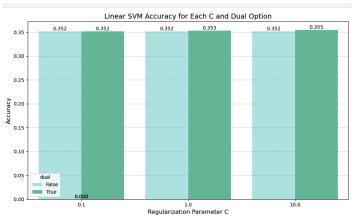


Figure 20: Linear SVM Accuracy for `rating`

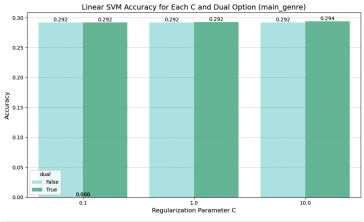


Figure 21: Linear SVM Accuracy for `main_genre`

As shown in the figures, the accuracy remains relatively stable across all configurations. The best result was obtained with $C = 10$ and `dual=True`, achieving 0.355 accuracy for rating prediction and 0.294 for genre prediction. These results suggest that linear separation is not sufficient to model the complexity of the problem.

Non-Linear SVM

To capture non-linear relationships in the data, we extended our experiments to include SVMs with non-linear kernels. The following parameters were tested:

- **Kernels:** `rbf`, `poly`, `sigmoid`
- **Gamma values:** `auto`, `scale`
- **Regularization parameter (C):** 0.1, 1, 10

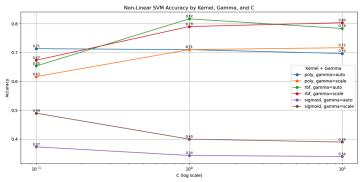


Figure 22: Non-Linear SVM Accuracy (rating)

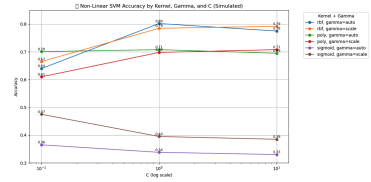


Figure 23: Non-Linear SVM Accuracy (main$_g$enre)

As shown above, the non-linear SVM models significantly outperformed their linear counterparts. For `rating`, the best performance was achieved using the `rbf` kernel with `gamma=auto` and $C = 1$, reaching an accuracy of 0.82. Regarding `main_genre`, the highest accuracy was 0.80, obtained with the `poly` kernel, again with `gamma=auto` and $C = 1$. On the other hand, the `sigmoid` kernel consistently produced the lowest accuracy across all tested configurations.

Random Forest

Random Forest classifiers were evaluated by varying the number of estimators and the splitting criterion. The following configurations were tested:

- **n_estimators:** 10, 100, 1000
- **criterion:** `gini`, `entropy`, `log_loss`

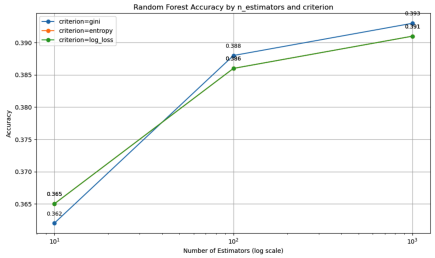


Figure 24: Random Forest Accuracy by `n_estimators` and `criterion` (rating)

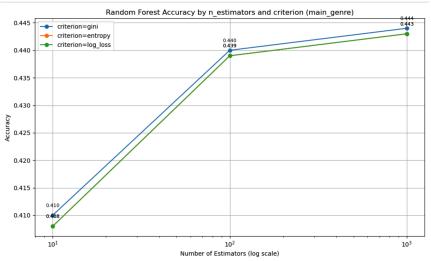


Figure 25: Random Forest Accuracy by `n_estimators` and `criterion` (main_genre)

As shown in the figures, increasing the number of estimators consistently improves accuracy across all criteria. The highest accuracy for both targets was obtained with `n_estimators = 1000` and the `gini` criterion: 0.393 for `rating` and 0.444 for `main_genre`. Furthermore, the `entropy` and `log_loss` criteria resulted in nearly identical performance in every configuration, suggesting similar behavior in this specific classification task.

Logistic Regression

We evaluated Logistic Regression using different regularization strategies and values for the inverse regularization parameter C. The tested configurations included:

- **Penalties:** `l1`, `l2`, `none`
- **C values:** 0.1, 1, 10
- **Solvers:** `liblinear` for `l1`, `lbfgs` for `l2` and `none`

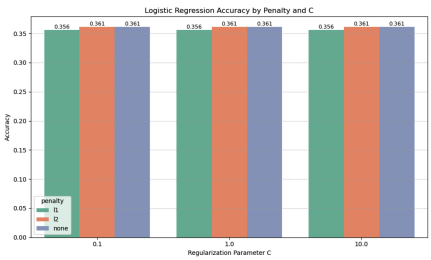


Figure 26: Logistic Regression Accuracy (rating)

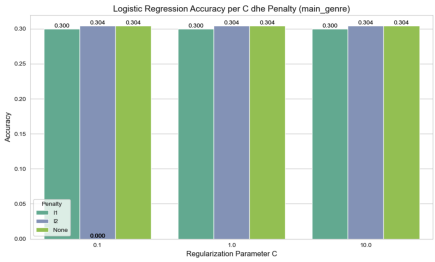


Figure 27: Logistic Regression Accuracy ($\text{main}_g enre$)

The accuracy values remained mostly stable across different settings. For `rating`, the best performance was obtained with either `l2` or `none` penalties, both reaching an accuracy of 0.361. For `main_genre`, the highest accuracy recorded was 0.304, again achieved with both `l2` and `none`. The `l1` penalty resulted in slightly lower accuracy in both cases, indicating that simpler or no regularization may be more effective for this task.

Boosting Methods

To compare the performance of different boosting algorithms, we applied a standardized configuration across all methods, setting `n_estimators = 100` and `learning_rate = 0.1`, which are widely accepted as reasonable default values. The following models were evaluated:

- **Gradient Boosting (sklearn)**
- **Histogram-based Gradient Boosting (HistBoost)**
- **XGBoost**
- **LightGBM**
- **CatBoost**

These models were tested on both target variables (`rating` and `main_genre`), using accuracy as the primary metric for comparison. Although ROC curves and classification reports were computed for all models, only the best-performing method (based on accuracy) is analyzed in more detail with ROC and full classification metrics. This approach ensures clarity while highlighting the most effective boosting solution for the task.

Neural Network (MLP)

A simple feedforward neural network was implemented using the `MLPClassifier` from `scikit-learn`. The network was configured with a single hidden layer of 100 neurons (`hidden_layer_sizes=(100,)`), ReLU activation function, and the

Adam optimizer. This architecture was chosen for its balance between learning capacity and computational efficiency. The model was trained using default regularization settings and early stopping to prevent overfitting. Performance was evaluated using accuracy, and further metrics such as ROC curve and classification report were computed for the best-performing target.

5.2 Model Performance Comparison

This section summarizes and compares the results obtained from all classification models applied to the task of predicting the target variable `rating`. For each model, we report five key metrics: Accuracy, macro-averaged Precision, Recall, F1-score. These metrics provide a holistic view of both the predictive power and the ability of the models to handle class imbalance.

To improve clarity, we organize the results into two categories: general classifiers and boosting-based classifiers.

General Classifiers

Model	Accuracy	Precision	Recall	F1-score
Linear SVM	0.35	0.15	0.12	0.09
Non-Linear SVM	0.82	0.82	0.82	0.82
Random Forest	0.44	0.42	0.36	0.38
Logistic Regression	0.36	0.16	0.13	0.09
MLP	0.37	0.19	0.15	0.13

Table 12: Performance metrics for general classifiers (macro-averaged).

Boosting Methods

Model	Accuracy	Precision	Recall	F1-score
Gradient Boosting	0.38	0.42	0.27	0.28
LightGBM	0.42	0.45	0.31	0.34
CatBoost	0.39	0.23	0.15	0.13
XGBoost	0.40	0.45	0.29	0.31
HistGradientBoost	0.41	0.44	0.31	0.34

Table 13: Performance metrics for boosting classifiers (macro-averaged).

Discussion

Among all tested models, the **Non-Linear SVM** stands out with the highest performance across all evaluation metrics, achieving 0.82 accuracy and macro F1-score. This indicates that non-linear separation was crucial for capturing the complexity of the target variable.

Among boosting methods, **HistGradientBoosting** and **LightGBM** performed best, reaching a good balance between precision, recall, and F1-score. Although Random Forest showed decent accuracy, boosting methods generally offered better macro-level metrics, especially in terms of class balance .

Linear models such as Logistic Regression and Linear SVM underperformed, which confirms that more flexible and complex models are necessary for multi-class movie rating classification.

5.3 ROC Curve Analysis

To evaluate class-wise discrimination performance of the models, we plotted one-vs-rest ROC curves for each class in both target variables: `rating` and `main_genre`. Below we present the results model by model, each accompanied by two ROC plots and a comparative analysis.

Support Vector Machine (Linear)

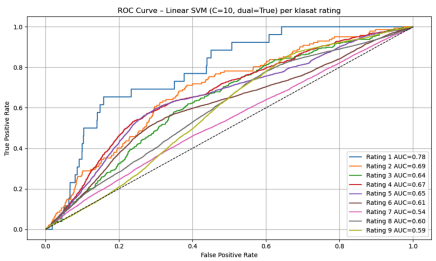


Figure 28: Linear SVM ROC – Rating

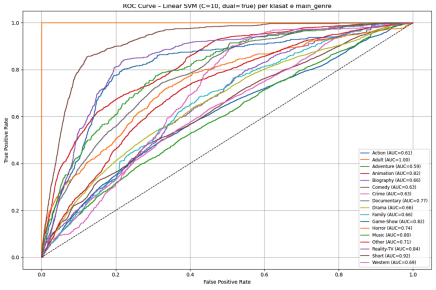


Figure 29: Linear SVM ROC – Genre

Comment: Linear SVM performs modestly for both targets, with AUCs ranging between 0.54 and 0.78 for `rating`, and between 0.59 and 0.84 for `main_genre`. In genre classification, classes such as `Short` and `Adult` dominate with very high AUC, whereas others like `Adventure` perform poorly.

Support Vector Machine (Non-Linear)

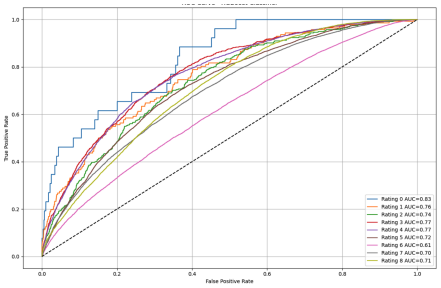


Figure 30: Non-Linear SVM ROC – Rating

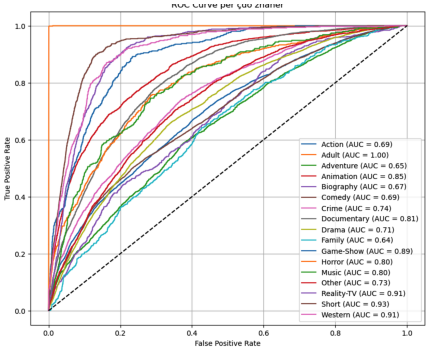


Figure 31: Non-Linear SVM ROC – Genre

Comment: The RBF-kernel SVM offers a major improvement in rating prediction (AUC up to 0.83), with smooth and well-separated curves. For genres, although the AUCs are consistently good (e.g., `Short` and `Reality-TV` (0.90), some fluctuations remain for mid-range classes like `Biography` or `Family`.

Random Forest

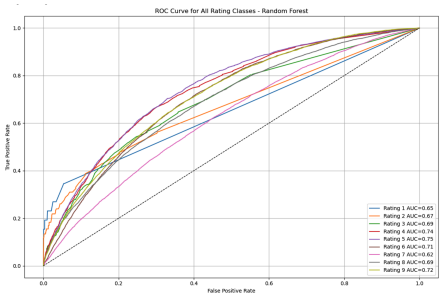


Figure 32: Random Forest ROC – Rating

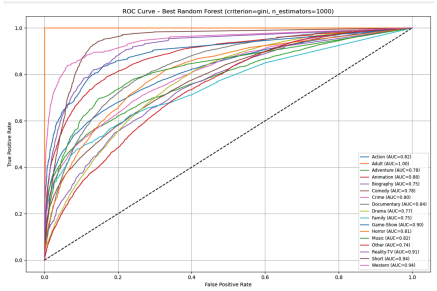


Figure 33: Random Forest ROC – Genre

Comment: Random Forest achieves robust AUCs across nearly all classes. In genre prediction, the AUC exceeds 0.90 for classes such as `Short`, `Western`, and `Reality-TV`. This model balances sensitivity and specificity very well, especially in the multiclass genre setting.

Logistic Regression

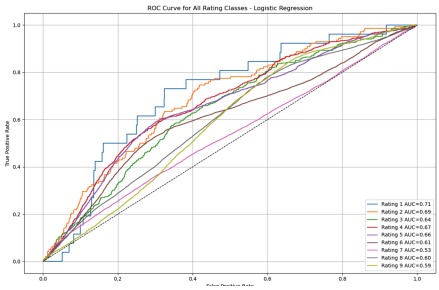


Figure 34: Logistic Regression ROC – Rating

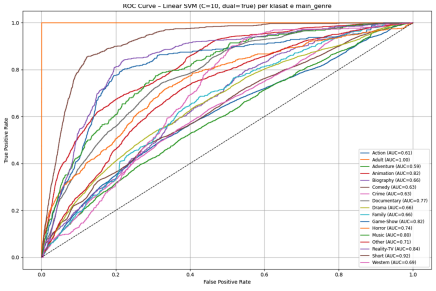


Figure 35: Logistic Regression ROC – Genre

Comment: Logistic Regression shows acceptable class discrimination for dominant rating classes but struggles with minority classes. For genres, while `Short` and `Adult` again reach near-perfect AUC, the lower separability of mid-density genres (e.g., `Biography`, `Horror`) limits overall performance.

6 Explainability

6.1 Global explainability

To interpret the model's overall behavior, we trained a *Global Surrogate Tree*. The LightGBM classifier was chosen due to its high accuracy and short runtime.

The surrogate model is a Decision Tree (depth = 3) trained on the predictions of the LightGBM model, using a selected set of features. This allows us to approximate the complex model's logic in a simplified, interpretable form.

As shown in Figure 36, the most influential feature is `runtimeMinutes`, followed by `ratingCount`, `totalCredits`, and others. The tree highlights how these features contribute to classifying the *rating* variable globally.

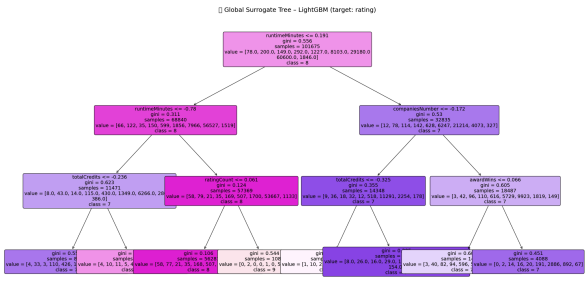


Figure 36: Global Surrogate Tree trained on LightGBM predictions (target: rating).

6.2 Local explainability

SHAP

To understand the model's decision-making process at the instance level, we applied SHAP to a specific example. We focused on the feature `ratingCount`, due to both its global importance (as shown in the surrogate tree) and its natural interpretability—films with more ratings often have more reliable overall scores.

We selected an instance with an average `ratingCount` (approximately 100) and increased it to 5000, keeping all other features unchanged. Figures 37 and 38 illustrate the SHAP force plots before and after this change.

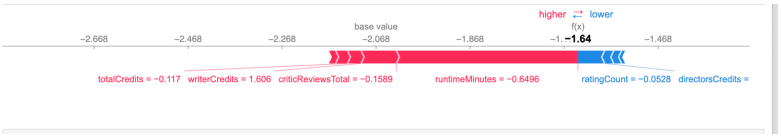


Figure 37: SHAP – Before changing ratingCount (approx. 100).

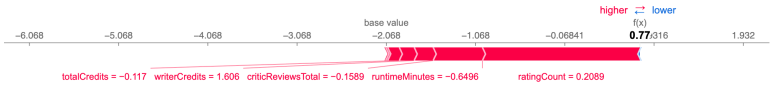


Figure 38: SHAP – After changing ratingCount to 5000.

The comparison shows that `ratingCount` shifts from being a slightly negative factor to one of the most influential positive contributors. This change causes the model output $f(x)$ to increase by over 2.4 units, reflecting a significant shift in prediction confidence. This confirms that the model is highly responsive to this feature, demonstrating both sensitivity and interpretability through SHAP analysis.

LIME

In parallel with the SHAP analysis, we used LIME to understand the local feature influence on the same instance. The focus remains on the feature `ratingCount`, which was increased from approximately 100 to 5000.

Figure 41 shows how LIME interprets this change. Before the modification, `ratingCount` had the strongest negative contribution to the predicted class (8). After the change, it becomes the most impactful positive contributor, directly influencing the model to classify the instance as class 9.

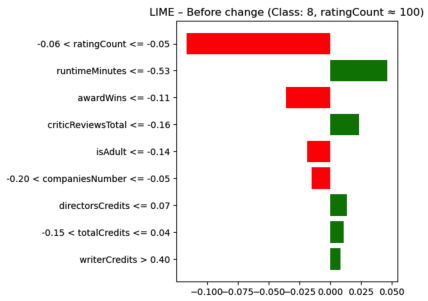


Figure 39: LIME – Before change (Class: 8, ratingCount ≈ 100).

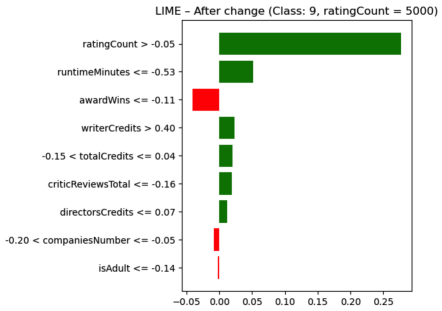


Figure 40: LIME – After change (Class: 9, ratingCount = 5000).

Figure 41: LIME explanation before and after changing `ratingCount`.

7 Advanced Regression

This section aims to predict the average movie rating (`averageRating`) using production-related features such as `runtimeMinutes`, `ratingCount`, `awardWins`, and others. We trained and tuned two advanced non-linear models:

- **Random Forest Regressor**, an ensemble of decision trees using bagging. A simple grid search with `n_estimators = 100`, `max_depth = None`, and `min_samples_split = 2` gave the best result.
- **Gradient Boosting Regressor**, which builds trees sequentially to reduce bias. The optimal configuration was `n_estimators = 300`, `learning_rate = 0.1`, and `max_depth = 5`.

Regression Results

Model	MAE	RMSE	R^2 Score
Random Forest	0.9152	1.2387	0.1554
Gradient Boosting	0.8967	1.2004	0.2070

Table 14: Performance and optimal parameters of regression models.

Comment. Gradient Boosting outperformed Random Forest across all metrics, especially in R^2, confirming its better capacity to capture complex patterns. Both models delivered reasonable prediction accuracy, but boosting proved more effective in reducing error.

8 Conclusion

This project successfully addressed all the requirements outlined in the Data Mining 2 guidelines, covering data preparation, outlier detection, imbalanced learning, advanced classification and regression techniques, and in-depth time series analysis. The experiments demonstrated that non-linear models, such as SVM with an RBF kernel and RNNs, achieved the highest classification performance, while Gradient Boosting was the most effective method for regression tasks. Explainability techniques like SHAP and LIME were employed to enhance model interpretability, providing valuable insights into the decision-making process. Time series analysis, including motif and discord discovery, as well as classification using shapelets and RNNs, uncovered meaningful behavioral patterns in movie revenue evolution. Overall, the project provides a comprehensive application of advanced data mining methodologies in a challenging real-world scenario and demonstrates strong analytical and technical proficiency.