Data Mining 2 Project

Lorenzo Albani, Luigi Ascione, Tommaso Maitino

UNIVERSITÀ DI PISA

June 25, 2025

Contents

1 Introduction 3

2 Data Understanding and Preparation 3
2.1 Tabular Dataset . . . 3

3 Outliers 3
3.1 Extended Isolation Forest . . . 4
3.2 Handling outliers . . . 5

4 Imbalanced Learning 5
4.1 Undersampling . . . 6
4.2 Oversampling . . . 6
4.3 Combined Approach . . . 6
4.4 Balancing at the Algorithm Level . . . 7

5 Advanced Classification 7
5.1 Logistic Regression . . . 7
5.2 Support Vector Machines . . . 8
5.2.1 Linear SVM . . . 9
5.2.2 Non-Linear SVM . . . 9
5.3 Neural Networks . . . 10
5.4 Ensemble Methods . . . 13
5.5 Gradient Boosting Machines . . . 15
5.6 Advanced Classification Results . . . 16

6 Advanced Regression 17
6.1 Random Forest Regressor . . . 17
6.2 XGBoost Regressor . . . 18
6.3 Advanced Regression Results . . . 18

7 Explainability 18
7.1 SHAP . . . 18
7.2 LORE . . . 20

8 Time Series 21
8.1 Preparation . . . 21
8.2 Motifs/Discords . . . 22
8.3 Clustering . . . 23
8.3.1 K-means . . . 24
8.3.2 Hierarchical Clustering . . . 25
8.3.3 Conclusions . . . 25
8.4 Classification . . . 27
8.4.1 Instance-based models . . . 27
8.4.2 Shapelets-based model . . . 27
8.4.3 Other methods . . . 29
8.4.4 Final considerations . . . 29

1 Introduction

This paper presents an advanced data mining analysis of an IMDb dataset containing information on movies, TV shows, and other visual content.

2 Data Understanding and Preparation

2.1 Tabular Dataset

The IMDb dataset compiles detailed information about movies, TV shows, and other visual entertainment formats, including community-generated ratings. Each entry contains essential details like the original title, release year, runtime, and the total number of user votes. The dataset also offers insights into significant factors such as awards, nominations, user reviews, and statistical ratings—ranging from top to lowest ratings, along with critic review counts. Additionally, it includes metadata such as the country of origin, genre, the number of images and videos associated with each title, castNumber, CompaniesNumber, externalLink, quotesTotal, writerCredits, directorCredits and soundMixes.
From the general statistics, it can be seen that the mean for `awardWins` is 0.31, while the median is 0. This highlights that at least 75% of units have never won an award, as indicated by the value of the third quartile. This attribute has a low standard deviation, meaning that the data are clustered near the mean. For `numVotes`, the average is much higher than both the median and the third quartile, suggesting that many titles have a low number of votes, while a few have a significantly high number of ratings. In this case, there is substantial variability, with the data distributed far from the mean.
Regarding `totalCredits`, the mean is somewhat higher than the median, with 75% of the titles having at least 64 credits. The high standard deviation value indicates that the data points are widely dispersed from the mean.
For the total count of critic and user reviews, it can be seen that, as before, the median is 0, while the mean is higher, meaning that at least 50% of samples have no reviews. On average, user reviews are more numerous than critic reviews. The statistics for `ratingCount` are very similar to those for `numVotes`, with only a few data points differing between the two variables. For the following new attributes—`externalLinks`, `writerCredits`, `directorsCredits`, and `quotesTotal`—the maximum values are significantly higher than the third quartile.
Due to the high frequency and to the range of the data the distribution is not clear enough, the data needs to be transformed for a better analysis.
For data quality, the variable `Endyear` has 96% missing values and has therefore been removed; for runtime minutes "NA" it's filled with 0 and for `countryofOrigin` it's substitute with $'/N'$, so a category of country of origin not classificated.
To make the distributions more homogeneous and reduce the impact of outliers without losing too much information, a logarithmic transformation is applied to some attributes. A small constant is added to all values to handle cases where the value is 0. The attributes transformed, for the above reasons, are respectively: `numVotes`, `awardWins`, `AwNmExWins`, `criticReviewsTotal`, `userReviewsTotal`, `totalImages`, `totalVideos`, `castNumber`, `companiesNumber`, `externalLinks`, `writerCredits`, `directorsCredits`, `quotesTotal`.
After calculating the correlation matrix it's observed that the correlation between the variables `ratingCount` and `numVotes` is 1, and 88% of the rows are identical, so `ratingCount` has been removed. Regarding other variables, there is a positive correlation of 0.83 between `castNumber` and `totalCredits`, so if the first variable increases the other one increases in average. Then there is a strong positive connection between `externalLinks` with `CriticReviewsTotal` (0.92) and `userReviewsTotal` with `numVotes` (0.80).

3 Outliers

In this section, different types of anomaly detection methods are applied. In particular, **HBOS** (Histogram-based outlier), **LOF** (Density-based approach), **CBLOF** (Clustering-based approach),

Extended Isolation Forest (Model-based approach), and **kNN distances** (Distance-based approach) are used (Figure 1).
For all methods, only numeric variables are selected, data are standardized through the `StandardScaler()` function, and then outlier detection is performed. Specifically for the last method, due to the high number of instances, the dimensionality of the dataset is reduced using PCA (5 components), and then the distance-based approach is applied.
After calculating outlier scores for each method, these scores are standardized in turn. Therefore, values with units greater than a threshold of 3 can be classified as outliers due to their very high outlier scores.

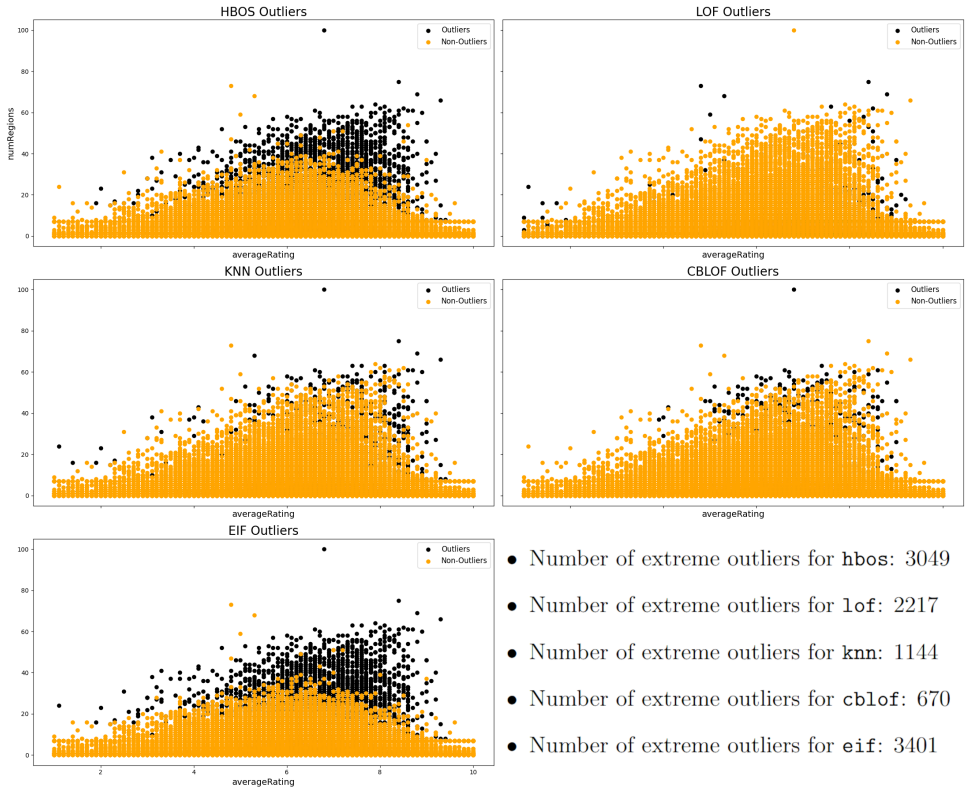


Figure 1: HBOS, LOF, KNN, CBLOF, EIF outliers

Due to the limitations of two-dimensional visualization, selected outliers are not always clearly discernible in the plots. Nevertheless, similarities emerge between HBOS and EIF methods, as well as between KNN and CBLOF, with EIF exhibiting the highest outlier detection rate. Methods are compared, and points classified as outliers by at least three techniques are designated as general outliers. For a better visualization of the multidimensional dataset, a PCA in two components is performed and the anomalies are highlighted in black in Figure 2.

3.1 Extended Isolation Forest

Specifically, the use of the **Extended Isolation Forest** is explored in depth. The following parameters are used for the model: `ntrees = 200`, `sample_size = 256`, `ExtensionLevel = 1`. A compromise is sought between result accuracy and computation time. The graph in Figure 3a shows a radial visualization of the first 100 trees generated by the Extended Isolation For-

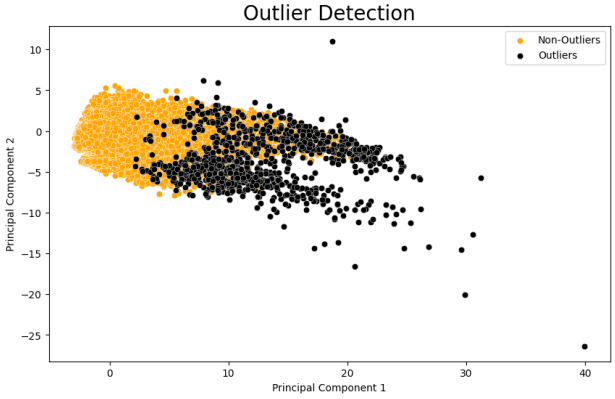
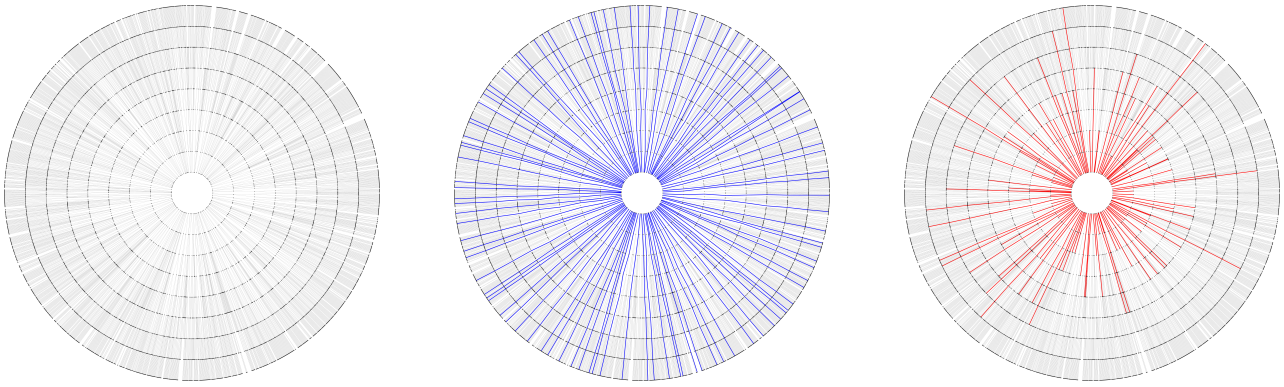


Figure 2: Outliers detected visualized after PCA.

est model. Each spoke represents a branch (path) of a tree, and the concentric circles indicate different node depth levels. It provides a general representation of the structure of the model's trees.
In Figure 3b, the paths taken by a normal instance (non-anomaly) are highlighted in blue. The lines extend much farther outward, indicating deeper traversal paths: the non-anomaly is isolated only after more steps, as expected for a "normal" point well embedded in the data distribution.
In Figure 3c, the paths taken by an anomaly within each tree are highlighted in red. The lines stop quickly (near the center), indicating that the instance is isolated in just a few steps: a typical characteristic of anomalies, which tend to reside in sparse and isolated regions of the dataset.
This graphical comparison illustrates the model's discriminative behavior.



(a) (b) (c)

Figure 3: (a) Plot of the first 100 trees of Isolation Forest. (b) Overplot of the paths for one non-anomaly. (c) Overplot of the paths for one anomaly.

3.2 Handling outliers

It's chosen to eliminate a total of 907 records given by the usage of the detection methods. This step is crucial to ensure the integrity and accuracy of the dataset. By removing these records, we mitigate the risk of false positives and negatives, which could otherwise skew the results of our analysis. The combination of multiple detection methods provides a robust mechanism to identify and exclude anomalous data points.

4 Imbalanced Learning

Imbalanced learning techniques have been applied to a new variable, `has_award`. This variable is created from the `awardWins` variable, distinguishing between titles that have won at least one award and those that have not won any. Subsequently, the `awardWins` variable was removed to eliminate correlations with the newly created variable. Finally, the data are normalized using the `MinMaxScaler()` function. The class distribution is shown in Figure 4.

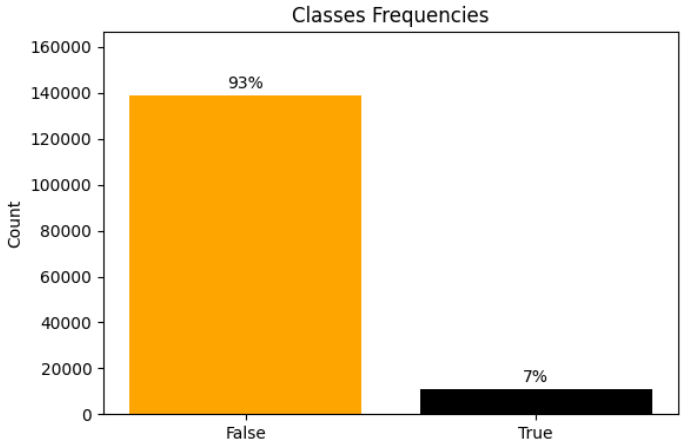
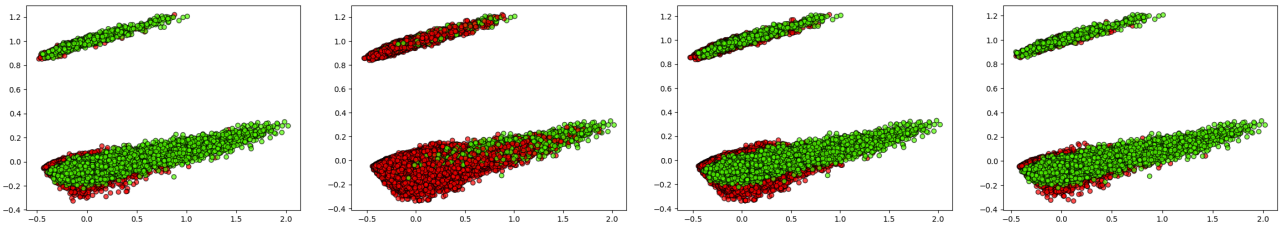


Figure 4: `has_award` classes frequencies. class 0: 95885 records, class 1: 8151 records.

4.1 Undersampling

Undersampling is performed using different techniques, such as **Random Undersampling**, **Tomek Links**, **Edited Nearest Neighbors**, **Cluster Centroids**.
In Figure 5 the distribution of the two classes for each transformation can be seen graphically using PCA.



(a) RandomUndersampl. (b) Tomek Links. (c) EditedNN. (d) Cluster Centroids.

Figure 5: PCA visualization on two principal components of the two classes of the target variable `has_award`, following the application of imbalanced learning techniques.

A decision tree classifier is used with `min_samples_leaf=3` to evaluate the variations in results caused by undersampling.
For Random Undersampling and Cluster Centroids, in both cases the undersampling threshold is set to the number of records in the minority class. Neither method led to an improvement in performance metrics.
As for Tomek Links, no noteworthy results are observed, also because only 1.85% of records are removed from the majority class. As for the Edited Nearest Neighbours method, slightly better results were observed for the minority class. Both of these methods will be further investigated in Section 4.3.

4.2 Oversampling

Various techniques are used also for the oversampling, such as **Random Oversampling**, **SMOTE**, **ADASYN**.
In this case, the results were very similar for all three methods, with minimal differences in all cases between the initial metrics and the post-transformation metrics.

4.3 Combined Approach

After individually testing various transformations, a combination is chosen to attempt to achieve better results. This combination consists of applying **ADASYN** to oversample the minority class, followed by **Tomek Links** to remove borderline samples from the majority class while retaining the most informative ones, and finally, **Edited Nearest Neighbours** to clean noise at the boundary between the two classes. This sequence led to a slight improvement in performance. The metrics before and after the transformations are reported in Table 1. After the transformations the number of records for each class is 95510 for class 1 and 77960 for class 0.

	Precision		**Recall**		**F1-score**	
	before	**after**	**before**	**after**	**before**	**after**
0	0.95	0.97	0.96	0.90	0.95	0.93
1	0.46	0.36	0.43	0.70	0.45	0.48
Accuracy					0.92	0.88
Macro Avg	0.71	0.67	0.70	0.80	0.70	0.71
Weighted Avg	0.91	0.92	0.92	0.88	0.91	0.90

Table 1: The table shows the resulting metrics from a decision tree classifier before and after imbalanced learning transformations.

A slight decrease in the f1-score of the majority class and a slight improvement in the f1-score of the minority class can be observed. Moreover, there is a decrease in the overall accuracy of the model. However, by analyzing the ROC curves before and after the transformations, reported in Figure 6, a clear improvement can be noticed. The model, by beginning to predict the minority class as well,

experiences a slight decrease in overall accuracy due to an increase in errors on the majority class. However, it achieves a notable improvement in ROC AUC score, reflecting a better ability to distinguish between the two classes, along with a significant increase in the recall of the minority class.

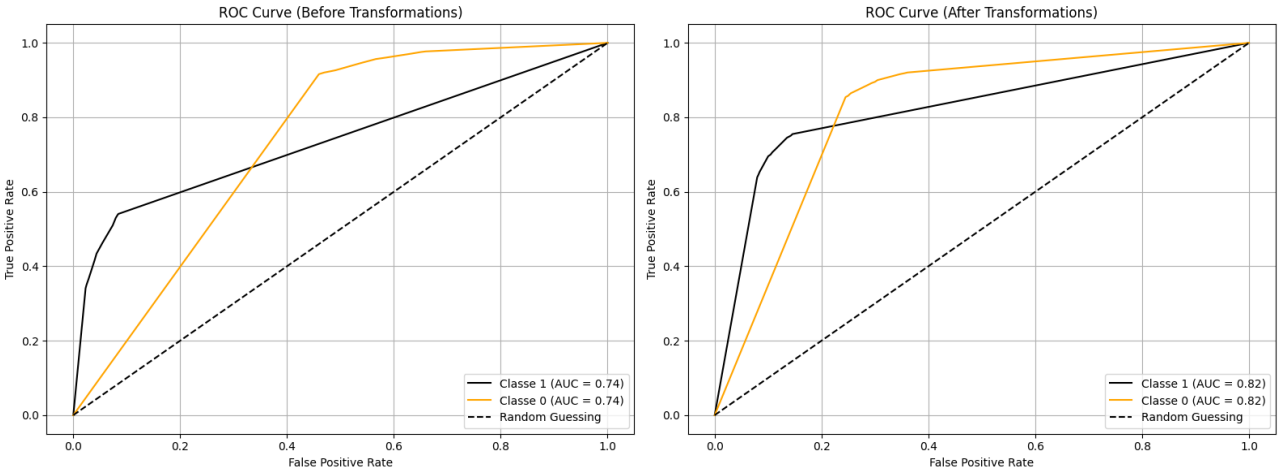


Figure 6: `has_award` decision tree ROC curves, before (left) and after (right) combined approach of rebalancing techniques.

4.4 Balancing at the Algorithm Level

Finally, an attempt is made to balance the class weights directly at the algorithm level. In this case as well, a decision tree classifier is used. Through a grid search, various class weights and different values for `min_samples_leaf` are tested. The best parameters configuration is: `{'class_weight': {0: 3, 1: 1}, 'min_samples_leaf': 5}`. However, the results are disappointing, as the classifier did not significantly improve the prediction of the minority class.

5 Advanced Classification

For classification, Logistic Regression, Support Vector Machine, Neural Networks, Ensemble Methods, and Gradient Boosting Machine are implemented. To maintain continuity with the Data Mining 1 analysis, the target variables used are `rating` and `titleType` remapped as [TV Series, Videogame, Movie, Shorts, TV Special, Video]. For all classifiers, all numerical variables are used; however, in the classification of the `rating` variable, `averageRating` was excluded, as `rating` is a discretized version of `averageRating` and its inclusion would have made the analysis less meaningful. Before building each model, data are divided in train (70%) and test (30%), and the train has been standardized.

5.1 Logistic Regression

Rating
To build the first model, the optimal regularization term (C) is chosen between the values `0.001, 0.01, 0.1, 1, 10, 100`. This is achieved through 5-fold cross-validation combined with grid search. The grid search returns `C = 1` as the best hyperparameter, a moderate regularization showing an accuracy of 35%. Table 2 indicates that the model does not perform well: many classes are not classified correctly, while others have slightly higher values. In particular, the class `(7,8]`, the one with the larger support, has the highest value in precision, but still low, a high recall value of 0.78 and a f1-score of 0.50. Therefore, the model is good at finding the true examples of this class, but it is not very precise in doing so, which suggests there are many false positives. The ROC curve in Figure 7 shows for each class that the model has a better capacity of distinguishing and classifying than to random guessing, with an AUC score of 0.70.

	precision	recall	f1-score	support
(0, 1]	0.00	0.00	0.00	26
(1, 2]	0.00	0.00	0.00	142
(2, 3]	0.00	0.00	0.00	355
(3, 4]	0.00	0.00	0.00	1033
(4, 5]	0.10	0.00	0.00	2726
(5, 6]	0.31	0.07	0.11	6398
(6, 7]	0.31	0.34	0.32	11709
(7, 8]	0.37	0.78	0.50	14567
(8, 9]	0.24	0.00	0.00	6620
(9, 10]	0.00	0.00	0.00	1284
accuracy			0.35	
mac avg	0.13	0.12	0.09	44860
weig avg	0.29	0.35	0.26	44860

Table 2: Performance metrics of the logistic regressor with C = 1, `rating`.

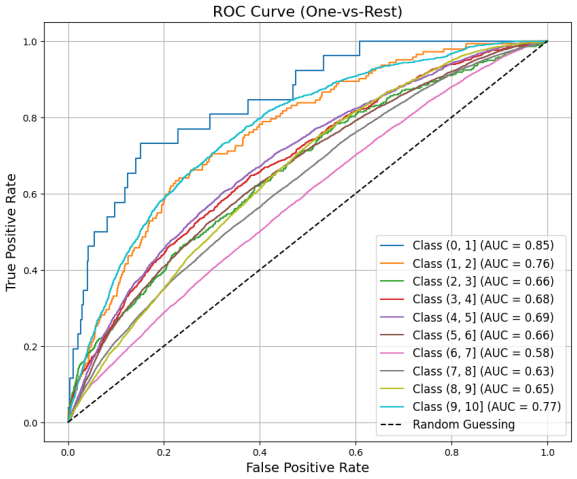


Figure 7: ROC Curve of the Logistic Regression, `rating`, AUC Score = 0.70.

TitleType

For `titleType`, as for `rating` variable, to find the best regularization term, a 5-fold cross-validation is done. Also in this case the best C is 1. The model has high accuracy, but it can only classify some examples correctly. None of the `TV Special` examples are classified correctly. The `Video` and `Videogame` classes have low recall, meaning the model is not recognizing many instances of these classes. Meanwhile, the logistic regression is recognizing `Movies` and `TV Series` well. The ROC curve shows that the model has a very high capability to distinguish between classes, precision-recall curve shows low values for the `titleType` mentioned earlier, not correctly classified.

5.2 Support Vector Machines

As with Logistic Regression, the **SVM classifier** is applied to the features `rating` and `titleType`. For a better visualization of the class distribution in the data, these are transformed using PCA into two components, reported in Figure 8 and 9.

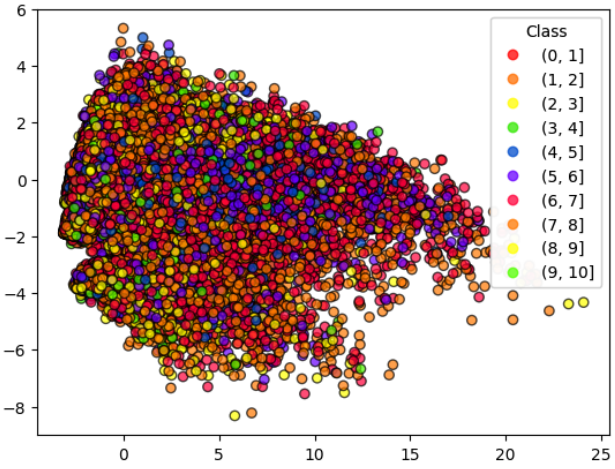


Figure 8: PCA transformed dataset highlighting `rating` classes distribution.

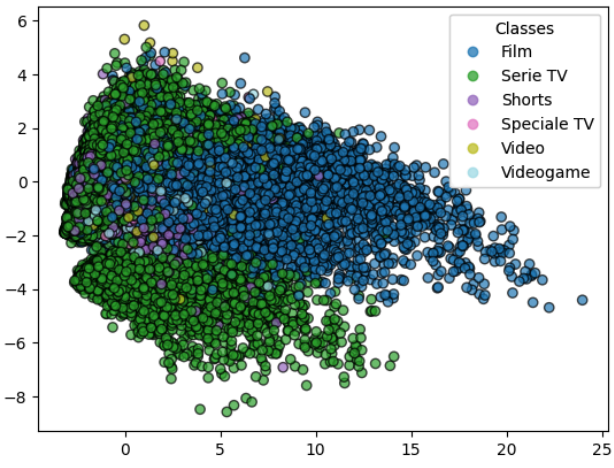


Figure 9: PCA transformed dataset highlighting `titleType` classes distribution.

It can be observed that the data for the `rating` class are more mixed, while the observations for the `titleType` class form different clusters and seem to be non-linearly separable.

5.2.1 Linear SVM

For the **Linear SVM**, a 5-fold cross-validation (`n_splits=5`) was performed within a Grid Search exploring the values `[0.001, 0.01, 0.1, 1, 10, 100]` for the hyperparameter `C`, ranging from low to high regularization terms. The value that generate the higest accuracy on test set is `{'C': 0.001}`, a low penalization, so the model allows a larger margin of errors.
The results are the same as those of the logistic regression, with the same accuracy and f1-score for the classes. Therefore, the two classifiers built very similar models, but with different regularization.

5.2.2 Non-Linear SVM

For the training of non-linear SVMs, due to the long duration of k-fold grid search for parameter tuning, a different approach is tried:

1. the dataset is divided into training and test sets with stratification of the target variable and the training set is standardized;
2. the training set is randomly split ten times into stratified subsamples, each containing 10% of the data;
3. a holdout grid search is performed on each subsample, returning the best hyperparameters;
4. ten models are obtained, and the overall best hyperparameters are calculated as the median of all values;
5. the model is trained on the entire training set using the best hyperparameters and tested on the test set.

With this method, different models are trained on small parts of the data in a faster time, achieving the same results. This process is applied to various non-linear SVM models, such as **radial SVM**, **sigmoid SVM** and **polynomial SVM**.

	rbf			**sigmoid**			**poly**			
	precision	**recall**	**f1-score**	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**	**support**
(0, 1]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	26
(1, 2]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	142
(2, 3]	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.09	0.10	355
(3, 4]	0.41	0.02	0.04	0.00	0.00	0.00	0.00	0.00	0.00	1033
(4, 5]	0.36	0.12	0.17	0.00	0.00	0.00	0.22	0.04	0.06	2726
(5, 6]	0.35	0.19	0.25	0.22	0.04	0.06	0.27	0.07	0.12	6398
(6, 7]	0.36	0.35	0.35	0.34	0.28	0.31	0.33	0.28	0.31	11709
(7, 8]	0.41	0.76	0.53	0.37	0.87	0.52	0.38	0.83	0.52	14567
(8, 9]	0.34	0.07	0.12	0.00	0.00	0.00	0.00	0.00	0.00	6620
(9, 10]	0.71	0.07	0.13	0.00	0.00	0.00	0.00	0.00	0.00	1284
accuracy			0.39			0.36			0.36	
mac. avg	0.29	0.16	0.16	0.09	0.12	0.09	0.13	0.13	0.11	44860
weig. avg	0.38	0.39	0.33	0.24	0.36	0.26	0.26	0.36	0.27	44860
C		1			10			10		
Gamma		0.1			0.001			1		
coef0		-			-0.9			0		
Degree		-			-			2		

Table 3: Performance metrics of SVM for different kernel functions, `rating`.

From the results reported in Table 3, it is evident that the RBF model has the highest values for precision, recall, and f1-score overall. However, the RBF model fails to classify any instances in the first three classes. Only the SVM with polynomial functions can correctly classify few instances in the `(2, 3]` class. All three models have relatively high f1-scores for the `(7, 8]` class. This is likely due to the high support for this class, which helps the models classify it correctly.
The best hyperparameters found show that the sigmoid kernel has a high penalization term, allowing less error, the polynomial kernel has a relatively high penalization term, and the RBF kernel has a moderate one. Gamma controls the influence of every training example, leading to more complex decisions. The polynomial kernel has a high gamma value, the RBF kernel has a moderate one, and

the sigmoid kernel has a very low one. Therefore, considering accuracy and the three indicators, RBF is the best model for this target label.
The ROC curve and precision-recall curve are shown respectively in Figure 10 and Figure 11.
Looking at the two curves, it can be observed that all classes have a not so high AUC score and low values on precision-recall curve. Only class `(7,8]` has a slightly high AP score. This means that the class stands out in terms of average precision, indicating its superior performance in precision-recall metrics.

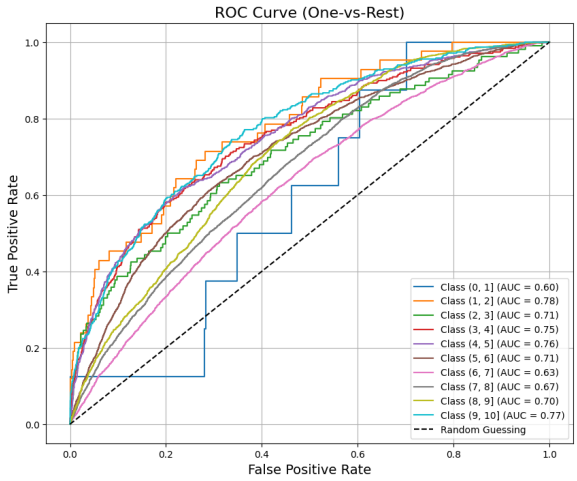


Figure 10: ROC curve of SVM classifier with RBF kernel. AUC = 0.71.

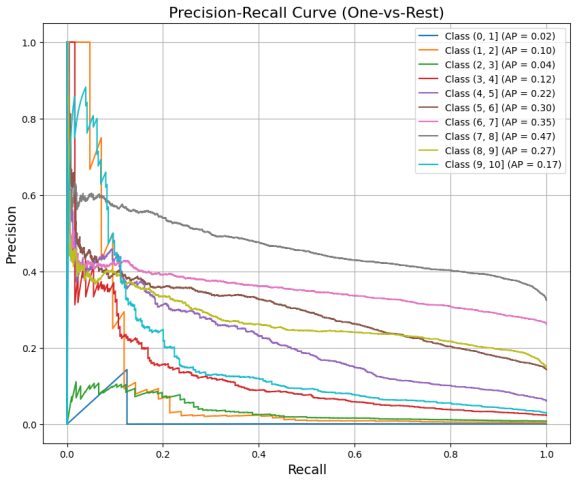


Figure 11: Precision-recall curve of SVM classifier with RBF kernel.

TitleType
For the target `titleType`, a k-fold cross-validation grid search is performed using the same hyperparameters as the previous model. With a C value of 1, a model is built with an accuracy of 82%, but only half of the classes are well-classified with a high f1-score. `TV special`, `Video`, and `Videogame` have low values in recall and f1-score.
For the non-linear SVM, the same approach used for the `rating` variable is applied. Three models are generated, each with distinct characteristics: the polynomial model correctly classifies only the `TV series` class; the sigmoid model performs very similarly to the linear one but achieves slightly better results across all three metrics; the RBF model stands out as the best, with an accuracy of 92% and high F1-scores for all classes except `TV special` (0.21) and `Video` (0.53). Regarding the best hyperparameters, the sigmoid model uses a high regularization term of 100, the polynomial model uses 10, and the RBF model uses 36.

5.3 Neural Networks

Regarding neural networks, the training set (70% of the total) is first normalized using `StandardScaler`.

Rating
Initially, the **MLPClassifier** function from `scikit-learn` IS used to build and train the neural network model. To prevent overfitting, the early stopping technique is employed, with the stopping criterion set to halt training if the accuracy on the validation set does not improve by more than 10^{-4} for 20 consecutive epochs. A hyperparameter tuning process is carried out using `RandomizedSearchCV`, with the following search space defined for the model parameters:
`{'hidden_layer_sizes': [(256, 128, 64), (128, 64, 32), (64, 32), (16,), (8,), (16, 8), (32, 16)], 'alpha': [0.1, 0.01, 0.001], 'learning_rate_init': [0.01, 0.001, 0.0001], 'activation': ['tanh', 'relu']}`.
The best combination of hyperparameters found is:
`{'learning_rate_init': 0.001, 'hidden_layer_sizes': (128, 64, 32), 'alpha': 0.01, 'activation': 'relu'}`.

Using the best hyperparameters obtained from the randomized search, the model did not achieve better results compared to the previous classifiers.
Thus, from this point onward, PyTorch is used. Label encoding is applied to the target variable to make it compatible with PyTorch, as the target needs to be a tensor of integers. All the models tested are initialized with `epochs = 300` and `early stopping = 30`. Three different neural network models are implemented:

- **Deep MLP:** this architecture includes three hidden layers with ReLU activation functions between them. A `Dropout(0.3)` layer was applied after the second hidden layer to help prevent overfitting.
- **MLP with Batch Normalization:** in this version, a `BatchNorm1d` layer is applied after each hidden layer to normalize the outputs and promote faster and more stable convergence. As in the previous model, ReLU activations are used between layers to introduce non-linearity.
- **Residual MLP:** this architecture incorporates two residual blocks, each composed of two hidden layers followed by `BatchNorm1d`. The use of residual connections allows the gradient to skip certain layers and flow more easily through the network. This helps mitigate the vanishing gradient problem and results in a more stable training process. Also in this case ReLU activations are used between every layer. The Residual MLP neural network structure is graphically represented in Figure 12:

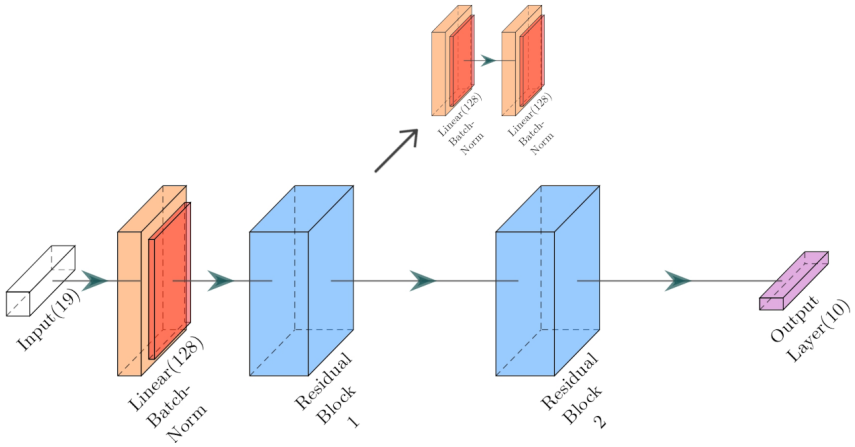


Figure 12: Residual MLP structure

Among the three models, the **Deep MLP** achieved the best overall performance, outperforming the classifiers discuss. Table 4 reports the results of the Deep MLP classifier, while Figure 13 shows the related ROC curve.

	precision	recall	f1-score	support
(0, 1]	0.00	0.00	0.00	26
(1, 2]	0.60	0.11	0.18	142
(2, 3]	0.11	0.01	0.03	355
(3, 4]	0.29	0.05	0.09	1033
(4, 5]	0.38	0.15	0.22	2726
(5, 6]	0.37	0.23	0.29	6398
(6, 7]	0.35	0.45	0.40	11709
(7, 8]	0.44	0.70	0.54	14567
(8, 9]	0.42	0.10	0.16	6620
(9, 10]	0.46	0.09	0.14	1284
accuracy			0.41	
mac avg	0.34	0.19	0.21	44860
weig avg	0.40	0.41	0.37	44860

Table 4: Performance metrics of Deep MLP for the target variable `rating`.

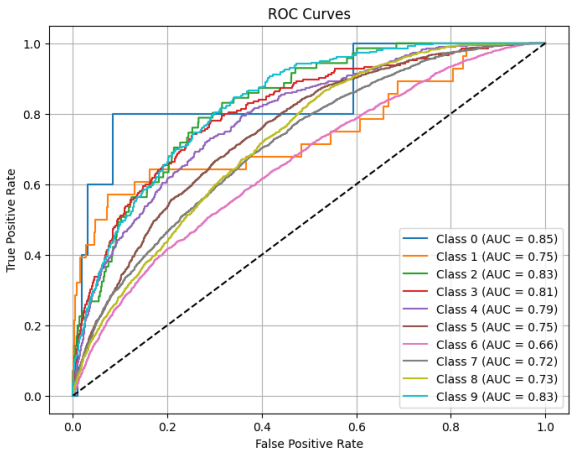


Figure 13: ROC curve of the Deep MLP classifier for `rating`. AUC Score = 0.77.

Compared to the classifiers described in the previous sections, the model demonstrates a good ability to distinguish the different classes. Even for a class with low support such as `(1, 2]`, we have a decent f1-score when compared to the other classes and their respective support. Moreover, looking at Figure 13, we notice an excellent AUC Score for this model.
Focusing specifically on the model, the graph showing the training and validation loss can be observed in Figure 14.

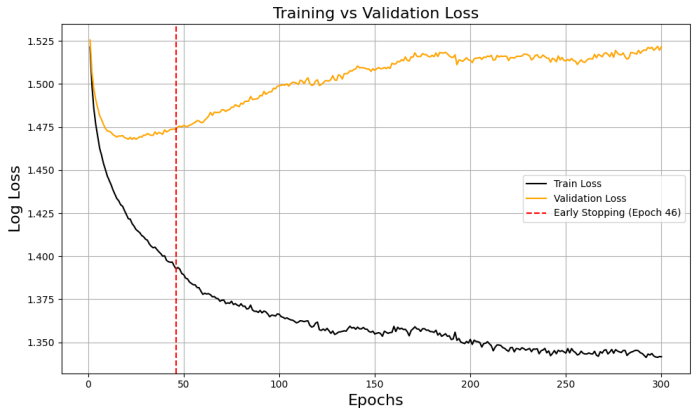


Figure 14: Training and Validation Loss function with Early Stopping, Deep MLP

Early stopping stops the training at epoch 46, which means that the model, between epoch 46 and epoch 76, did not achieve any significant improvement in accuracy. As expected, after epoch 46, a progressive increase in validation loss and a corresponding decrease in training loss are observed, indicating the onset of overfitting.
Batch Normalization MLP performance appears slightly more balanced, but it worsens on the majority classes compared to the Deep MLP. It's possible that batch normalization slightly improves the model's generalization, but not enough to outperform the base model.
Residual MLP underperforms, probably because it is too complex or poorly optimized for this specific problem. However, it stands out as the best performer so far in predicting the `(3, 4]` rating class, that frequently yielded an f1-score close to zero.

TitleType
The same neural network architectures used for the `rating` variable are also applied to the `titleType` variable. Specifically:

- **MLPClassifier**: this model achieves the best performance in predicting the `titleType` target variable. It is able to correctly distinguish most classes, with the exception of `TV Special`, which once again proves difficult to differentiate from the others across all classifiers tested. The model's evaluation metrics are presented in Table 5.

	precision	**recall**	**f1-score**	**support**
Movies	0.91	0.93	0.92	12947
Shorts	0.96	0.98	0.97	4994
TV Series	0.88	0.90	0.89	24496
TV Special	0.47	0.16	0.24	345
Video	0.75	0.44	0.56	1545
Videogame	0.78	0.80	0.79	533
accuracy			0.93	
mac. avg	0.79	0.70	0.73	44860
weig. avg	0.92	0.93	0.92	44860

Table 5: MLPClassifier Performance metrics `titleType`

- **Deep MLP**: this model demonstrates predictive capabilities very similar to those of the MLP-Classifier, but struggled even more with identifying the `TV Special` class.
- **Batch Normalization MLP**: delivers strong performance, comparable to that of the MLP-Classifier and Deep MLP, though slightly less accurate overall.
- **Residual MLP**: once again, this model shows the weakest performance among all neural network variants evaluated.

5.4 Ensemble Methods

Rating

For the ensemble classifier implementation, a Random Forest is used. For this specific task, the categorical variable `titleType` is added to the other ones, one-hot encoded. Hyperparameter tuning is performed using a randomized search with cross-validation on `max_depth`, `min_samples_split`, and `min_samples_leaf`, identifying the optimal configuration as `max_depth`=None, `min_samples_split`=2, and `min_samples_leaf`=1. Each tree employs a feature subset sized at the square root of the total, built via BAGGING, with the final forest comprising 300 trees (`n_estimators`). Performance metrics are summarized in Table 6, while Figure 15 shows the relative ROC curve and Figure 16 the precision-recall curve.

Class	Precision	Recall	f1-score	Support
(0, 1]	0.60	0.23	0.33	26
(1, 2]	0.62	0.13	0.21	141
(2, 3]	0.34	0.04	0.07	353
(3, 4]	0.35	0.06	0.10	1027
(4, 5]	0.35	0.14	0.20	2719
(5, 6]	0.37	0.30	0.33	6372
(6, 7]	0.40	0.46	0.43	11643
(7, 8]	0.48	0.67	0.56	14463
(8, 9]	0.44	0.25	0.31	6563
(9, 10]	0.61	0.18	0.28	1281
accuracy			0.43	44588
macro avg	0.46	0.25	0.28	44588
weighted avg	0.43	0.43	0.41	44588

Table 6: Classification metrics of the Random Forest classifier, Rating.

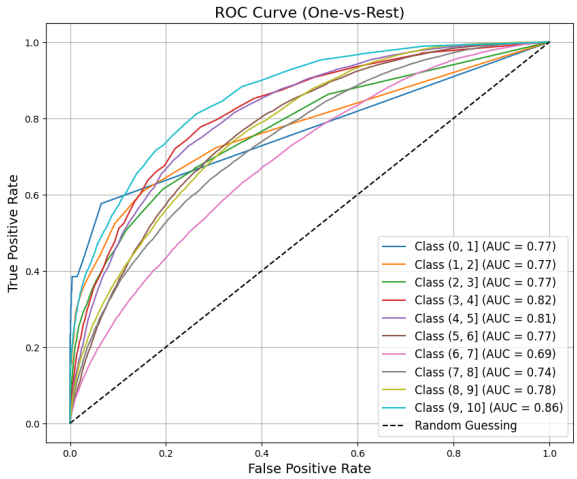


Figure 15: ROC curve of the Random Forest classifier for the target variable `rating`. AUC = 0.78

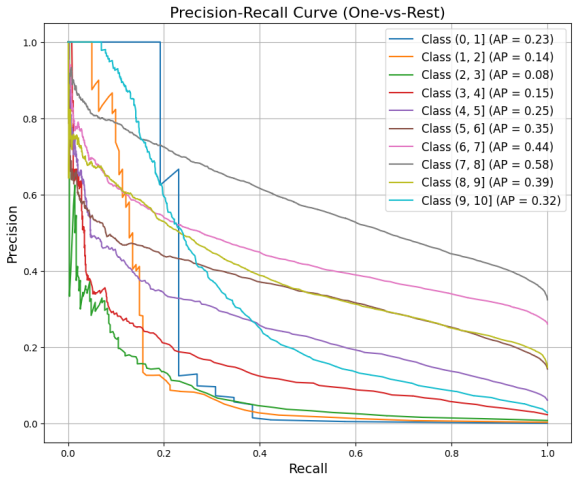


Figure 16: Precision-recall curve of the Random Forest classifier for the target variable `rating`.

A global feature importance analysis is done, integrating two methodologies: Mean Decrease in Impurity (MDI) to quantify average impurity reduction in splits, shown in Figure 17, and permutation feature importance, which consists of randomly shuffling the values of a single feature and observing the resulting degradation of the model's accuracy. Figure 18 shows the results of ten permutations. The comparison between the two methods for assessing feature importance in the Random Forest classifier reveals both significant convergences and differences. MDI highlights `numVotes`, `startYear`, and `totalCredits` as the most relevant variables for the model, followed by `castNumber` and `runtimeMinutes`. These same features are also among the most important with the permutation feature importance, confirming their central role in the model's predictive ability. However, a variation in the ranking of features is observed: for example, `TV Series` is among the least important according to MDI, yet emerges as the most relevant feature according to permutation importance. This discrepancy suggests that MDI may be influenced by the structure of the trees and the presence of correlations among variables, while permutation importance provides a more direct measure of each feature's impact on predictive performance. The observed discrepancy for the `TV Series` variable suggests two primary underlying mechanisms. First, `TV Series` exhibits hidden correlations with other numerical features in the dataset. While the Random Forest algorithm preferentially selects these correlated

numerical variables for splitting decisions during tree construction, `TV Series` represents the fundamental underlying driver of the predictive relationship. This phenomenon occurs because tree-based algorithms tend to favor numerical features over categorical ones when both carry similar predictive information, leading to an underestimation of the categorical variable's true importance in the MDI metric. Second, `TV Series` functions as a confounding variable that simultaneously influences both other features and the target variable, creating a complex causal structure that MDI fails to capture adequately. For instance, `TV Series` inherently possess different structural characteristics compared to movies, including episodic format, reduced runtime, and lower vote counts, which ultimately affect their rating patterns. The Random Forest algorithm identifies and utilizes these intermediate causal factors for splitting decisions, while the original categorical cause (title type) remains less visible in the tree structure. This masking effect results in `TV Series` receiving lower importance scores in MDI despite being the primary causal factor in the predictive chain. The integrated analysis of both approaches makes it possible to identify key features as well as variables that may be underestimated or overestimated depending on the chosen metric. This underscores the importance of employing multiple evaluation criteria to achieve a comprehensive and reliable understanding of the role of features within a Random Forest model.

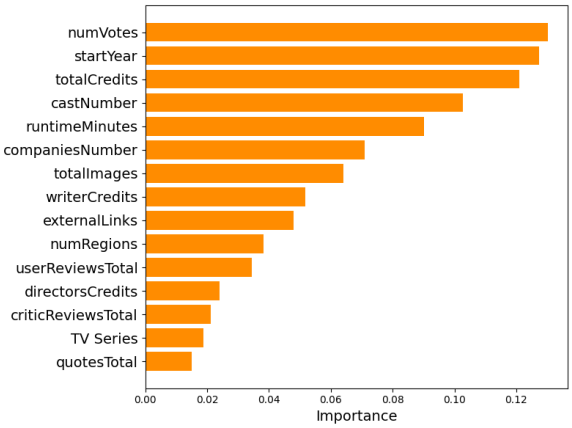


Figure 17: Bar plot of the first fifteen most important features for the Random Forest classifier for the target variable `rating`. The importance is computed with the Mean Decrease in Impurity.

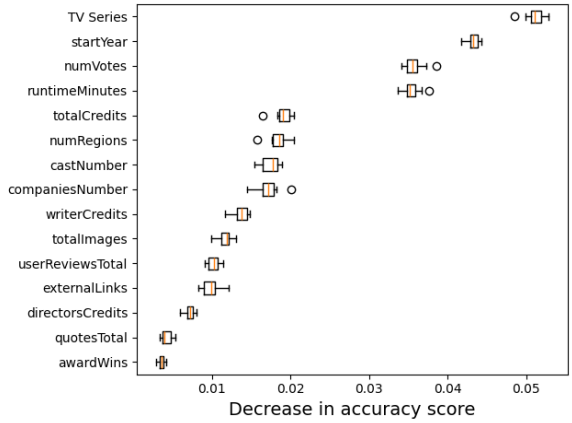


Figure 18: Fifteen most important features for the Random Forest classifier for the target variable `rating`, based on feature permutation importance. Box plots represent the distribution of importance scores across ten independent permutation iterations for each feature.

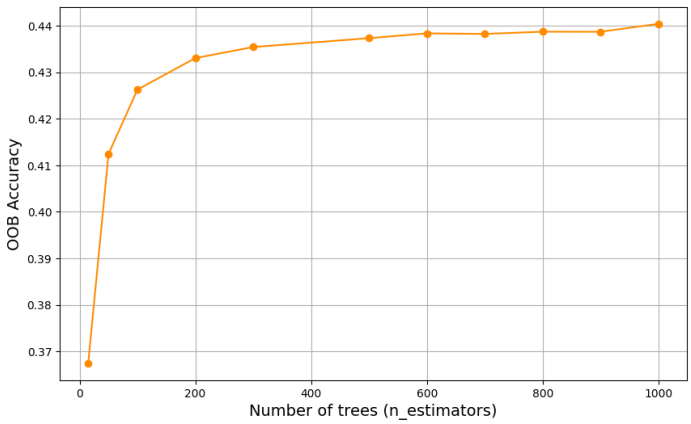


Figure 19: Out-of-bag accuracy w.r.t. the variation of the number of base models for the Random Forest classifier.

Finally, a study on how the number of base models impacts the performance of the classifier is executed. This analysis is performed via out-of-bag accuracy, the average accuracy for each training record calculated using predictions from the trees that do not contain that record in their respective bootstrap sample: as shown in Figure 19, after an obvious significant increase in accuracy for a low number of estimators, from 100 to 1000 there is an increase in accuracy of slightly more than 1%. `n_estimators`=300 seems a reasonable choice since increasing more increases the training time without a significant improvement in performance.

TitleType

The same approach followed for `rating` is also applied in the same way for the target variable `titleType`. The classifier evaluation metrics are shown in the table 7, and the ten most important features with their scores are shown in Table 8.

Class	Precision	Recall	F1-score	Support
Movies	0.91	0.95	0.93	12835
Shorts	0.88	0.92	0.90	4992
TV Series	0.96	0.98	0.97	24351
TV Special	0.75	0.08	0.14	343
Video	0.84	0.39	0.53	1541
Videogame	0.88	0.68	0.77	526
Accuracy			0.93	44588
Macro avg	0.87	0.67	0.71	44588
Weighted avg	0.93	0.93	0.93	44588

Table 7: Classification metrics per class of the Random Forest classifier for the target variable `titleType`.

Feature	**Importance**
runtimeMinutes	0.29
numRegions	0.15
startYear	0.07
averageRating	0.07
totalCredits	0.06
castNumber	0.06
directorsCredits	0.05
externalLinks	0.04
numVotes	0.03
writerCredits	0.03

Table 8: Top 10 feature importances, based on MDI, of the Random Forest classifier for the target variable `titleType`. The scores are normalized.

It is observed that, in this case, the feature `runtimeMinutes` is clearly the most important, contributing almost one-third of the reduction in impurity of tree splits. This result confirms expectations, since `runtimeMinutes` is the most distinctive feature among the various title types.

5.5 Gradient Boosting Machines

For Gradient Boosting Machines, the XGBoost algorithm is selected due to its power, efficiency, and flexibility in tuning a wide range of hyperparameters through grid search with k-fold cross-validation. As in previous experiments, the dataset is split into training (70%) and testing (30%) sets, with stratification based on the target variable. All features are standardized.

Rating

Grid search is performed to tune key hyperparameters such as tree depth, learning rate, and regularization terms. After fifty iterations, the best configuration includes a high L2 regularization term of 10, a moderately high L1 regularization term of 1, a medium learning rate of 0.1, and a tree depth of 7. This combination results in a model that is well-balanced between complexity and generalization. The strong regularization helps prevent overfitting, while the moderate depth and learning rate allow the model to learn effectively without converging too slowly. Performance metrics are summarized in Table 9, while Figure 20 shows the ROC curve and Figure 21 the precision-recall curve.

Class	**Precision**	**Recall**	**F1-score**	**Support**
(0, 1]	0.67	0.23	0.34	26
(1, 2]	0.43	0.06	0.11	141
(2, 3]	0.38	0.04	0.07	353
(3, 4]	0.31	0.05	0.09	1027
(4, 5]	0.38	0.13	0.20	2719
(5, 6]	0.38	0.26	0.30	6372
(6, 7]	0.38	0.38	0.38	11643
(7, 8]	0.43	0.75	0.55	14463
(8, 9]	0.38	0.13	0.19	6563
(9, 10]	0.49	0.14	0.22	1281
Accuracy			0.41	44588
Macro avg	0.42	0.22	0.25	44588
Weighted avg	0.40	0.41	0.37	44588

Table 9: Classification metrics of the XGB classifier, Rating.

With an overall accuracy of 0.41, the highest F1-score is observed in the (7, 8] class, while performance is lower across the remaining classes. Feature importance is also computed and will be discussed in Section 7. Interestingly, unlike the Random Forest model, the most influential features for predicting rating are `runtimeMinutes` and `numRegions`.

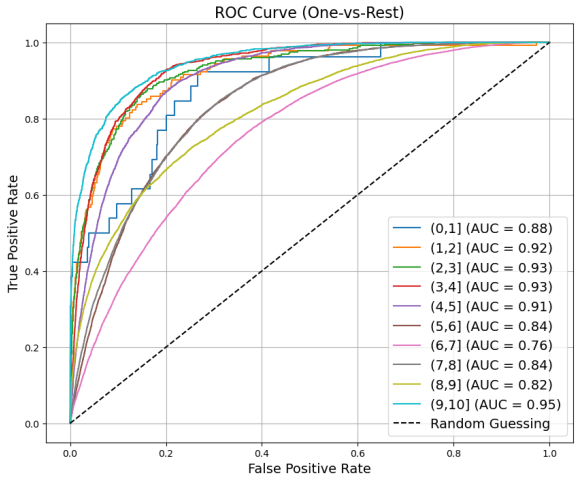


Figure 20: ROC curve of the XGB classifier, `rating`. AUC = 0.78

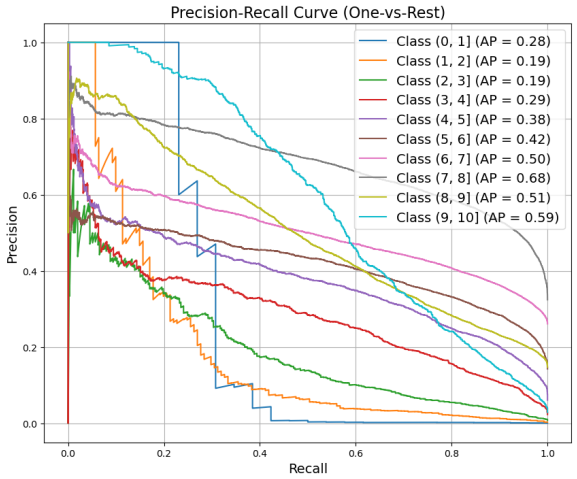


Figure 21: Precision-recall curve of the XGB classifier, `rating`.

Title Type

For the second target variable, grid search is again applied. In this case, the model does not require any L1 or L2 regularization, suggesting that regularization is not necessary to achieve strong performance. The optimal configuration includes a tree depth of 9 and a relatively high learning rate of 0.3, enabling faster learning.

The model achieves excellent results, reported in Table 10, with an overall accuracy of 0.94 and F1-scores above 0.90 for most classes. The only exceptions are the `TV Special` and `Video` classes, which show significantly lower performance. Overall, the model proves to be highly effective and well-calibrated for this target variable. Feature importance will again be explored in the explainability section.

Class	Precision	Recall	F1-score	Support
Film	0.91	0.95	0.93	12835
TV Series	0.97	0.98	0.97	24351
Shorts	0.89	0.93	0.91	4992
TV Special	0.59	0.23	0.33	343
Video	0.76	0.47	0.58	1541
Videogame	0.85	0.80	0.83	526
Accuracy			0.94	44588
Macro avg	0.83	0.73	0.76	44588
Weighted avg	0.93	0.94	0.93	44588

Table 10: Classification metrics of the XGB classifier, Title Type.

5.6 Advanced Classification Results

Rating

By comparing the results of all classifiers used, it can be stated that the best performer is the **Random Forest**. It achieves strong results across all classes, including those less represented in the dataset. Also the **Deep MLP** and **XGBoost** perform well: XGBoost attains results very similar to Random Forest in correctly classifying the target variable's classes. The Deep MLP performs well on the most represented classes but struggles to correctly distinguish the less represented ones, with the exception of the `(1, 2]` class.

Focusing specifically on the classes of `rating`: the first three classes are the most difficult to classify, which is not surprising given that they also have the lowest support. Some classifiers outperformed others on these classes, with the Deep MLP excelling in the `(1, 2]` class, and Random Forest along with XGBoost performing best on the `(0, 1]` class. Random Forest's bootstrap sampling naturally creates diverse training subsets where minority classes achieve varying representation across individual trees, enabling better capture of rare class patterns. In contrast, SVM and logistic regression optimize global decision boundaries that can be distorted by the limited and potentially noisy samples from low-support classes.

The most represented classes in the dataset, namely `(6, 7]` and `(7, 8]`, are the ones with the highest f1-scores across both the best and the worst classifiers.

Classes, `(8, 9]` and `(9, 10]`, generally have a lower f1-score, with good separation from the other classes particularly within the Random Forest model.
As for the ROC AUC scores of the individual classes, these are influenced by the true negatives and can therefore give a misleading impression of good performance in the less represented classes.

TitleType
As for the classification of the target variable `titleType`, the best-performing classifier is **XGBoost**. Compared to the project for Data Mining 1, an improvement in the final accuracy result is obtained, going from the previous maximum of `0.90` (achieved by Decision Tree) to the new `0.94`.
Also in this case, Random Forest proves to be an excellent classifier, this time alongside scikit-learn's **MLPClassifier** and the **RBF SVM**.
The `TV Special` class, which is also the least represented in the dataset (0.7%), remains the most difficult to classify. The best result is achieved by XGBoost with an f1-score of 0.33.
The `Movies`, `TV Series`, and `Shorts` classes remain the ones generally classified best, with high performance even among the worst classifiers.
The `Videogame` class shows excellent performance with both Random Forest and MLPClassifier, but reaches an f1-score of 0.83 only with XGBoost.
Finally, the `Video` class has an f1-score ranging between 0.50 and 0.60 across all the models analyzed. This indicates that the class is often distinguished from the others, probably the smaller classes, and therefore has enough informative signal to achieve a good level of precision and recall, but not a strong enough signal to go beyond that, likely due to a lack of discriminative features.
Regarding this aspect, it is interesting to note how records belonging to `TV Special` and `Video` are very often misclassified as `Movies`. This is most likely because the classifiers, in moments of 'uncertainty', tend to fall back on a dominant class, and compared to `TV Series`, `Movies` is more similar to these two classes, especially with regard to the variable `runtimeMinutes`, which, as we will see, has a major impact on the decisions made by the models.
These aspects of the classification will be further explored later in Section 7.

6 Advanced Regression

This section addresses regression for the target variable `averageRating`; all numerical features of the dataset are used as predictors. The dataset is divided into train and test, with a 70%-30% proportion. Two different regressors are employed: a Random Forest and an XGBoost model. The aim is to evaluate the predictive capabilities of these models in estimating the average rating, analyzing their performance and interpreting the results obtained.

6.1 Random Forest Regressor

For the Random Forest regressor, a randomized search with cross-validation is implemented to optimize the hyperparameters `max_depth`, `min_samples_leaf`, and `min_samples_split`, targeting the minimization of the mean squared error (MSE). The optimal configuration identified is `max_depth=None`, `min_samples_split=2`, and `min_samples_leaf=1`, maximizing the model's predictive capacity while maintaining full flexibility in tree growth. Resulting performance metricsare summarized in Table 11, along with those of the XGBoost regressor.
This implementation demonstrates moderate performance on the analyzed dataset: the coefficient of determination R^2 reaches a value of 0.35, indicating that the model is capable of explaining approximately 35% of the variance present in the target data: the MSE stands at 1.18, while the mean absolute error (MAE) equals 0.79. The MSE, being based on squared errors, amplifies predictions with significant deviations, suggesting the presence of outliers or particularly imprecise predictions in the dataset. The MAE value implies that, on average, the model's predictions deviate from the actual value by approximately 0.79 units in the original measurement scale of the target variable. These values suggest that the decision tree-based ensemble model succeeds in partially capturing underlying patterns in the data, while still presenting significant room for improvement in predictive capability.

6.2 XGBoost Regressor

For the XGBoost regressor, a grid search with cross-validation is performed to determine, in particular, the optimal learning rate, exploring values from 0.01 to 0.3: the analysis identifies a learning rate of 0.2 as the most effective. From the other parameters, we can see that the model uses a high L2 regularization value of 10, a medium-high L1 regularization of 1, and a high maximum tree depth of 9. This configuration builds a powerful and complex model with strong regularization to help prevent overfitting. Evaluation results, are detailed in Table 11.
The obtained R^2 value is 0.31, corresponding to 31% of explained variance. The MAE of 0.82 suggests that XGBoost predictions deviate on average from the actual value by approximately 0.82 units.

6.3 Advanced Regression Results

The comparative analysis highlights the superiority of Random Forest over XGBoost across all considered metrics. Random Forest outperforms XGBoost by four percentage points in terms of R^2, registers a lower MSE by 0.07 units, and presents a reduced MAE by 0.03 units. This difference, while not dramatic, proves consistent across all evaluated metrics. The superior performance of Random Forest could be attributed to the nature of the dataset, which might benefit more from the bagging approach and variance reduction typical of random tree ensembles, rather than the sequential optimization of boosting implemented in XGBoost. Despite XGBoost being generally recognized for its superior generalization capabilities and gradient optimization, in this specific application context it fails to surpass the performance of traditional Random Forest.

Metric	Random Forest	XGBoost
R^2	0.35	0.314
MSE	1.18	1.25
MAE	0.79	0.82

Table 11: Performance metrics of the regressor models

7 Explainability

7.1 SHAP

Rating
This section analyzes explainability, aiming to explain the classification of the XGBoost classifier, described in Section 5.5, which has proven to be one of the best among those tested. The analysis focuses both on trying to provide a global explanation to understand the overall model behavior and on examining individual cases to offer local, detailed interpretations of classification decisions.
For the explainability analysis of the developed XGBoost model, the SHAP method is applied using the optimized TreeExplainer implementation, specifically designed for tree-based models. Figure 22 displays a summary plot of SHAP values for the target class `(7,8]`, where each point represents a test set observation, showing both feature impacts on predictions (horizontal axis) and their z-normalized value distributions (color scale). Concurrently, Table 12 reports global feature importance defined as the average gain across all splits the feature is used in. Comparative analysis reveals significant similarities and differences: critical features like `runtimeMinutes` (highest importance) and `StartYear` show consistency between local and global relevance, as do marginal features like `totalVideos`. However, `isAdult` exhibits a marked discrepancy: while ranking among the top five global features, its local contribution proves negligible for the `(7,8]` class, likely reflecting the feature's extreme imbalance in the dataset. Conversely, `NumVotes`, absent from the global top five, demonstrates systematically high local impact, suggesting a contextually crucial role in discriminating the specific class. This dichotomy emphasizes the necessity to integrate global and local perspectives for comprehensive model behavior understanding.
The analysis then focuses on explaining the classifier's decision for two specific records, both belonging to the `(7,8]` class, identified as the easiest to predict. One record is correctly classified, while the other is misclassified as belonging to the `(6,7]` class, despite both having a relatively high predicted probability of 75% for their assigned class. Figures 23 and 24 present the waterfall plots of the features that most influenced the model's decision for each record, respectively for the misclassified and

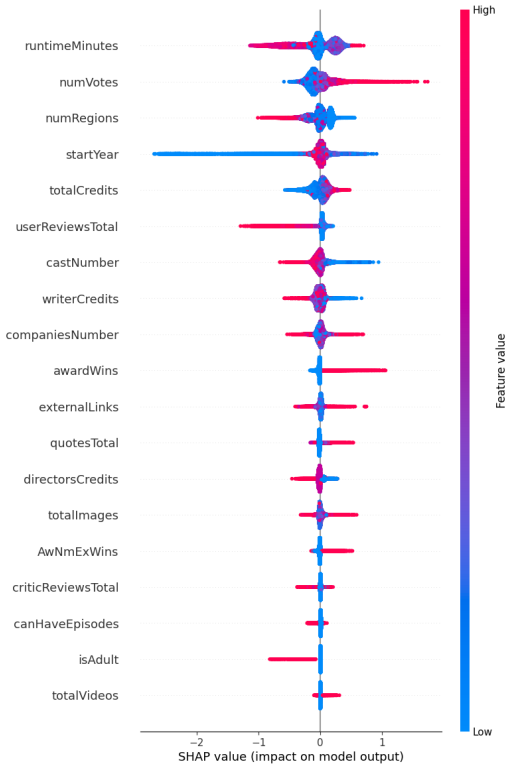


Figure 22: Summary plot for rating interval class (7,8].

Feature Importance	
runtimeMinutes	3.45
numRegions	3.07
isAdult	2.93
startYear	2.59
awardWins	2.53
canHaveEpisodes	2.21
AwNmExWins	1.93
numVotes	1.89
writerCredits	1.81
directorsCredits	1.80
quotesTotal	1.76
totalCredits	1.53
userReviewsTotal	1.52
externalLinks	1.52
companiesNumber	1.45
castNumber	1.44
totalImages	1.33
criticReviewsTotal	1.32
totalVideos	1.08

Table 12: Feature importance values for the XGBoost classifier, defined as the average gain across all splits the feature is used in.

correctly classified cases. The expected value is the classifier base value for the predicted class, while $f(X)$ represents the predict probability, both in terms of log-odds. For each displayed feature the z-normalized value is reported. Comparative analysis reveals a significant difference in the features with the greatest predictive impact: in the correctly classified record, a low value of `numRegions` pushes the prediction toward the correct class, consistent with the summary plot findings. Conversely, in the misclassified record, `numRegions` is the most influential feature, with a high relative value that directs the prediction toward the incorrect class. Additionally, the `awardWins` feature has a high value in the misclassified record and contributes positively to classification in the `(6,7]` class; however, as shown in the summary plot, high values of this feature also support the `(7,8]` class, indicating that it may introduce ambiguity and confusion in the classifier's decision-making process.

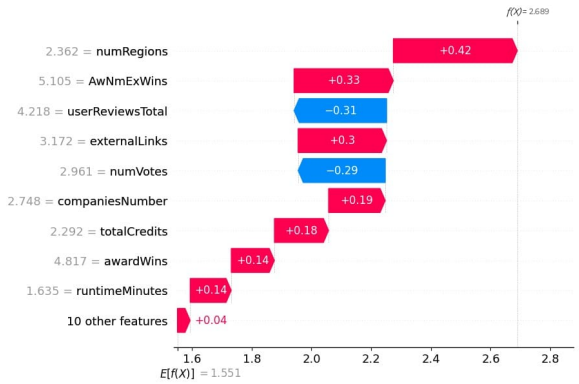


Figure 23: Waterfall plot of the features impact of a class (7,8] instance wrongly classified as class (6,7]. The predict probability of this record is 0.75.

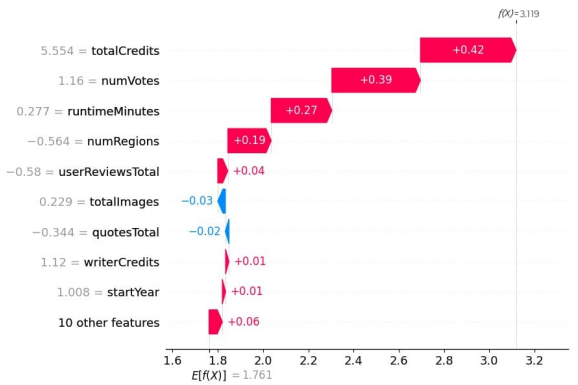


Figure 24: Waterfall plot of the features impact of a class (7,8] instance correctly classified. The predict probability of this record is 0.75.

TitleType

Regarding the `titleType` variable, where it is noted that the `TV Special` class is frequently misclassified as `Movies`, as said in Section 5.6. This issue is further investigated using **SHAP** within the XGBoost classifier, applying the same methodology previously described for the `rating` variable. Initially, as in the analysis of `rating`, a comparison is made between the global feature importance and the feature importance obtained through SHAP for the `TV Special` class, again using the optimized TreeExplainer implementation. As for SHAP, and therefore for `TV Special`, the most important features are, respectively:
`runtimeMinutes=0.57, numRegions=0.39, castNumber=0.38, writerCredits=0.31, startYear=0.30`. The values reported for each feature correspond to the mean absolute value of the SHAP values for that specific class. As for the global features of XGBoost, the most important ones are, in order: `canHaveEpisodes=41.2, isAdult=33.9, runtimeMinutes=33.7, numRegions=28.0, directorsCredits=17.1`.
We immediately notice that, while at a global level the most important feature by far is `canHaveEpisodes`, which naturally distinguishes the `TV Series` class from the others, for the `TV Special` class the most important local feature is `runtimeMinutes`. This feature in particular appears to differentiate `TV Special` from the other classes, as seen by SHAP summary plots, especially from `TV Series`, `Shorts`, and `Videogame`, when it takes on a high value, but at the same time brings it closer to the `Movies` class. A further analysis is carried out on this aspect by selecting two records from the dataset belonging to `TV Special`, both with a predicted probability of 75%: the first misclassified as `Movies`, Figure 25, and the second correctly classified, Figure 26. It is immediately evident that the `runtimeMinutes` feature plays a crucial role in classifying records as either `TV Special` or `Movies`, confirming the initial hypothesis that this feature aids the models in correctly distinguishing the `TV Special` class from others, while also causing classifiers' more uncertain decisions to be mistakenly assigned to the `Movies` class.

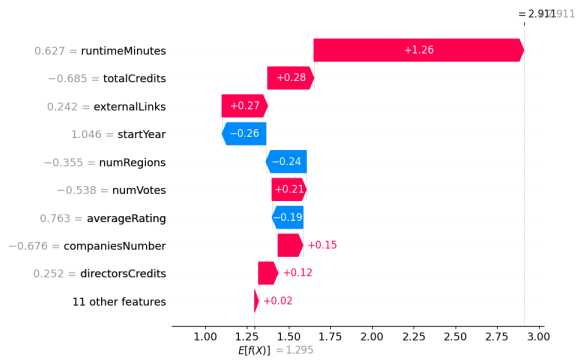


Figure 25: Waterfall plot of the features impact of a class 'TV Special' instance wrongly classified as class 'Movies'. The predict probability of this record is 0.75.

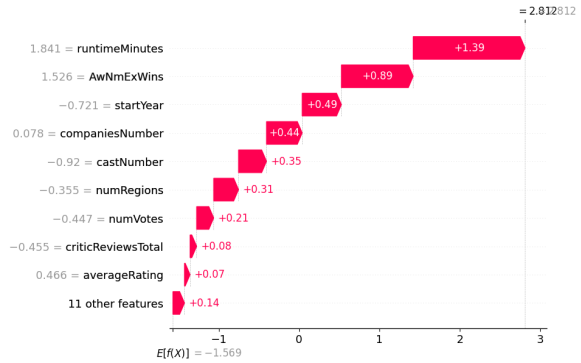


Figure 26: Waterfall plot of the features impact of a class 'TV Special' instance correctly classified. The predict probability of this record is 0.75.

7.2 LORE

To explain the classification results of `rating` provided by the XGBoost classifier, another explainability method, **LORE**, is also used. For comparison with the SHAP results, the same two instances shown in Figures 23 and 24 are explained. For the instance that is misclassified into the class `(6,7]`, this is the explanation provided by LORE:
`r = startYear <= 1967.50, titleType != TV Series -> rating: (6, 7] .`
As for the instance that is correctly classified, LORE provides the following explanation:
`r = titleType != Movies, numVotes > 1.08, criticReviewsTotal <= 0.35 -> rating: (7, 8] .`
The most interesting aspect of these results is the presence of `numVotes > 1.08`; in fact, in the SHAP

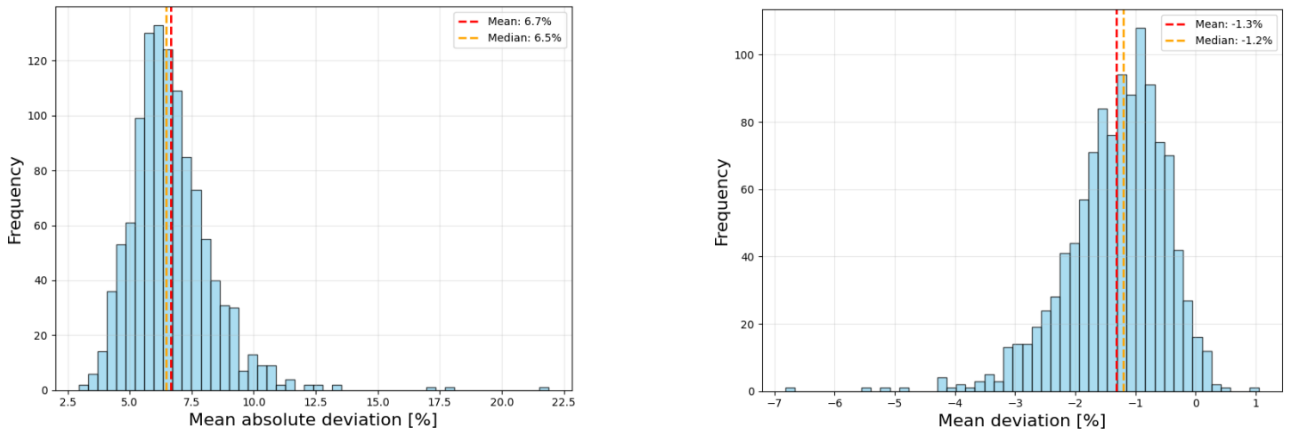
explanations, a value of `1.16` for the `numVotes` variable in the correctly classified record contributes significantly to the correct classification. Similarly, in the misclassified record, a value of `2.961` for `numVotes` pushes the classifier's decision in the correct direction.

8 Time Series

For the time series analysis, a dataset containing the daily domestic income (USA and Canada) of 1134 movie titles is extracted, spanning a period of one hundred days from their theatrical release. The data source is the IMDbPro website. Since some titles have fewer than one hundred days of data, the missing days were filled using noise-augmented mean imputation. The dataset also includes the movie ID, rating, rating category, and genre. Using an API, the data are enriched with movie titles retrieved via their IDs.

8.1 Preparation

In the data preparation phase, the dataset is analyzed and preprocessed for the subsequent tasks.
From a simple plot of the mean and standard deviation of the time series, useful characteristics emerges: generally, movies show high income values in the first few days after release, followed by recurring patterns every fourteen days that gradually diminish. In reality, these recurrences occur every seven days, corresponding to weekends when cinema attendance is higher. This biweekly trend is due to the initial synthethic data augmentation. Even for movies with time series longer than a hundred days on IMDbPro, it is observed, via comparison on the website, that only the first 50 days are considered, then extended to a hundred using noise-augmented mean. To identify the presence of data augmentation in the dataset, an analysis of relative deviations of odd-indexed values with respect to the mean of adjacent values is conducted. The analysis reveales a mean absolute deviations of 6.7% (σ = 1.6%) and signed deviations of -1.3% (σ = 0.8%), indicating a systematic negative bias. The distributions of mean deviations, shown in Figure 27 exhibit quasi-normal shapes, suggesting a controlled perturbation process rather than natural variability. These results provide quantitative evidence of synthetic data obtained through interpolation with additive noise, confirming the hypothesis of systematic data augmentation of the original dataset.



(a) (b)

Figure 27: Histrogram of the distribution of mean deviations: (a) absolute relative deviations, (b) signed relative deviations.

In any case, the time series are kept at this length, analyzing the dataset as it was provided.
Regarding the standard deviation, income values deviate more significantly at the beginning and during weekly recurrences, but these deviations stabilize over time.
Each time series is checked for missing values, but none is found. For every time series also anomalies are checked using a cross-validation approach with the Hampel Filter, Grubbs' Test, and the IQR

method. If a data point is flagged as an anomaly by at least two out of the three methods, it's removed: no such anomalies are detected.
The data are then normalized using Z-score normalization to make the time series, which are on different scales, comparable for clustering and classification.
Before proceeding to the next phases, three hundred and seventeen non-stationary time series are identified and detrended to remove trend components and make them more suitable for statistical and predictive modeling.
Finally, a SAX (Symbolic Aggregate approXimation) transformation is applied to reduce complexity, using a dimensionality of 50 and 6 distinct symbols. However, as will be discussed in Section 8.3, the non-approximated series are preferred due to the relatively low original dimensionality and better performance.
The original `ratingCategory` classes are five: High, Medium High, Medium, Medium Low, and Low; however, since records classified as Low account for less than 1% of the dataset (ten instances), this class is merged with Medium Low to ensure a more balanced distribution of observations. This aggregation is performed both for classification purposes and to analyze the distribution within the clusters.

8.2 Motifs/Discords

The next step involves identifying motifs and discords to detect recurring patterns and unique behaviors in the series. In both cases, a sliding window of size 7 is used, based on the observed fourteen days peaks.
Motifs are computed for each title, along with the matrix profile. A maximum of four motifs are found per statistical unit, with an average of about three motifs per unit.

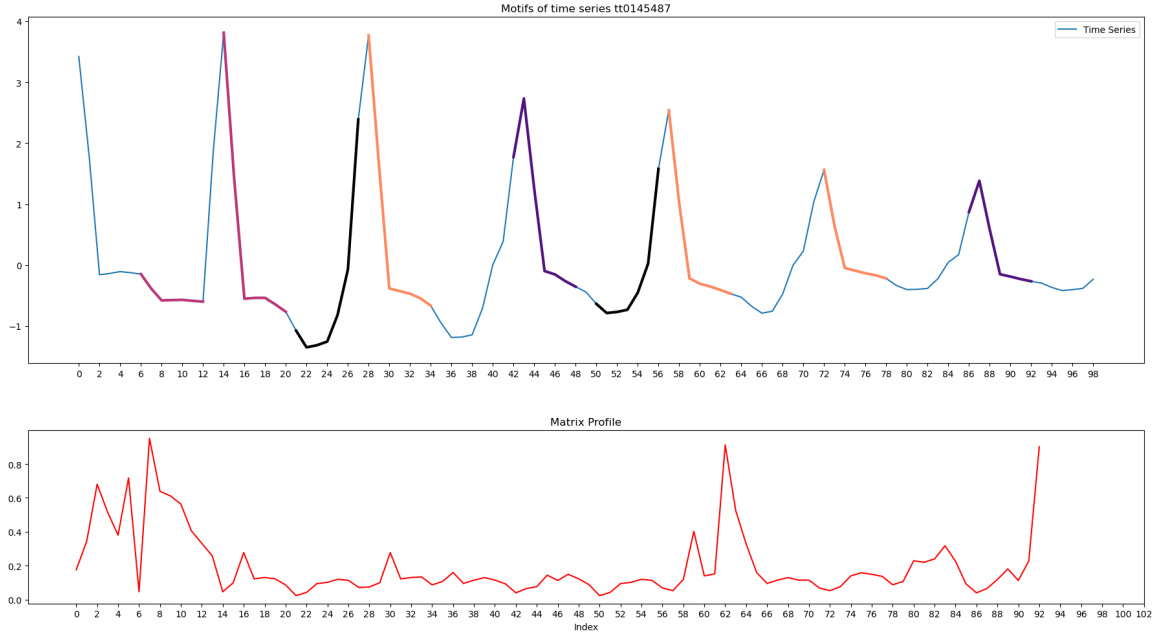


Figure 28: Motifs (top) and Matrix Profile (bottom) of a time series (Spider-Man: 2002)

Figure 28 shows an example of a time series with four motifs, shown in different colors, along with the matrix profile. Notably, income values show similarities at the end of weekends, when they decline, and a few days later, as Tuesdays in the USA and Canada often feature promotional offers to attract customers. Other similarities occur at the beginning of weekends when attendance increases. From the graph, it appears that the motifs at time points 5 and 14 may not be visually accurate. This is due to some noise present in the early days of the data.
As for discords, the three most anomalous subsequences are selected for each title, using an exclusion zone of three. Figure 29 shows the discords found on the same time series of Figure 28.

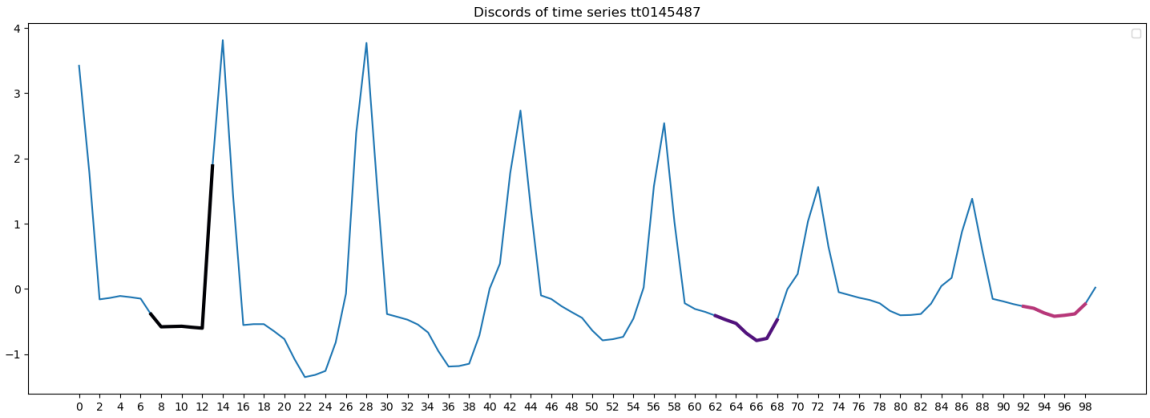


Figure 29: Discords on a time series (Spider-Man: 2002)

For instance, as expected due to data noise, a discord appears at timestamp 6, overlapping with a previous motif subsequence. The second anomalous subsequence at timestamp 62 stands out due to a spike in income a few days earlier. The third discord at time 92 is attributed to a flattening of box office revenues.
To investigate the nature of discriminative patterns, an analysis of the spatial relationship between motifs, discords and shapelets, the extraction of which will be discussed in section 8.4, is conducted. The analysis focuses on temporal overlap between these elements, defined as the shapelet length that coincides with a motif or discord at the same temporal position. Results show significant overlaps between discriminative shapelets and both motifs and discords, indicating that both pattern types contribute to rating category classification. Overlaps with motifs suggest that specific recurring patterns in box office progression constitute reliable indicators for certain categories, likely reflecting typical distribution strategies or consumption cycles. Simultaneously, overlaps with discords highlight the importance of exceptional events, such as unexpected revenue peaks or sudden drops, that distinctively characterize some movie rating categories. These findings indicate that an effective classification approach must consider both the presence of standardized patterns and the occurrence of significant anomalies in the temporal evolution of commercial performance. An example of these overlaps is shown in Figure 30 and Figure 31.

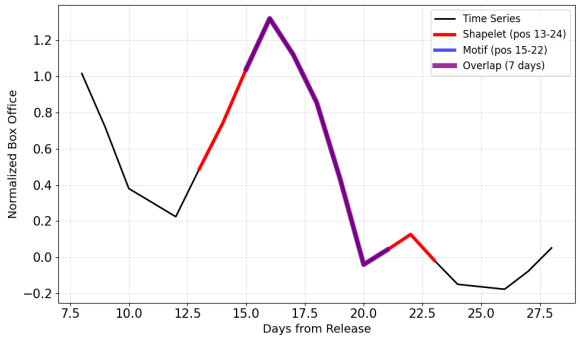


Figure 30: Zoom of a time series on the overlap region between a shapelet and a motif.

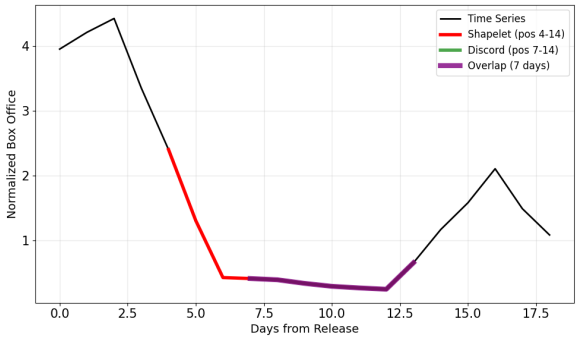


Figure 31: Zoom of a time series on the overlap region between a shapelet and a discord.

8.3 Clustering

As previously mentioned, both the approximated and non-approximated datasets—already standardized—are used for clustering. The non-approximated dataset yields better results.
In this phase, a distance-based clustering approach is applied using K-means, as well as a hierarchical clustering after transforming the dataset into new features using TSFresh. Additionally, another feature transformation using BORF is also tested.

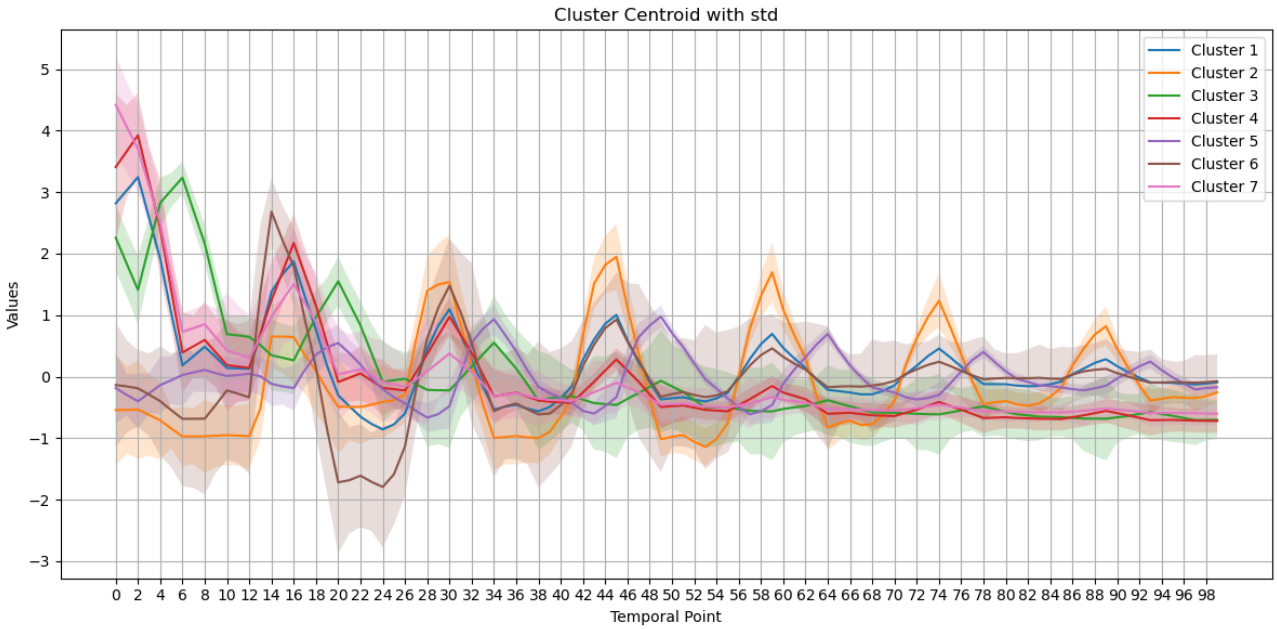


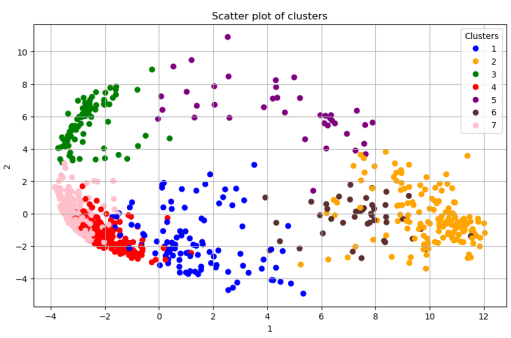
Figure 32: Centroids of the clusters found with K-means.

8.3.1 K-means

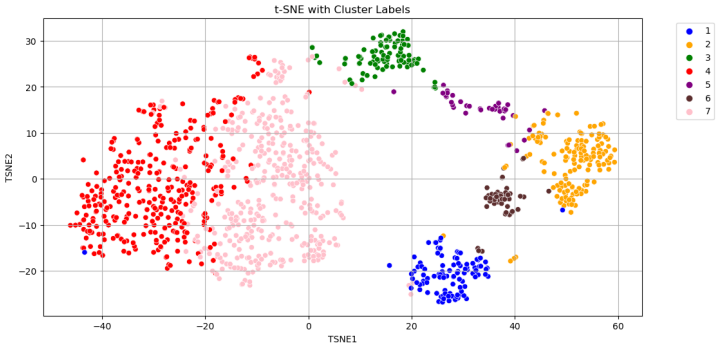
For the non-approximated dataset, the `TimeSeriesKMeans` algorithm is used with Euclidean distance. This distance metric is chosen because it is simple and fast to compute, the time series are of equal length, and they are fairly well aligned, even at the peak points.
First, the optimal number of clusters is determined by computing the silhouette scores and the SSE for cluster counts ranging from two to ten. The chosen number of clusters is seven, with a silhouette score of 0.36 and an SSE of 3353. The algorithm is then applied, and the centroids of the time series for each cluster are visualized in Figure 32.
The following characteristics can be observed for each cluster:

- **Cluster 1**: starts with high values but tends to flatten over time.
- **Cluster 2**: includes time series with initially low income that later exhibit higher and more substained peaks.
- **Cluster 3**: similar to cluster 1 but shifted forward by one day.
- **Cluster 4**: similar to cluster 1, but income flattens much more quickly.
- **Cluster 5**: starts low with cyclic patterns shifted forward by one day; has high standard deviation due to the diversity of titles.
- **Cluster 6**: starts low, initially shows high peaks both upward and downward, then flattens.
- **Cluster 7**: starts high, but has the lowest cyclic patterns among all clusters.

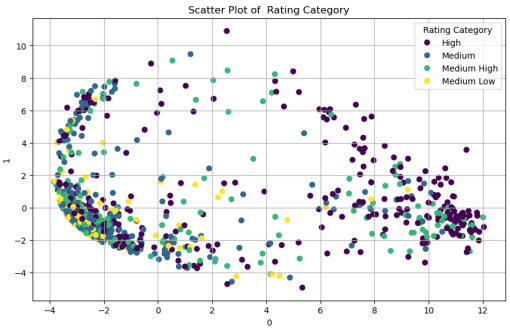
Dimensionality reduction and cluster visualization are performed using PCA and t-SNE. The distribution of these categorical variables within the clusters is then analyzed. From the plots in Figure 33, it can be observed that clusters are more clearly defined in the t-SNE visualization. However, in both visualizations, clusters 4 and 7 appear very close and slightly overlapping in the PCA plot. Additionally, in the t-SNE plot, cluster 2 (in yellow) shows tighter grouping of points.
Regarding the distribution of rating categories within the clusters, both visualizations show a mix of values, indicating that the clusters are not well-separated by rating. The only exception is cluster 2, which predominantly contains titles from the `High` rating category.
For the approximated dataset k-means leads to six clusters, slighter lower silhoutte score of 0.34 and the majority of points in only two clusters.



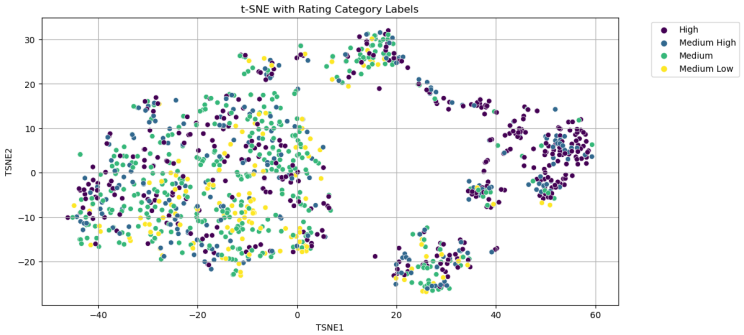
(a) Clusters K-Means - PCA



(b) Clusters K-Means - t-SNE



(c) Rating K-Means - PCA



(d) Rating K-Means - t-SNE

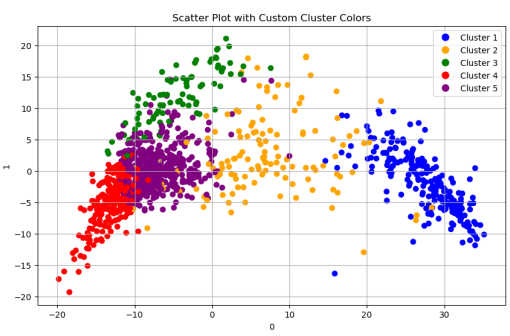
Figure 33: Visualization of clusters obtained with K-means and rating categories using PCA and t-SNE. Points Distribution: C1: 9%; C2: 13%; C3: 8%; C4: 29%; C5: 3%; C6: 4%; C7: 34%.

8.3.2 Hierarchical Clustering

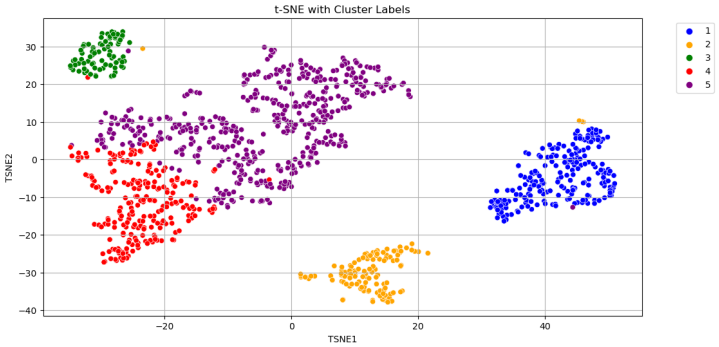
As a second clustering method, hierarchical clustering is applied using Ward's method, aiming to group the data into more compact and homogeneous clusters by minimizing the increase in within-cluster variance. Before applying the algorithm, the time series are transformed into features using TSFresh. Clustering is then performed both on the raw features set and on the dataset after applying PCA, with the latter yielding better results in terms of silhouette score and cluster purity, as will be shown.
The explained variance of the components is computed, and the dataset is reduced to six principal components. Clustering is then performed, resulting in five groups by cutting the dendrogram, with a silhouette score of 0.32. As before, clusters are visualized using PCA and t-SNE in Figure 34. In this case, the clusters appear slightly more separated than in the previous clustering method. Specifically, in the top-left of the PCA plot, cluster 2 shows more dispersed points, while in the t-SNE plot, they are much closer together. This suggests that the points are spread across multiple directions in the high-dimensional space but are locally similar. From a visual inspection of the rating distribution, cluster 1 contains a large number of `High` rating class titles and appears more clearly defined and separated compared to the previous algorithm. The remaining clusters show a mix of categories.
When hierarchical clustering is applied without PCA, four clusters are obtained, with a silhouette score of 0.28 and less well-defined groupings compared to the reduced-dimensionality dataset.
Meanwhile, when it's used the approximated dataset (post SAX), for hierarchical clustering with PCA, it results in four clusters, with a lower silhouette score of 0.29 and most points concentrated in only two clusters. However, the outcome is very similar to that of the original dataset.

8.3.3 Conclusions

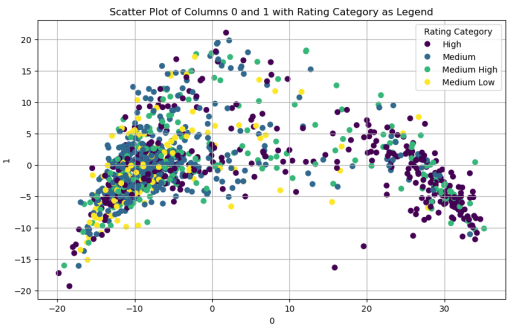
To understand the characteristics and interpretability of each clustering solution, both the `ratingCategory` and `genre` distributions are analyzed within the clusters, along with computing the average purity score. Class distribution within clusters is preferred over the opposite approach due to the class imbalance



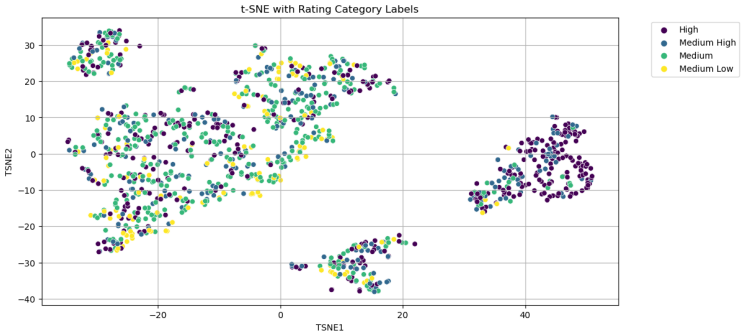
(a) Hierarchical Clustering - PCA



(b) Hierarchical Clustering - t-SNE



(c) Rating Distribution - PCA

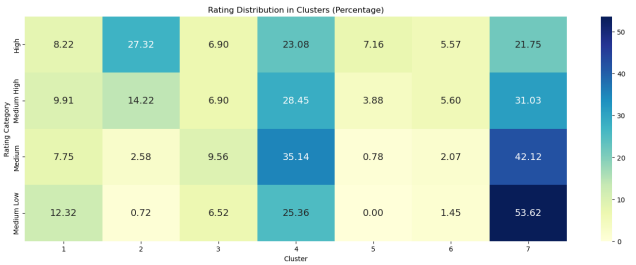


(d) Rating Distribution - t-SNE

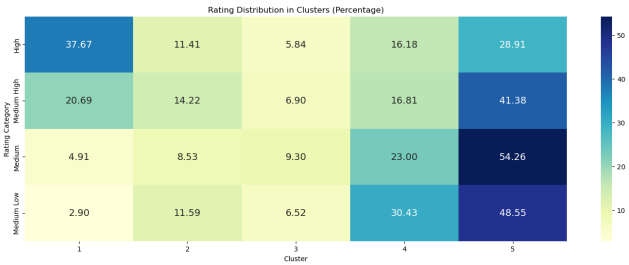
Figure 34: Visualization of clusters obtained with hierarchical clustering and rating categories using PCA and t-SNE. Points Distribution: C1: 19%; C2: 12%; C3: 7%; C4: 20%; C5: 42%.

in the dataset.

Comparing the two clustering methods, both show a slight ability to separate rating categories. With K-Means, high-rated titles are well isolated in Cluster 2, while `Medium` and `Medium High` are mainly found in clusters 4 and 7, though with some overlap. `Medium Low` is well represented in Cluster 7.
In the hierarchical clustering, cluster 1 contains nearly 40% of `High` rated titles, with some `Medium High` as well. Cluster 4 is dominated by `Medium Low`, and cluster 5 by `Medium` (54%) and `Medium High`. Cluster 7 has a more mixed composition. In summary, hierarchical clustering provides clearer segmentation for extreme classes but more overlap in central clusters. Figure 35 show the distribution of the `ratingCategory` classes among the clusters.
Regarding genre, excluding the last cluster which is more mixed, hierarchical clustering yields more interpretable groupings: cluster 1 includes genres like documentary, biography, war, and history—often associated together—while cluster 4 includes darker genres such as horror, sci-fi, thriller, mystery,

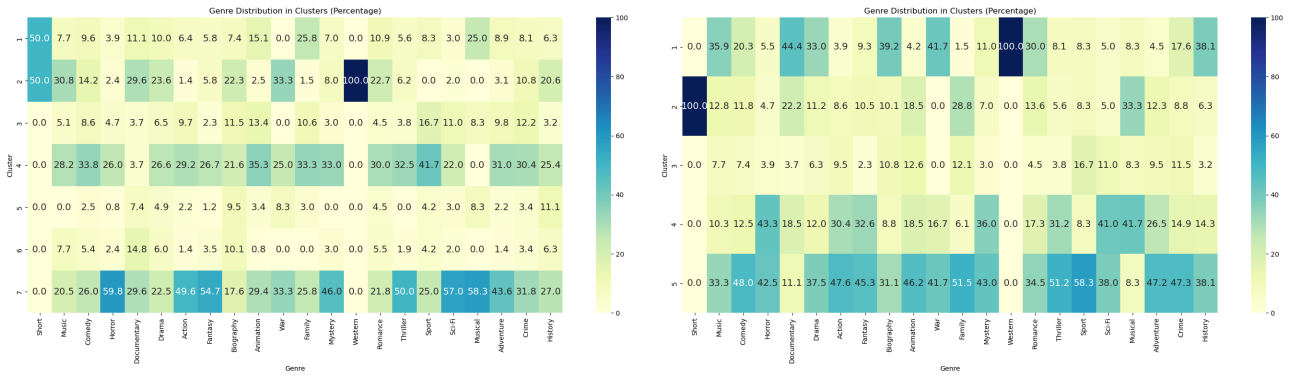


(a)



(b)

Figure 35: `ratingCategory` distribution in clusters obtained with K-means (a) and hierarchical clustering (b). Average Purity for k-means: 0.42, Average Purity for hierarchical: 0.37



(a) (b)

Figure 36: `genre` distribution in clusters obtained with K-means (a) and hierarchical clustering (b). Average Purity for k-means: 0.15, Average Purity for hierarchical: 0.16

fantasy, and, interestingly, musical. Both k-means and hierarchical clustering show similar clustering behavior for western films, though this consistency may be largely attributed to the extremely small sample size of only two western records in the dataset. Figure 36 show the distribution of the `genre` classes among the clusters.
During the clustering phase, HDBSCAN and hierarchical clustering with BORF-transformed features followed by SVD are also tested. However, these approaches perform worse, with a silhouette score of 0.30 and only two non-homogeneous clusters along with several very small groups.
Therefore, considering also the class distribution within the clusters, the best-performing clustering algorithm is the hierarchical one applied after feature extraction and dimensionality reduction.
Simple descriptive statistics are computed for each cluster of this method. Cluster 5 has the most entries (482), while cluster 3 has the fewest (82). Cluster 1 contains the highest-rated titles with an average rating of 7.2, while clusters 4 and 5 have the lowest average ratings at 6.4.

8.4 Classification

This section addresses the task of multiclass classification applied to the time series dataset, using `rating_category` as the target variable. The dataset is split into training and test sets using a stratified 70%-30% division with respect to the target variable, in order to preserve class proportions in both partitions. Several classification methods from different algorithmic families are employed, comparing their performances.

8.4.1 Instance-based models

Two KNN classifiers are analyzed, differentiated by the distance metric used: euclidean distance and dynamic time warping (DTW). For both models, hyperparameter tuning is performed using grid search combined with 5-fold cross validation, exploring the number of neighbours (3, 5, 7, 9, 11) and the weighting scheme for neighbour votes (uniform or distance-weighted) as parameters. For the DTW-based classifier, a window size of 7 with the Sakoe-Chiba band constraint is adopted to account for possible variations in the day of the week when the movie is released. The best parameter configuration found for both classifiers is `n_neighbors=9` and `weights='distance'`.
Tables 13 and 14 report the evaluation metrics for the two classifiers. It can be observed that the classifier using Euclidean distance performs slightly better than the other.

8.4.2 Shapelets-based model

This section presents a shapelet-based classifier. The shapelets are retrived using a random shapelet extraction approach, selecting 2000 candidate subsequences with a minimum length of 6 and a maximum length of 21, thus covering a time range from one to three weeks. The extraction is performed through the `RandomShapeletTransform` method from the Python `sktime` library. The extracted shapelets are

Class	Precision	Recall	F1-score	Support
High	0.55	0.50	0.53	113
Medium	0.44	0.68	0.53	116
Medium High	0.26	0.14	0.19	70
Medium Low	0.26	0.12	0.16	42
Accuracy			0.44	341
Macro avg	0.38	0.36	0.35	341
Weighted avg	0.42	0.44	0.41	341

Table 13: Classification metrics for the KNN classifier with Euclidean distance.

Class	Precision	Recall	F1-score	Support
High	0.53	0.50	0.51	113
Medium	0.42	0.61	0.50	116
Medium High	0.24	0.14	0.18	70
Medium Low	0.22	0.12	0.15	42
Accuracy			0.42	341
Macro avg	0.35	0.34	0.34	341
Weighted avg	0.39	0.42	0.39	341

Table 14: Classification metrics for the KNN classifier with DTW.

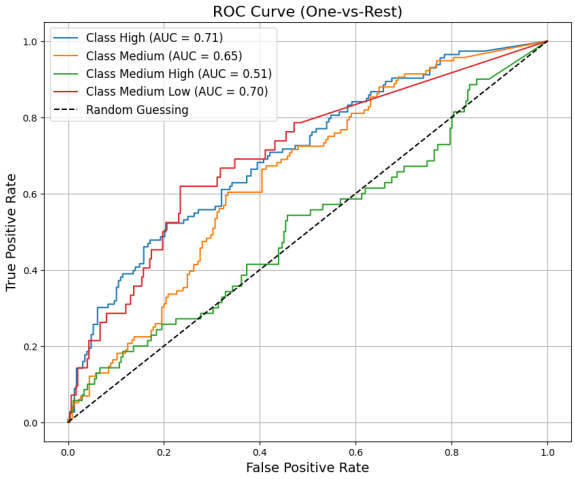


Figure 37: ROC-curve for the KNN classifier with Euclidean distance. AUC = 0.64.

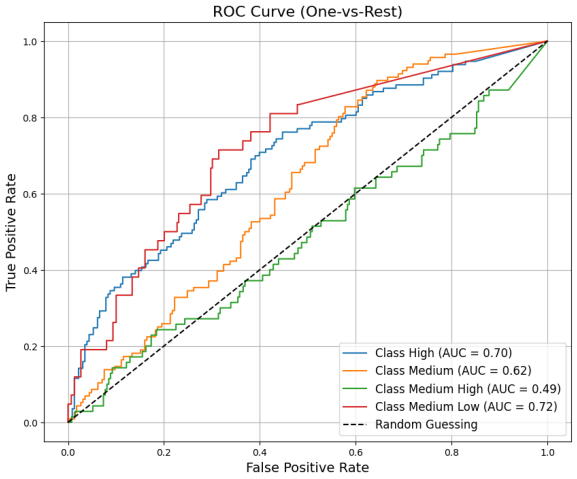


Figure 38: ROC-curve for the KNN classifier with DTW. AUC = 0.63.

then used as features for a logistic regression classifier. The five shapelets with the highest information gain belong four to series of the `High` class and one to a series of the class `Medium`, which are also the classes with the largest number of records and the best classification performances. Table 15 reports the classifier evaluation metrics. Figure 39 shows the most representative shapelets for each class based on Information Gain. Their analysis shows that:

- **Class High:** the representative shapelet shows a very pronounced peak at the beginning of the subsequence (z-normalized value above 2.5), followed by a rapid decline that brings the series below zero within the first three time stamps. Subsequently, an oscillating trend is observed without reaching the initial levels again. This pattern suggests that, locally, high-rated films often experience a phase of very intense and sudden revenue, followed by a quick decrease, which is a distinctive feature of this class.
- **Class Medium:** the shapelet exhibits an even more marked initial peak (above 3), followed by a rapid decline and an oscillating phase, with a relative second peak around the middle of the subsequence and a further decline towards the end. This indicates that films in this class also have local segments of very high revenue followed by a decrease, but with greater variability in the latter part of the shapelet. The pattern suggests phases of revenue recovery after the initial drop, which may distinguish this class from High.
- **Class Medium High:** the shapelet is characterized by a rising trend in the first half of the subsequence, reaching a maximum at the seventh point, followed by a decline and stabilization near zero. This local pattern suggests that, for this class, discriminative segments are characterized by a progressive increase in revenue rather than an immediate peak. The less pronounced shape and smaller amplitude compared to other classes reflect greater heterogeneity in local patterns, consistent with the classification difficulties observed for this category.
- **Class Medium Low:** the most informative shapelet shows a very high initial peak (above 3), followed by a sharp decline and a long stabilization phase at negative or near-zero values. This pattern highlights that, locally, even lower-rated films can have segments of very high revenue,

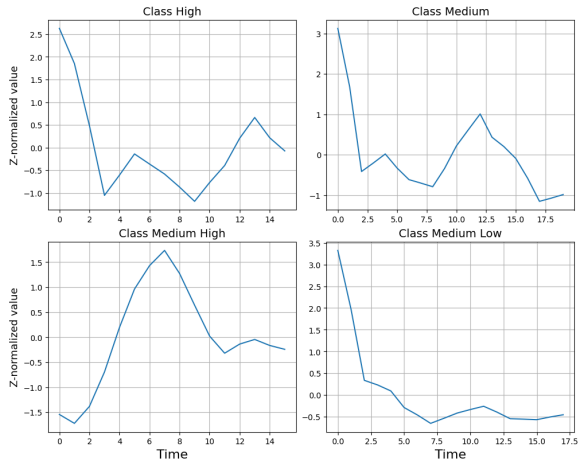


Figure 39: Most representative shapelets per class, based on Information Gain.

Class	Precision	Recall	F1-score	Support
High	0.53	0.54	0.54	113
Medium	0.46	0.64	0.54	116
Medium High	0.31	0.14	0.20	70
Medium Low	0.32	0.26	0.29	42
Accuracy			0.46	341
Macro avg	0.41	0.40	0.39	341
Weighted avg	0.44	0.46	0.44	341

Table 15: Classification metrics of the shapelets-based classifier.

Class	Precision	Recall	F1-score	Support
High	0.56	0.62	0.59	113
Medium	0.42	0.56	0.48	116
Medium High	0.36	0.21	0.27	70
Medium Low	0.38	0.19	0.25	42
Accuracy			0.46	341
Macro avg	0.43	0.40	0.40	341
Weighted avg	0.45	0.46	0.45	341

Table 16: Classification metrics of the WEASEL classifier.

Class	Precision	Recall	F1-score	Support
High	0.55	0.59	0.57	113
Medium	0.48	0.47	0.47	116
Medium High	0.33	0.27	0.30	70
Medium Low	0.36	0.43	0.39	42
Accuracy			0.46	341
Macro avg	0.43	0.44	0.43	341
Weighted avg	0.46	0.46	0.46	341

Table 17: Classification metrics of the BORF-based classifier.

but these are followed by a rapid decline and a phase of low interest, distinguishing them from other classes.

8.4.3 Other methods

A WEASEL classifier, a dictionary-based approach, is employed for time series classification. Hyperparameter tuning is performed through a grid search with 5-fold cross validation, exploring the parameters `alphabet_size` with values [2,4,6,8] and `window_inc` with values [1,3,5]: the best configuration found is `alphabet_size`=8 and `window_inc`=1. Table 16 reports the classification results obtained with WEASEL.
The analysis is extended with an additional classifier that combines the **BORF** transformer with logistic regression, replicating the WEASEL classification pipeline.meters for the **BORF** transformation are: `min_window_size=4`, `alphabets_min_symbols=3`, `alphabets_max_symbols=4` and `min_dilation=1`. The classification metrics per class for this model are presented in Table 17, while Figure 40 shows the related ROC curve and Figure 41 the precision-recall curve.
Finally, the CIF classifier, an ensemble interval-based model, configured with `n_estimators`=200, is employed; its results are reported in Table 43.

8.4.4 Final considerations

All classifiers exhibit comparable performance in terms of accuracy, but a more detailed analysis reveals substantial differences among them. As expected, all models perform better on the more represented classes, while the Medium High class proves to be the most challenging to classify. In particular, for both KNN classifiers, as highlighted by the ROC curves in Figures 37 and 38, the classification of the Medium High class is essentially equivalent to a random guess. The shapelets-based classifier, although not the best in terms of accuracy, highlights interesting patterns, showing that the most distinctive shapelet of the Medium High class is not particularly discriminative. The best classifiers in these

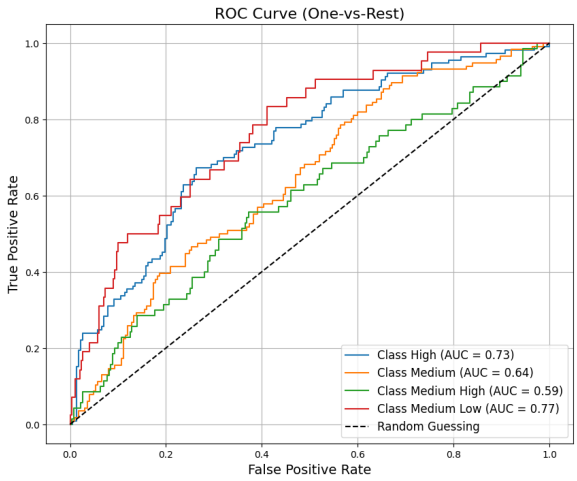


Figure 40: ROC-curve for the BORF-based classifier. AUC = 0.68.

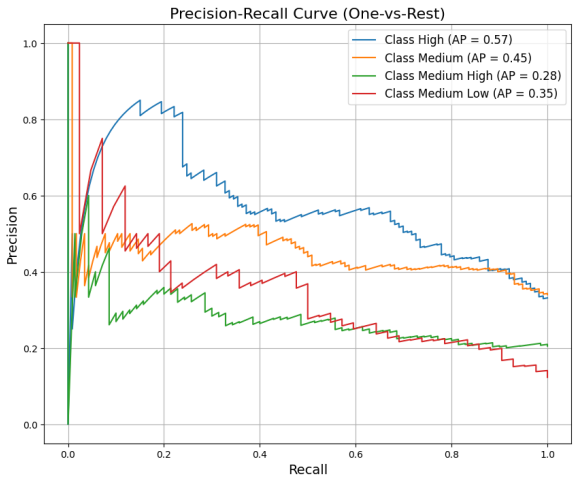


Figure 41: Precision-recall curve for the BORF-based classifier.

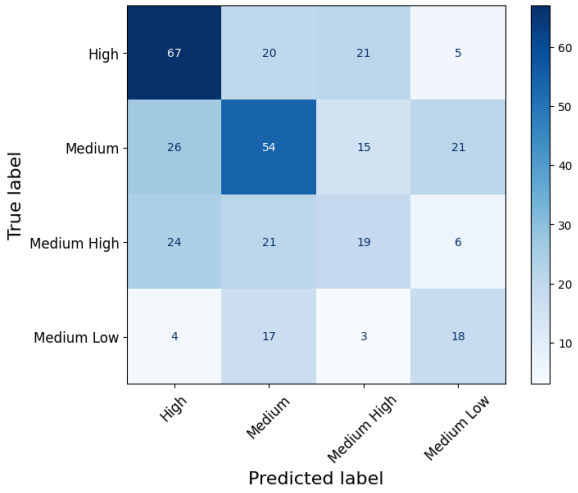


Figure 42: Confusion Matrix of the BORF-based classifier.

Class	Precision	Recall	F1-score	Support
High	0.60	0.59	0.60	113
Medium	0.44	0.76	0.56	116
Medium High	0.25	0.04	0.07	70
Medium Low	0.59	0.24	0.34	42
Accuracy			0.49	341
Macro avg	0.47	0.41	0.39	341
Weighted avg	0.47	0.49	0.44	341

Figure 43: Classification metrics of CIF classifier.

experiments are the dictionary-based ones, in particular the model based on the BORF transformation. The CIF classifier achieves the highest accuracy but performs significantly worse on the Medium High class and has a training time approximately an order of magnitude higher than WEASEL and BORF.

Class	Precision	Recall	F1-score	Support
High	0.50	0.52	0.51	113
Medium	0.43	0.43	0.43	116
Medium High	0.25	0.21	0.23	70
Medium Low	0.35	0.38	0.36	42
Accuracy			0.41	341
Macro avg	0.38	0.39	0.38	341
Weighted avg	0.41	0.41	0.41	341

Table 18: Classification metrics of the BORF-based classifier on the dataset without data augmentation.

To assess the impact of synthetic points on classification performance, the same approach used for the original dataset is applied to the dataset without augmentation. The BORF-based classifier remains superior in this configuration as well. Results, reported in Table 18, show worse performance compared to the augmented dataset in every class and for every metric. The dimensional increase probably provides additional information enabling better class distinction, suggesting that synthetic interpolation captures relevant aspects of temporal patterns. Without data augmentation, the time series may be too short to capture their inherent complexity.