Data Mining II Project:

Analyzing Data Insights from the IMDb Platform

Authors:

Bruno Barbieri, Chiara Ferrara, Ankit Kumar Bhagat

Academic Year 2024/2025

Contents

Introduction

The goal of this report is to illustrate the caracteristics of the given IMDb dataset, and to show key insights that can be obtained from it. In particular, the focus of many of the observations is based on aspects that could be useful for a product's creation and marketing, in order to optimize the chances of success of a product in the market.

1 Data Understanding and Preparation

TODO: distrib graphs The dataset contains around 1.6 million of titles of different types. For each title, the dataset contains information regarding many different aspects. Table 1 lists the initial categorical features.

Feature	**Description**
`originalTitle`	Original title, in the original language (?)
`isAdult`	Whether or not the title is for adult
`canHaveEpisodes`	Whether the title can have episodes
`isRatable`	Whether the title can be rated by users
`titleType`	Type of the title (e.g., movie, tvseries)
`countryOfOrigin`	Countries where the title was primarily produced
`genres`	Genres associated with the title
`regions`	Regions for this version of the title
`soundMixes`	Technical specification of sound mixes
`worstRating` (ordinal)	Worst title rating
`bestRating` (ordinal)	Best title rating
`rating` (ordinal)	IMDB title rating class

Table 1: Initial categorical features of the IMDb dataset

Of the initial categorical attributes, the following were removed:

- `originalTitle`, as it did not provide particularly useful information;
- `isAdult`, as it was almost completely correlated with the *Adult* genre, so a logical OR operation was performed, and the genre only was kept; Interesting to note the fact that in our representation, being that the genres are represented through freq enc, we don't have the info
- `canHaveEpisodes`, as it was completely correlated with the title type being *tvSeries* or *tvMiniSeries*;
- `isRatable`, as it was always true;
- `worstRating` and `bestRating`, as they were always 1 and 10, respectively;
- `rating`, as it was obtainable from the `averageRating` continuous attribute, through a simple discretization.

`soundMixes` was also removed, as it required some domain knowledge to be understood, as well as having issues with the values it contained.
Because of their very similar meaning, `regions` and `countryOfOrigin` were merged through a simple union operation. The resulting feature was then represented trhough frequency encoding on the entire list, as well as counts of the number of countries from each continent. This resulted in eight new features (six continents, one for unknown country codes, and the last for the frequency encoding).
While inspecting the `genre` attribute, it was observed that each record contained up to three genres, listed in alphabetical order—indicating that the order did not convey any semantic information about the title. To represent this information, three separate features were created, each corresponding to one of the genres. These features were encoded using frequency encoding, sorted in descending order of frequency across the dataset. A value of 0 was used to indicate missing genres—either when no genres were present or to fill the remaining slots when fewer than three were available.

The initial numerical features are listed in Table 2.
`endYear` was removed due to it not being meaningful for non-Series titles, and having around 50% of missing values for *tvSeries* and *tvMiniSeries*.
`totalImages`, `totalVideos` and `quotesTotal` were merged through a simple sum operation into a single feature

Feature	Description
`startYear`	Release year of the title (series start year for TV)
`endYear`	TV Series end year
`runtimeMinutes`	Primary runtime of the title, in minutes
`numVotes`	Number of votes the title has received
`numRegions`	Number of regions for this version of the title
`totalImages`	Total number of images for the title
`totalVideos`	Total number of videos for the title
`totalCredits`	Total number of credits for the title
`criticReviewsTotal`	Total number of critic reviews
`awardWins`	Number of awards the title won
`awardNominations`	Number of award nominations excluding wins
`ratingCount`	Total number of user ratings submitted
`userReviewsTotal`	Total number of user reviews
`castNumber`	Total number of cast individuals
`CompaniesNumber`	Total number of companies that worked for the title
`averageRating`	Weighted average of all user ratings
`externalLinks`	Total number of external links on IMDb page
`quotesTotal`	Total number of quotes on IMDb page
`writerCredits`	Total number of writer credits
`directorCredits`	Total number of director credits

Table 2: Initial numerical features of the IMDb dataset

(`totalMedia`) because of their similar semantic meaning, as well as heavy right skewness. The same was true for `awardWins` and `awardNominations`, as well as `userReviewsTotal` and `criticReviewsTotal`, merged with the same procedure into `totalNominations` and `reviewsTotal`, respectively.
castNumber, writerCredits, directorCredits with and without totalCredits; deltacredits
`runtimeMinutes` had a very high number of missing values (add %). Since the feature had high relevance in the domain, it was imputed with random sampling from a interquartile range, separately for each title type.
eventually, add description of the imputation procedure for tasks which involved titleType

2 Outliers

2.1 COF

2.2 Isolation Forest

2.3 ABOD

3 Imbalanced Learning

3.1 Undersampling

3.2 Oversampling

3.2.1 SMOTE

4 Advanced Classification

In this section, classification results are showcased for two target variables: `averageRating` (properly binned into 5 classes), and `titleType` (with 6 classes).
We then applied multiple classification models using the data as preprocessed previously (see Section **??**).
The first target variable is `titleType`, which includes six categories: *movie*, *short*, *tvEpisode*, *tvSeries*, *tvSpecial*, and *video*. The classes are not equally distributed, with *tvSpecial* and *video* being significantly under-represented compared to the others. Entries labeled as *videoGame* were removed from both the training and testing sets, as they were too few to be useful for classification. The remaining categories were merged into broader groups according to the following mapping: *movie* and *tvMovie* were grouped as *movie*, *short* and *tvShort* were grouped as *short*, *tvSeries* and *tvMiniSeries* were grouped as *tvSeries*, while *tvEpisode*, *tvSpecial* and *video* were left unchanged. All feature columns were standardized using a *StandardScaler*. In addition, the

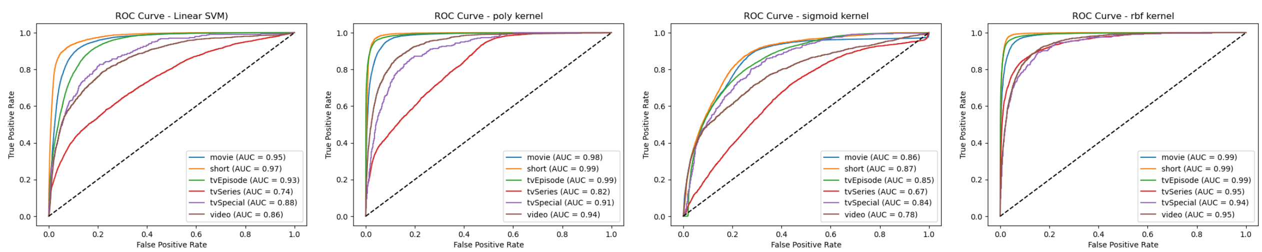
variable `canHaveEpisodes` was removed prior to training, since it provides direct information about the target `titleType` and could therefore introduce data leakage.

4.1 Support Vector Machines

We applied Support Vector Machines (SVM) to the `titleType` classification task. Both linear and non-linear kernels were explored in order to evaluate how decision boundary complexity influences predictive performance. The first experiment used a Linear SVM trained on the full dataset. A grid search with five-fold cross validation was carried out on the parameters $C \in \{0.01, 0.1, 1, 10, 100\}$ and $max\_iter \in \{1000, 5000, 10000\}$. The optimal configuration, with $C = 100$ and $max\_iter = 1000$, achieved a test accuracy of 0.81. While precision and recall were high for majority classes (*movie*, *short*, *tvEpisode*), the classifier failed on *tvSeries*, *tvSpecial*, and *video*, indicating that a linear decision boundary is insufficient for this problem.
Non-linear kernels were then evaluated. A grid search was first performed on a stratified 10% subset of the training set to efficiently explore a wide range of hyperparameters for each kernel, since a full search on the complete dataset would have been computationally prohibitive. For the RBF kernel, C was varied from 0.01 to 1000 and γ between `scale` and `auto`. The polynomial kernel was tested with C from 0.01 to 100, degree 2–4, γ as `scale` or `auto`, and `coef0` 0 or 1. The sigmoid kernel was explored over C 0.01–100, γ `scale`/`auto`, and `coef0` 0 or 1. Remember to fix C!
The best configuration for each kernel, reported in Table 3, was then retrained on the full dataset and evaluated on the test set. Both RBF and polynomial kernels reached approximately 0.90 test accuracy, substantially outperforming the linear baseline and sigmoid. The RBF kernel was selected as the reference non-linear model due to slightly more stable results and improved recall on the under-represented classes.
ROC curves were used to evaluate class separability (Figure 1), showing excellent separation for majority classes, although minority categories remained problematic.
I will change the text and explain the figures better.
To address class imbalance, the RBF kernel was retrained with `class_weight=balanced`, which penalizes misclassification of under-represented classes. This model reached a slightly lower overall accuracy of 0.84, but recall for *tvSpecial* and *video* improved, providing a more equitable classification across categories. Confusion matrices (Figure 2) illustrate that *tvSpecial* and *video* ... Analysis of the support vectors confirmed this effect. In the unbalanced RBF, nearly all points of minority classes became support vectors, while in the balanced model the total number of support vectors increased and was more evenly distributed across classes, indicating a more complex but fairer decision function.
Table 3 summarizes the main results, including the parameters used for each kernel and the corresponding test performance.

Table 3: Comparison of SVM models on the IMDb classification task.

Model	Best Params (main)	Test Accuracy	Macro F1-score
Linear SVM	$C = 100$, $max\_iter = 1000$	0.81	0.45
RBF kernel	$C = 10$, $\gamma =$ scale	0.90	0.64
Polynomial kernel	$C = 10$, degree=3, $\gamma =$ auto	0.90	0.64
Sigmoid kernel	$C = 0.1$, $\gamma =$ auto	0.65	0.36
RBF (balanced)	$C = 10$, $\gamma =$ scale, balanced	0.84	0.65

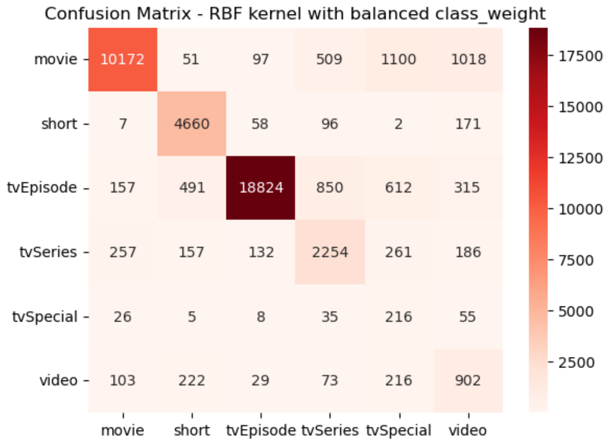


(a) Linear SVM (b) Polynomial kernel (c) Sigmoid kernel (d) RBF kernel

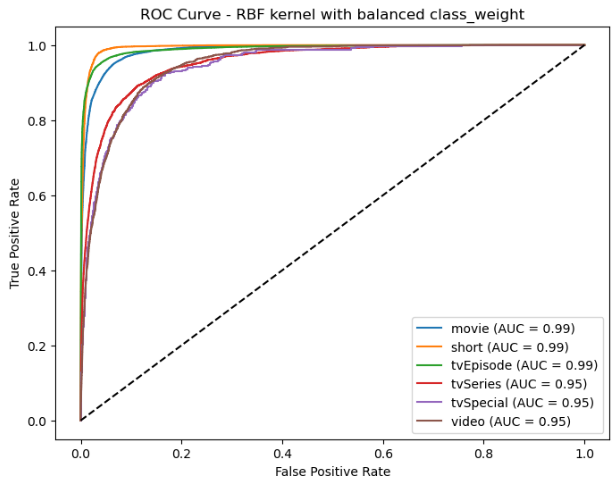
Figure 1: ...

4.2 Ensemble methods

Boosting and Random Forest models were trained on the classification, while being optimized via Stratified Randomized Search with 5-fold cross-validation over a predefined hyperparameter space.



(a) Confusion Matrix RBF balanced



(b) ROC RBF balanced

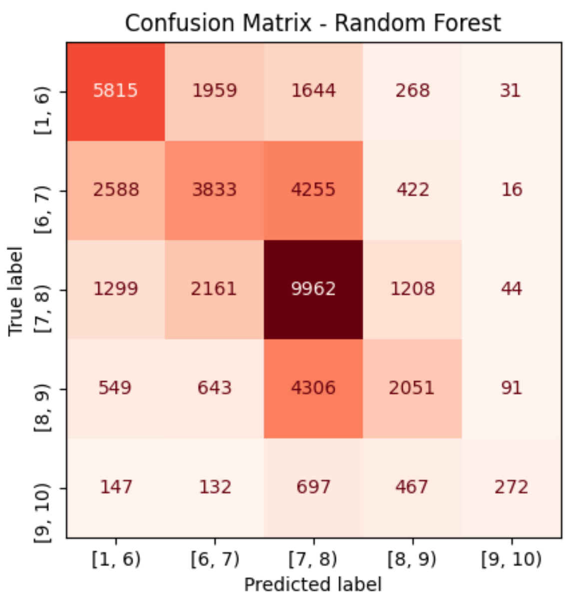
Figure 2: ...

For the `averageRating` classification task, the best hyperparameters for the Random Forest model were: `n_estimators=42`, `max_depth=19`, `min_samples_split=4`, `min_samples_leaf=3`, `max_features=0.74`, `criterion='gini'`, `class_weight=None`.
For the AdaBoost model, the best hyperparameters were: `n_estimators=56`, `learning_rate=0.47`, `estimator__max_depth=`
`estimator__min_samples_split=16`, `estimator__min_samples_leaf=16`.
The table below summarizes the classification report for both models.

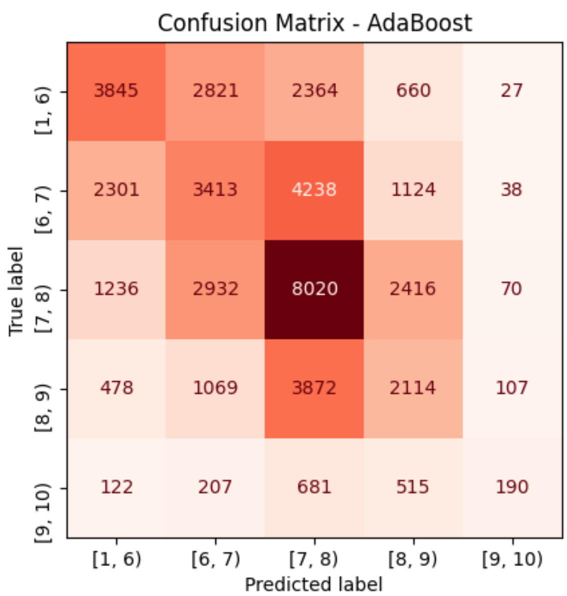
Model	Accuracy	Macro Precision	Macro Recall	Macro F1-score
Random Forest	0.49	0.51	0.41	0.42
AdaBoost	0.39	0.40	0.33	0.34

Table 4: Performances of the models on the `averageRating` classification task.

The Random Forest model outperforms AdaBoost in all metrics. The biggest difference is found in the recall. Figure 5a and Figure 5b show the confusion matrices for the models.



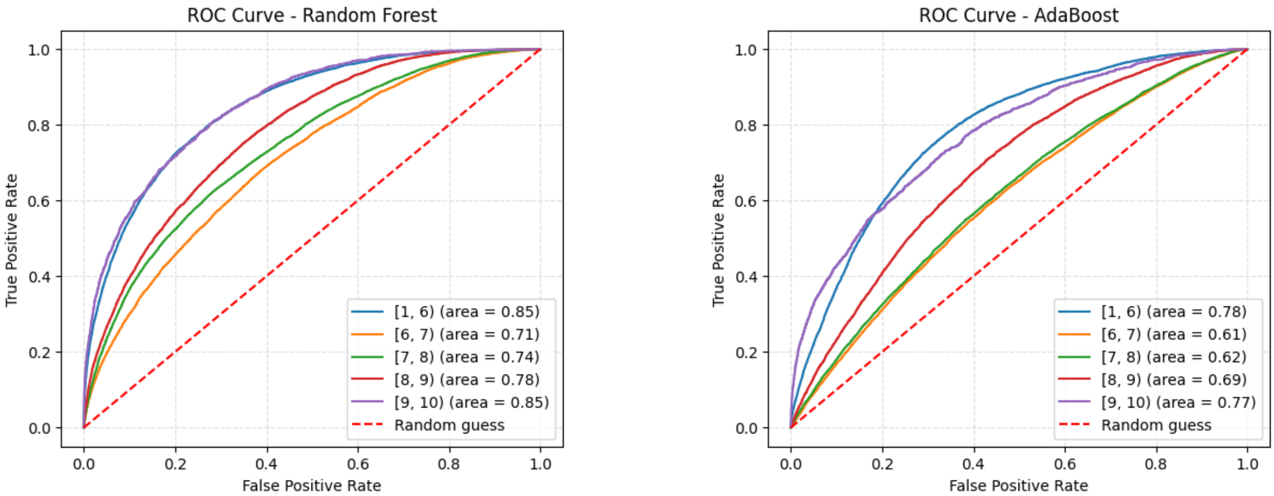
(a) Confusion Matrix - Random Forest



(b) Confusion Matrix - AdaBoost

Figure 3: Confusion matrices for the ensemle models on the `averageRating` classification task.

From these, it can be seen why the recall is much lower for AdaBoost, with regards to Random Forest: the former tends to classify less aggressively as the most represented class. It's also worth noting that Random Forest tends to assign most of the misclassifications to the adjacent classes, while AdaBoost spreads them more evenly across all classes.
Figures 6a and 6b show the ROC curves for the two models. From these representations, it can be seen



(a) ROC Curve - Random Forest

(b) ROC Curve - AdaBoost

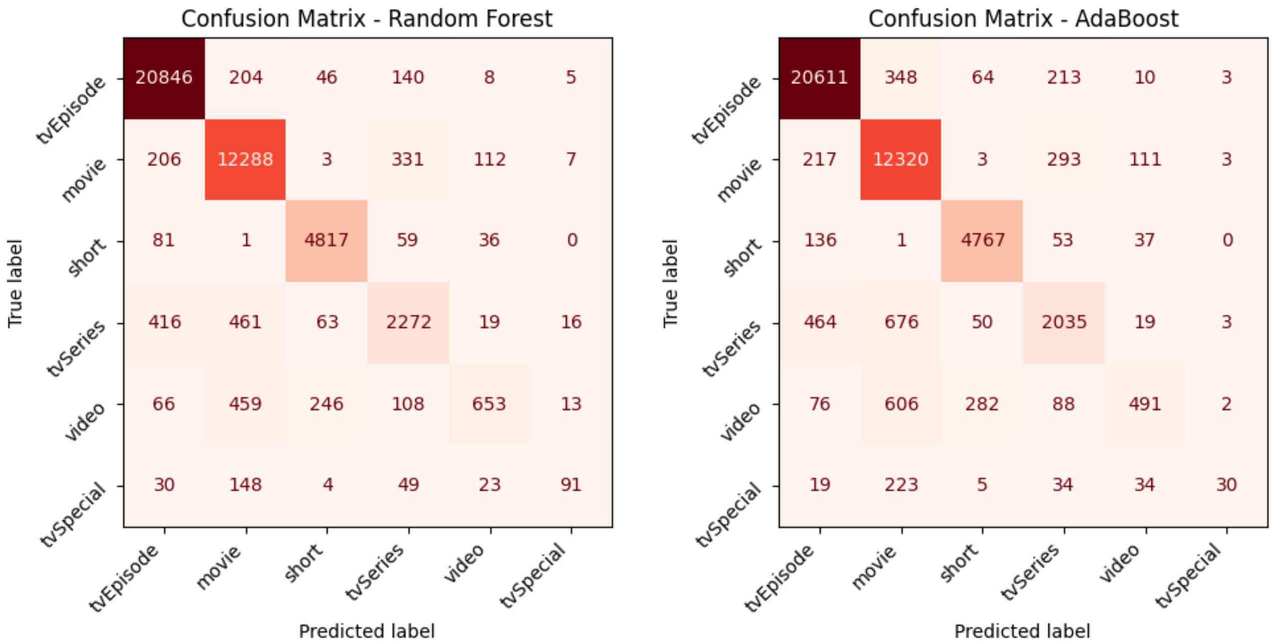
Figure 4: ROC curves for the ensemble models on the `averageRating` classification task.

that AdaBoost has poorer performances in all classes, but especially struggles with the under-represented ones, which lead to the biggest difference in Area Under The Curve (AUC).
On the `titleType` classification task, the best hyperparameters obtained for the Random Forest model were: `n_estimators=42`, `max_depth=19`, `min_samples_split=4`, `min_samples_leaf=3`, `max_features=0.74`, `criterion='gini'`, `class_weight=None`.
For the AdaBoost model, the best hyperparameters were: `n_estimators=56`, `learning_rate=0.47`, `estimator__max_depth=` `estimator__min_samples_split=16`, `estimator__min_samples_leaf=16`.
The table below summarizes the classification report for both models.

Model	Accuracy	Macro Precision	Macro Recall	Macro F1-score
Random Forest	0.92	0.84	0.71	0.75
AdaBoost	0.90	0.82	0.65	0.68

Table 5: Performances of the models on the `titleType` classification task.

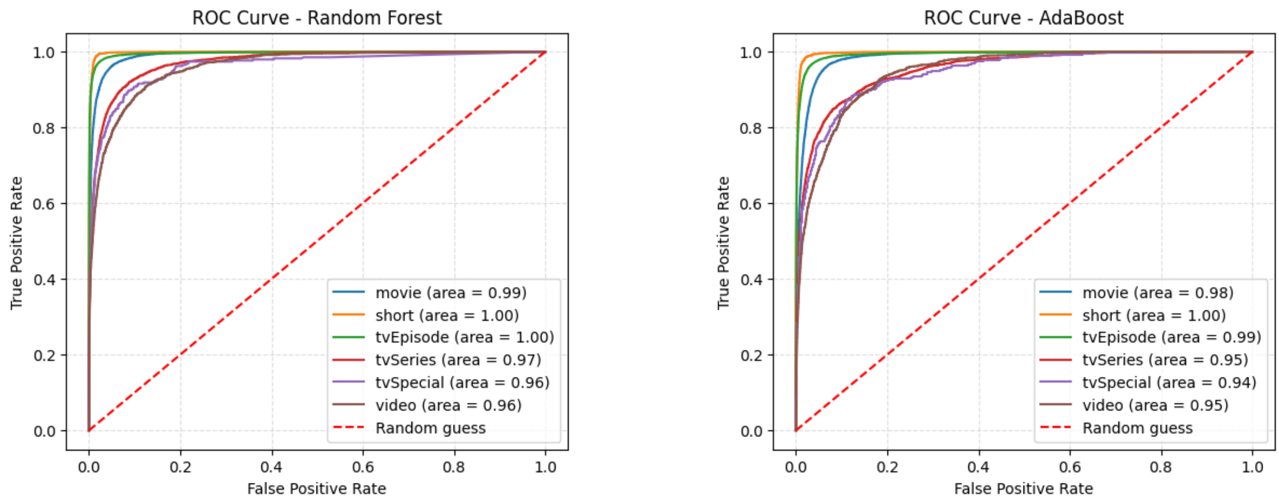
Figure 5a and Figure 5b show the confusion matrices for the models.



(a) Confusion Matrix - Random Forest
(b) Confusion Matrix - AdaBoost

Figure 5: Confusion matrices for the ensemle models on the `titleType` classification task.

Both models perform well on the first four classes, while struggling with /texttttvideo and /textttttvSpecial, which are the most under-represented classes. In general, the two models show similar performances, with Random Forest generally outperforming AdaBoost by a slight margin. Contrary to the results, the models base their decisions on different feature importances: while Random Forest assigns over half of the importance to `runtimeMinutes`, AdaBoost spreads the importance evenly across multiple features, with the top being `runtimeMinutes` with around 15%.
The ROC curves for the two models are shown in Figure 6a and Figure 6b.



(a) ROC Curve - Random Forest
(b) ROC Curve - AdaBoost

Figure 6: ROC curves for the ensemble models on the `titleType` classification task.

Again, similar performances are observed. The biggest difference seems to be found in the under-represented classes, which seem to have a bigger difference in Area Under The Curve (AUC).

4.3 Neural Networks

4.4 Model Comparison

5 Advanced Regression