University of Pisa

Data Mining and Machine Learning

Project Documentation

CinemAI

Marco Bologna
Andrea Ruggieri

A.Y. 2023-2024

Abstract

According to Gower Street Analytics, film industry is expected to generate over $32.3 billion in revenue in 2024. Despite the increasing use of streaming services and experiencing a decline compared to 2023, which saw revenues of $33.9 billion also thanks to blockbusters like Barbie and Oppenheimer, the cinema industry is likely to continue to generate significant numbers, remaining (for at least another year) as one of the most lucrative industries in every country. Given this situation, professionals are continuously looking for the highest possible marginal profit for movie's revenue, so project's aim was provide an as complete as precise study of industry available data helping production company answer the question:

Is it possible to predict the economic success of a movie before hitting theaters?

Project involved different Machine Learning and Data Mining techniques to a dataset containing movie's features to predict movie's revenue, exploiting regression models for a continuous value, then clustering for revenue discretization and finally classification for revenue rank-based classes.

Code developed, including presentation, is already available on following GitHub page:

CinemAI GitHub link

Since some datasets were too heavy according to GitHub space available, following link provide access to the missing ones from GitHub project's page:

Google Drive link

Contents

1 Dataset overview

This section helps reader understanding data construction process and features obtained, starting from multiple raw dataset merged into single one on which pre-processing techniques were applied as explained in next chapter.

1.1 Introduction

This section introduce dataset used for studies made:

- IMDB dataset
  Founded on Kaggle dataset is obtained via web scraping on IMDB website, a kind-of social network for movies rating including following film features:

Feature	Description
imdb_title_id	movie id on IMDB website
title	movie title
original_title	movie original title
year	year in which film was released or expected to be released
status	year in which movie was released
date_published	date in which movie was released
genre	movie genre
duration	screenplay movie time
country	country in which movie was shot
language	languages spoken
director	responsible of movie's direction
writer	movie's script writer(s)
production_company	production company which produced movie
actors	actors involved in movie
description	movie description
avg_vote	average movie's vote according to IMDB community
votes	vote count
budget	budget spent producing movie
usa_gross_income	US revenue generated by movie
worldwide_gross_income	worldwide revenue generated by movie
production_companies	production company produced movie
reviews_from_users	number of reviews made by users
reviews_from_critics	number of reviews made by critics

Table 1: IMDB dataset description

- TMDB (The Movie DataBase)
  Founded on Kaggle website too, is one of most common used data source, this dataset was made by scraping The Movie DataBase website from which only features also included in IMDB were maintained and the others were discarded (e.g. vote, average_vote, etc.)

Next subsections helps for both increasing number of entries with useful values for budget and including prizes information related to actors, directors and writers.

1.2 Data integration

Since this point, world_wide_gross_income is considered as *revenue*.

1.2.1 Revenue and Budget

	All	Without Budget	Without Revenue
TMDB	1030555	981645	1017647
IMDB	81273	58746	70944

Table 2: Comparazione tra IMDB e TMDB

Preliminary analysis just made points out many useless entries, including null and non zero values for revenue and budget, so since these are the most important ones for the whole project Data integration was first step used for number of correct values of revenue and budget increasing: starting from IMDB dataset, this was merged with the other one obtaining a complete raw dataset with 20825 entries.

1.2.2 Inflation and value's adjustement

Since database contains movies made from 1930 to 2019, with revenue and budget values expressed in both dollars and foreign currencies, a conversion into unique dollar currency and a non-inflated value was mandatory.

1. Find table with Consumer Price Index for every country in raw dataset. Since value was reported in different format depending on different countries (e.g. Japan has monthly updates, Germany yearly and so on), all values was converted into standard yearly format, using mean value for the ones with different format

2. An append file was used to convert currency id of raw database and country in CPI file into a single format, since single ids were associated to different countries (e.g. USD not in US only) and same country were expressed using different format (e.g. Czechia and Checz Republic)

3. Once a single value of CPI was obtained for every year (from 1930 to 2023) for every country in raw database, following formula was applied computing right value of movie revenue:

$$\text{Intermediate Value} = \frac{\text{Revenue} * \text{CPI\_2023}}{\text{CPI\_year\_of\_movie\_release}} \tag{1}$$

4. Once Intermediate value was obtained the conversion into USD was need as final step

Be mind that budget didn't need same conversion process since all column values were already expressed in dollars, only requiring inflation adjustement from past year to now days.

1.2.3 Number of films

Due to the lack of movie information available and the impossibility of a-priori knowledge of features, *actor* feature in the dataset containing the list of actors involved in the movie was used to add a column 'movies made before', which values represents number of movies made before by the whole cast. This number works as kind-of as popularity index, incremented for every film made by every actor: considering for example "Critters 3", DiCaprio brings none points for related column since was his first film, but the star will bring a +1 for the next one, "Poison Ivy", a +2 for "What's eating Gilbert Grape", the third, and continuous for next ones.

1.2.4 Prizes nomination and win

For same reasons as before, lack of movie's information and hope of adding useful features, the idea of including prizes information on the original dataset was developed as it follows:

1. Main awards recognized by the whole cinema industry including Oscar, Golden Globe, Emmy and BAFTA were founded on Kaggle

2. Accurate pre-processing phase in which null values were removed

3. Useless nominations and related wins, for example all including 'Best Film' which award only the product itself and not the actor, writer or director involved, were removed

4. Information obtained were merged with the original dataset, adding following columns:

Award/Nomination	**Value**
dir_emmy_nom	# Emmy nomination obtained by director
dir_emmy_won	# Emmy won by director
writer_emmy_won	# Emmy nomination obtained by writer
writer_emmy_nom	# Emmy won by director
act_emmy_nom	# Emmy nomination obtained by whole cast
act_emmy_won	# Emmy won by whole cast
dir_oscar_nomination	# Oscar nomination obtained by director
dir_oscar_won	# Oscar won by director
writer_oscar_nomination	# Oscar nomination obtained by writer
writer_oscar_won	# Oscar won by writer
cast_oscar_nomination	# Oscar nomination obtained by whole cast
cast_oscar_won	# Oscar won by whole cast
dir_globe_nomination	# Golden Globe nomination obtained by director
dir_globe_won	# Golden Globe won by director
cast_globe_nomination	# Golden Globe nomination obtained by whole cast
cast_globe_won	# Golden Globe won by whole cast
BAFTA_act_nom	# BAFTA nomination obtained by whole cast
BAFTA_act_won	# BAFTA won by whole cast
BAFTA_dir_nom	# BAFTA nomination obtained by director
BAFTA_dir_won	# BAFTA won by director
BAFTA_writer_nom	# BAFTA nomination obtained by director
BAFTA_writer_won	# BAFTA won by writer

Table 3: Overview of Awards and Nominations

Information obtained must be considered according to movies made before with an idea similar to the number of movies made by an actor seen above: if a director was awarded by Golden Globe in 2012, all previous movies entries has zero on both dir_globe_nomination and dir_globe_won, but a +1 for all movies made after considered year.

1.2.5 Production Companies

Current subsection is dedicated to Production Company column, which contains the name of the production company made the movie presenting one main issue for which the feature has too many possible unique values due to the reality model itself: one big production company, for example Disney, can have multiple subsidiaries, as Walt Disney Studios, Walt Disney Pictures, Disney and Co., ESPN, Marvel Entertainment, Marvel Enterprises, Marvel Studios, Pixar Animation Studios and Touchstone Pictures. According to Wikipedia, subsidiaries were manually merged on single value, giving 'Disney' to all companies mentioned above. Even bringing unique values of the column reduced the number of unique ones, still too much were left so the solution adopted was the avoidance of the non big companies in one single possible valued labeled as 'Other', maintaining the only top companies.

2 Preprocessing

Firstly, following features were removed:

- *imdb_title_id, title, original_title*: instances identifiers, useless information for further studies
- *description*: useless for model training tasks
- *average_vote, votes, reviews_from_users, reviews_from_critics*: numerical information cannot be used since a-posteriori ones
- *usa_gross_income*: feature discarded since the presence of *world_wide_gross_income* and many movies not distributed in US

2.1 Handling Missing Values

Missing values were handled with different strategies depending on the column:

- **Budget**: The *converted_budget* column had 119 missing values out of 20'825 rows. Given the relatively small number of missing entries, these rows were removed maintaining the integrity of financial data without significantly impacting the overall dataset.
- **Production Company**: After feature changes explained in sub-section 1.2.5, *Production Company* 629 missing values left were transformed into 'Other' value.
- **Country and Language**: There were 5 rows where both the *Country* and *Language* columns had missing values.
  These rows were removed from the dataset due to the lack of critical information, ensuring that analyses are not compromised by incomplete entries. For the additional 100 rows where only the *Language* was missing, the same approach used for the Production Company was applied: missing values were filled with "Other." This strategy was used to retain these entries while still addressing the missing data issue.

2.2 Handling Categorical features

Dataset includes several categorical features, as *Country*, *Language*, *Genre*, *Release Date* and *Production Company*, which required careful handling to

ensure effective use in following analysis and modelling.
Since *Country* feature contains hundreds of unique values with a highly imbalanced distribution, the top 15 most frequent countries were retained, as these accounted for the majority of observations.

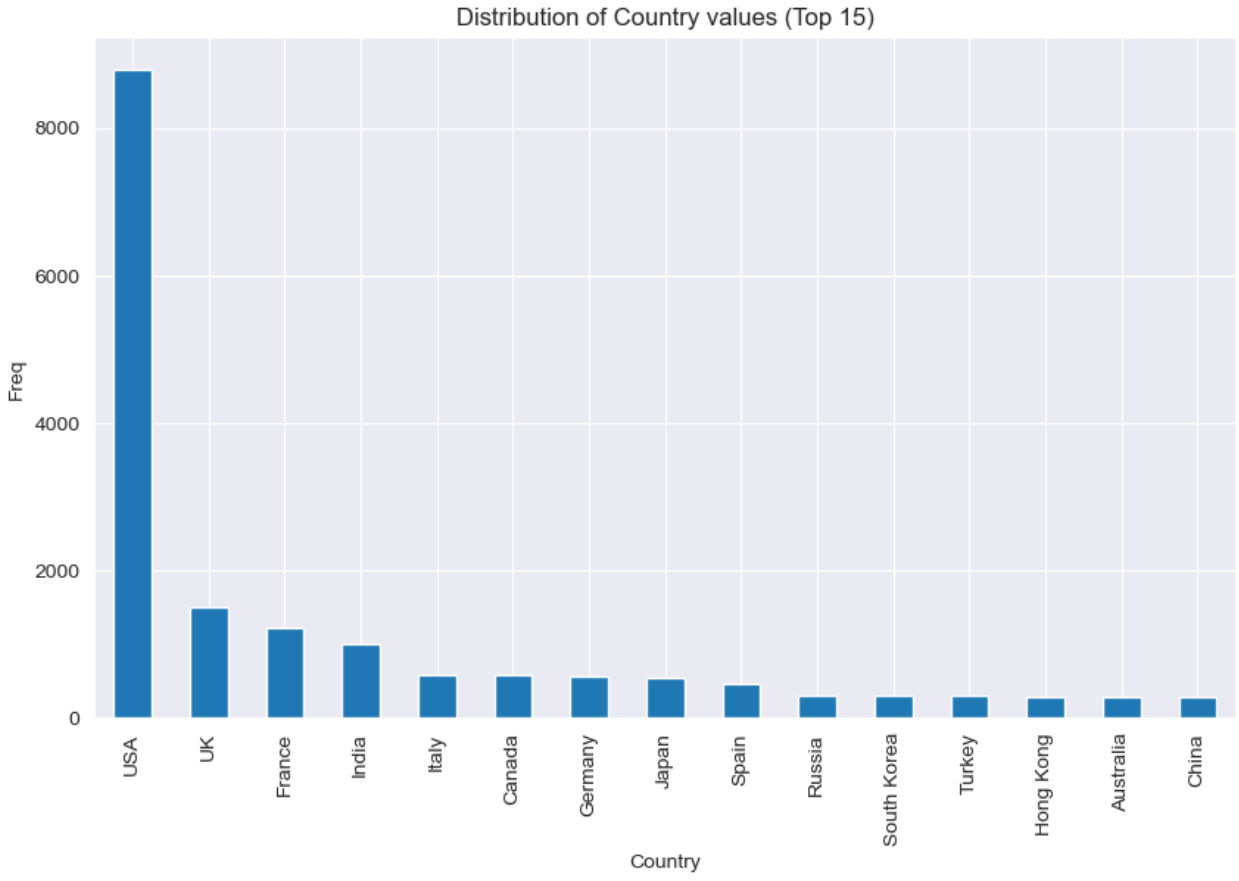

Figure 1: Country distribution

Remaining countries, which had -relatively- few entries, were grouped under a single category labeled "Other" simplifying dataset while still capturing the most of country information.
Same strategy was applied to *Language* and *Production Company*, reducing complexity and ensuring less frequent entries do not disproportionately impact the analysis.

Release Date feature was transformed to extract the month of release since can be a significant predictor of a movie's performance, as different period can have different levels of audience attendance (e.g. most of blockbuster hits theaters during Christmas Holidays). This new categorical feature, *release_month*, captures seasonal trends and helps in forecasting revenues.

2.2.1 One-Hot Encoding

Following these changes, all categorical variables were treated using one-hot encoding, a technique converting categorical variables into a series of binary columns, each one representing a possible category and assigning 1 if category is present, 0 otherwise. Especially, for the Genre feature, one-hot encoding effectively manages movies with multiple genres by assigning them to multiple binary columns, ensuring that all relevant genre information is captured.

2.3 Feature Engineering

Through feature engineering, we aimed to enhance the dataset's predictive power while simplifying its structure. Leveraging statistical tests and correlation analysis, redundant features were identified and the most informative ones were selected for further analysis.

2.3.1 Categorical Features

To assess the relationship between categorical variables, **Chi-Square test** was chosen and as expected, the test revealed a high degree of similarity between the Country and Language variables.

$$\text{Chi-square} = \sum \frac{(\text{Observed Frequencies} - \text{Expected Frequencies})^2}{\text{Expected Frequencies}} \tag{2}$$

Given this redundancy and the inherent complexities associated with handling multiple highly correlated features, *Country* feature was removed from the dataset.

2.3.2 Numerical Features

Correlation analysis was performed also on numerical features, observing high correlations within the section related to awards between nomination and related winners.

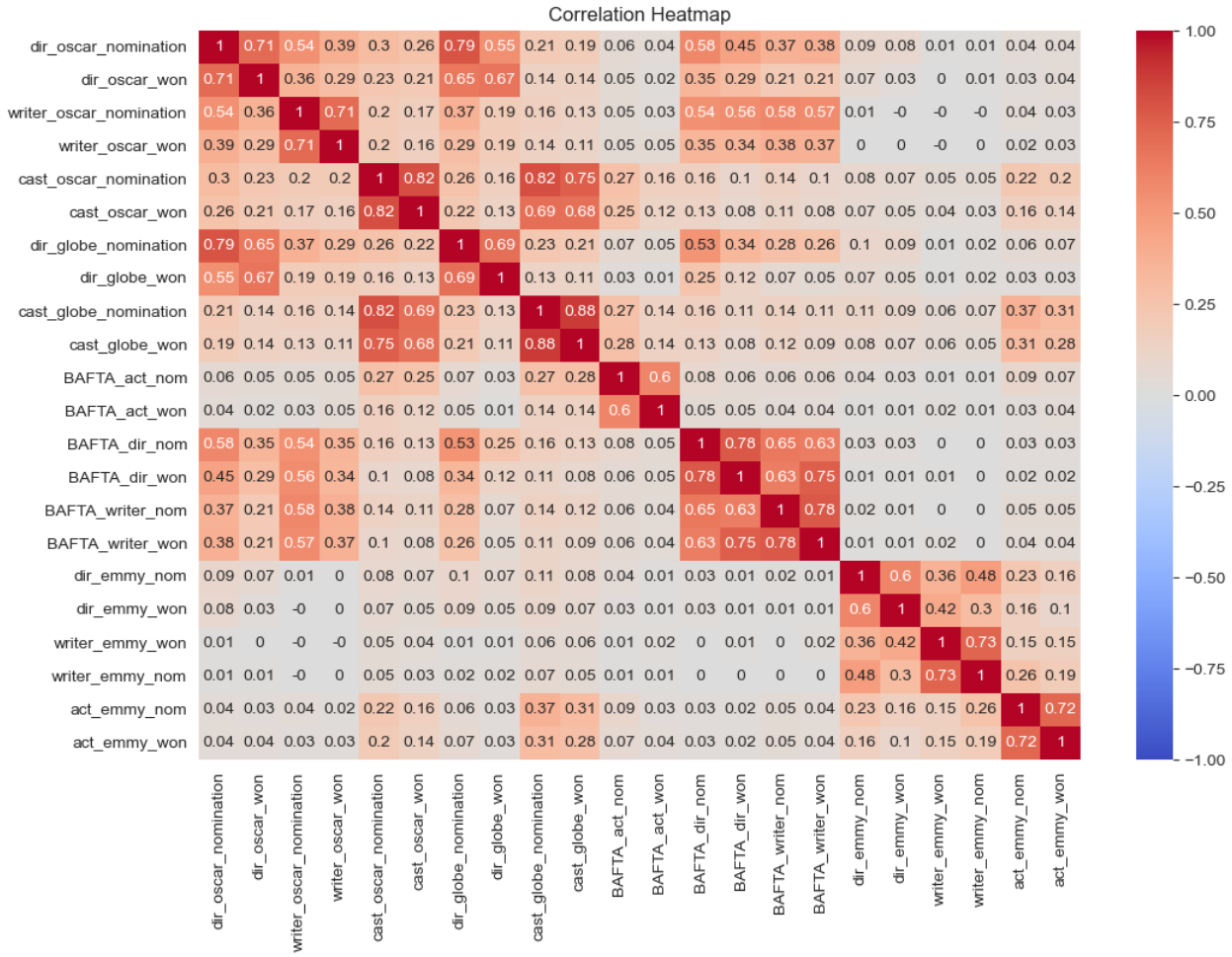

Figure 2: Correlation Matrix focuses on prizes features

Consequently, only nominations related columns were kept, as they provide valuable information while mitigating multicollinearity concerns.

Moreover, important correlation between Oscars and Golden Globe awards was noticed so the columns for which more films had non-zero values was selected, offering greater variability and, consequently, more predictive power in subsequent analyses.

3 Regression analysis

Considering a numerical value as target, this chapter presents different regression models testing phase to predict the revenue of movies using the dataset presented above. Following subsections presents the model selection and evaluation procedure for the six different ones tested:
Decision Tree, *Random Forest*, *AdaBoost*, *GradientBoosting*, *K-Nearest Neighbours* and *ElasticNet*.

3.1 Cross-Validation and Hyperparameters tuning

Before every model test step, dataset was divide into training test, for model training phase, and test set, for model test and evaluation phase, maintaining the 80% for the first and the 20% for the latter. For each one of regression model tested, flat cross-validation technique was used, considering a 10-fold cross-validation on the training set on which Flat cross validation algorithm was applied.
Flat cross validation algorithm, considering K=10:

1. For K=1, .., 10
   (a) Splits training set into K subsets, obtaining $S_1, S_2, .., S_{10}$
   (b) Trains model using K-1 subsets
   (c) Test model using K-left subset
2. After K times, final result obtained is the mean of K results obtained

Using flat cross validation, result obtained was mean result and reduced the probability of chance inflation. Since search of best model is biased by initial parameters configuration, a **Grid Search** was implement for hyperparameters tuning, working through multiple combination and taking best result only for each model. Then, model was trained on the entire training set and subsequently tested it on related test set using this two-step process which ensures well tuning and suitable evaluation method.

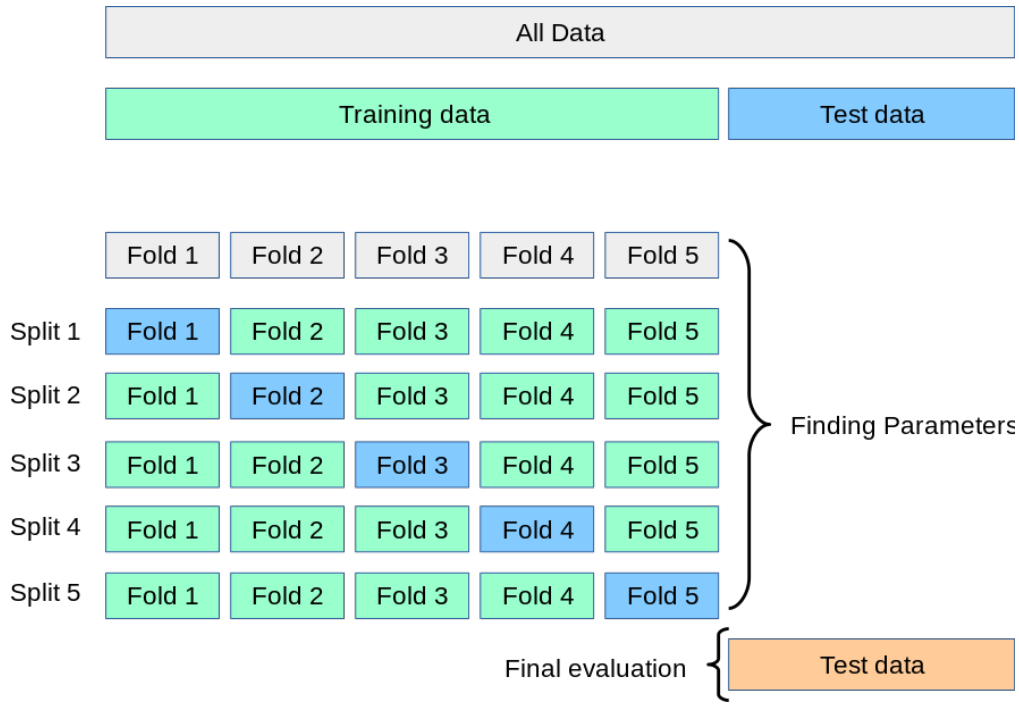

Figure 3: Flat Cross Validation

3.2 Pipeline for Each Step

At each step, the pipeline consisted of the following stages:

- **Preprocessing**: For preprocessing, we used a *Standard Scaler* for numerical features. Standard scaling involves transforming the data to have a mean of zero and a standard deviation of one. This step is crucial for ensuring that numerical features are on the same scale, which is particularly important for regression models sensitive to the scale of input features.

- **Feature Selection**: We applied feature selection techniques to identify the top 20 features that most significantly contribute to predicting the revenue. Feature selection helps in reducing the dimensionality of the dataset, improving model performance, and reducing overfitting by eliminating less informative features.

- **Model Training and Testing**: Finally, we trained the regression model using the selected features and the preprocessed data. The model was then tested on the test set to evaluate its performance.

All this steps were executed for each one of models further explained.

3.3 Regression Models

For model evaluation, four main parameters were considered:

- **Mean Squared Error (MSE):** square of the mean of difference between observed and predicted values, representing error under estimation where higher values represent higher errors

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\text{observed}_i - \text{predicted}_i)^2 \tag{3}$$

- **Root Mean Squared Error (RMSE):** is the squared root of MSE; squared root allows pay more attention on biggest error, providing the error size estimation

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{4}$$

- **Mean Absolute Error (MAE):** is the mean of absolute value of errors, representing error without weighgeting highest errors so is less sensitive to outliers

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\text{observed}_i - \text{predicted}_i| \tag{5}$$

- **R-Squared (R^2):** represents the proportion of explained variance over the whole explained variance of the problem, so basically *how much of the problem was captured*; highest values (~ 1) mean the model can efficiently understand data and predict without residuals, low values instead (~ 0) means model can understand independent variables so basically value predicted is mean value

$$R^2 = 1 - \frac{\text{Sum of Squares of Residuals}}{\text{Total Sum of Squares}} \tag{6}$$

3.3.1 Random Forest

As suggested by the name itself, Random Forest regression model creates multiple decision tree via bagging, so using bootstrap aggregating sampling technique. During tree construction step, model only considers a random subset of features and final prediction is made considering the mean of all predictions made by the whole forest. Main advantage of the model is feature's weight estimation, making overfitting less dangerous and giving more accurated weighted predictions.

Parameters	**Possible Values**
regressor__n_estimators	[50, 300]
regressor__max_depth	[4, 10]
regressor__min_samples_split	[2, 10]
regressor__min_samples_leaf	[1, 4]
regressor__max_features	['sqrt', 4]
regressor__bootstrap	[True, False]
regressor__criterion	['squared_error', 'poisson']

Table 4: Parameter Grid for Random Forest Regressor

Parameter	**Best Value**
Bootstrap	True
Criterion	Squared Error
Max Depth	None
Max Features	sqrt
Min Samples Leaf	1
Min Samples Split	10
N_Estimators	500
Random State	42

Table 5: Best Parameters for Random Forest Regressor

Metric	**Test Set Results**	**Train Set Results**
Final RMSE	208,057,585.44	163,416,971.14
Final MAE	82,534,253.37	58,024,217.05
Final MSE	4.329×10^{16}	2.671×10^{16}
Final R^2	0.418	0.684

Table 6: Comparison of Test and Train Set Results

3.3.2 Decision Tree

Decision Tree model predicts target value with simple decision rules inferred from the data features, learning from data to approximate a curve with a set of if-then-else decision rules: the deeper the tree, the more complex the decision rules and the fitter the model.

Parameters	**Possible Values**
regressor__max_depth	[None, 10, 20, 30]
regressor__min_samples_split	[2, 5, 10]
regressor__min_samples_leaf	[1, 2, 4]

Table 7: Parameter Grid for Decision Tree Regressor

Parameter	**Best Value**
regressor__max_depth	10
regressor__min_samples_leaf	4
regressor__min_samples_split	2

Table 8: Best Parameters for Decision Tree Regressor

Metric	**Test Set Results**	**Train Set Results**
Final RMSE	234,341,231.22	192,282,060.24
Final MAE	88,080,157.98	71,758,403.95
Final MSE	5.492×10^{16}	3.697×10^{16}
Final R^2	0.262	0.562

Table 9: Comparison of Test and Train Set Results

3.3.3 AdaBoost

Adaptive Boosting algorithm allow the construction of a regression model creating k models in k rounds, where for each round a model M_k is developed using a subset of tuples obtained via sample with replacement and probability of being extraced is weighted by the last time tuple was extraced: once a tuple's value is correctly predicted the probability of being selected decreases, otherwise increases, giving more attention on misclassified tuples.

Parameters	Possible Values
regressor__n_estimators	[50, 100, 150]
regressor__learning_rate	[0.01, 0.1, 1.0]
regressor__loss	['linear', 'square', 'exponential']

Table 10: Parameter Grid for AdaBoost Regressor

After grid search implementation, following configuration was selected as the one provided best results:

Learning Rate	Loss	N_Estimators
0.01	Exponential	50

Table 11: Best Parameters for AdaBoost Regressor

Metric	Test Set Results	Train Set Results
Final RMSE	213,078,427.55	214,334,476.40
Final MAE	85,459,858.06	81,809,219.25
Final MSE	4.540×10^{16}	4.594×10^{16}
Final R^2	0.390	0.456

Table 12: Comparison of Test and Train Set Results

3.3.4 Gradient Boosting

Gradient Boosting regression model may be powerful considering also nonlinear relationship between model target and features and dealing with missing values, outliers, and high cardinality categorical values without any special treatment.

Parameters	Possible Values
regressor__n_estimators	[50, 100, 150]
regressor__learning_rate	[0.01, 0.1, 1.0]
regressor__max_depth	[3, 5, 7]
regressor__min_samples_split	[2, 5, 10]
regressor__min_samples_leaf	[1, 2, 4]

Table 13: Parameter Grid for Gradient Boosting Regressor

Parameter	Best Value
Learning Rate	0.1
Max Depth	3
Min Samples Leaf	1
Min Samples Split	5
N_Estimators	50
Random State	42

Table 14: Best Parameters for Gradient Boosting Regressor

Metric	Test Set Results	Train Set Results
Final RMSE	218,257,105.34	199,650,305.70
Final MAE	83,351,584.60	77,359,532.15
Final MSE	4.764×10^{16}	3.986×10^{16}
Final R^2	0.360	0.528

Table 15: Comparison of Test and Train Set Results

3.3.5 ElasticNet

ElasticNet main feature is merging two ways of linear regression principles: Ridge, for correlated values reducing risk of input variations bring high output variations, and Lasso, weighting features simplifying model.

Parameters	Possible Values
regressor__alpha	[0.1, 0.5, 1.0]
regressor__l1_ratio	[0.1, 0.5, 0.9]

Table 16: Parameter Grid for ElasticNet Regressor

Parameter	Best Value
regressor__alpha	0.1
regressor__l1_ratio	0.9

Table 17: Best Parameters for the ElasticNet Regressor

Metric	Test Set Results	Train Set Results
Final RMSE	255,072,458.63	273,842,741.70
Final MAE	119,316,715.33	118,242,345.18
Final MSE	6.506×10^{16}	7.499×10^{16}
Final R^2	0.126	0.112

Table 18: Comparison of Test and Train Set Results

3.3.6 K-Nearest Neighbour

K-NN regression model is one of the easiest way to implement regression and classification, using distance (e.g. Manhattan, Euclidean) from a point and nearest neighbours and giving as output mean values. Even thought model is simple, giving right input parameters isn't easy so a complex grid search was implemented:

Parameters	Possible Values
regressor__n_neighbors	[3, 5, 7, 9, 11]
regressor__weights	['uniform', 'distance']
regressor__algorithm	['auto', 'ball_tree', 'kd_tree', 'brute']
regressor__leaf_size	[20, 30, 40, 50]
regressor__p	[1, 2]

Table 19: Parameter Grid for KNN Regressor

Parameter	Best Value
regressor__algorithm	kd_tree
regressor__leaf_size	30
regressor__n_neighbors	11
regressor__p	2
regressor__weights	uniform

Table 20: Best Parameters for the KNN Regressor

Metric	Test Set Results	Train Set Results
Final RMSE	236,898,685.34	233,574,372.14
Final MAE	89,388,114.62	80,122,849.92
Final MSE	5.612×10^{16}	5.456×10^{16}
Final R^2	0.246	0.354

Table 21: Comparison of Test and Train Set Results

3.4 Result's evaluation

- Worst result was obtained with ElasticNet, which seemed too simple to handle multiple features in small dataset
- Random Forest was not evenly considered due to the high difference between train and test results, highlighting overfitting problem already discussed
- Best result was obtained using AdaBoost classifier
- One possible reason why Decision Tree and AdaBoost models have better performance may be the weighing, since first ones improve performance over multiple runs and the latter typically have worst results in presence of noise and outliers. All in all, these models give better results using simple models in an iterative an weighted way, reducing bias and focusing over complex dependencies and though errors

Since even best result was not satisfying due to low R^2 value, the problem was transformed from revenue value prediction into range of possible revenue prediction using classification technique, as described in next chapter.

Models	**Train Set**				**Test Set**			
	R^2	**MAE**	**RMSE**	**MSE**	**R^2**	**MAE**	**RMSE**	**MSE**
Random Forest	0.684	580242	1634697	2.67	0.418	855342	2080575	4.39
Decision Tree	0.562	717584	1922820	3.69	0.262	880801	2343412	5.49
AdaBoost	0.456	818092	2143344	4.59	0.39	854598	2130784	4.54
Gradient Boosting	0.528	773595	1996503	3.98	0.36	833518	2182571	4.76
KNN	0.354	801228	2335743	5.45	0.246	893881	2368986	5.16
ElasticNet	0.11	1182423	2738427	7.5	0.126	1193167	2550724	6.5

Table 22: Model's Performance Comparison

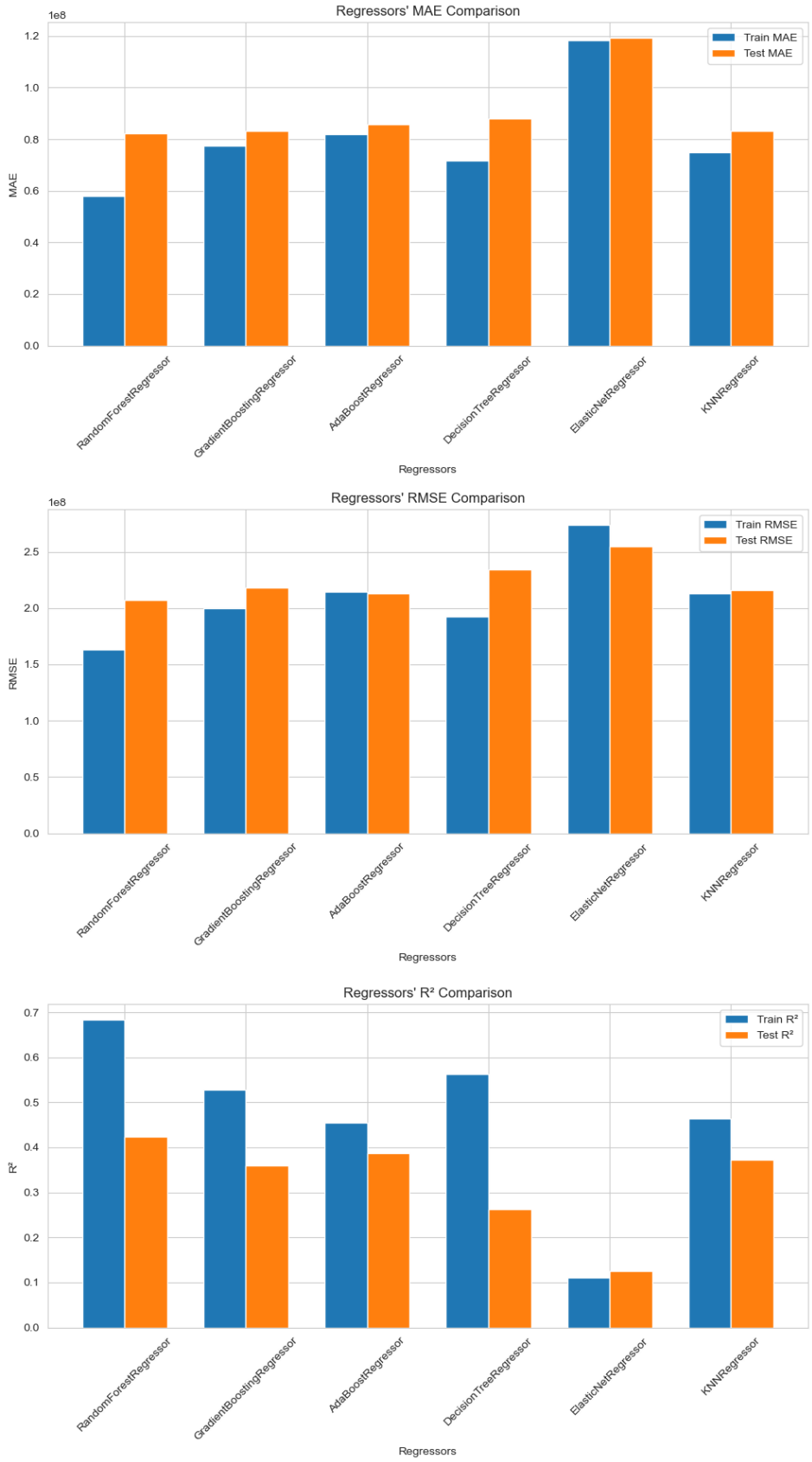

Figure 4: Regressors MAE, RMSE and R^2

4 Classification Analysis

Since the objective of the study was helping production companies for an a-priori prediction of the possible revenue and non suitable values of R^2 and were Mean Squared Error obtained from regression models, classification was the supervised learning technique used to change the problem from predicting revenue value into predicting class revenue, a kind-of possible range in which future revenue can be.
Preliminary step was the discretization of the target variable in order to transform the revenue in classes. Following subsection contains models used, presenting model itself and results obtained including final considerations.

4.1 Data discretization

According to results over regression models presented before, an approach to classification models was thought as possible solution for changing the task to predict movie revenue to predict class of film according to revenue. Since Classification is a supervised learning techniques and no labels were provided in the original dataset, different data discretization ways were taken for changing revenue into discrete value, trying to obtain multiple classes based on revenue.
First way of data discretization was implemented studying distribution of revenue, extracting homogeneous ranges according to values and number of instances per class.

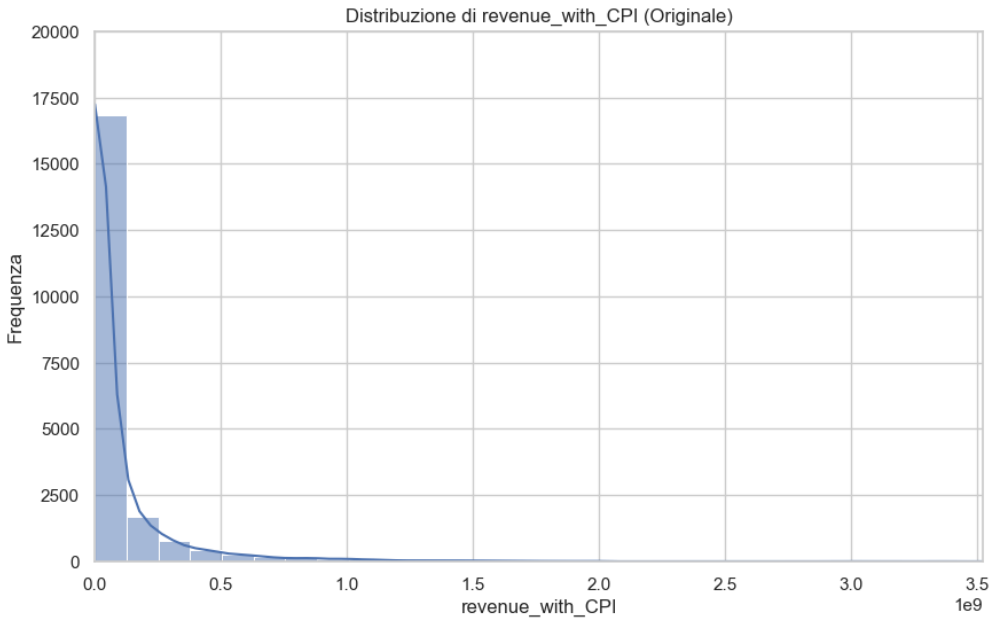

Figure 5: Revenue distribution plot

As show above in Figure 5, most of values are in N-M range, so was tried to find sub-divisions in there but following results obtained were not considered satisfying, so another data discretization via clustering described in next sub-chapter was adopted.

4.1.1 Data Discretization using KMeans Clustering

Clustering is unsupervised learning technique used to find classes from no-labeled data. For project's aim, single feature discretization was adopted trying to obtain different ranges of revenue.
K-Means algorithm was adopted thanks to simplicity, flexibility, interpretability and computational efficiency, considering also the spherical clusters handling with similar number of instances in there. According to problem's theory, nor other algorithms were tested due to lack of acceptable results, in terms of both precision (measured using silhouette) and number of instances per cluster. Results are related to different values of number of clusters K, the only parameter algorithm needs, and revenue value handling, trying with standard one and logarithmic transformation:

		Standard Revenue		**Log Transformation**	
n_clusters	**Cluster**	**Silhouette Score**	**Number of Elements**	**Silhouette Score**	**Number of Elements**
2	Average	0.8949	–	0.6074	–
2	1	0.9104	20075	0.6485	13382
2	2	0.4001	630	0.5323	7323
3	Average	0.8621	–	0.5548	–
3	1	0.8877	19474	0.6018	9470
3	2	0.4543	49	0.5142	3793
3	3	0.4562	1182	0.5158	7442
4	Average	0.8576	–	0.5308	–
4	1	0.8849	19380	0.5656	6857
4	2	0.7015	6	0.5183	4831
4	3	0.4555	1257	0.5105	6933
4	4	0.4784	62	0.5122	2084

Table 23: Comparison of Clustering Results: Standard Revenue vs. Log Transformation

Clustering made clear one of main problem of the dataset, faced already in the preliminary clustering analysis with a very low value of Hopkins Static Test, which is related to the high number of low revenue movies comparing to the low number of high revenue ones, making difficult for clustering recognize the label of the latter. Even though, discretization step was concluded maintaining labels obtained using clustering.

Cluster	From	To	Explanation
0	1.20	113087.10	Low
1	113341.70	3452707.50	Medium-Low
2	3458347.00	54779022.20	Medium-High
3	54795903.10	8813184891.00	High

Table 24: Overview of Clusters

4.2 Cross validation and pipeline for each step

Classification phase adopts the same approach of regression for hyperparameter tuning, using flat cross-validation. However, due to the nature of classification tasks, the pipeline underwent some changes, starting from the preprocessed dataset but with the categorical features not encoded yet (except for the genre[1]), at each step, the classification pipeline includes all following stages:

- **Oversampling**: To address class imbalance issues, the **SMOTENC**[2] algorithm was adopted. This technique aimed to synthetically generate new instances from the minority class, making the dataset more balanced.
- **Data Preprocessing**: *One-hot encoding* was applied to the entire dataset to encode categorical variables into a binary format and *Standard Scaler* was applied to the numerical ones.
- **Feature Selection**: Once preprocessing was completed, feature selection techniques were applied to identify the most 20 informative features.
- **Model Training and Testing**: Subsequently, the selected features and the sampled preprocessed data were used to train the classification model. The model was then tested on the test set to evaluate its performance.

[1]For the genre column only, One hot encoding was applied before SMOTENC and then all the boolean genre columns were passed to the algorithm as categorical features; in this way it is possible to create synthetic observations containing more that one value for the genre, as happens in real observations

[2]*Synthetic Minority Over-sampling Technique for Nominal and Continuous data*

4.3 Classification Models

4.3.1 Random Forest

The functioning of the model is analogous to what was illustrated in the regression section.

Parameters	**Possible Values**
classifier__n_estimators	[100,300]
classifier__max_depth	[None, 4, 10]
classifier__min_samples_split	[2,10]
classifier__max_features	[None, 'log2', 'sqrt']
classifier__bootstrap	[true, false]
classifier__criterion	['gini', 'entropy']

Table 25: Parameter Grid for Random Forest Classifier

Parameters	**Best Value**
classifier__n_estimators	300
classifier__max_depth	10
classifier__min_samples_split	10
classifier__max_features	'sqrt'
classifier__bootstrap	true
classifier__criterion	'gini'

Table 26: Best Parameters for Random Forest Classifier

Metric	**Test Set Results**	**Train Set Results**
Final Accuracy	0.4894	0.5545
Final F1 Score	0.4853	0.5535

Table 27: Comparison of Test and Train Set Results

4.3.2 AdaBoost

The functioning of the model is analogous to what was illustrated in the regression section.

Parameters	**Possible Values**
classifier__n_estimators	[50, 100, 150]
classifier__learning_rate	[0.01, 0.1, 1.0]
classifier__algorithm	['SAMME', 'SAMME.R']

Table 28: Parameter Grid for AdaBoost Classifier

Parameters	**Best Value**
classifier__n_estimators	100
classifier__learning_rate	0.1
classifier__algorithm	'SAMME.R'

Table 29: Best Parameters for AdaBoost Classifier

Metric	**Test Set Results**	**Train Set Results**
Final Accuracy	0.5006	0.5074
Final F1 Score	0.5004	0.5077

Table 30: Comparison of Test and Train Set Results

4.3.3 GaussianNB

This classifier uses Bayes' theorem with the assumption of conditional independence between features, assigning predicted classes by calculating the conditional probability of each class given an input vector, using a Gaussian distribution to model the features.

Parameters	**Possible Values**
classifier__var_smoothing	[1e-09, 1e-08, 1e-07]

Table 31: Parameter Grid for GaussianNB Classifier

Parameters	**Best Value**
classifier__var_smoothing	1e-9

Table 32: Best Parameters for GaussianNB Classifier

Metric	Test Set Results	Train Set Results
Final Accuracy	0.3298	0.3281
Final F1 Score	0.3383	0.3337

Table 33: Comparison of Test and Train Set Results

4.3.4 K-Nearest Neighbour

The functioning of the model is analogous to what was illustrated in the regression section.

Parameters	Possible Values
classifier__n_neighbors	[3, 5, 9]
classifier__weights	['uniform', 'distance']
classifier__algorithm	['ball_tree', 'kd_tree', 'brute', 'auto']
classifier__leaf_size	[10, 20, 30]
classifier__p	[1, 2]

Table 34: Parameter Grid for K-Nearest Neighbour Classifier

Parameters	Best Value
classifier__n_neighbors	9
classifier__weights	'uniform'
classifier__algorithm	'ball_tree'
classifier__leaf_size	20
classifier__p	1

Table 35: Best Parameters for K-Nearest Neighbour Classifier

Metric	Test Set Results	Train Set Results
Final Accuracy	0.4609	0.5812
Final F1 Score	0.4658	0.5847

Table 36: Comparison of Test and Train Set Results

4.3.5 Logistic Regression

Logistic regression is a classification model that uses the logistic function to estimate the probability that an instance belongs to a particular class.
It is based on linear regression and applies the logistic function to transform the output into a value between 0 and 1, which can be interpreted as the probability of belonging to a class.

Parameters	**Possible Values**
classifier__penalty	['l1', 'l2']
classifier__C	[0.001, 0.01, 0.1, 1, 10, 100
classifier__solver	['liblinear']

Table 37: Parameter Grid for Logistic Regression

Parameters	**Best Value**
classifier__penalty	'l2'
classifier__C	0.1
classifier__solver	'liblinear'

Table 38: Best Parameters for Logistic Regression

Metric	**Test Set Results**	**Train Set Results**
Final Accuracy	0.4716	0.4757
Final F1 Score	0.4529	0.4605

Table 39: Comparison of Test and Train Set Results

4.3.6 SVC

This classifier seeks to find a hyperplane in such a way that it optimally separates the data points of the classes. It uses support vectors, which are the points closest to the hyperplane, to define the separation between classes. SVC can be adapted for both linear and non-linear classification problems through the use of different kernels.

Parameters	**Possible Values**
classifier__C	[0.1, 1, 10]
classifier__kernel	['poly', 'rbf', 'sigmoid', 'linear']
classifier__gamma	['auto', 'scale']
classifier__degree	[2, 3, 4]

Table 40: Parameter Grid for SVC Classifier

Parameters	**Best Value**
classifier__C	0.1
classifier__kernel	'poly'
classifier__gamma	'auto'
classifier__degree	2

Table 41: Best Parameters for SVC Classifier

Metric	**Test Set Results**	**Train Set Results**
Final Accuracy	0.4549	0.4602
Final F1 Score	0.4701	0.4758

Table 42: Comparison of Test and Train Set Results

4.4 Results' evaluation

Among the various models tested, the Random Forest, SVC and AdaBoost demonstrated the most robust and balanced performance.
However, Random Forest shows some signs of overfitting, as evidenced by the gap between the training and test accuracy.

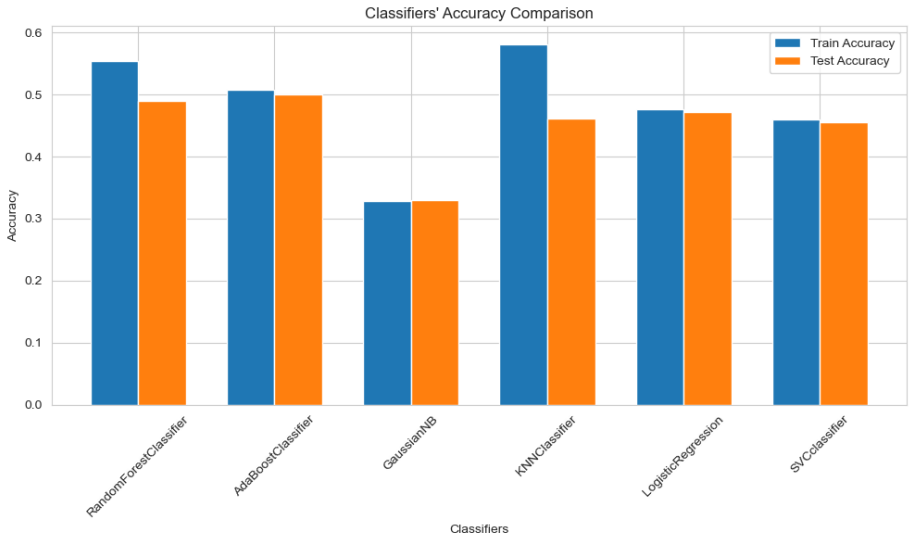

Figure 6: Accuracy and Classification comparison

One of the critical challenges in this analysis was the imbalanced nature of the dataset, since imbalanced classes can skew performance metrics making models appear less effective for minority classes. Problem just introduced is clearly evidenced in the lower precision and recall for classes 0 and 3 across all models.

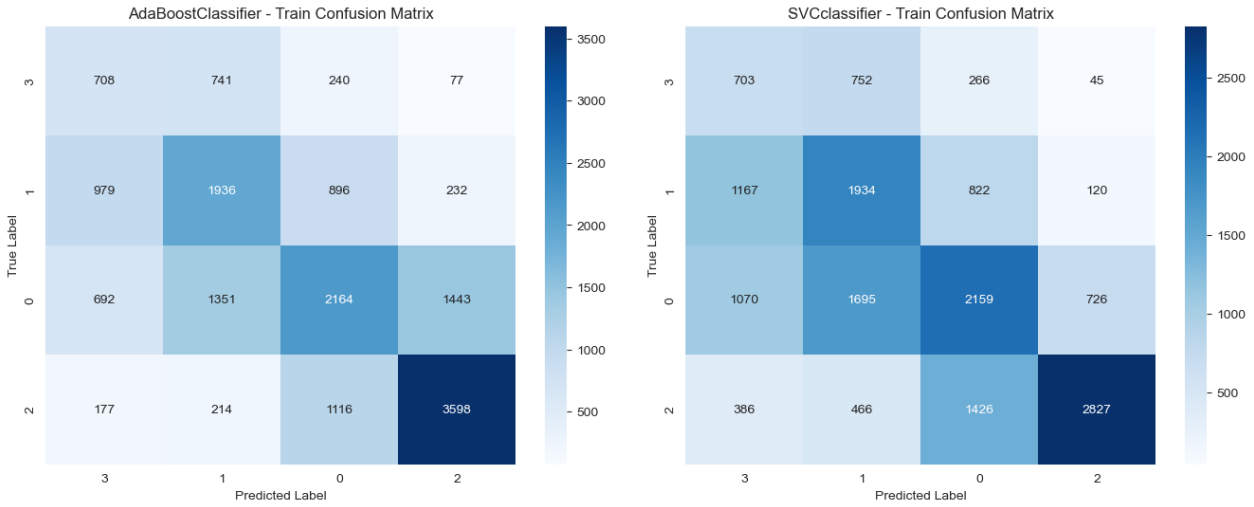

Figure 7: AdaBoost and SVC Train Confusion Matrix

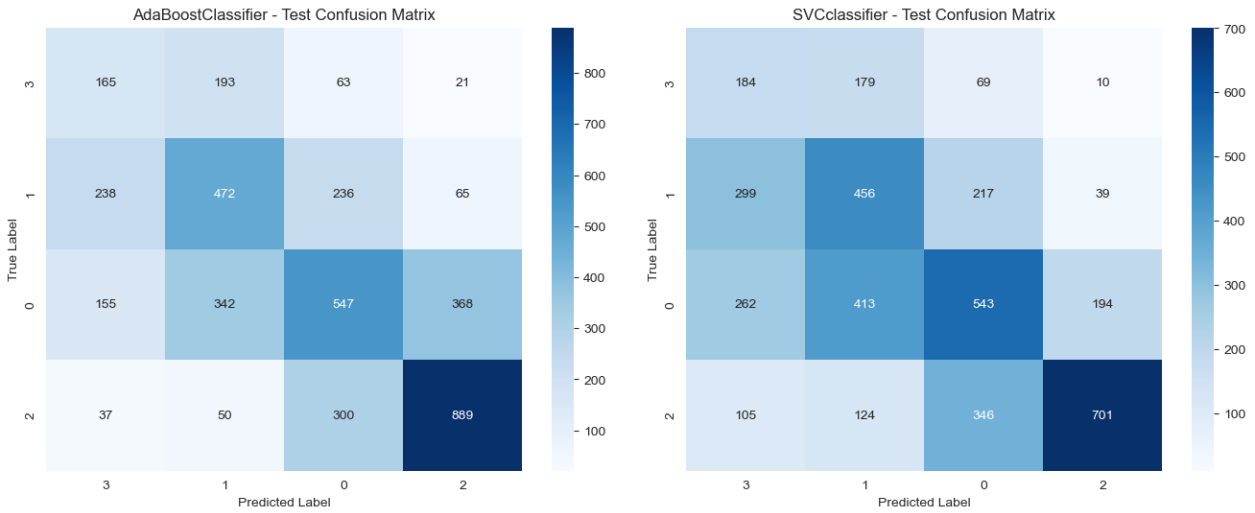

Figure 8: AdaBoost and SVC Test Confusion Matrix

In observing the confusion matrix of the top classifiers, it is noted that most errors occur by assigning a class close to the actual class.
This fact can be measured using the **one-away accuracy**, as depicted in the following graph:

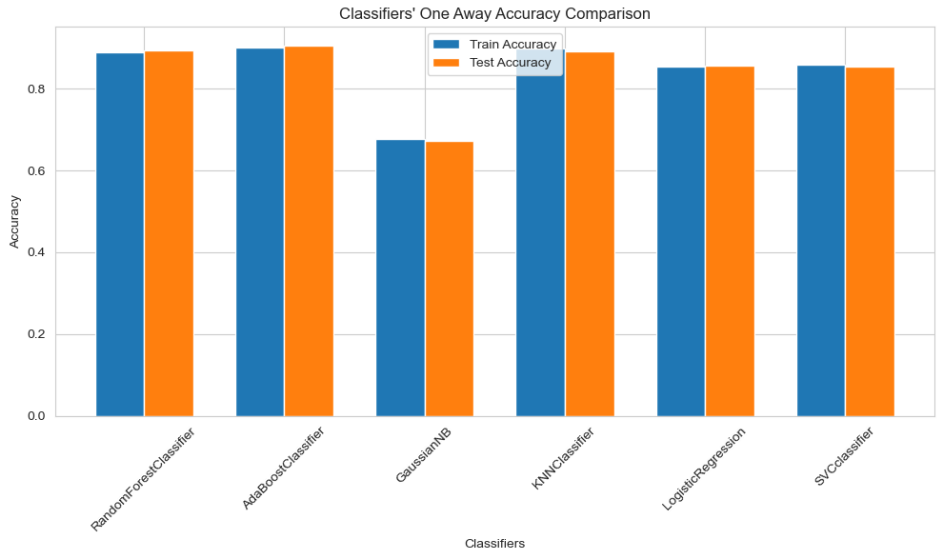

Figure 9: Classifiers' One-Away Accuracy

Looking at the graph, a significant increase in the accuracy of the models is noticeable, particularly AdaBoost proves to be better than SVC, as it makes fewer severe errors.

The **ROC** (Receiver Operating Characteristic) **curve** provided additional insights into model performance, particularly useful in the context of imbalanced datasets. Unlike accuracy, which can be misleading when classes are imbalanced, the ROC curve evaluates the trade-off between the true positive rate (sensitivity) and the false positive rate across various threshold settings. The **AUC** (Area Under the Curve) score derived from the ROC curve gives a single scalar value summarizing the model's ability to discriminate between positive and negative classes.

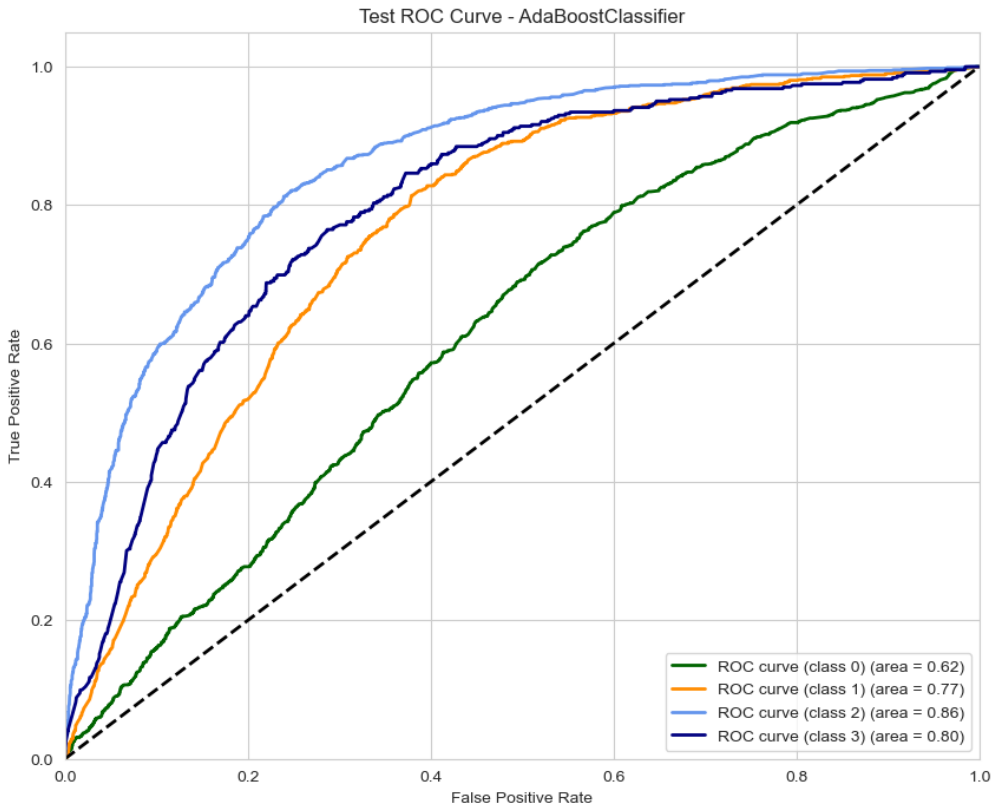

Figure 10: AdaBoost ROC curve 1 vs rest

Interestingly, the ROC curve results for a one-vs-rest approach showed significantly better performance compared to traditional metrics like accuracy, precision, and recall. This discrepancy highlights the importance of using appropriate evaluation metrics that can provide a more nuanced view of model performance.
The ROC-AUC score indicated that the models had a better capacity to distinguish between classes than what was suggested by accuracy alone, particularly in scenarios where the dataset was imbalanced.

5 Conclusions

- Since the beginning, problem was though to be explained due to missing available features, considering for example the absence of marketing campaigns which can justify a movie with low budget and an highest revenue, explaining the reason why firstly data integration process was made including new features to the ones already founded on raw dataset. However, studies made was considered useful since firstly classification models don't have bad results, then results are comparable to the ones obtained in previous similar works made on similar data.
  For example, the article "A Machine Learning Approach to Predict Movie Box-Office Success" provides a classifiers' comparison over a reduced version of our dataset of only 755 movies, discretizing the revenue in 5 classes and obtaining 48.44% of accuracy with SVC classifier.

- Regression models:

  - None of the models developed can be considered satisfying, since even best model AdaBoost has an explained variance R^2 too low to admit the the model has understood the problem and also error is not acceptable since is too high

- Classification models:

  - Imbalanced nature of the revenue feature brings bad results over less represented classes
  - Due to imbalanced nature of revenue feature, the Area Under the ROC Curve (AUC) was adopted as evaluation metric for the models shown AdaBoost as best one
  - Even though high values for models evaluations were not obtained, classification results were thought quite acceptable considering nature of the problem, objectives of the project and benchmark

- By the end of the project, including documentation, application user (identified as production company) is able to find a reasonable prediction of movie's revenue as numeric value using AdaBoost regression model and revenue class based on revenue with AdaBoost classification model, giving a suitable overview of future incomes

6 Application

Since movie's industry is always looking for highest marginal revenue, CinemAI is an application thought for workers for workers, suggesting possible income for new coming films.

6.1 Application guidelines

Since techniques applied for value prediction are well-explained in previuous chapter, this section introduce user to application guidelines.

- **Input**:
  - Actor: at lest one and at most ten actors can be specified, awards obtained are implicitly computed using specific dataset
  - Writer: at least one and at most three writers can be specified, awards obtained are implicitly computed using specific dataset
  - Director: one and only one director must be specified, awards obtained are implicitly computed using specific dataset
  - Budget: starting budget reserved for the movie expressed in USD
  - Runtime: movie duration expressed in minutes
  - Production company: production company funding movie
  - Language: main language spoken during runtime
  - Month published: month by which movie will be released
- **Output**:
  - Class of revenue in which movie would be, according to AdaBoost classification model
  - Movie's revenue predicted using AdaBoost regression model
  - Only for highest revenue prediction, posters of movies in same class as the one predicted

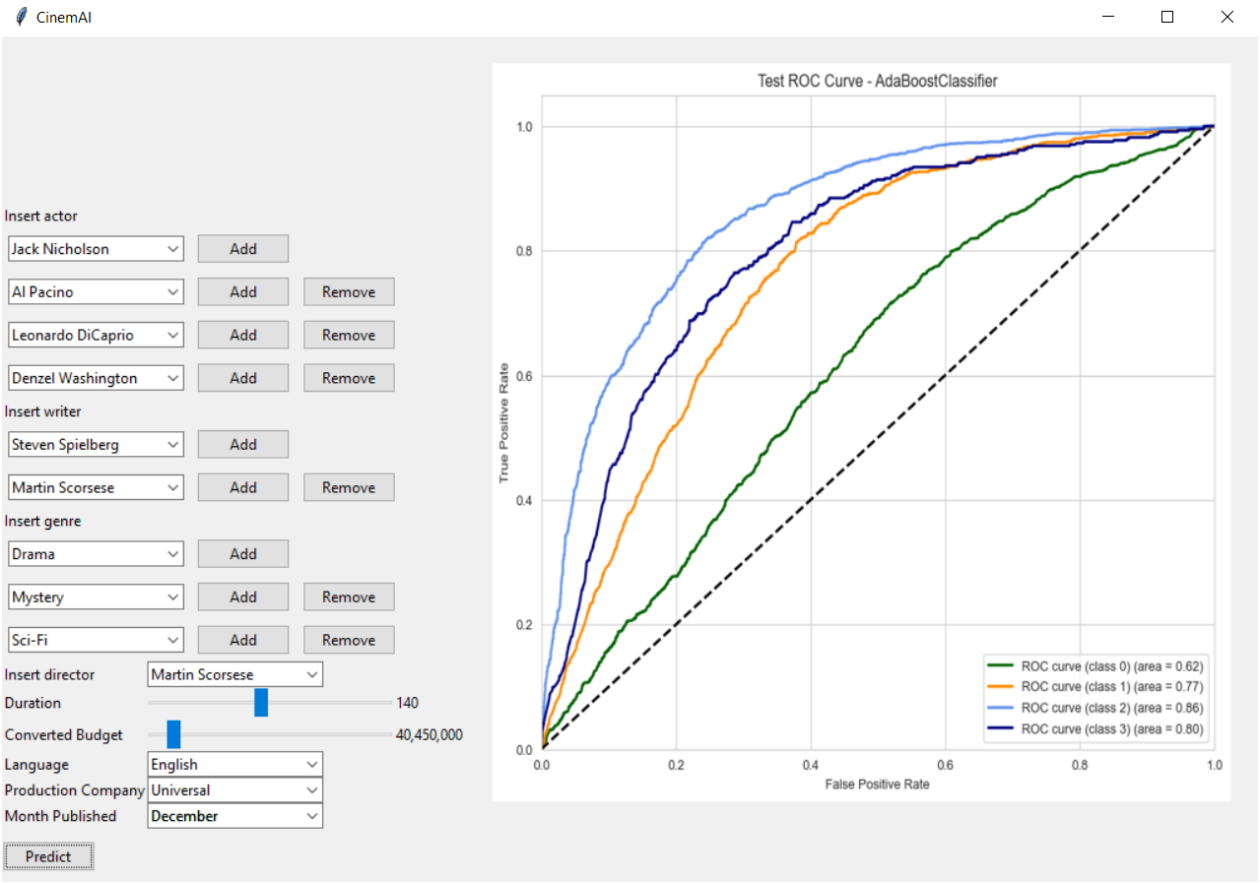

Figure 11: Application starting page

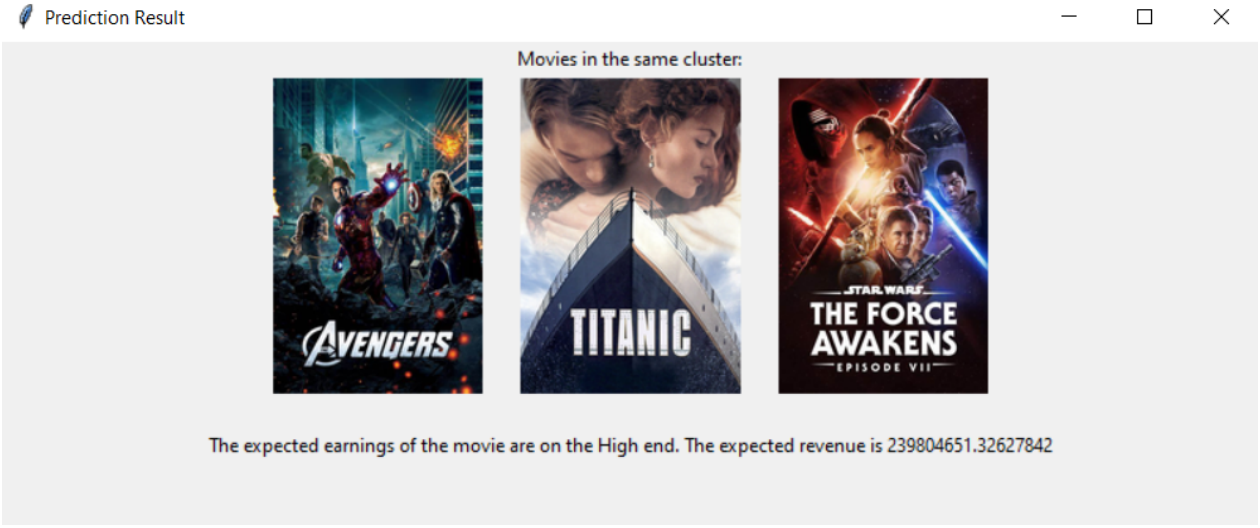

Figure 12: Application output page