Machine Learning Project

1.Introduction

In the following work, we apply machine learning algorithms by combining two datasets. The first dataset consists of movie features, while the second dataset includes user ratings for these movies. Additionally, by utilizing a third dataset provided to us, we manage to leverage data available from the internet.

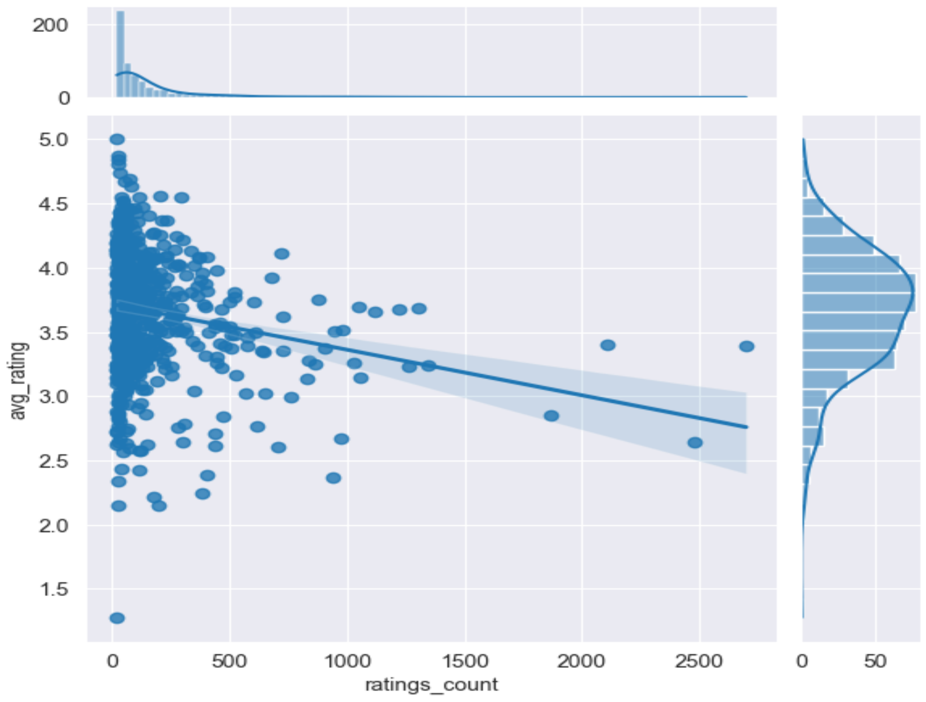
2. Description of the datasets

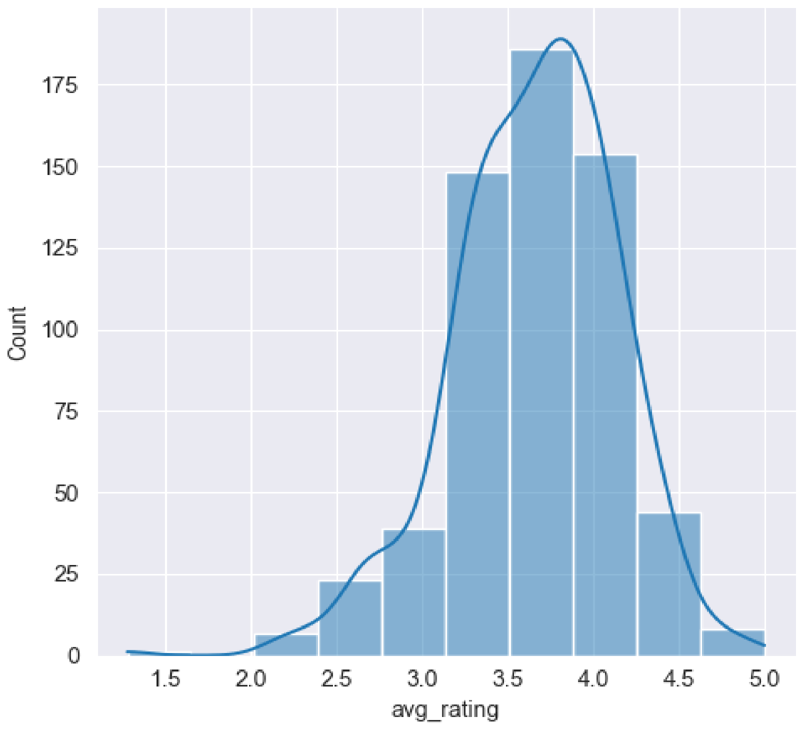
In the movie dataset, there are 9,742 records, each representing a movie with individual characteristics such as a unique identifier, the movie title, the year of its first release in theaters, and the genres it belongs to, such as adventure, horror, comedy, drama, etc.

In the user ratings dataset, there are 100,836 records, each representing a user's rating for the movies they have watched, along with the timestamp of the rating. It's worth noting that the ratings come from 610 different users.

A brief analysis of the two datasets yields the following conclusions:

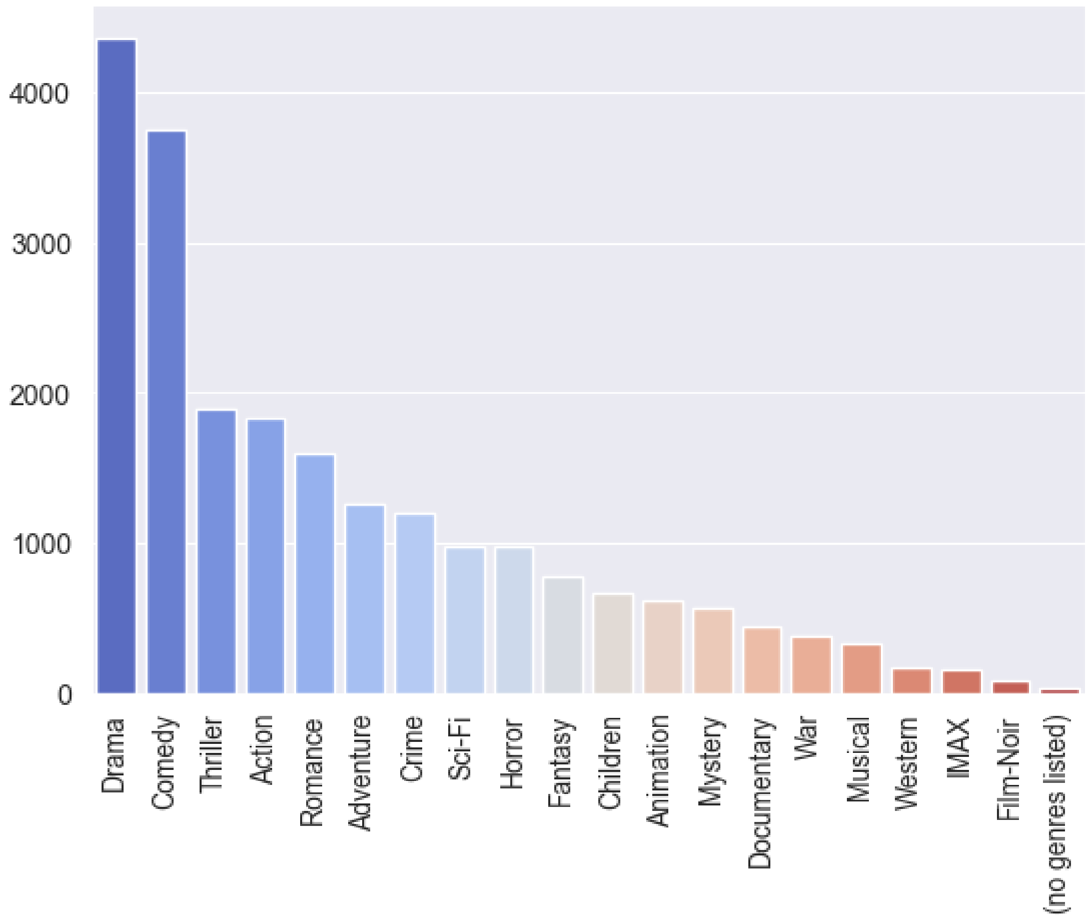
1. Since a movie can belong to more than one genre, there are multiple genre combinations, resulting in 951 unique combinations in the dataset. Therefore, our approach was to create a binary variable for each genre, assigning 1 if the movie falls into that category and 0 otherwise.
2. Regarding the ratings, we found the average rating per user and observed that the distribution of mean ratings follows a normal distribution. Additionally, it's interesting to note that users with mean movie ratings close to extreme values (0.5 or 5) are those who have watched and rated few movies. Conversely, as the number of movies watched by users increases, the mean rating tends to converge towards intermediate values on the rating scale (e.g., 3 or 3.5). Finally, the vast majority of users have watched fewer than 500 movies, while there are some users who have watched 2000 or 2500 movies (which could be considered outliers, but in this case, we decided not to exclude them from the dataset as it would remove a significant number of records).



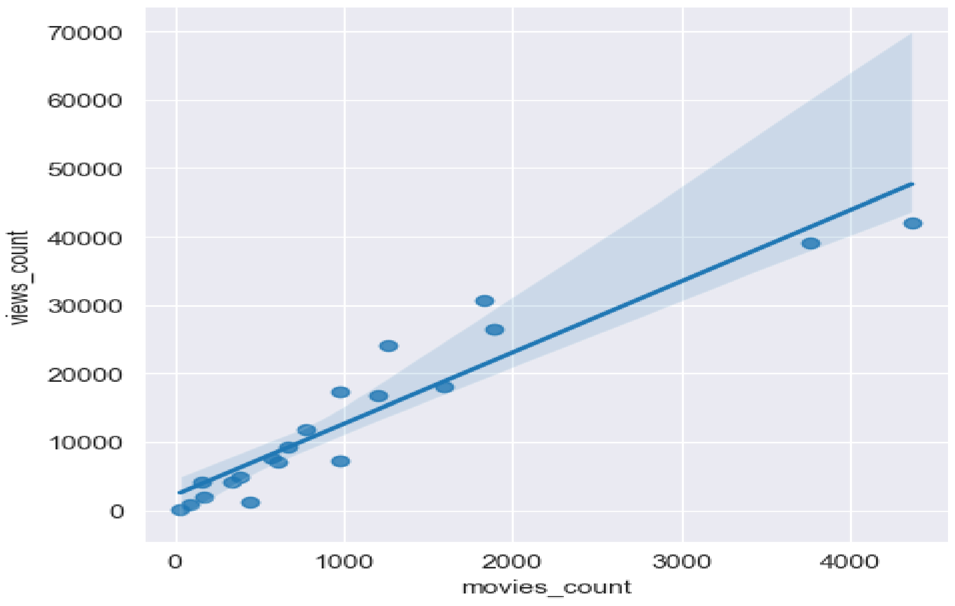


1. By combining the movie genres with user ratings, we draw the following conclusions:

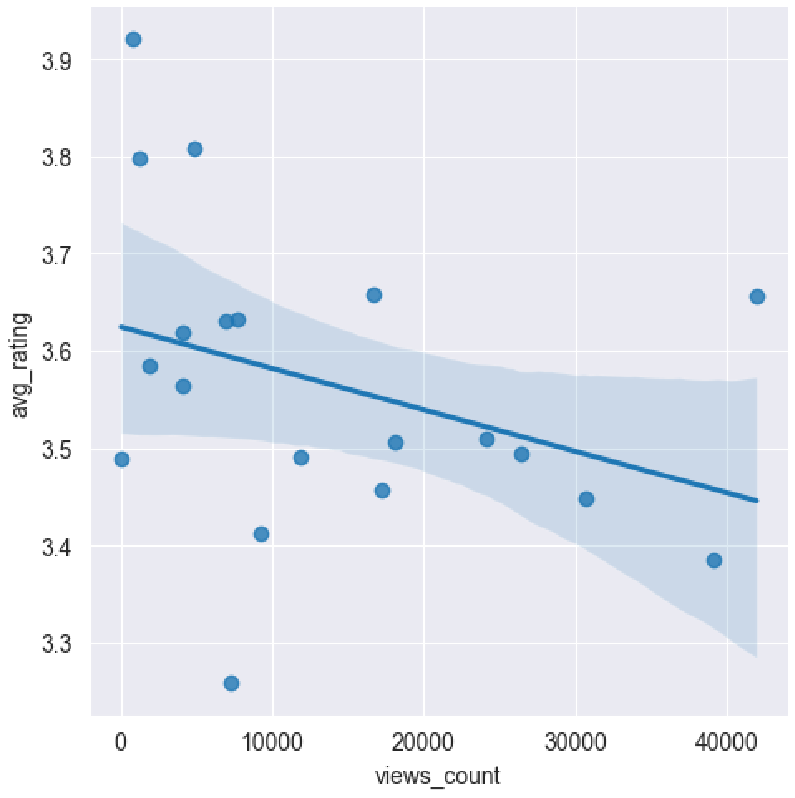
- The genres with the most movies are drama, followed by comedy, and significantly lower in third place are horror movies. The genres with the fewest movies are film_noir, IMAX, Western, and musical.



- With a rather simplified approach (graphically), we found that there is a correlation between the number of movies per genre and the number of screenings per genre. This correlation mainly aids in understanding the data and is not used to support any hypothesis. Additionally, it's important to note that we cannot determine the direction of causality. On one hand, it's logical that as the number of movies in a genre increases, the total number of screenings for that genre also increases. On the other hand, we could interpret the same correlation as suggesting that the popularity of a genre drives producers to create more movies in those genres.



- Finally, another observation is that the average rating per genre seems to decrease as the number of views increases. However, this correlation appears to be weak and exhibits significant variability, making it impractical to draw statistically reliable conclusions.



3.Clustering

The descriptive statistical analysis of the dataset was followed by an attempt to segment the movie audience based on how much they enjoy watching movies, how much pleasure they derive from consuming this commodity, and their preferences regarding movie genres.

To achieve this, we initially found the total number of movies (indicating how much of a movie enthusiast each user is) and the average rating (indicating the enjoyment derived from the commodity) per user. Then, we determined how many movies each user has watched in each category (related to genre preferences). At this point, it should be clarified that the sum of movies per genre exceeds the total number of movies per user because most movies belong to more than one category. To make the values of the fields representative of what we desired, we divided the number of movies each user has watched per category by the total number of movies the user has watched.

Next, we divided the total number of movies per user by the maximum value in the column (i.e., by the biggest movie enthusiast) and followed a similar approach to compute the average rating per user, dividing all values by the maximum of the column.

For clustering, we utilized both distance-based algorithms (such as K-means), which would interpret clusters as consumers with different preferences and behaviors, and density-based algorithms. In the latter case, besides proximity in preferences and behaviors, the number per category played a significant role (density of each population). Such an algorithm could be useful in identifying the largest segments of the movie market. On-demand movies would likely be those with the largest number of users within the cluster, but the boundaries of this cluster should be examined to avoid significant heterogeneity within it. Regarding this study, the aim is to predict the user's rating for each movie, so a clustering algorithm based on distance seems more appealing due to its findings (segmentation based on consumer preference profiles). However, this did not discourage us from trying both categories of algorithms to reach conclusions by evaluating both cases.

KMeans

Finally, regarding preprocessing for the proper application of clustering, normalizing distances so that each feature (column) has the same weight in the final result is crucial. Using StandardScaler, where each value in the column is divided by its standard deviation, achieves this. In this case, the positives of applying the algorithm include distinguishing clusters of consumers with different preferences. The main means of differentiation should be the distance they have from certain preferences represented by the respective centroids of the algorithm. However, negatives of the algorithm include forcing all elements to belong to some cluster, resulting in outliers not being filtered. Additionally, the final result is vulnerable to the initialization of the cluster count. To find the appropriate number of clusters, two tools were used.

1. The "Elbow" criterion aims to minimize the sum of squared distances between each point x*x* and the centroid μ*μ* of the cluster to which x*x* belongs, and identifies the point at which the graphical representation forms an "elbow".
2. The Silhouette Score measures how close the data points are within a cluster. It calculates the average distance between a data point and all other points in the same cluster. Lower values indicate better cohesion, suggesting that the points within the cluster are tightly packed. It also measures how different a cluster is from other clusters. For each data point, it computes the average distance from the point to all points in the nearest neighboring cluster. Again, lower values indicate better isolation, meaning that the clusters are distinct from each other. The silhouette score for the entire dataset is the average of the silhouette scores for each data point, ranging from -1 to 1:
    - 1: Indicates that the cluster is dense and well-separated.
    - 0: Means that the clusters overlap.
    - -1: Indicates incorrect clustering, where data points might have been assigned to the wrong clusters.

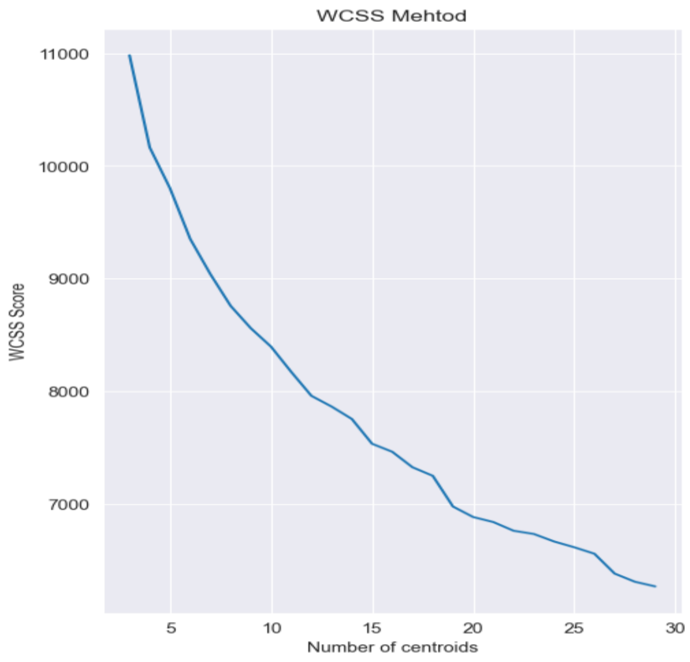
For the second problem of initializing clusters, we applied two variations of the k-means algorithm:

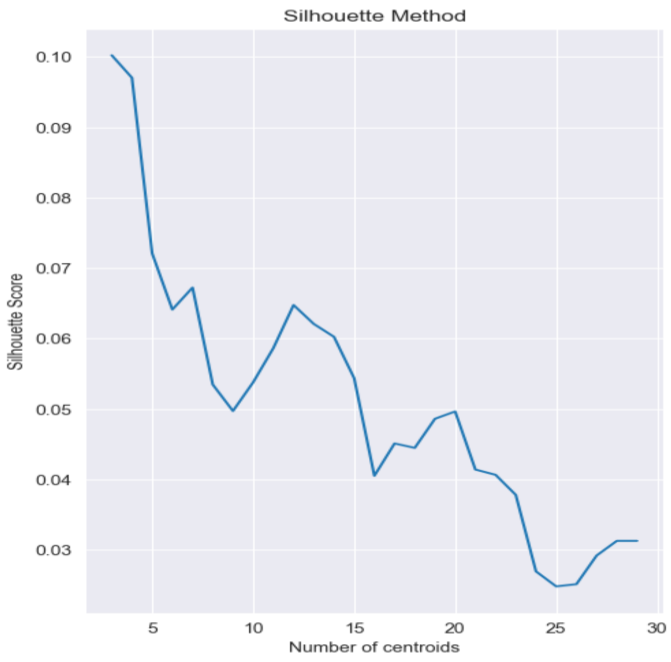
1. Bisection: It splits one of the existing clusters using K-means and is less sensitive to initialization.
2. Selection of initial means via sampling: A variation of K-means that produces divisive clustering hierarchically.

   kmeans = KMeans(n_clusters = 7, init='k-means++',random_state = 42, n_init= 10)

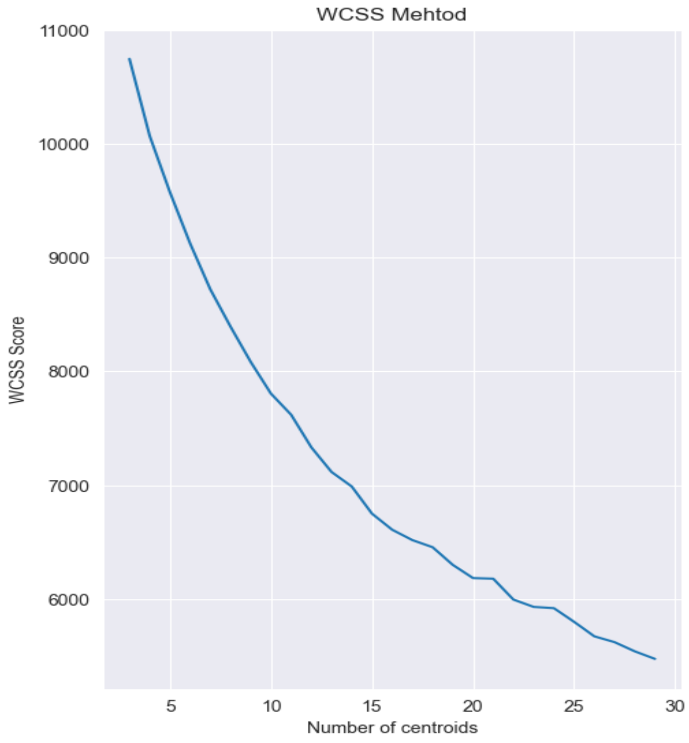
Based on the final results, we chose to use k-means++ with sampling and a number of clusters set to 15, as seen in the plots of the methods below.

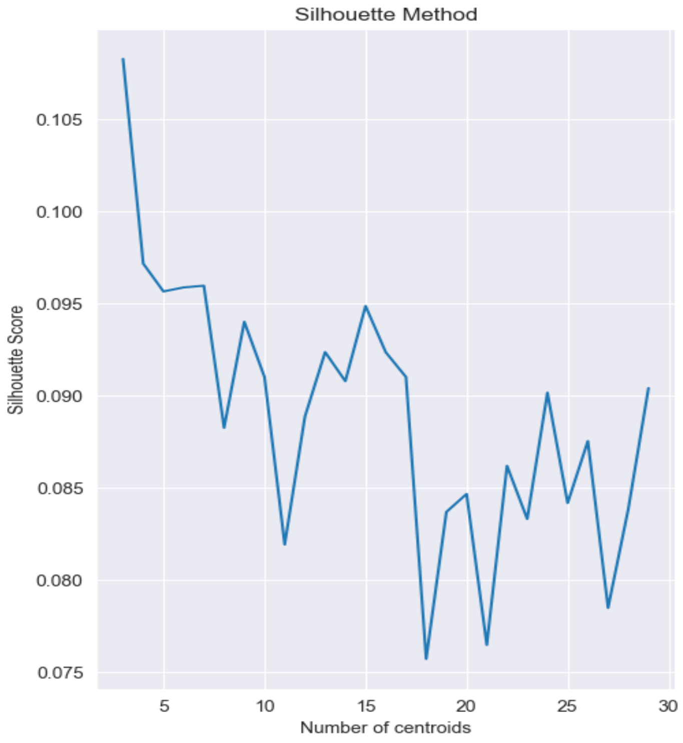
k-means++



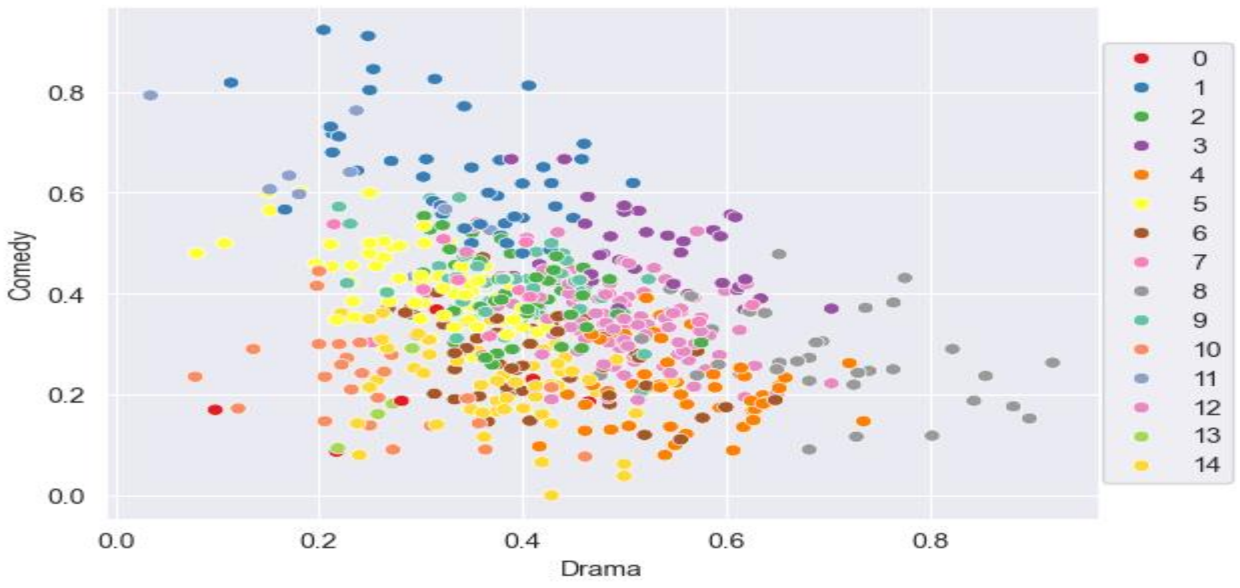


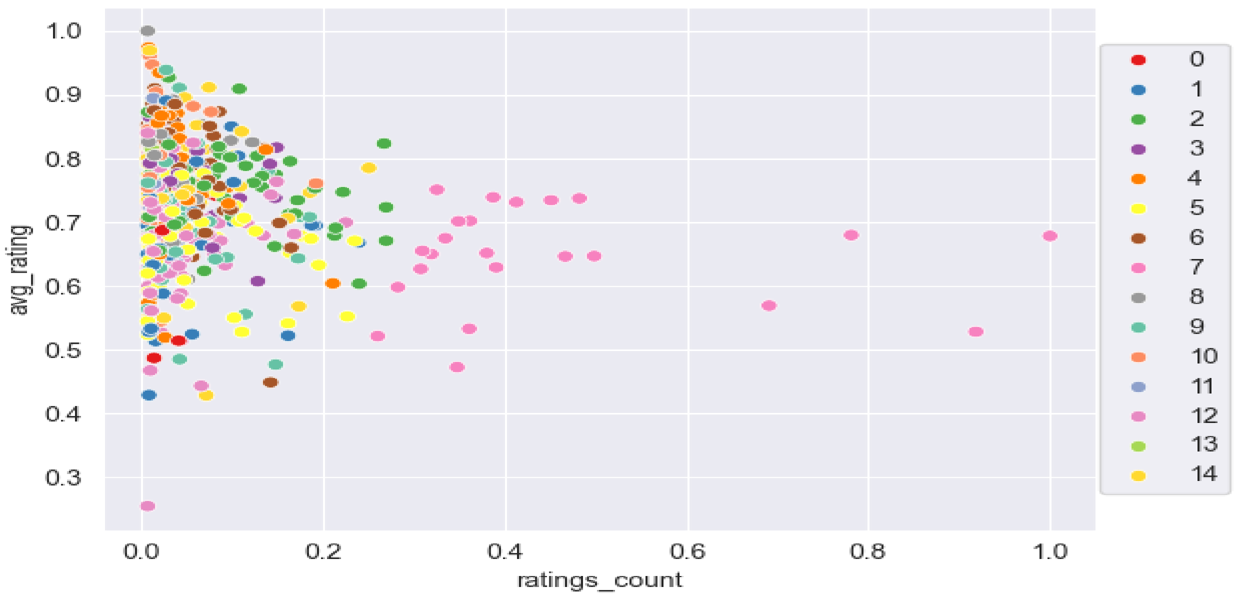
Bisection





Therefore, using k-means with sampling, due to the multiple clustering criteria, we randomly selected the top 2 most popular movie genres and the number of movies per user with their average rating for these movies.



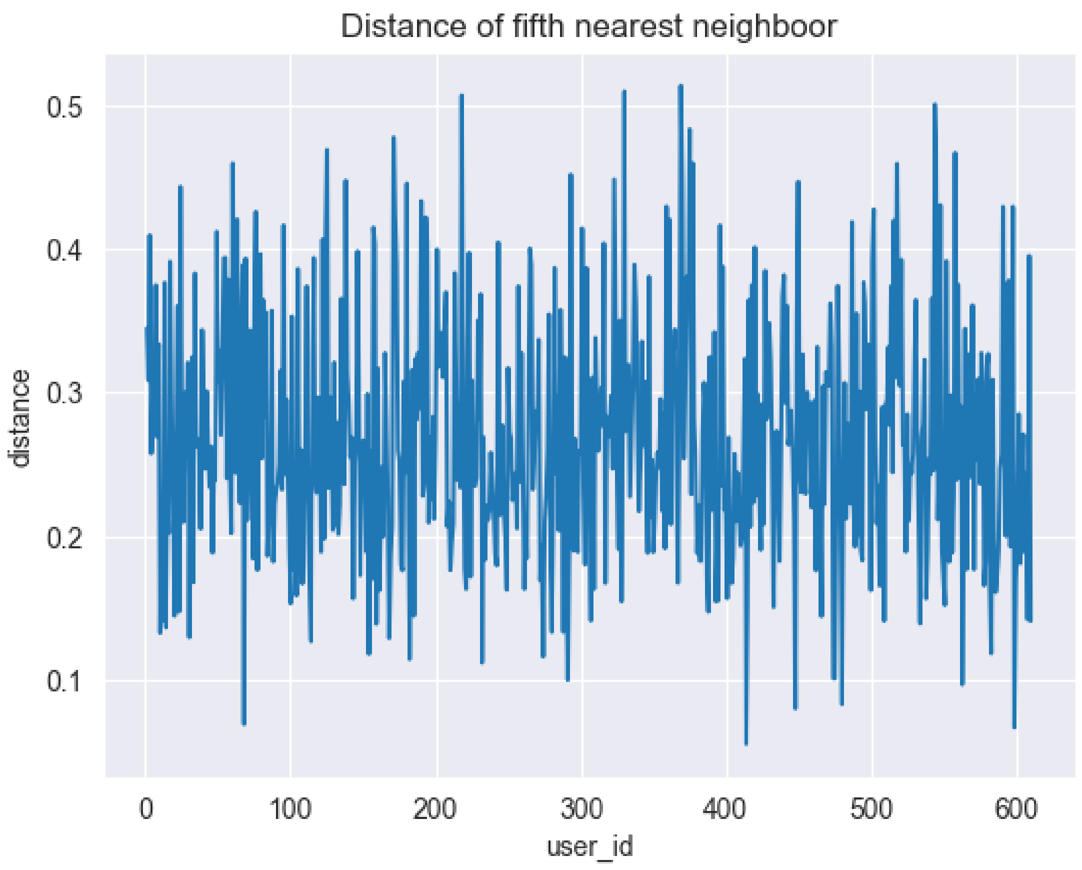


DBSCAN - OPTICS

The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm is a density-based clustering algorithm.

The DBSCAN algorithm operates as follows:

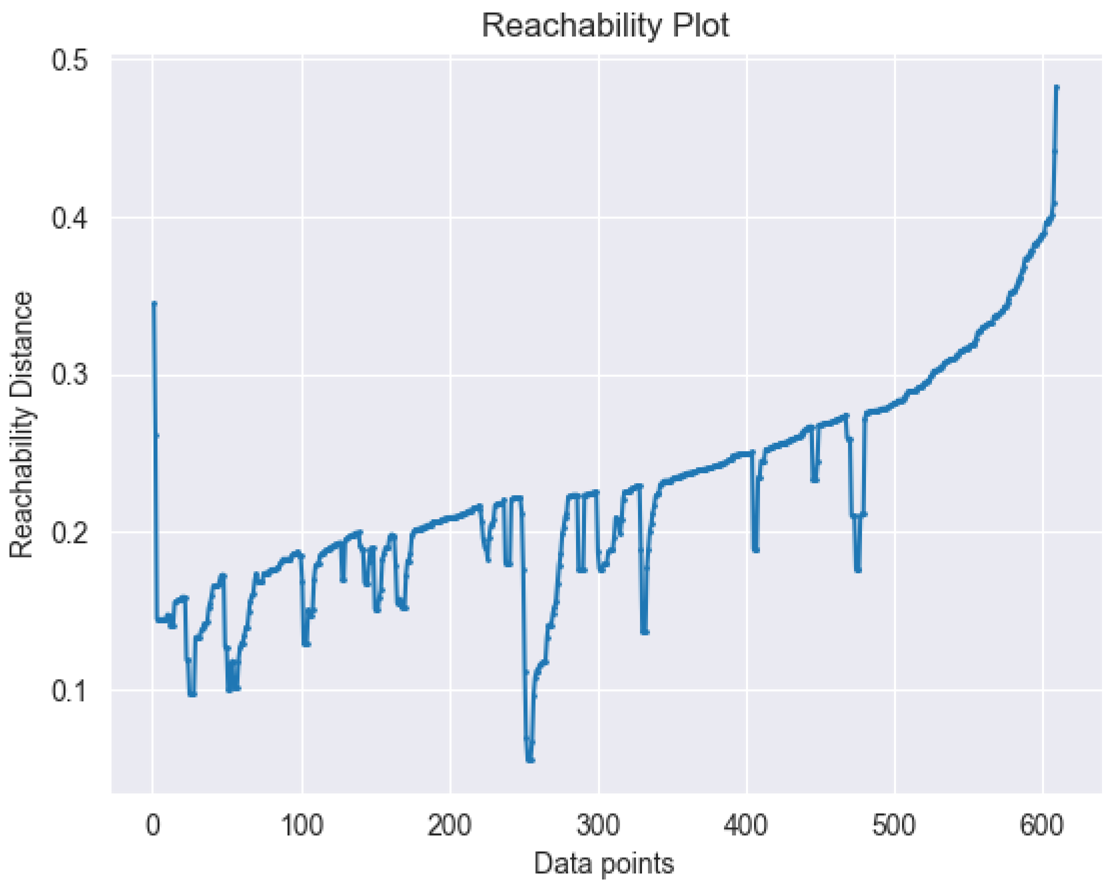
Density Estimation: DBSCAN identifies core points, which are data points in areas of high density, and border points, which lie on the edges of dense regions. Group Creation: It creates groups by connecting core points that are within a specified distance ε of each other. Core points that are close enough to each other are considered part of the same group. Noise Identification: Points that are neither core points nor within distance ε of any core point are considered noise points or outliers. DBSCAN's ability to identify clusters based on density makes it robust to noise and capable of handling clusters with arbitrary shapes and sizes. To properly parameterize DBSCAN, we executed some additional algorithms to compute the distances of the 5 nearest neighbors for each user. Then, we calculated the mean of the 5th nearest neighbor for each user. The algorithm was executed using the built-in "NearestNeighbors" package in scikit-learn. Finally, it is worth mentioning that the cosine distance was used as the distance metric for both DBSCAN and OPTICS, which we applied below.



The OPTICS (Ordering Points To Identify the Clustering Structure) algorithm is a density-based clustering algorithm that extends the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm.

The OPTICS algorithm operates as follows:
First, it calculates the reachable distance of each point with respect to its nearest neighbor. The reachable distance is a measure of how densely the points are distributed among themselves.
Next, it constructs a reachability plot, which represents the reachable distances of all points sorted in ascending order. This plot captures the density-based clustering structure of the data.

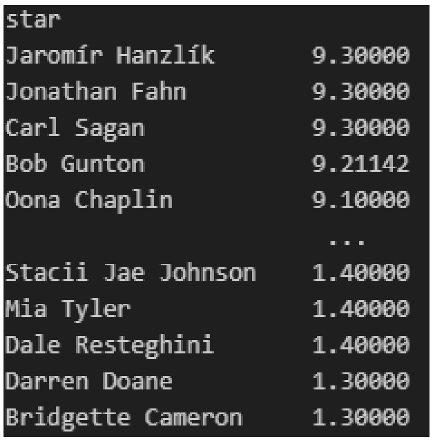
Then, OPTICS identifies clusters by extracting peaks and valleys from the reachability plot. Peaks correspond to cluster centers, while valleys represent areas between clusters or noise.
Finally, it assigns each point to a cluster based on the reachable distance and predefined parameters, such as the minimum number of points and maximum distance.
Overall, OPTICS provides a more flexible and scalable approach to clustering, especially for datasets with variable densities and irregular shapes.



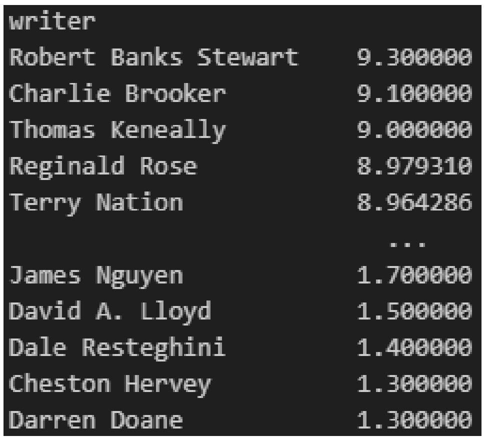
Τα συμπεράσματα μας για τους δυο αλγορίθμους που λειτουργούμε την πυκνότητα ήταν αρκετά απογοητευτικά, με την έννοια ότι δεν μπορούσαν να διακρίνουν τις διαφορετικές συστάδες χρηστών και τις περισσότερες παρατηρήσεις να τις χαρακτηρίζουν ως θόρυβο. Επίσης τα σύνορα των συστάδων που αναγνώρισε επέτρεπαν αρκετά ετερογενής προτιμήσεις λόγω των πολυάριθμων συνδυασμών προτιμήσεων ταινιών που συχνά μοιράζονται έναν, δύο ακόμα και τρεις τύπους ταινιών. Έτσι, η διαφορά στις προτιμήσεις δεν είναι εύκολο να εντοπιστεί.

<u>4.Τελευταία στάδια προπαρασκευής και εξόρυξης δεδομένων</u>

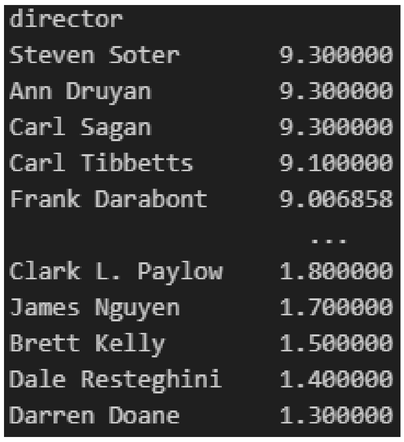
In the final analysis that we chose to perform, data for actors, writers, and directors were needed. These data could be obtained from IMDb using the "links.csv" for finding movie data and then merging it with existing data. Therefore, an extra code was created for data collection from IMDb, which exists in the files as "imdb_scraper.ipynb". Having collected ratings for each movie, the names of the top 3 actors, all writers, and directors, as well as the IMDb ratings of the specific movie, which are stored in the file "imdb_data.csv", we were able to find the average rating per actor, the average rating per writer, and the average rating per director. Below are some results:



```
star
Jaromír Hanzlík        9.30000
Jonathan Fahn          9.30000
Carl Sagan             9.30000
Bob Gunton             9.21142
Oona Chaplin           9.10000
                         ...
Stacii Jae Johnson     1.40000
Mia Tyler              1.40000
Dale Resteghini        1.40000
Darren Doane           1.30000
Bridgette Cameron      1.30000
```



```
writer
Robert Banks Stewart    9.300000
Charlie Brooker         9.100000
Thomas Keneally         9.000000
Reginald Rose           8.979310
Terry Nation            8.964286
                           ...
James Nguyen            1.700000
David A. Lloyd          1.500000
Dale Resteghini         1.400000
Cheston Hervey          1.300000
Darren Doane            1.300000
```



```
director
Steven Soter        9.300000
Ann Druyan          9.300000
Carl Sagan          9.300000
Carl Tibbetts       9.100000
Frank Darabont      9.006858
                       ...
Clark L. Paylow     1.800000
James Nguyen        1.700000
Brett Kelly         1.500000
Dale Resteghini     1.400000
Darren Doane        1.300000
```

Next, we calculated the average rating of the combination of the top 3 actors in the specific movie, the writers, and the directors, by summing up their average ratings and dividing by their count. With the necessary data at hand, we can now proceed to the part of creating prediction algorithms.

5. Αλγόριθμοι Προβλέψεων

RANDOM FOREST VS LINEAR REGRESSION

5.1 Random Forests

Our first attempt in this project was to unlock the secrets of user preferences and recommend movies they would truly love using the power of RandomForest algorithm. The RandomForest algorithm is based on mechanisms used by decision trees. To better understand the mechanism of a decision tree, we can liken it to a movie critic. Each "critic" (a decision tree) analyzes various aspects of movies (genre, director, actors), the user's profile, and their history (past ratings) to predict a rating. However, the peculiarity of RandomForests is that they consist of many decision trees, each of which decides separately and differently since they use different evaluation criteria or even the same criteria but in a different order. Once all the trees make their conclusions, the most frequent answer is chosen. In simpler terms, we can say that the RandomForest examines the same problem from different perspectives, just like when many critics decide instead of one. Through this mechanism, the problem of bias, which is a common phenomenon in machine learning algorithms, especially in the simple decision tree algorithm, is largely resolved.

Advantages of the Algorithm:

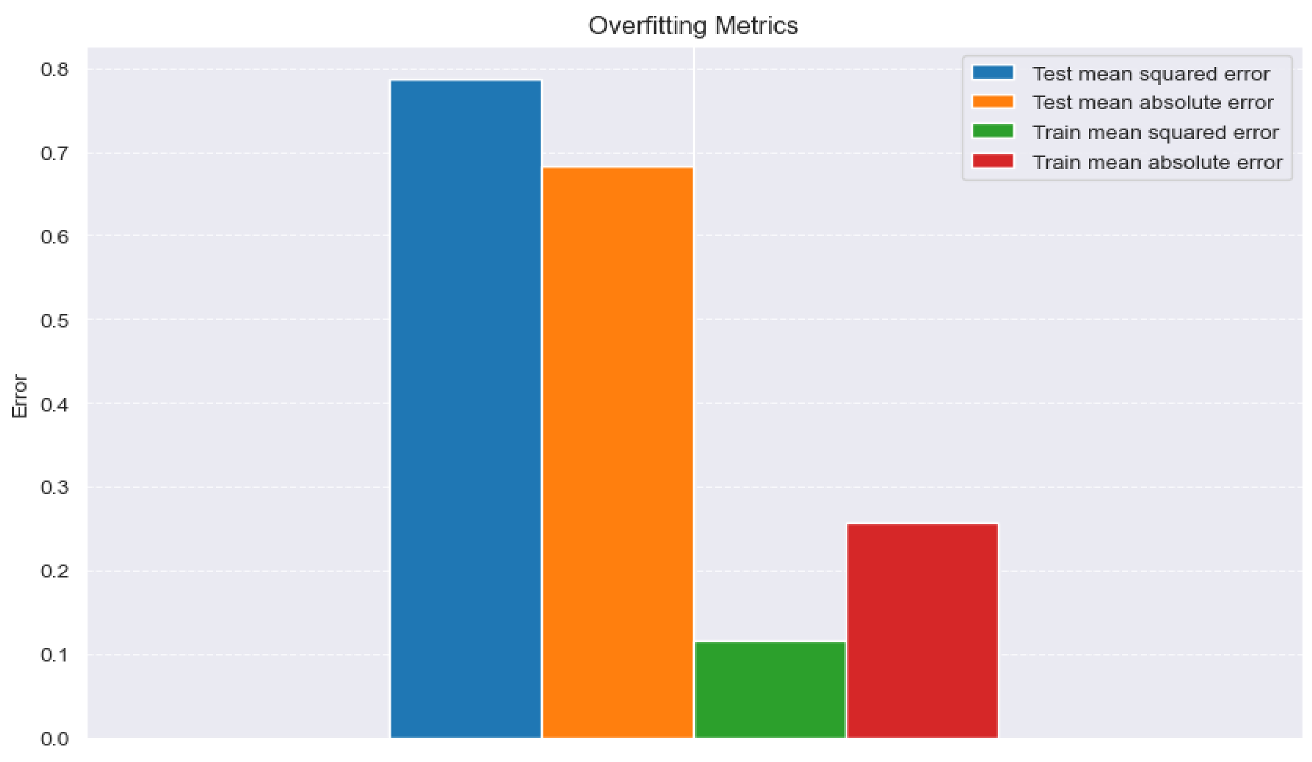
- It is robust to outliers, so there is no need for extensive data preprocessing.
- Ability to model nonlinear relationships between dependent and independent variables.
- Can handle large datasets.

- Provides information about the importance of each feature in the final decision.

Disadvantages of the Algorithm:

- Although it provides information about the importance of each feature, it does not clearly indicate the degree of contribution of the feature to the final decision.
- High computational cost.
- Regarding interpretability, the model is characterized as a "black box" similar to neural networks, as it is difficult to understand the exact reasoning behind each prediction.
- Handling noise requires proper parameter tuning, which in the case of this algorithm is complex and may require experimentation.

In the initial application of the algorithm, parameter selection was mainly based on empirical rules, and we encountered the problem of overfitting as there was very good performance in predicting the training data, while the performance was quite disappointing when tested on unseen data. Below is a diagram showing the mean squared error and mean absolute error for the training and evaluation sets.



At this point, it became evident that the model's parameters were not appropriate, requiring readjustment of their values. However, we found it useful to dedicate a small section to explain how each of these parameters affects the model's structure and, consequently, its effectiveness.

1. **max_features**: This parameter determines the number of columns (features) to be considered when splitting data at the nodes of the trees. The more features used, the greater the complexity and flexibility of the model, allowing it to capture data intricacies. However, there's a trade-off:

increasing features may lead to model bias and over-dependence on training data, resulting in poor performance on new data.

2. **max_depth**: This parameter sets the maximum depth to which a tree can grow. When its value is None, trees can grow as needed to minimize the loss function. These trees are called "bagged trees" and are very effective in predicting data they have already seen (training set). However, they again lead to increased complexity and model bias, negatively affecting performance on new data.
3. **min_samples_split** and **min_samples_leaf**: These parameters determine the minimum number of data points required to split internal nodes or form leaves, respectively. A large value can prevent over-dependence but may also lead to an overly simplified model unable to capture the relationship between features and the target variable (movie rating in our case).

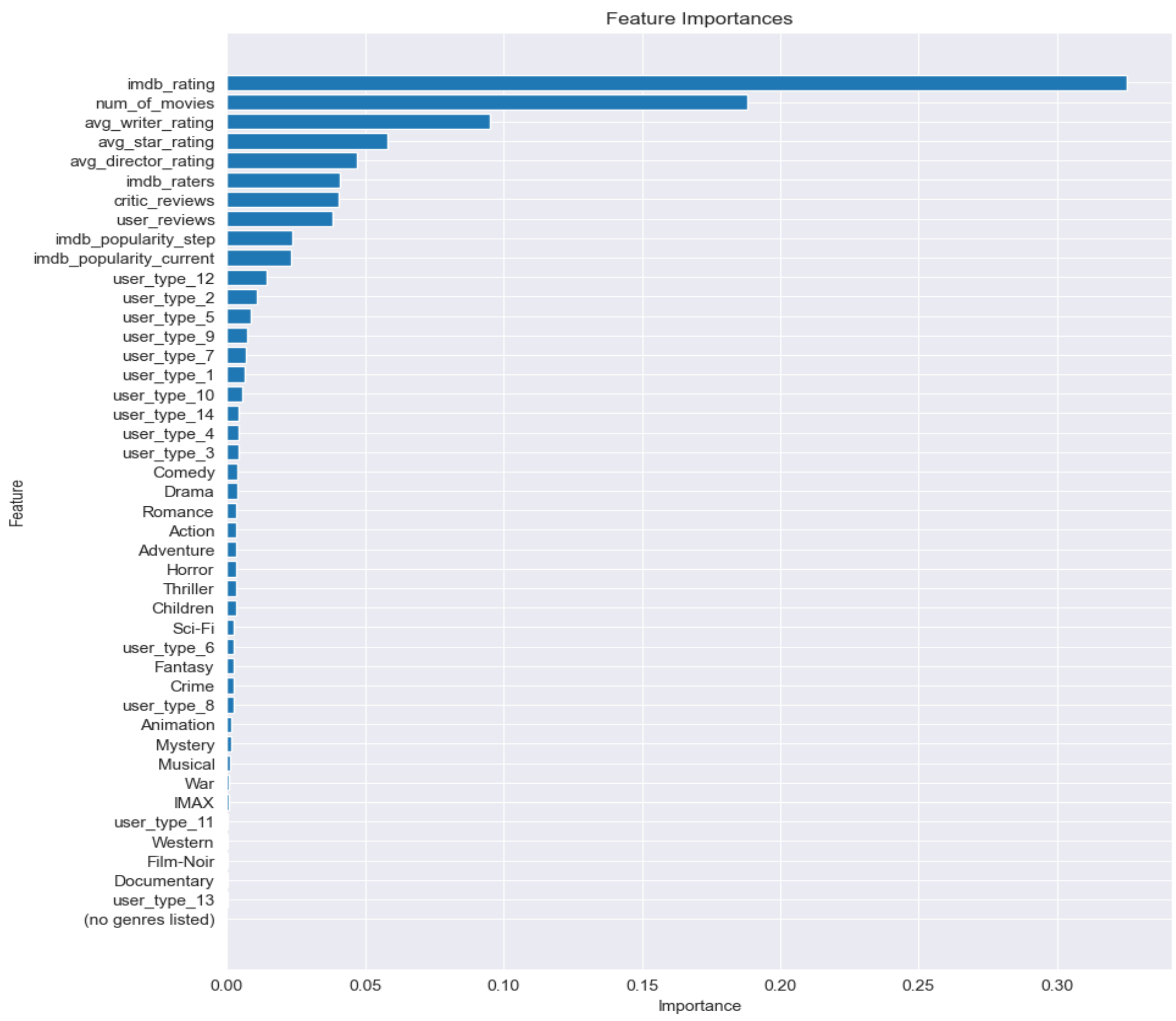
Regarding finding suitable values for these parameters, it should be emphasized that there is no deterministic method for all cases. Crucial factors include the number of columns, complexity, interdependencies among variables, and the number of data points.

A commonly followed rule is to try different combinations of values passed to the parameters. This is the approach we followed using an integrated functionality of scikit-learn called GridSearchCV, which creates a "grid" of parameter combinations and tests all possible combinations. In addition to trying all possible parameter combinations, the algorithm applies the technique of cross-validation to evaluate the generalization of a model. GridSearchCV uses a dictionary as arguments describing the parameters to be tested and the type of model for training. Also, the CV (cross-validation) parameter determines the number of subsets of the dataset used for training and evaluating the model. Training is done on (n-1) subsets, while the model is evaluated on the subset not used during training. Lastly, it's worth noting that this technique is time-consuming and computationally intensive. Below are the evaluation metrics and the importance of each feature in the final prediction.

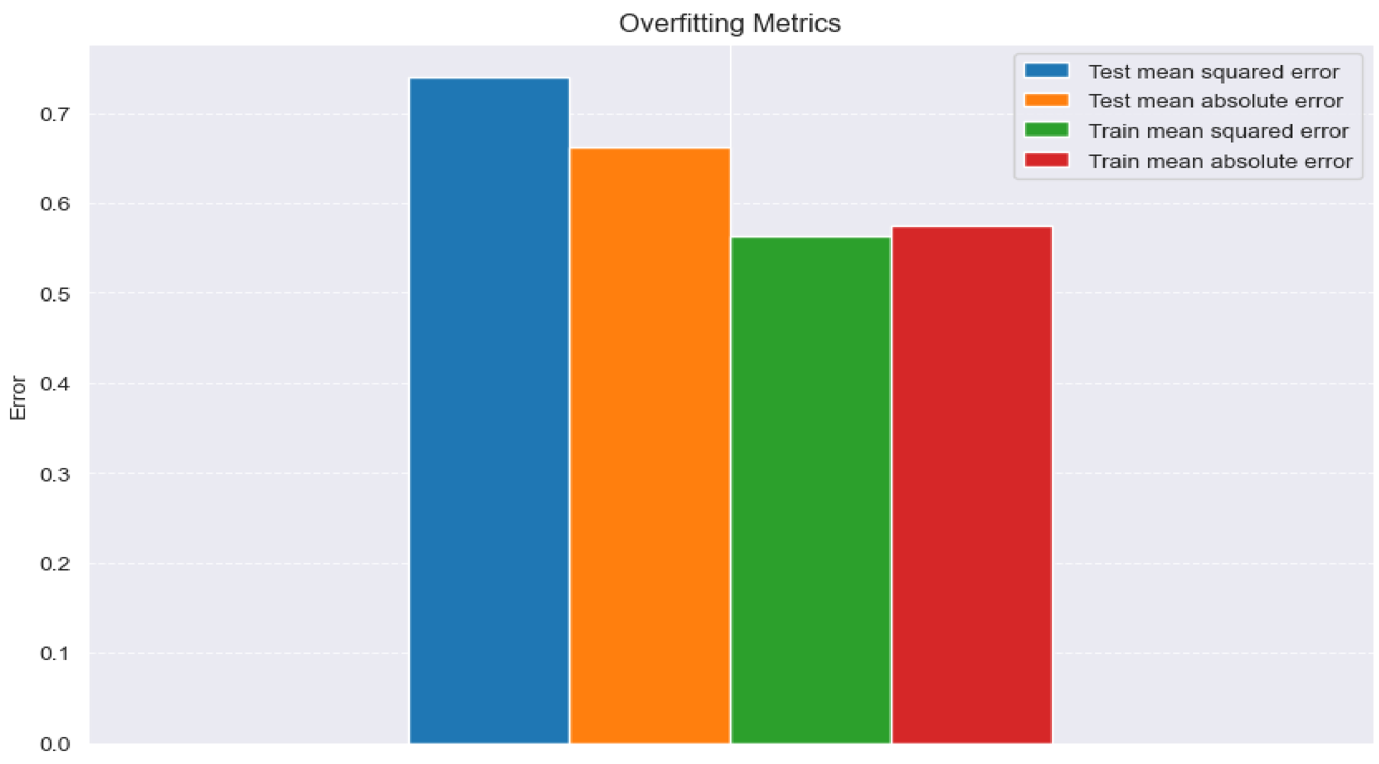
	Test Set	Train Set
MSE	0.7406	0.5632
MAE	0.6631	0.5741

From the perspective of a classification problem, the metric used was the accuracy with a value of 0.2520834030590047.

Regarding the importance of the coefficients:



And as for addressing overfitting:



5.2 Linear Regression

Our first approach focused on leveraging the power of Random Forest to uncover user preferences and recommend movies that would excite them. Now, let's explore linear regression, an alternative method with different advantages and disadvantages.

Linear regression is a statistical method that models the relationship between a dependent variable and one or more independent variables. The basic idea is to find the "best" straight line that fits the data. The "best" straight line is the one that minimizes the sum of squared residuals between the data points and the line.

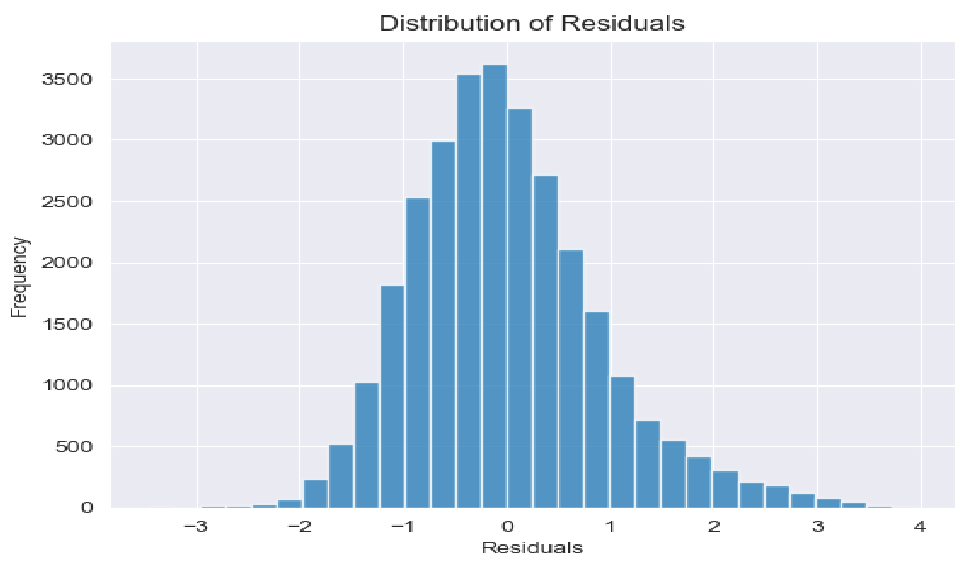
Advantages:

- **Simplicity**: Linear regression is easily understandable and interpretable.
- **Low computational cost**: Training a linear model is fast and computationally inexpensive.
- **Flexibility**: It can be applied to various types of data.
- **Explainability**: Linear regression provides clear information about the contribution of each feature to the prediction.

Disadvantages:

- **Linearity assumption**: Linear regression assumes a linear relationship between variables. In cases of non-linearity, the model's accuracy is negatively affected.
- **Sensitivity to outliers**: Extreme values in the data can significantly distort the model's results.
- **Limited modeling capability**: Linear regression cannot capture complex relationships between variables. Also, its effectiveness depends on how well it meets the Gauss-Markov assumptions, which can be quite restrictive.

In the initial execution of the linear regression model on the same data as used in the RandomForest algorithm, we observed many redundant variables through the p-value, while multicollinearity was also present (see source code file). To address this issue, redundant variables that were statistically insignificant at a 5% confidence level were removed. It's important to note that the error term follows a normal distribution, so the statistical properties of least squares estimators apply.



Finally, the following metrics are presented for evaluating the model:

	Test Set
MSE	0.83199
MAE	0.705445
R-squared	0.222631

** For further improvement of the results, the Principal Component Analysis method was also tested. However, no improvement was observed, while at the same time, the ability to interpret the results was lost.**

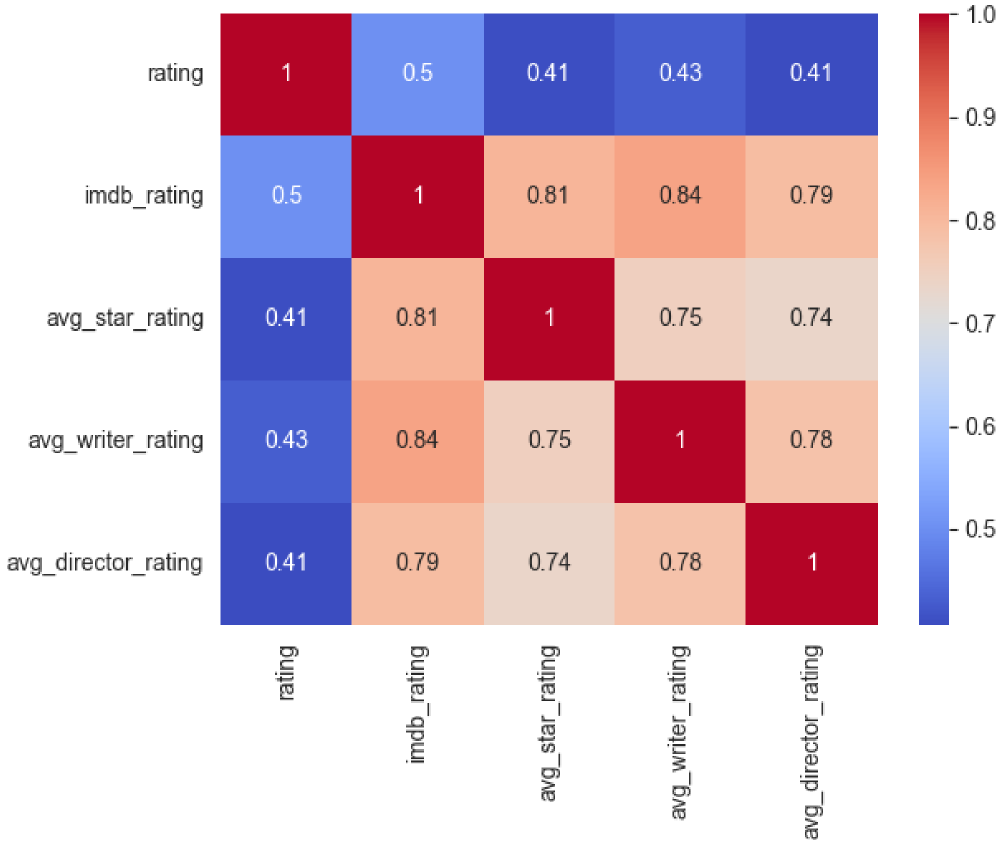
Σύνοψη των πλεονεκτημάτων και των μειονεκτημάτων

Characteristic	Linear Regression	Random Forest
Simplicity	High	Low
Computational cost	Low	High
Flexibility	High	High
Explainability	High	Low
Condition of linearity	Necessairy	Not Necessairy
Sensitivity to outliers	High	Low
Modelling Capability	Limited	High

6. Συμπεράσματα

Based on the results of the two models, we observe that their predictive ability is at similar levels.

However, in the case of linear regression, we have the ability to better interpret the contribution of each feature, while it is also computationally more efficient. If we want to interpret the results of the two algorithms, it is that there are no different preferences and consumer profiles but good and bad movies, as the IMDb rating has the highest correlation with the dependent variable, as seen in the heatmap. Looking at the diagram also raises the question of how much actors, writers, and directors matter for the success of a movie.



So, what we can conclude from the results is that for a movie to be successful, the writer matters first, followed closely by the actors, and lastly the director. Although all of them have a fairly high correlation