

**Logistic Regression:**

Logistic Regression is a supervised learning algorithm for binary classification tasks, where the goal is to predict a movie or show success based on a threshold IMDB score. We could convert the IMDB scores into a binary format by setting a threshold value where we deem the movie or show successful if the IMDB score  $\geq 7$ . Logistic Regression is chosen for this problem because it's a simple, interpretable algorithm that can model binary outcomes.

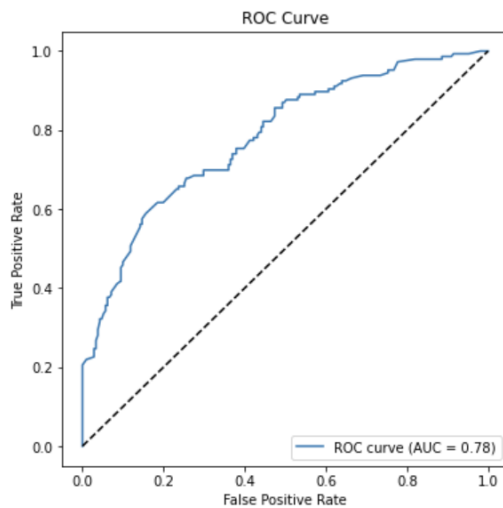
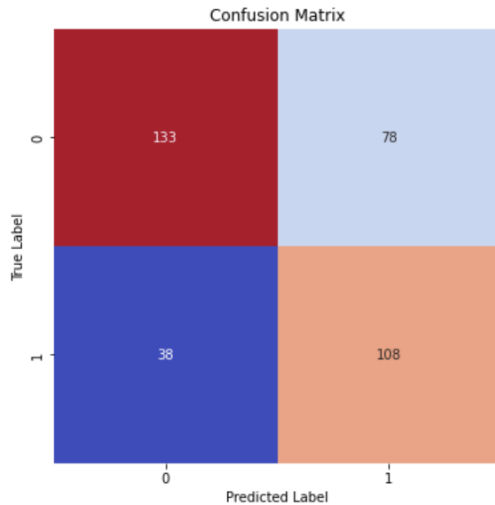
As explained earlier, we converted the IMDB scores into a binary format by setting a threshold value. Then, we split the cleaned dataset into training and testing sets using an 80-20 ratio to train the model. Afterward, we can train the Logistic Regression model using the training set and evaluate its performance on the testing set.

To assess the effectiveness of the Logistic Regression algorithm, we can use metrics such as accuracy, precision, recall, F1-score, and the area under the Receiver Operating Characteristic curve. These metrics help us understand the model's ability to classify movies correctly and shows as successful or not and balance false positive and false pessimistic predictions.

By analyzing the relevant metrics and visualizations, such as a confusion matrix heatmap, we can assess the model's effectiveness and gain insights into the relationships between the features and the binary outcome of the movie or show being successful or not. This intelligence can help us understand which factors contribute to a movie or show's success, make predictions about future productions based on their attributes, and help give insight on a predictive measure to know if the movie or show is a good investment for the company.

Logistic Regression can help Netflix classify movies and shows as successful or not based on a threshold IMDB score where we deem the movie or show to be successful if IMDB score  $\geq 7$  and unsuccessful and vice versa. By training a binary classification model on various features, Netflix can predict the probability of success for potential content investments. This can aid Netflix in prioritizing content that is more likely to be watched by its audience, maximizing user watch time, and minimizing the risk of content underperforming.

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.78      | 0.63   | 0.70     | 211     |
| 1            | 0.58      | 0.74   | 0.65     | 146     |
| accuracy     |           |        | 0.68     | 357     |
| macro avg    | 0.68      | 0.69   | 0.67     | 357     |
| weighted avg | 0.70      | 0.68   | 0.68     | 357     |



### **Linear Regression:**

We used a linear regression model to predict the 'imdb\_score' based on the 'tmdb\_score.' Linear regression is a simple and widely used algorithm for predicting a continuous target variable based on one or more input features. We chose linear regression because it is easy to understand, implement, and interpret, provides insights into the relationship between input features and target variables, and allows quick model training and evaluation.

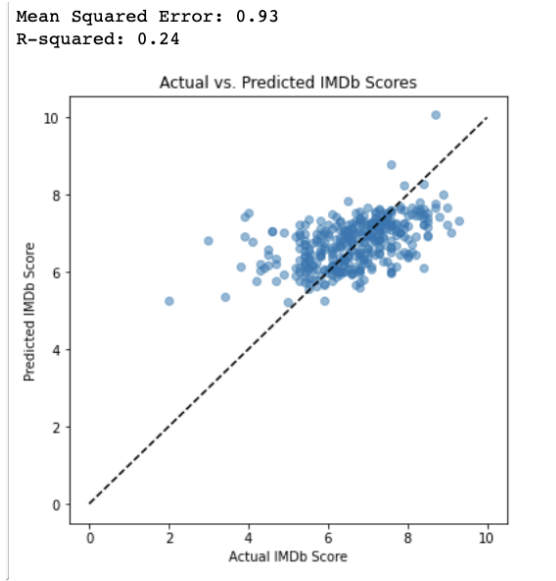
We first split the cleaned dataset into training and testing sets using an 80-20 ratio to train the model. We then used the training set to fit the linear regression model and evaluated its performance on the testing set.

We analyzed how well the linear regression model performs by looking at both metrics, the mean squared error (MSE) and the R-squared score. We used both metrics to determine how well our model predicts the IMDB scores using the TMDB scores. We can consider trying different algorithms if the model's performance isn't up to par. The MSE helps us understand the average squared difference between the predicted and actual values. A lower MSE means that our model is better fitting the data. The R-squared score shows the percentage of the variation in the target variable that can be explained by our model's input features. It ranges from 0 to 1, with a higher value indicating a better fit.

The linear regression model helps address the problem statement by identifying the relationship between features such as 'tmdb\_score' and the target variable 'imdb\_score.' By analyzing user ratings and understanding the impact of various factors on the scores, the model can provide insights informing Netflix's investment decisions for creating, acquiring, or streaming TV shows and movies. This, in turn, can help Netflix make data-driven decisions to satisfy and attract consumers, ultimately preventing significant revenue losses and optimizing its content portfolio.

The linear regression model can help Netflix by uncovering the relationships between various features, such as user ratings or content attributes, and the target variable, like IMDB scores. By analyzing these relationships, Netflix can identify trends and patterns that are associated with successful and well-received content. This knowledge can guide the company in making more informed investment decisions regarding the creation, acquisition, or streaming of TV shows and movies. Moreover, the model can help Netflix optimize its content portfolio and tailor its offerings to better match viewer preferences, which can lead to increased user satisfaction, higher viewer retention, and ultimately, improved business performance. The simplicity and interpretability of the linear regression model make it an effective starting point for exploring these relationships and extracting valuable insights from the available data.

As a result, the linear regression algorithm provides a simple and interpretable way to explore the relationship between the 'tmdb\_score' and 'imdb\_score.' While there may be more powerful or accurate algorithms for this problem, it is a helpful starting point for understanding the data and guiding further analysis. By evaluating the model's effectiveness using relevant metrics, we can decide whether to refine the current model or explore other algorithms to improve predictions.



### **K-means:**

The K-means algorithm is used for clustering tasks where the goal is to identify groups or patterns in the data. For our situation, we grouped movies and shows with similar features, such as runtime, release year, and genre, to understand the relationships between these features and their performance in IMDb scores.

K-Means Clustering can help Netflix group similar movies and shows based on their features. By identifying clusters of content with similar characteristics, Netflix can gain insights into audience preferences and identify trends or patterns in successful content. This information can guide Netflix's content acquisition and production decisions, as well as help them make more targeted recommendations to users, which can enhance user satisfaction and engagement.

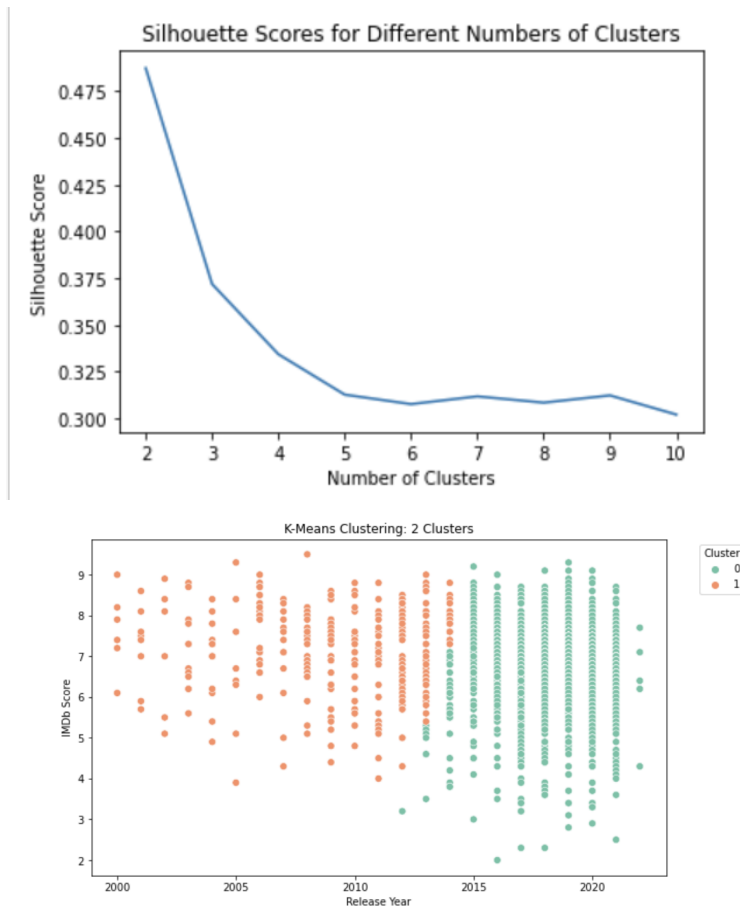
One reason we chose the K-means algorithm is due to its simplicity and scalability, making it suitable for large datasets. Additionally, it can help us uncover hidden patterns and structures in the data. We first need to preprocess the dataset to train and tune the K-means model. This involves selecting appropriate features, encoding categorical variables, and scaling numerical variables. We then determine the optimal number of clusters (K) using techniques like the elbow method or silhouette score, which involves running the K-means algorithm with different K values and evaluating the within-cluster sum of squares (WCSS) or average silhouette score.

After finding the optimal K value, we train the K-means model and assign each data point to the nearest cluster. We can then analyze the cluster centroids and the distribution of data points within each cluster to gain insights into the characteristics of different groups of movies and shows.

We can use metrics such as WCSS or silhouette score to assess the effectiveness of the K-means algorithm. Lower WCSS values or higher silhouette scores indicate better clustering performance. However, these metrics are not directly related to our problem statement (predicting IMDB scores) but can give us an idea of how well the algorithm has grouped similar movies and shows together.

By applying the K-means algorithm to our data, we can gain intelligence about the relationships between the features and the IMDB scores indirectly through the analysis of clusters. We can study the characteristics of each cluster to understand which features tend to be associated with higher or lower IMDB scores and use this information to inform our decision-making process, such as recommending movies or shows, targeting marketing efforts, or making production decisions.

The K-means analysis is not designed to predict target variables such as IMDB scores directly, but uncover patterns and relationships within the data that can be further analyzed to gain insights. If the primary goal is to predict IMDB scores, a supervised learning algorithm like Linear Regression, as discussed earlier, might be more appropriate.



### **K-NN:**

The k-Nearest Neighbors (k-NN) algorithm is a supervised learning method that can be used for both classification and regression tasks. It is based on the idea that data points with similar features will likely have identical target values. We aim to predict the IMDB scores of movies and shows based on their attributes, making k-NN a potential candidate for our problem.

K-NN can be used by Netflix to make recommendations based on content similarity. By finding the "nearest neighbors" of a given movie or show based on their features, Netflix can recommend similar content to users who enjoyed a particular title. This can improve the user experience on the platform by providing more accurate and tailored recommendations, which in turn can increase user engagement and satisfaction.

One reason to choose k-NN is its simplicity and ease of implementation. It is a non-parametric, instance-based learning algorithm that does not require any assumptions about the underlying data distribution. This can be particularly beneficial when dealing with complex, non-linear relationships between features and target values.

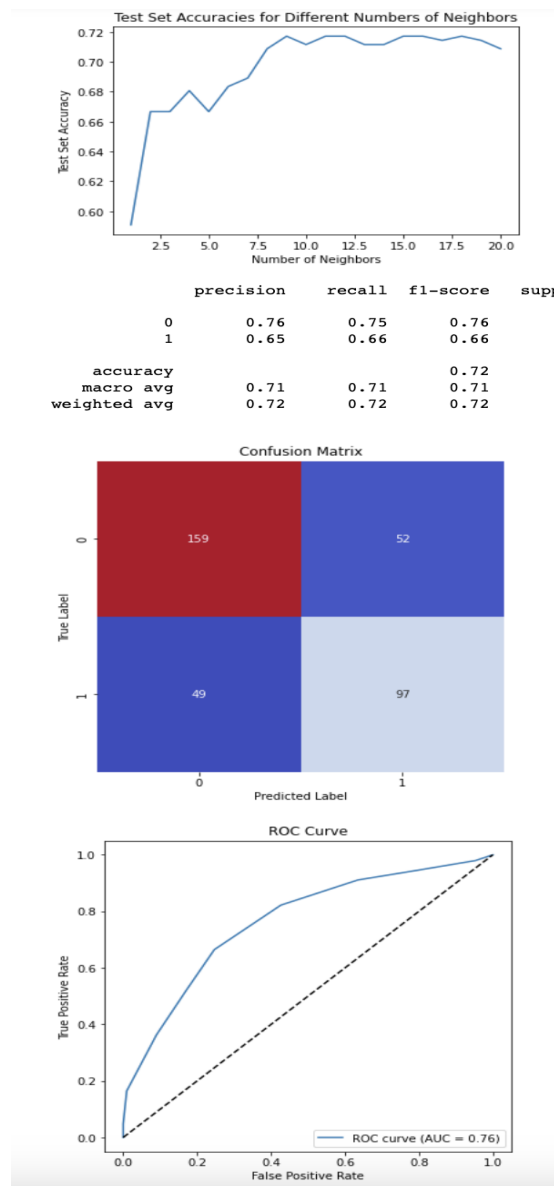
Training and tuning the k-NN model involved selecting appropriate features, encoding categorical variables, and scaling numerical variables. We then split the dataset into training and testing sets to train the model and evaluate its performance.

The primary hyperparameter to tune in a k-NN model is the number of neighbors (k). By varying the value of k, we can control the trade-off between model complexity and smoothness. A

smaller value of  $k$  can result in a more flexible model that captures local patterns, while a larger value of  $k$  can lead to a smoother decision boundary that is less sensitive to noise.

To find the optimal value of  $k$ , we used techniques like validation set to compare the performance of different  $k$  values and chose the one with the best results. When applied to our dataset, the effectiveness of the  $k$ -NN algorithm depended on its ability to capture the relationships between the input features and the IMDB scores. The algorithm had subpar accuracy in predicting the IMDB scores based on the given features. This was able to help us answer questions related to our problem statement and gain insights into the factors influencing the success of movies and shows.

In summary, the  $k$ -NN algorithm can be an appropriate choice for our problem due to its simplicity and ability to handle non-linear relationships. By tuning the model's hyperparameters and evaluating its performance using relevant metrics, we can assess its effectiveness and derive valuable intelligence from applying the algorithm to our dataset.



## Decision Tree:

The Decision Tree algorithm is a versatile supervised learning method for classification and regression tasks. It recursively splits the dataset into subsets based on the most informative feature at each step, resulting in a tree-like structure with decision nodes and leaf nodes representing the predicted target variable. Choosing a Decision Tree algorithm for predicting IMDB scores offers several advantages, including interpretability, non-linearity, and minimal data preprocessing. Decision Trees are easy to understand and visualize, enabling users to gain insights into feature-target relationships. They can handle non-linear relationships in complex



datasets and simplify the preprocessing step by accommodating missing values without requiring feature scaling or encoding.

To train and tune a Decision Tree model for our problem, we first preprocess the data by selecting relevant features, encoding categorical variables, scaling numerical variables, and handling missing values using imputation or other techniques. Next, we split the dataset into training and testing sets for model training and performance evaluation. We then create and train a Decision Tree model by fitting it to the training data, which involves recursively splitting the data and constructing the tree structure. Since Decision Trees can be prone to overfitting, we adjust hyperparameters such as the maximum depth, minimum samples per split, and minimum samples per leaf to balance model complexity and performance. Finally, we evaluate the model by predicting IMDB scores for the test dataset and comparing these predictions to the actual IMDB scores, assessing the model's effectiveness using relevant metrics like mean squared error (MSE) for regression tasks, or accuracy, precision, recall, and F1 score for classification tasks.

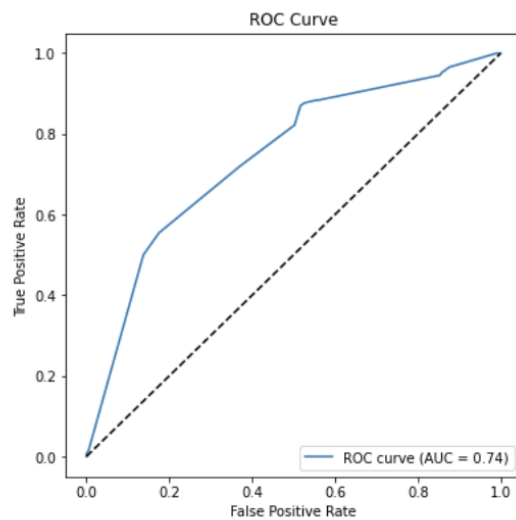
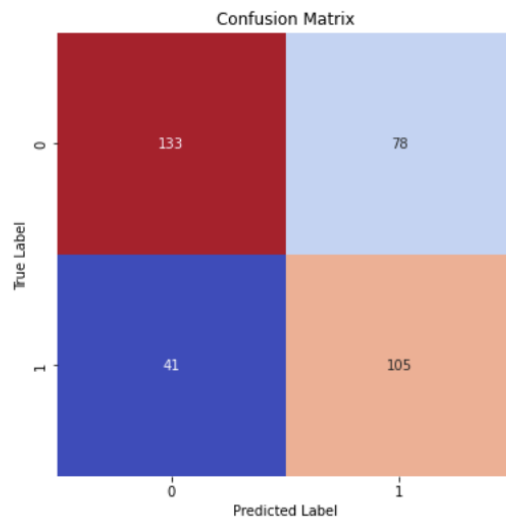
When applied to our data, the Decision Tree algorithm can help us answer questions related to our problem statement by providing an interpretable model that captures the relationships between the input features and the target variable (IMDB scores). By analyzing the relevant metrics and visualizing the decision tree structure, we can learn which features have the most significant impact on IMDB scores and how these features interact.

A Decision Tree model is a robust and interpretable machine learning algorithm that can help address the problem statement by predicting the success of TV shows and movies based on various features. Here's how the Decision Tree model can help with each part of the problem statement:

By training a Decision Tree model on historical data, Netflix can identify which features contribute most to the success of a TV show or movie. The model can help predict the potential success of a new production based on factors such as genre, production budget, cast, director, and more. This information can guide Netflix in making informed decisions about which projects to invest in and which to avoid, thereby maximizing return on investment.

In summary, the Decision Tree algorithm is suitable for our problem due to its interpretability, ability to handle non-linearity, and minimal data preprocessing requirements. By tuning and training the model effectively, we can gain valuable insights into the factors influencing IMDB scores and make accurate predictions.

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.76      | 0.63   | 0.69     | 211     |
| 1            | 0.57      | 0.72   | 0.64     | 146     |
| accuracy     |           |        | 0.67     | 357     |
| macro avg    | 0.67      | 0.67   | 0.66     | 357     |
| weighted avg | 0.69      | 0.67   | 0.67     | 357     |



**Citation:** *Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.*