30/03/2025

Project By: Erez Levy

ML Model Prediction - TMDB

The TMDB Dataset

The TMDB (The Movie Database) is a widely-used resource for movie and TV show data, providing valuable information such as ratings, plot summaries, and more.

This dataset contains a collection of 150,000 tv shows from the TMDB database, collected and cleaned.

Using the Google Colab platform I suggest a predictive model to determine the success of a TV show based on features like vote count, vote average, and popularity. We'll approach this as a regression problem, where we predict a continuous success metric, and we'll use multiple regression models

The Project Overview:

This dataset opens up a wide range of possibilities for data analysts and data scientists. Here are some ideas to get you started:

Explore trends in TV show popularity based on vote count and average. Analyze TV show genres to identify the most popular genres or combinations of genres. Investigate the relationship between TV show ratings and the number of seasons and episodes. Build a recommendation system that suggests TV shows based on a user's favorite genres or languages. Predict the success of a TV show based on features like vote count, average, and popularity. Identify the most prolific TV show creators or production companies based on the number of shows they have created. Explore the distribution of TV show run times and investigate whether episode duration affects the overall ratings. Investigate TV show production trends across different countries and networks. Analyze the relationship between TV show language and popularity, and investigate the popularity of non-English shows. Track the status of TV shows (in production or not) and analyze their popularity over time. Develop a language analysis model to identify sentiment or themes from TV show overviews.

The Approach:

1. Data Preparation :

- **Uniting Tables:** Combine different data sources into a unified dataset, ensuring consistency and alignment of data points**.**
- **Reduce Large Categories:** Simplify datasets by reducing the number of categories in categorical variables where necessary. This might involve grouping smaller categories into an "Other" category.
- **Clean Text:** Perform text cleaning by removing stop words, punctuation, special characters, and normalizing text (e.g., lowercasing). • Transform/Manipulate data

2. Exploratory Data Analysis (EDA):

- **Instant Reports:** Generate initial reports to get a quick overview of the dataset, including summary statistics and distributions**.**
- **Descriptive Analysis:** Use statistical methods and visualizations to understand the data, identify patterns, trends, and potential issues.
- **Correlation and other relationship Analysis:** between different features to identify multicollinearity and feature importance.

3. Data Cleansing :

- **Outliers:** Detect and handle outliers which might skew the data analysis and model performance. This might involve removing, transforming, or capping outlier values. •
- **Missing Values:** Address missing data through imputation methods (e.g., mean, median, mode) or by removing incomplete records, depending on the context and importance of

the missing information.

4. Feature Engineering & Feature Selection

- **Enriching:** Create new features from existing ones to better capture the underlying patterns in the data. This can involve mathematical transformations, aggregations, or domain-specific knowledge.
- **Normalization/Standardization:**Scale numerical features to ensure they have similar ranges, which helps certain algorithms perform better**.**
- **Feature Selection:** Choosing the most valuable features that predict the target value by running penalty models.

5. One-Hot Encoding

- Convert categorical variables into a format that can be provided to ML algorithms to do a better job in prediction. This typically involves creating binary columns for each category in a categorical feature.

6. Model Selection and Fine Tuning

- **Model Selection:** Choose appropriate machine learning models based on the problem at hand (e.g., regression, classification, clustering).
- **Cross-Validation:** Use cross-validation techniques to assess model performance and ensure robustness.
- **Fine Tuning:** Optimize model hyperparameters to improve performance, typically using methods like grid search, random search, or Bayesian optimization.

7. Model Evaluation

- **Performance Metrics:** Evaluate the model using appropriate metrics such as accuracy, precision, recall, F1-score, ROC-AUC for classification, or RMSE, MAE for regression.
- **Validation:** Validate the model on a separate validation set to ensure it generalizes well to unseen data.

The Methods

1.Data Collection & Preperation:

The first step in dataset preparation is data collection. This involves gathering all relevant data that will be used to train and test the machine learning model. Sources might include:

- **Databases:** Structured data from existing databases, such as customer profiles and transaction records.
- **APIs:** Data from public or private APIs, like social media or customer feedback platforms.

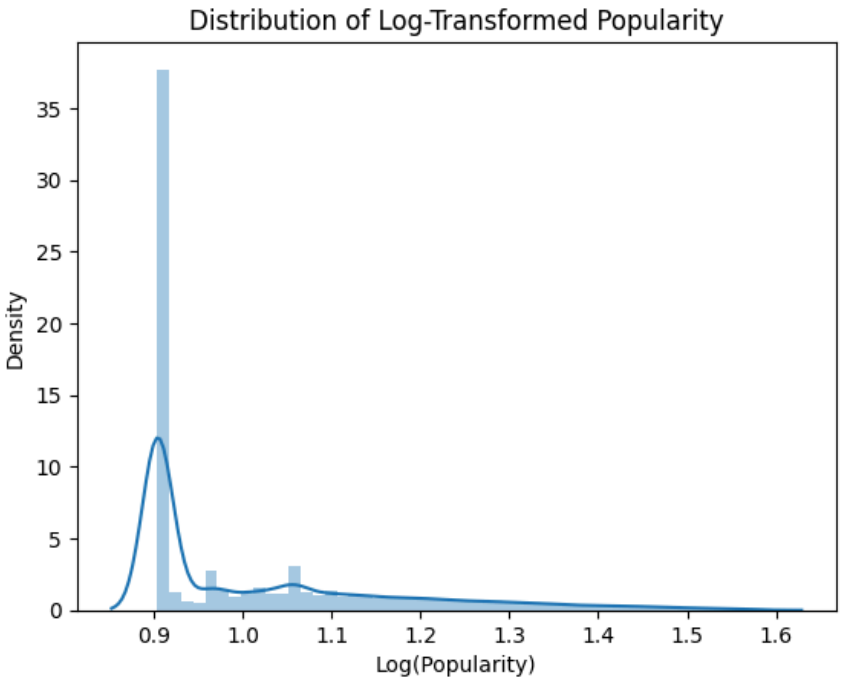
- **CSV/Excel Files:** Data stored in spreadsheets, possibly containing historical customer interaction data.
- **Web Scraping:** Extracting data from websites if necessary, to gather competitive market analysis.

Target Value prediction

Before diving into data preparation, it's essential to have a clear understanding of the problem the model aims to solve. This helps in identifying the relevant data features that will be crucial for the model's predictive capabilities.My project focuses on predicting customer churn in a telecommunications company, making it vital to select features such as customer usage patterns and contract details.

Based on the potential insights and business value, I would suggest focusing on predicting the popularity value.

Target Value-Popularity histogram:



It is a complex and dynamic metric that reflects overall success. Vote Average (Rating): It captures audience satisfaction and critical acclaim. Both of these targets have valuable real-world implications and can be approached with a variety of machine learning models.

Important Considerations:

Feature Engineering: Carefully select and engineer features from the TMDB data that you think will be most relevant to your chosen target variable.

Model Selection: Experiment with different machine learning models (regression for popularity or ratings, classification for status/renewal) to find the best performer.

Evaluation: Use appropriate metrics (like RMSE for regression or accuracy for classification) to assess the performance of your predictive model

2. Exploratory Data Analysis (EDA):

Descriptive statistics are used to summarize and describe the main features of a dataset. They provide a concise overview of the data, helping you understand its central tendency, dispersion, and shape.

1. Measures of Central Tendency:

- **Mean:** The average of all values in a dataset.
- **Median:** The middle value when the data is sorted.
- **Mode:** The most frequent value in a dataset.

These measures tell you where the "center" of your data lies.

2. Measures of Dispersion (or Variability):

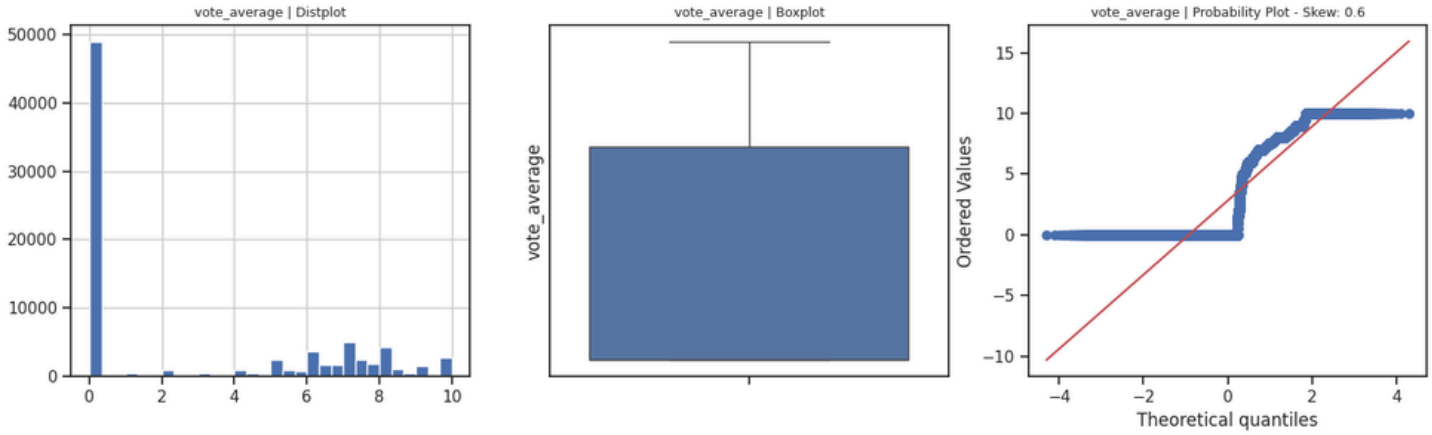
- **Range:** The difference between the maximum and minimum values.
- **Variance:** The average of the squared differences from the mean.
- **Standard Deviation:** The square root of the variance.

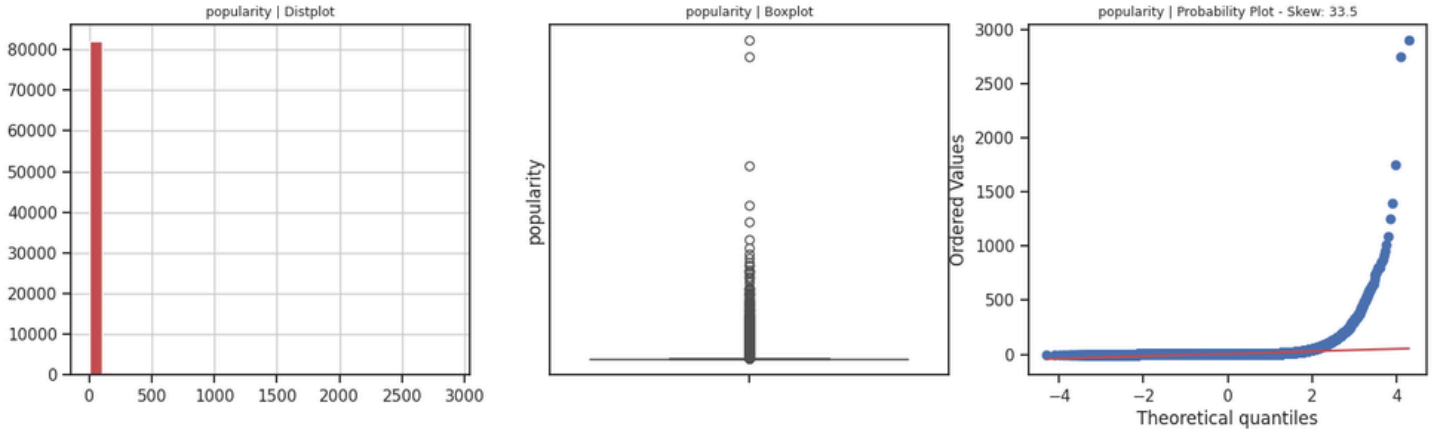
These measures describe how spread out the data is. A higher standard deviation indicates more variability.

3. Shape of Distribution:

- **Skewness:** Measures the asymmetry of the data distribution.
- **Kurtosis:** Measures the "peakedness" of the data distribution.

These measures provide insights into the overall shape of the data distribution.





Box Plots and Outliers

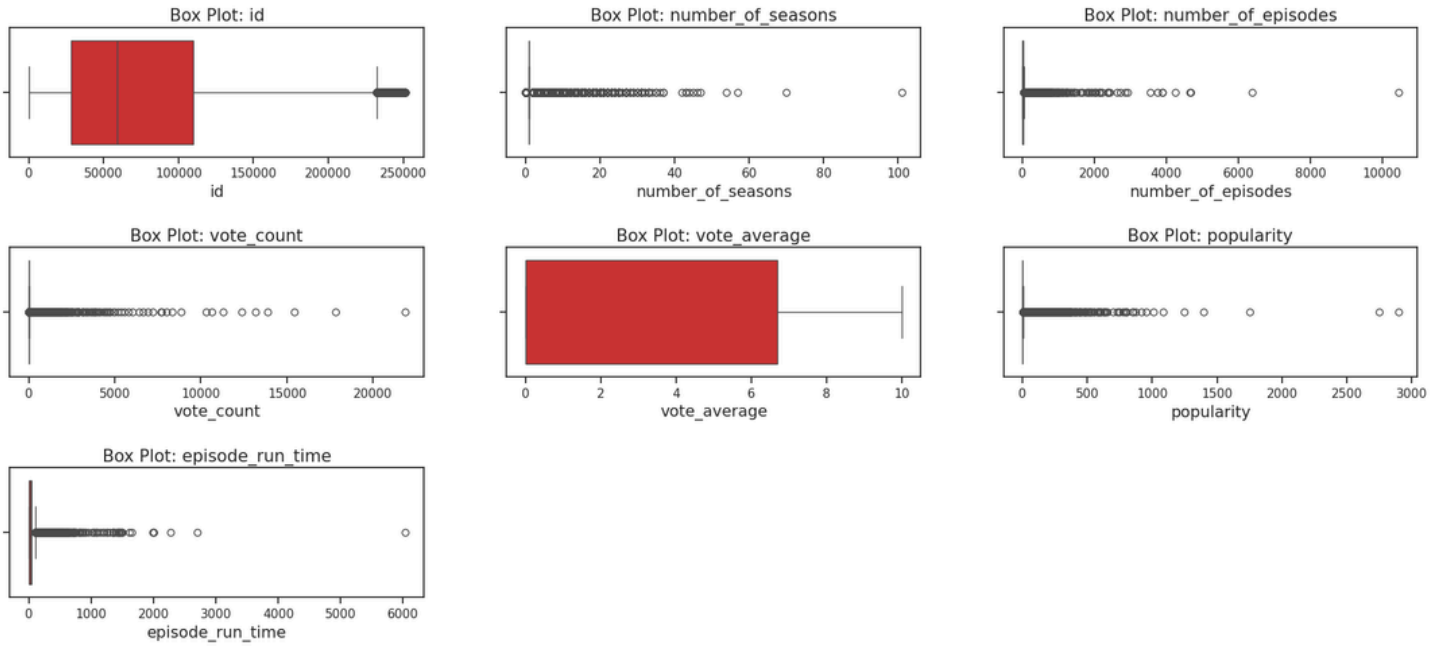
A box plot (also called a box-and-whisker plot) is a graphical representation of the distribution of a dataset. It displays the following key elements:

- **Box:** Represents the interquartile range (IQR), which contains the middle 50% of the data.
- **Line inside the box:** Represents the median (the middle value).
- **Whiskers:** Extend from the box to the minimum and maximum values within a certain range.
- **Outliers:** Data points that fall outside the whiskers are considered potential outliers and are plotted as individual points.

Identifying Outliers using Box Plots

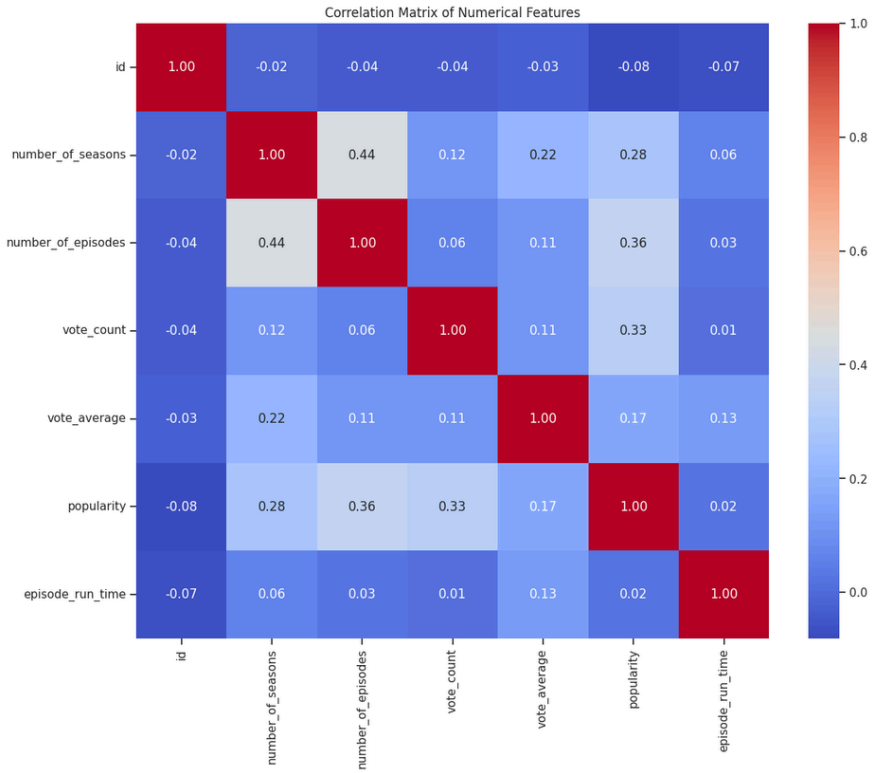
Outliers are typically identified using the following rule:

1. **Calculate the IQR:** The IQR is the difference between the third quartile (Q3) and the first quartile (Q1).
2. **Determine the upper and lower bounds:**
   - Upper bound: Q3 + 1.5 * IQR
   - Lower bound: Q1 - 1.5 * IQR
3. **Identify outliers:** Any data point that falls above the upper bound or below the lower bound is considered a potential outlier.



Correlation and other relationship Analysis:

between different features to identify multicollinearity and feature importance.



3. Data Cleansing

Data cleansing, also known as data cleaning, is the process of identifying and correcting or removing errors, inconsistencies, and inaccuracies in a dataset. This is a crucial step in data analysis and machine learning as it ensures the quality and reliability of your data, leading to more accurate and meaningful results.

Once the data is collected, it’s crucial to clean it. Data cleaning involves:

- **Handling Missing Values:** Decide whether to impute missing values or remove records with missing data, based on their impact on model accuracy.
- **Removing Duplicates:** Ensure that duplicate entries are eliminated to avoid bias, particularly in customer records.
- **Correcting Errors:** Fix any errors or inconsistencies in the data, such as typos or incorrect values in billing information.
- **Dropping Incomplete Records (Optional):** I considered dropping rows with a significant number of missing values, but I ultimately decided against it.
- **Identifying Missing Values:** You used tmdb_eda.isnull().sum() to check for missing values in each column of your DataFrame.
- **Imputation:** You filled in missing numerical values using the mean or median, depending on the skewness of the data. Categorical missing values were imputed using the mode (most frequent value).
- **Detection:** You used Z-score and IQR methods to identify outliers in numerical features like 'popularity', 'number_of_episodes', 'number_of_seasons', 'vote_count', and 'episode_run_time'.
- **Treatment:** I employed techniques like capping outliers at certain percentiles and using transformations (e.g., log transformation for 'popularity') to reduce the impact of extreme values.

Method : Zscore

The zscore function, typically found in libraries like scipy.stats in Python, is used to calculate the Z-score for each element of an array or series.

What is a Z-score?

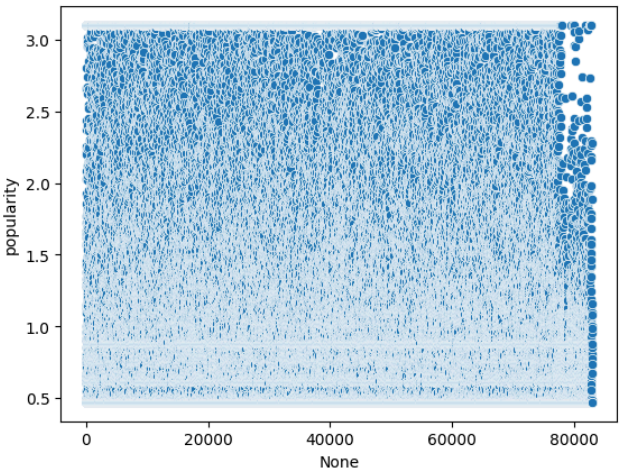
A Z-score (also called a standard score) is a statistical measure that tells you how many standard deviations away a data point is from the mean of its distribution.

1. **Centering:** The zscore function first centers the data by subtracting the mean (μ) of the distribution from each data point (x).
2. **Scaling:** It then scales the centered data by dividing by the standard deviation (σ) of the distribution.

The zscore insights:

- **Outlier Detection:** Z-scores help identify outliers. Data points with very high or very low Z-scores (e.g., above 3 or below -3) are often considered outliers.
- **Data Standardization:** Z-scores standardize your data, making it easier to compare and analyze data from different distributions.
- **Understanding Relative Position:** Z-scores help understand the relative position of a data point within its distribution.

The Dataset ‘popularity’ scatter plot:



Example: The ‘popularity’ zscore results:

0 0.748876

1 0.847949

2 0.748876

3 1.678422

4 0.748876

...

82867 0.092436

82868 0.407415

82869 0.027693

82870 0.363144

82871 0.408633

Name: popularity, Length: 82872, dtype: float64

Data Type Conversion:

- **Boolean to Integer:** You converted boolean columns to integers (0 and 1) for consistency.
- **Date/Time Conversion:** You converted 'first_air_date' and 'last_air_date' to datetime objects, calculated 'air_time' (duration in days), and added it as a new column.

Handling Inconsistent Data:

- No specific code for handling inconsistent data, like duplicate records or data entry errors. However, this can also be part of data cleansing.

Creating Dummies variables:

Dummy variables are a way to represent categorical data (like TV show genres, languages, or status) as numerical values that machine learning models can understand. They are essentially binary (0 or 1) variables that indicate the presence or absence of a particular category.

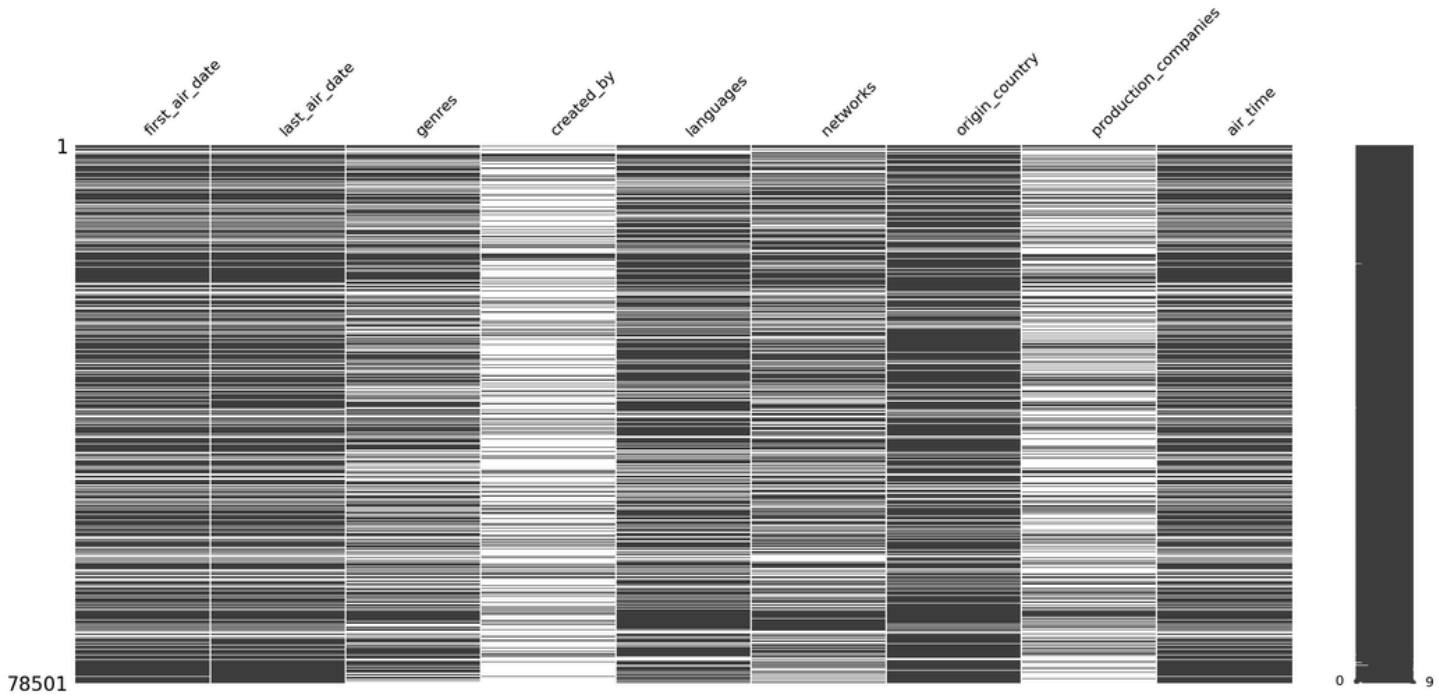
Most machine learning algorithms work with numerical data. They can't directly handle categorical data in its original form (e.g., text labels). Dummy variables convert categorical data into a numerical format that these algorithms can process.

- **Creating dummies for the ‘adult’ categorial column**.
- **Converting 'first_air_date' & 'last_air_date' to datetime objects & adding Column 'air_time'**

Data Type: Converting to datetime objects allows you to perform date and time-based calculations and analysis (like finding the difference between dates).

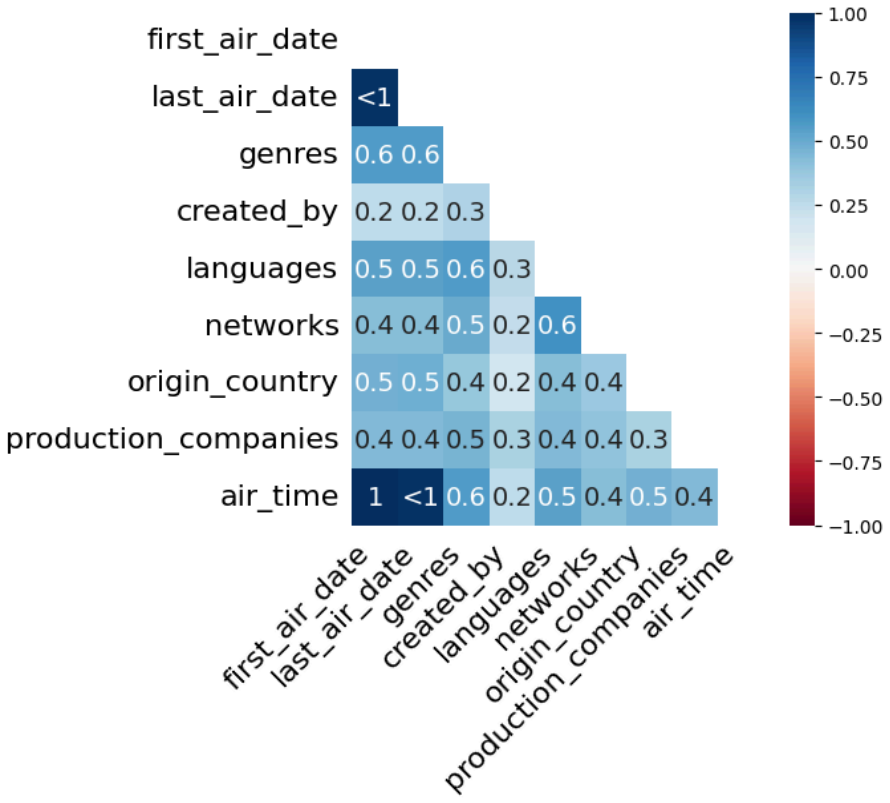
Feature Engineering: The 'air_time' column you create is a new feature that might be useful for your predictive model. It represents the total duration a TV show has been on air, which could be related to its popularity or success.

Dataset Missing Values Analysiss :



Missingness Correlation HeatMap

A missingness correlation heatmap is a visualization that helps you understand the relationships between missing values in different columns of your dataset. It shows the correlation between the presence or absence of data in one column and the presence or absence of data in another column.



Missing Values Imputation:

Missing values imputation is the process of filling in missing data points in your dataset with estimated values. It's an important step in data preprocessing because many machine learning algorithms cannot handle missing values directly.

Using two main imputation methods:

1.Mean/Median Imputation:

For numerical features: If the data is normally distributed (low skewness), missing values are filled with the mean of the column. If the data is skewed, missing values are filled with the median of the column.

Explanation: Mean imputation is appropriate for normally distributed data as it preserves the overall distribution.Median imputation is more robust to outliers and is preferred for skewed data.

2.Mode Imputation:

For categorical features:missing values are filled with the mode (most frequent value) of the column.

Explanation: Mode imputation is the most common approach for categorical data as it replaces missing values with the most likely category.

Imputation Concideration :

The reduction with the Original dataset forabout 25% I have decided not to use the new Imputed Dataset.

(The Analysis file)

Index: 78501 entries, 0 to 82871

Data columns (total 19 columns):

#	Column	Non-Null Count	Dtype
0	name	78501 non-null	object
1	id	78501 non-null	float64
2	number_of_seasons	78501 non-null	int64
3	number_of_episodes	78501 non-null	int64
4	vote_count	78501 non-null	int64
5	vote_average	78501 non-null	float64
6	first_air_date	58633 non-null	datetime64[ns]
7	last_air_date	59105 non-null	datetime64[ns]
8	popularity	78501 non-null	float64
9	type	78501 non-null	object
10	genres	78501 non-null	object
11	created_by	78501 non-null	object
12	languages	78501 non-null	object
13	networks	78501 non-null	object
14	origin_country	78501 non-null	object
15	production_companies	78501 non-null	object
16	episode_run_time	78501 non-null	int64
17	adult_True	78501 non-null	int64
18	air_time	78501 non-null	float64

dtypes: datetime64[ns](2), float64(4), int64(5), object(8)

memory usage: 12.0+ MB

<class 'pandas.core.frame.DataFrame'>

Index: 58587 entries, 1 to 82871

Data columns (total 19 columns):

#	Column	Non-Null Count	Dtype
0	name	58587 non-null	object
1	id	58587 non-null	float64
2	number_of_seasons	58587 non-null	int64
3	number_of_episodes	58587 non-null	int64
4	vote_count	58587 non-null	int64
5	vote_average	58587 non-null	float64
6	first_air_date	58587 non-null	datetime64[ns]
7	last_air_date	58587 non-null	datetime64[ns]
8	popularity	58587 non-null	float64
9	type	58587 non-null	object
10	genres	58587 non-null	object
11	created_by	58587 non-null	object
12	languages	58587 non-null	object
13	networks	58587 non-null	object
14	origin_country	58587 non-null	object
15	production_companies	58587 non-null	object
16	episode_run_time	58587 non-null	int64
17	adult_True	58587 non-null	int64
18	air_time	58587 non-null	float64

dtypes: datetime64[ns](2), float64(4), int64(5), object(8)

memory usage: 8.9+ MB

(58587, 19)

Data cleansing summery:

By performing these data cleansing steps, I improved the quality and consistency of the TV show data, which will be essential for building accurate and reliable predictive models in the subsequent stages of my project.

4.Feature Eng. & Feature Selection

Data transformation involves converting data into a suitable format or structure for analysis. This can include:

- **Normalization/Standardization:** Scaling numerical features to a standard range or distribution, ensuring that features such as call duration are on comparable scales.
- **Encoding Categorical Variables:** Converting categorical data into numerical format using techniques like one-hot encoding, particularly for features like customer region or type of service plan.
- **Feature Engineering:** Creating new features or modifying existing ones to improve model performance, such as aggregating usage over time to identify trends.

Enriching:

Creating new features from existing ones to better capture the underlying patterns in the data. This can involve mathematical transformations, aggregations, or domain-specific knowledge

Normalization/Standardization:

Bringing Features to a Common Scale -In machine learning, many algorithms perform better when numerical features have similar ranges. This is because features with larger ranges can disproportionately influence the model, leading to biased results. Normalization and standardization are two common techniques used to address this issue.

Benefits:

1. **Algorithm Performance:** Many machine learning algorithms, such as distance-based methods (k-Nearest Neighbors, K-Means), gradient descent-based algorithms (Linear Regression, Logistic Regression), and Support Vector Machines, are sensitive to the scale of features. When features have vastly different ranges, those with larger values can dominate the calculations, leading to suboptimal performance.
2. **Equal Feature Contribution:** Scaling ensures that all features contribute equally to the model's learning process. Without scaling, features with larger ranges might be given

more importance than those with smaller ranges, even if they are not inherently more informative.
3. **Faster Convergence:** In optimization algorithms like gradient descent, scaling can help the algorithm converge faster to the optimal solution. This is because the gradients are less likely to oscillate wildly when features are on a similar scale.

The Scale Numerical Features

Using StandardScaler - A Standardization Technique

StandardScaler is a preprocessing technique provided by the scikit-learn library in Python. It's used to standardize features by removing the mean and scaling to unit variance. In simpler terms, it transforms the data such that it has a mean of 0 and a standard deviation of 1.

The Method for StandardScaler

1. **Calculating Mean and Standard Deviation:** StandardScaler first calculates the mean (μ) and standard deviation (σ) of each feature in your dataset.
2. **Centering the Data:** It then centers the data by subtracting the mean from each value: X_centered = X - μ. This shifts the data so that the mean becomes 0.
3. **Scaling to Unit Variance:** Next, it scales the data by dividing each centered value by the standard deviation: X_scaled = X_centered / σ. This ensures that the standard deviation becomes 1.

Convert categorical variables:

Convert categorical variables into a format that can be provided to ML algorithms to do a better job in prediction. This typically involves creating binary columns for each category in a categorical feature.

Numeric (Continuous) Analysis:

Numeric (continuous) analysis involves examining numerical features in the dataset to understand their distributions, relationships, and potential impact on the machine learning models. It's a crucial step in the data exploration and preprocessing phase of a data science project.

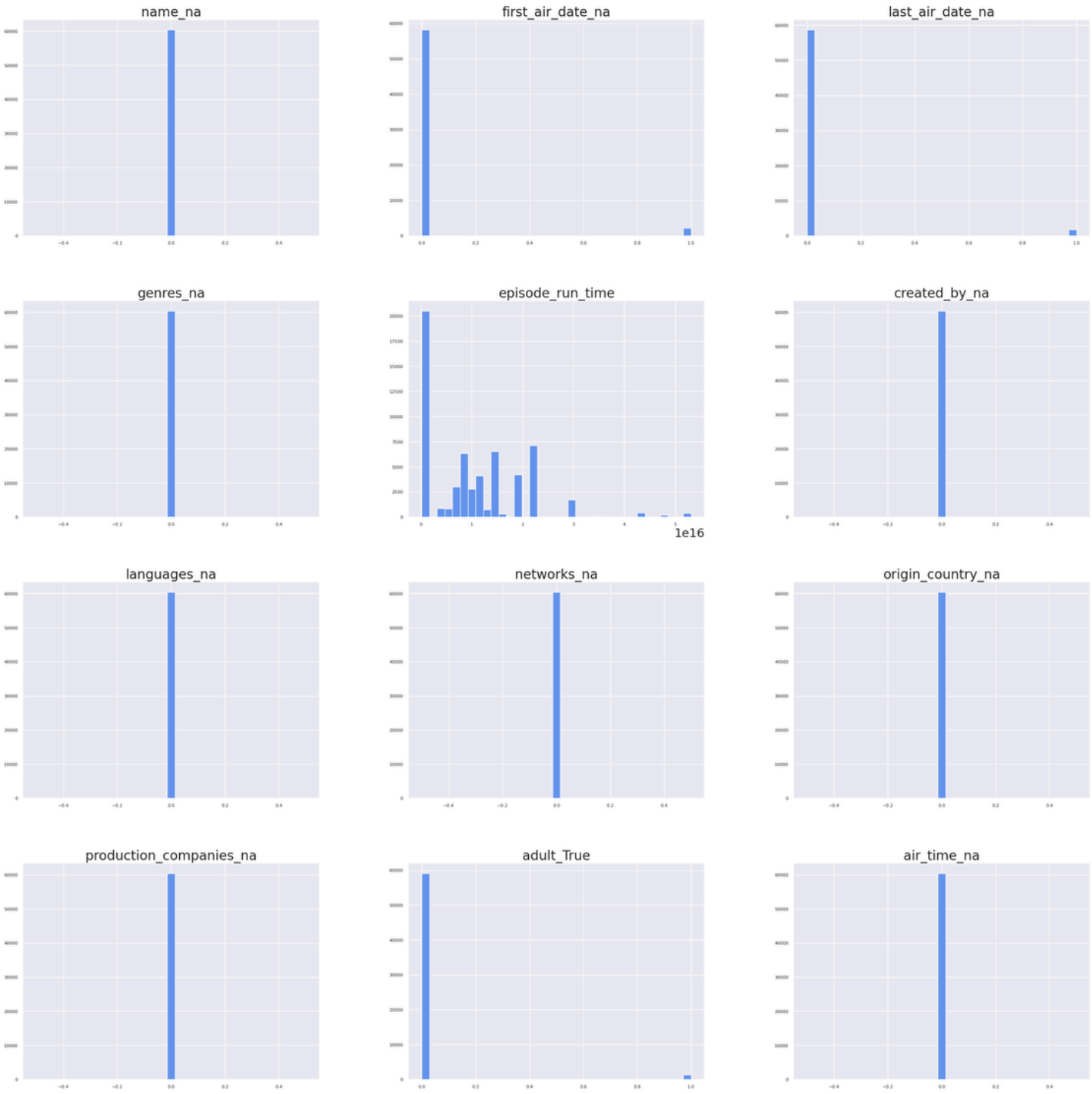
Convert specified columns to float:

In data analysis and machine learning, it's often necessary to ensure that numerical columns are represented as float data types. This is because many algorithms and operations expect or perform better with float values. Here are some reasons for conversion:

Mathematical Operations: Performing calculations like addition, subtraction, multiplication, and division on numerical data often requires float values for accuracy.

Algorithm Compatibility: Many machine learning algorithms expect features to be represented as floats for proper functionality and performance.

Data Consistency: Maintaining consistency in data types can simplify data manipulation and analysis.



Feature Selection: Choosing the Right Ingredients for the Model

Feature selection is the process of selecting a subset of relevant features (variables, predictors) to use in building a machine learning model. It's a crucial step in the model development process because it can:

Improve Model Performance: By removing irrelevant or redundant features, you can reduce noise and overfitting, leading to better generalization and accuracy.

Reduce Computational Cost: Fewer features mean faster training and prediction times, especially for complex models and large datasets.

Enhance Model Interpretability: Simpler models with fewer features are easier to understand and explain, making them more trustworthy and actionable.

Methods for Feature Selection:

There are three main categories of feature selection methods:

1. **Filter Methods:** These methods evaluate the relevance of features based on their statistical properties, independent of the chosen machine learning algorithm.
    - **Examples:**
        - **Correlation:** Selecting features highly correlated with the target variable.
        - **Chi-squared test:** Evaluating the dependence between categorical features and the target variable.
        - **ANOVA (Analysis of Variance):** Comparing the means of the target variable across different categories of a feature.
2. **Wrapper Methods:** These methods use a specific machine learning algorithm to evaluate the performance of different feature subsets. They are computationally more expensive but often lead to better results.
    - **Examples:**
        - **Recursive Feature Elimination (RFE):** Iteratively removing the least important features based on the model's coefficients or feature importance scores.
        - **Sequential Feature Selection (SFS):** Starting with an empty set and adding features one by one based on their contribution to model performance.
3. **Embedded Methods:** These methods incorporate feature selection as part of the model training process. They are often more efficient than wrapper methods and can identify interactions between features.
    - **Examples:**
        - **LASSO (Least Absolute Shrinkage and Selection Operator):** A linear regression model that uses regularization to shrink some coefficients to zero, effectively performing feature selection.
        - **Decision Tree-based algorithms:** These algorithms inherently select features based on their importance in splitting the data.

Choosing the Right Method:

The choice of feature selection method depends on several factors, including:

- **Dataset size and complexity:** Filter methods are generally faster for large datasets, while wrapper methods might be more suitable for smaller datasets.
- **Model type:** Some models have built-in feature selection capabilities, while others require external methods.
- **Desired level of accuracy:** Wrapper methods often achieve higher accuracy but are computationally more demanding.
- **Interpretability requirements:** Simpler models with fewer features are easier to interpret.

Using 5 models for evaluation :

```
'Feature': tmdb_eng.drop(columns=['popularity']).columns, # Use original column names
'Lasso': lasso_selected,
'SVM': svm_selected,
'GradientBoost': gb_selected,
'RandomForest': rf_selected,
'Ridge': ridge_selected
```

Feature Selection:

Found 13 columns for the prediction 'popularity' column:

```
<class 'pandas.core.frame.DataFrame'>
Index: 78501 entries, 0 to 82871
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  78501 non-null  float64
 1   number_of_episodes  78501 non-null  float64
 2   vote_average        78501 non-null  float64
 3   first_air_date      58633 non-null  datetime64[ns]
 4   last_air_date       59105 non-null  datetime64[ns]
 5   episode_run_time    78501 non-null  int64
 6   air_time            78501 non-null  float64
 7   log_popularity      78501 non-null  float64
 8   total_runtime       78501 non-null  float64
```

9 vote_count_to_episodes_ratio 60393 non-null float64

10 weighted_vote_average 78501 non-null float64

11 popularity_score 78501 non-null float64

12 popularity 78501 non-null float64

dtypes: datetime64[ns](2), float64(10), int64(1)

memory usage: 8.4 MB

Multivariable Analysis

Summarization and Selection of Variables

The Lasso penalty here is controlled by alpha = 0.01. This penalty forces some coefficients to shrink to zero, effectively performing feature selection. A higher penalty (larger alpha) would result in more coefficients being zeroed out, and a lower penalty (smaller alpha) would retain more features

Corroletion TMDB Dataset

Correlation measures the strength and direction of a linear relationship between two variables. In the TMDB dataset, this could be the relationship between features like:

vote_average and popularity:

Do movies with higher average ratings tend to be more popular?

number_of_episodes and total_runtime:

Does the number of episodes in a show correlate with its total runtime?

vote_count and popularity_score:

Is there a connection between the number of votes a show receives and its calculated popularity score?

Spearman's Rank Correlation

The method='spearman' argument specifies that Spearman's rank correlation is used. This method is non-parametric, meaning it doesn't assume a normal distribution of the data. It assesses the monotonic relationship between variables – whether they tend to increase or decrease together, regardless of the specific shape of the relationship.

Interpreting Correlation Values

The correlation coefficient ranges from -1 to +1:

+1: Perfect positive correlation. As one variable increases, the other increases proportionally. -1: Perfect negative correlation. As one variable increases, the other decreases proportionally. 0: No correlation. There is no linear relationship between the variables. Values between -1 and +1 indicate varying degrees of correlation:

Strong correlation: Values closer to -1 or +1 (e.g., -0.8, +0.7) Moderate correlation: Values around -0.5 or +0.5 Weak correlation: Values closer to 0 (e.g., -0.2, +0.3)

Example Interpretation

If the correlation between vote_average and popularity is +0.7, it suggests a strong positive correlation. This means that TV shows with higher average ratings tend to be more popular.

Important Considerations

Correlation does not imply causation:

Even if two variables are strongly correlated, it doesn't mean that one causes the other. There could be other factors influencing both variables.

Context is crucial:

The interpretation of correlation should be based on the specific dataset and domain knowledge. A correlation of 0.5 might be considered strong in some contexts but weak in others. How to use the correlation results:

Feature Selection:

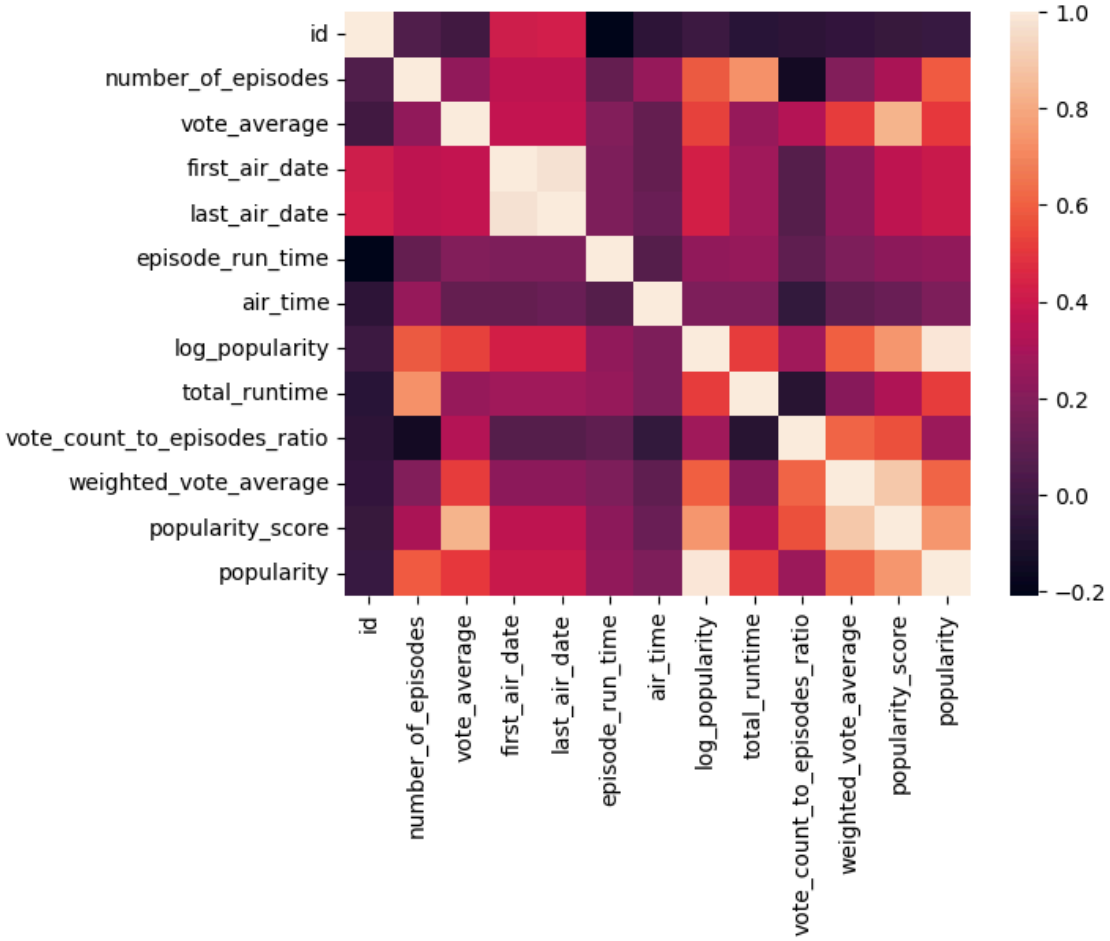
Identify features that are strongly correlated with your target variable (e.g., popularity). These features are likely to be important predictors in your machine learning model. Feature Engineering:

Consider creating new features by combining correlated features. For example, you could create a feature that combines 'vote_average' and 'vote_count' to capture a more comprehensive measure of audience approval.

Multicollinearity:

Very strong correlations between predictor features (features you are using to predict your target), you might want to remove one of the highly correlated features to avoid multicollinearity issues in your model. Multicollinearity can make it difficult to interpret the individual effects of features and can lead to unstable model estimates.

Heat Map TMDB Dataset



Chi-Square testing between episode_run_time & popularity

The Chi-Square test compares the observed frequencies of different combinations of categories in your data to the expected frequencies if there were no relationship between the variables. If the observed frequencies deviate significantly from the expected frequencies, it suggests that there is an association between the variables.

Chi-Square Test Results:

Chi-Square Statistic: 871193.5373

P-Value: 0.0000

Degrees of Freedom: 722190

Example (Correlation)

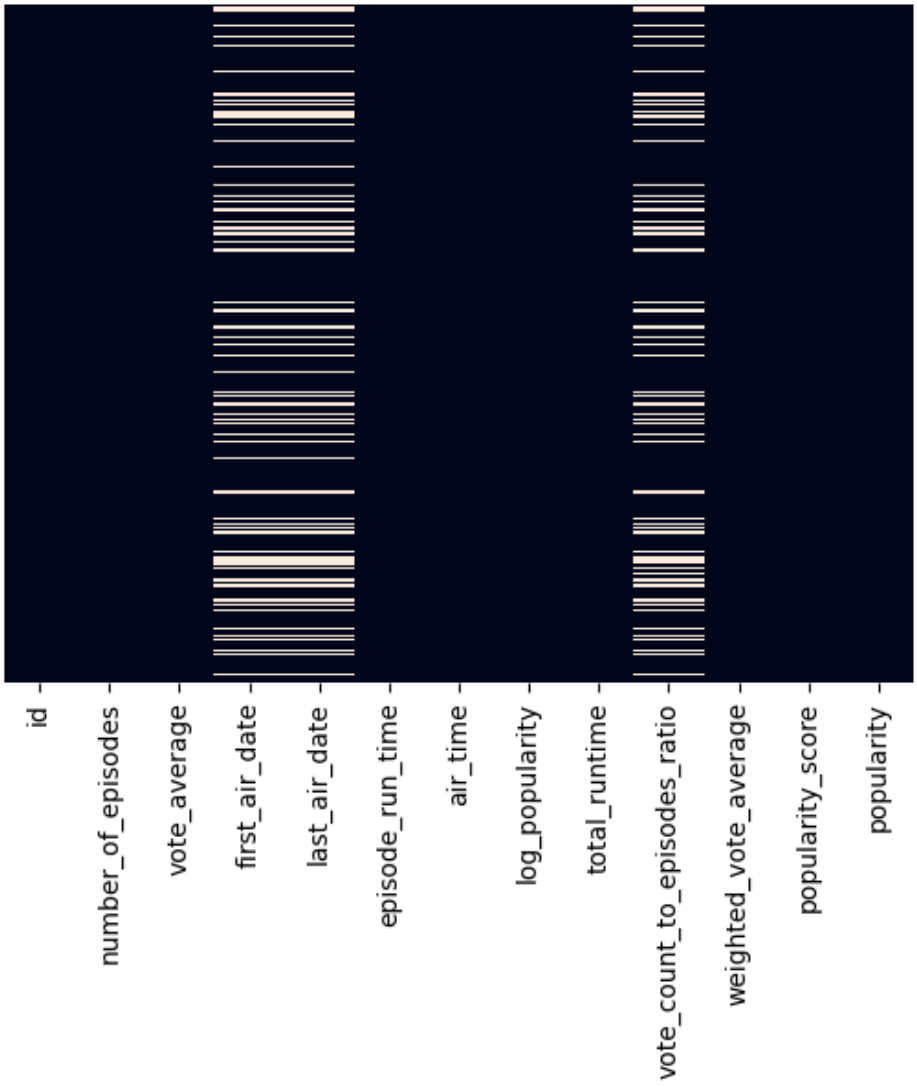
Analyzing the correlation between a TV show's and its popularity in tmdb_eda DataFrame. If the a p-value of 0.0000 for this correlation, it means there's a statistically significant relationship between popularity and vote average.

Missing values in TMDB

Missing data is a common occurrence in real-world datasets like TMDB. It means that for certain TV shows, some information might not have been collected or is unavailable. In your code, you are using the tmdb_hot DataFrame, which is a copy of your original dataset likely after some initial processing.

Identifying Missing Values : usesing these methods to identify missing data:

1. **tmdb_hot.isnull().head()**: This displays the first few rows of a DataFrame where True indicates a missing value and False indicates a non-missing value for each cell. It provides a quick visual check.
2. **sns.heatmap(tmdb_hot.isnull(), yticklabels=False, cbar=False)**: This creates a heatmap visualization where missing values are represented by a different color (usually darker). It gives a broader overview of the missing data pattern across your dataset.



Understanding Missing Data in TMDB

Missing data is a common occurrence in real-world datasets like TMDB. It means that for certain TV shows, some information might not have been collected or is unavailable. using the tmdb_hot DataFrame, which is a copy of the original dataset likely after some initial processing.

Handling Missing Data

While my code doesn't explicitly show how you're *handling* the missing data, here are common approaches to dealing with missing values, and how they might apply to your TMDB analysis:

1. **Deletion:**
    - **Row Deletion (Listwise Deletion):** If a row has any missing values, I can remove the entire row from the dataset. This is simple but can lead to a significant loss of data, especially if missing values are widespread.
    - **Column Deletion:** If a column has a high percentage of missing values, I might consider removing the entire column. However, you should be cautious as you might lose valuable information.
2. **Imputation:** This involves filling in missing values with estimated values. Common techniques include:
    - **Mean/Median/Mode Imputation:** Replacing missing values with the mean (for numerical data), median (for skewed numerical data), or mode (for categorical data) of the respective column.
    - **Regression Imputation:** Using a regression model to predict missing values based on other features.
    - **K-Nearest Neighbors (KNN) Imputation:** Using the values of the k-nearest neighbors to estimate the missing value.

Calculate correlations with outliers - Understanding the Impact of Outliers on Correlation.

Outliers are data points that significantly deviate from the overall pattern of your data. They can have a substantial impact on correlation calculations, potentially leading to misleading results. Here's why:

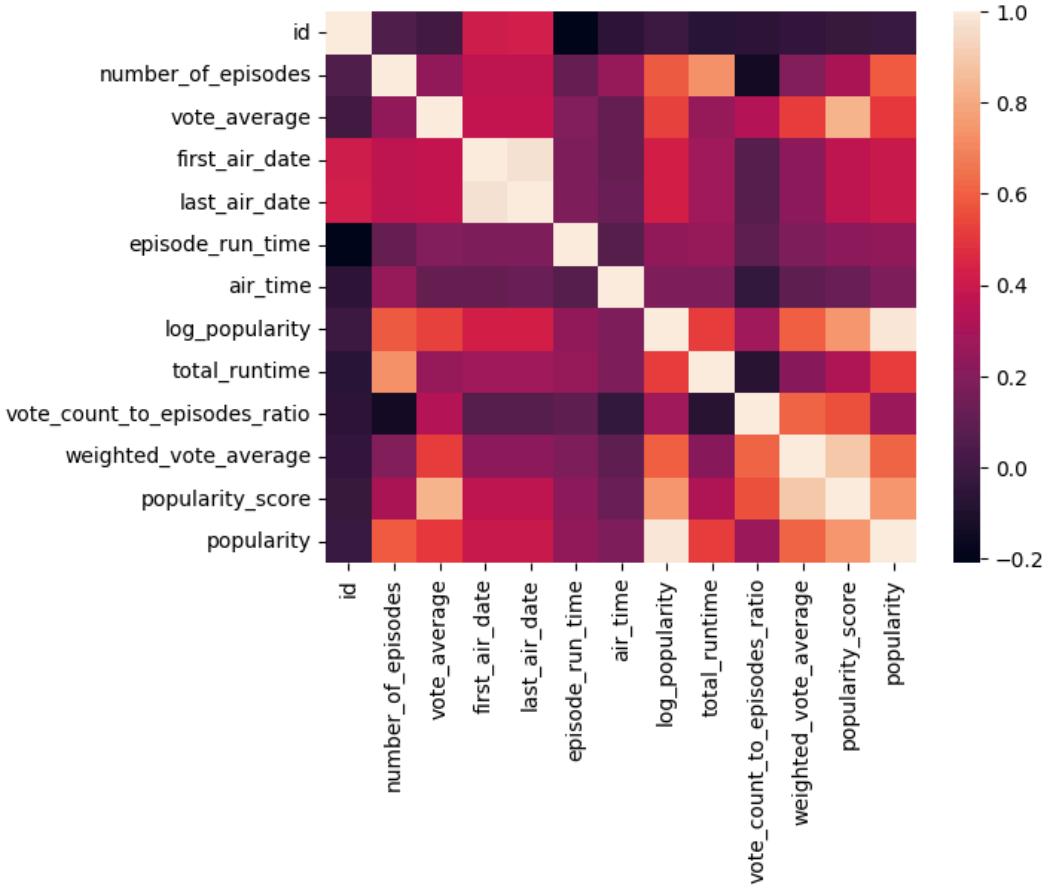
Inflated or Deflated Correlation:

Outliers can artificially inflate or deflate the correlation coefficient, making the relationship between variables appear stronger or weaker than it actually is.

Distorted Relationships:

Outliers can distort the linear relationship between variables, making it difficult to accurately assess the true correlation.

Heatmap-Correlation with Outliers



Robust Correlation Measures

Spearman's Rank Correlation:

Instead of Pearson's correlation (which is sensitive to outliers), use Spearman's rank correlation. It's a non-parametric method that considers the ranks of data points rather than their actual values, making it less affected by outliers

Spearman's Rank Correlation:

	id	number_of_episodes	vote_average \
id	1.000000	0.227290	0.054508
number_of_episodes	0.227290	1.000000	0.372206
vote_average	0.054508	0.372206	1.000000
first_air_date	0.603677	0.506448	0.352557
last_air_date	0.603684	0.520394	0.357738
episode_run_time	-0.259249	0.214567	0.217593
air_time	-0.236679	0.409553	0.128322

log_popularity	0.090573	0.682339	0.552825
total_runtime	-0.040125	0.676012	0.404107
vote_count_to_episodes_ratio	-0.165375	0.004140	0.869878
weighted_vote_average	0.024913	0.397420	0.947569
popularity_score	0.105238	0.580933	0.866341
popularity	0.090573	0.682339	0.552825

5.Imbalance Techniques & Model Selection & Fine Tuning & Model

Imbalance Techniecs for the tmdb_model Dataset

In machine learning, imbalanced datasets refer to datasets where the distribution of classes is not equal. This means that one class (the majority class) has significantly more instances than another class (the minority class). This imbalance can pose a challenge for training machine learning models because the model might become biased towards the majority class and perform poorly on the minority class.

To address this issue, various imbalance techniques are used to rebalance the dataset or adjust the learning process to improve the model's performance on the minority class. Here are some common imbalance techniques:

1. Resampling Techniques

- **Undersampling the Majority Class:** If we have a large number of TV shows with low popularity, you could randomly remove some of them to balance the dataset. However, this could lead to loss of information.
- **Oversampling the Minority Class:** If we have a small number of TV shows with very high popularity, you could duplicate or create synthetic samples of them to increase their representation.

2. Cost-Sensitive Learning

- **Adjusting Loss Function:** we can modify the loss function of your regression model to give more weight to errors on the minority class (TV shows with high popularity). This penalizes the model more for mispredicting popular shows, encouraging it to focus on them.

3. Transformation of Target Variable

- **Log Transformation:** If the 'popularity' data is highly skewed, applying a log transformation might help make the distribution more normal, potentially improving the performance of regression models.

Summery:

- **Regression Context:** The choice of imbalance technique depends on the specific characteristics of the dataset and the goals of theanalysis. Experiment with different approaches and evaluate their impact on model performance.
- **Evaluation Metrics:** For regression problems, useing appropriate evaluation metrics like Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared to assess the effectiveness of the imbalance techniques.
- **Data Understanding:** Before applying any technique, we carefully analyze the distribution of your target variable ('popularity') to understand the extent of the imbalance. This will help you choose the most appropriate approach.

The Classification Report

The classification report is a performance evaluation metric in machine learning that provides a comprehensive overview of how well a classification model is performing. It breaks down the model's predictions for each class, offering insights beyond just overall accuracy.

The difference between ROS (Random Over-Sampling) and RUS (Random Under-Sampling) techniques in the context of imbalanced datasets in machine learning.

Classification Report- Model HistGradientBoostingClassifier

The HistGradientBoostingClassifier is a powerful machine learning algorithm for classification tasks, introduced in scikit-learn. It's a variant of Gradient Boosting that uses histograms to bin continuous features, leading to significant speed improvements, especially for large datasets.

Results- `Create and train the HistGradientBoostingClassifier`

Accuracy: 0.9988

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	6762
1	1.00	1.00	1.00	6804
accuracy			1.00	13566
macro avg	1.00	1.00	1.00	13566
weighted avg	1.00	1.00	1.00	13566

ROS vs. RUS: Balancing Imbalanced Datasets

Both ROS and RUS are resampling techniques used to address the issue of imbalanced datasets, where one class (majority class) has significantly more instances than another class (minority class). However, they achieve balance through different approaches:

ROS (Random Over-Sampling):

- **Approach:** ROS increases the representation of the minority class by randomly duplicating existing instances of that class.
- **Effect:** It creates a larger dataset with a more balanced class distribution.
- **Advantages:**
  - Simple to implement.
  - Does not lose any information from the original dataset.
- **Disadvantages:**
  - Can lead to overfitting because the model might memorize duplicated instances.
  - May not generalize well to new, unseen data.
  - Increases the size of the dataset, potentially increasing training time.

RUS (Random Under-Sampling):

- **Approach:** RUS reduces the representation of the majority class by randomly removing instances from that class.
- **Effect:** It creates a smaller dataset with a more balanced class distribution.
- **Advantages:**
  - Can help improve the model's performance on the minority class.
  - Reduces the size of the dataset, potentially decreasing training time.
- **Disadvantages:**
  - Can lead to loss of valuable information from the majority class.
  - May not be suitable for datasets with a very small majority class.

Key Differences:

Feature	ROS	RUS
Target Class	Minority class	Majority class
Action	Duplicates instances	Removes instances
Dataset Size	Increases	Decreases
Information Loss	No	Yes
Overfitting Risk	Higher	Lower

Choosing Between ROS and RUS:

- **Dataset Size:** For very large datasets, RUS might be preferred to reduce training time. For smaller datasets, ROS might be better to avoid losing too much information.
- **Severity of Imbalance:** For highly imbalanced datasets, a combination of ROS and RUS might be considered.
- **Model Performance:** Experiment with both techniques and evaluate their impact on the model's performance using appropriate metrics.

Results- Create and train the RandomOver Sumpler (ROS)

Results for ROS: Accuracy: 0.9951

Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	6762
1	1.00	0.99	1.00	6804
accuracy			1.00	13566
macro avg	1.00	1.00	1.00	13566
weighted avg	1.00	1.00	1.00	13566

Results- Create and train the RandomUnder Sumpler (RUS)

Results for RUS: Accuracy: 0.6759

Classification Report:

	precision	recall	f1-score	support
0	0.61	1.00	0.75	6762
1	1.00	0.35	0.52	6804
accuracy			0.68	13566
macro avg	0.80	0.68	0.64	13566
weighted avg	0.80	0.68	0.64	13566

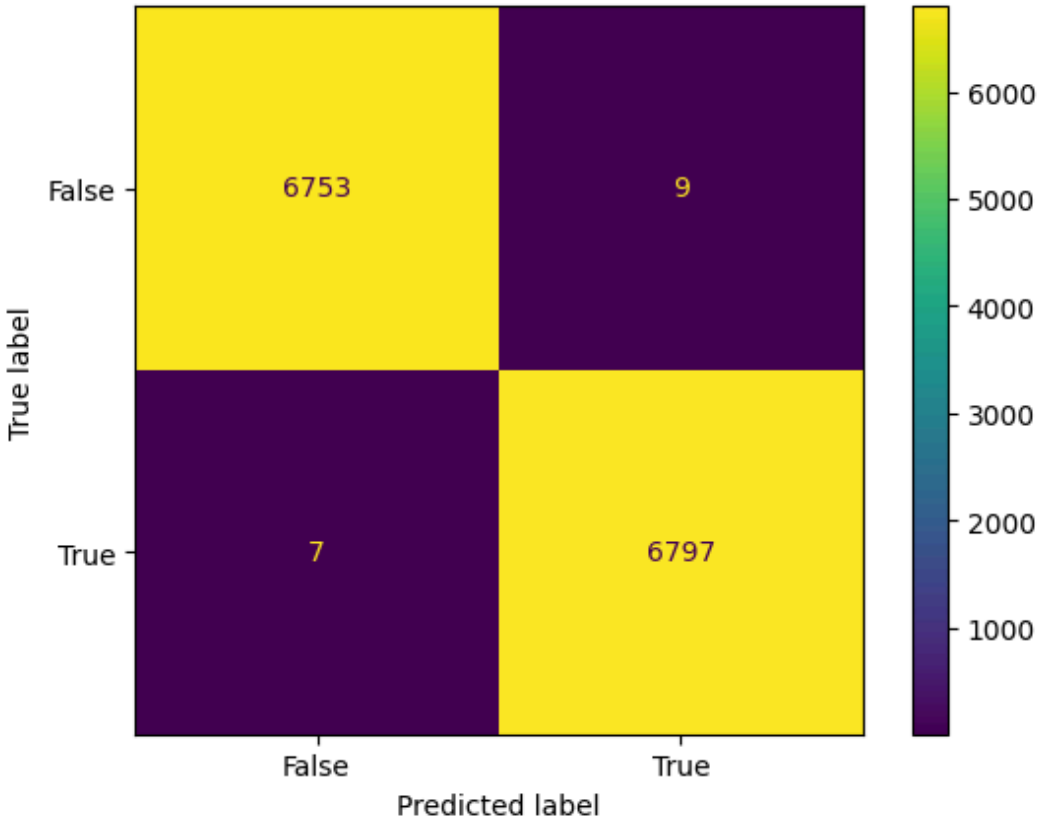
Support Vector Machine

SVM is a supervised ML Algorithm which can be used for both classification and regresstion models.

Confusion Metrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows us to visualize the performance of a supervised learning algorithm by showing the counts of:

True Positives (TP): Correctly predicted positive instances. True Negatives (TN): Correctly predicted negative instances. False Positives (FP): Incorrectly predicted positive instances (Type I error). False Negatives (FN): Incorrectly predicted negative instances (Type II error).



Gridsearch:

The machine learning find the optimal hyperparameters for a model. It helps improve model performance by systematically exploring different combinations of hyperparameter values and selecting the combination that yields the best results.

hyperparameter trade-off
Gread Search Cross-Validation
C - achieving a low training error and a low testing error.0.1 week regulation
gamma - controls the width of the radial basis function
RFB: Radial Basis Function (linear, sigmoid, poly)

Grid Search Cross-Validation:

Improved Model Performance:

By systematically exploring different hyperparameter values, you are more likely to find the optimal settings that lead to better model performance.

Reduced Overfitting: Cross-validation helps prevent overfitting by evaluating the model on multiple subsets of the data.

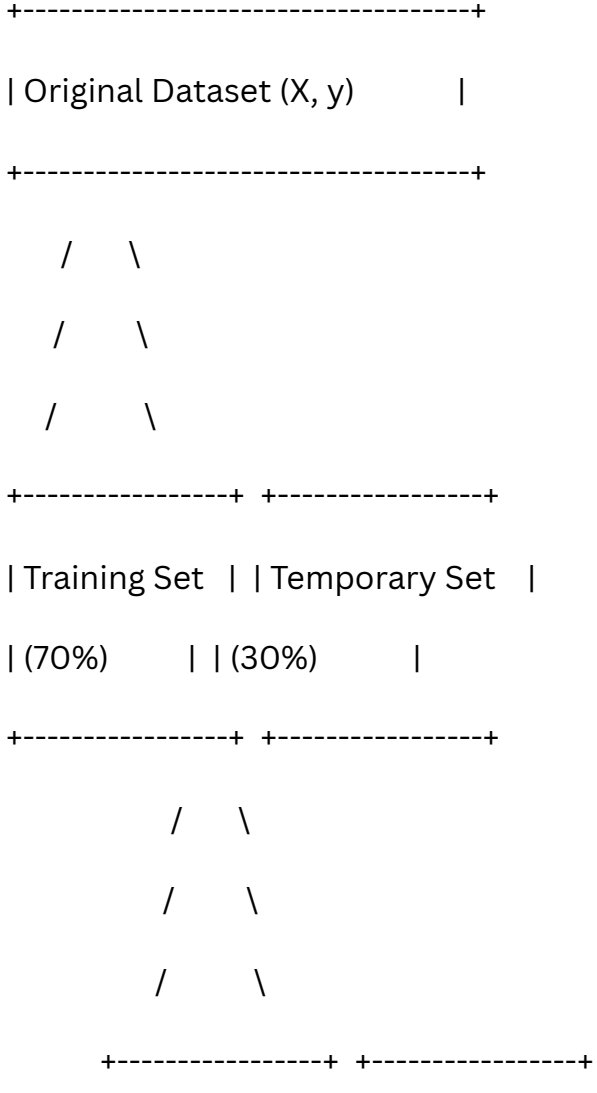
Automation: Grid search automates the process of hyperparameter tuning, saving you time and effort.

Data Splitting

For training and evaluating the model, the dataset should be split into different subsets:

- **Training Set:** The portion of data used to train the model, comprising about 70-80% of the dataset.
- **Validation Set:** Used to tune model parameters and prevent overfitting, typically around 10-15% of the dataset.
- **Test Set:** A separate set used to evaluate the model's performance, consisting of the remaining 10-15%.

Data Splitting Schem :



| Validation Set | | Test Set |

| (15%) | | (15%) |

+----------+ +----------+

The SVC Model

SVC, or Support Vector Classifier, is a powerful supervised machine learning algorithm used for classification tasks. It's part of the Support Vector Machine (SVM) family, which can be used for both classification and regression. SVC is particularly effective for binary classification problems (where you have two classes) but can be extended to multi-class scenarios as well.

The main goal of SVC is to find an optimal hyperplane that best separates data points into different classes. A hyperplane is a decision boundary that maximizes the margin between the classes. The margin is the distance between the hyperplane and the nearest data points of each class, called support vectors.

SVC Classifier Best Parameters - using Grid Serch techniques

Best parameters: {'classifier__C': 10, 'classifier__gamma': 1, 'classifier__kernel': 'linear'}

Grid Prediction Metrix

[[6758 4]

[1 6803]]

Accuracy: 0.9912

Classification Report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	6762
1	1.00	0.98	0.99	6804
accuracy			0.99	13566
macro avg	0.99	0.99	0.99	13566
weighted avg	0.99	0.99	0.99	13566

Support Vector Machine-SVM Model

SVM is a supervised ML Algorithm which can be used for both classification and regresstion models

Results :

Mean Absolute Error (MAE): 0.037

Mean Squared Error (MSE): 0.002

Root Mean Squared Error (RMSE): 0.048

R-squared (R^2): 0.995

Root Mean Squared Logarithmic Error (RMSLE): 0.104

Classification Models -Linear Regression

Type: Linear regression is a classic and widely used regression model that assumes a linear relationship between the features and the target variable (popularity).

Results :

Mean Squared Error: 0.001650061216726249

R-squared: 0.9963508974467066

Predicted Popularity for sample data: 451.1209113096185

Decision Tree

A Decision Tree is a supervised machine learning algorithm used for both classification and regression tasks. It's like a flowchart that makes decisions based on a series of questions (features) about the data. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Mean Squared Error (MSE): 0.000

Mean Absolute Error (MAE): 0.000

R-squared (R^2): 1.000

Random Forest Regressor

Type: An ensemble learning method that builds multiple decision trees and averages their predictions to improve accuracy and robustness.

Results:

Mean Squared Error: 1.2514478364850565e-08

R-squared: 0.9999999723242905

Predicted Popularity for sample data: 2.4988170788203123

The XGBoost model

XGBoost (Extreme Gradient Boosting) is a powerful and popular machine learning algorithm that belongs to the gradient boosting family. It's widely used for both classification and regression tasks and is known for its high accuracy and efficiency.

Results:

Root Mean Squared Error (RMSE): 0.009039797312510827

R-squared (R2): 0.9998197245701756

Feature Importance:

	Feature	Importance
7	log_popularity	0.485234
12	success	0.465747
11	popularity_score	0.039580
1	number_of_episodes	0.007311
2	vote_average	0.000755
9	vote_count_to_episodes_ratio	0.000630
10	weighted_vote_average	0.000366
3	first_air_date	0.000225
4	last_air_date	0.000095
0	id	0.000031
5	episode_run_time	0.000012
6	air_time	0.000007
8	total_runtime	0.000006

6.Model Selection

Model selection is the process of choosing the best machine learning model from a set of candidate models for a particular task. The goal is to find a model that generalizes well to unseen data and provides the most accurate and reliable predictions.

Model Selection Importance:

- **Performance:** Different models have different strengths and weaknesses. Model selection helps you find the model that performs best for your specific data and problem.
- **Overfitting:** Some models are more prone to overfitting (performing well on training data but poorly on new data). Model selection techniques help identify models that generalize well.
- **Efficiency:** Some models are more computationally expensive than others. Model selection helps you choose a model that balances performance and efficiency.
- **Interpretability:** Some models are easier to understand and interpret than others. Model selection can help you choose a model that is suitable for your needs in terms of interpretability.

Best Model Selection Metrics:

MSE - Mean Squared Error

RMSE Root Mean Squared Error

MAE Mean Absolute Error - Calculates the average of the absolute differences between predicted and actual values.

RMSLE Root Mean Squared Logarithmic Error

	model	MSE	RMSE	MAE	RMSLE
0	Linear Regression	3.651998e-01	6.043176e-01	4.314085e-01	7.134928e-01
1	Decision Tree	1.569423e-25	3.961594e-13	2.708953e-13	1.440164e-12
2	RandomForest	1.752239e-09	4.185975e-05	1.065316e-05	2.142479e-05
3	ADABoost	1.657772e-03	4.071575e-02	2.973612e-02	6.229777e-02
4	GBM	1.651109e-05	4.063384e-03	2.077484e-03	3.873567e-03
5	SVR	3.944364e-01	6.280417e-01	4.194485e-01	6.683359e-01
6	XGB	6.713038e-06	2.590953e-03	1.296956e-03	2.255282e-03

Best Model Ranking by MSE :

	model	MSE	RMSE	MAE	RMSLE
1	Decision Tree	1.569423e-25	3.961594e-13	2.708953e-13	1.440164e-12
2	RandomForest	1.752239e-09	4.185975e-05	1.065316e-05	2.142479e-05
6	XGB	6.713038e-06	2.590953e-03	1.296956e-03	2.255282e-03
4	GBM	1.651109e-05	4.063384e-03	2.077484e-03	3.873567e-03
3	ADABoost	1.657772e-03	4.071575e-02	2.973612e-02	6.229777e-02
5	SVR	3.944364e-01	6.280417e-01	4.194485e-01	6.683359e-01
0	Linear Regression	3.651998e-01	6.043176e-01	4.314085e-01	7.134928e-01

Best Model Ranking by RSME :

	model	MSE	RMSE	MAE	RMSLE
1	Decision Tree	1.569423e-25	3.961594e-13	2.708953e-13	1.440164e-12
2	RandomForest	1.752239e-09	4.185975e-05	1.065316e-05	2.142479e-05
6	XGB	6.713038e-06	2.590953e-03	1.296956e-03	2.255282e-03
4	GBM	1.651109e-05	4.063384e-03	2.077484e-03	3.873567e-03
3	ADABoost	1.657772e-03	4.071575e-02	2.973612e-02	6.229777e-02
0	Linear Regression	3.651998e-01	6.043176e-01	4.314085e-01	7.134928e-01
5	SVR	3.944364e-01	6.280417e-01	4.194485e-01	6.683359e-01

Best Model Ranking by RSMLE :

	model	MSE	RMSE	MAE	RMSLE
1	Decision Tree	1.569423e-25	3.961594e-13	2.708953e-13	1.440164e-12
2	RandomForest	1.752239e-09	4.185975e-05	1.065316e-05	2.142479e-05
6	XGB	6.713038e-06	2.590953e-03	1.296956e-03	2.255282e-03
4	GBM	1.651109e-05	4.063384e-03	2.077484e-03	3.873567e-03
3	ADABoost	1.657772e-03	4.071575e-02	2.973612e-02	6.229777e-02
5	SVR	3.944364e-01	6.280417e-01	4.194485e-01	6.683359e-01
0	Linear Regression	3.651998e-01	6.043176e-01	4.314085e-01	7.134928e-01

Summery : The best model by 3 Categories is - Decision Tree

7.Decision Tree Regressor - Fine Tuning

Fine-tuning a Decision Tree Regressor involves finding the optimal hyperparameter values that minimize the model's error and improve its predictive performance.

Best Parameters for DecisionTree Regressor :

1. Regressor_max_depth : 10
2. Regressor_min_samples_leaf: 1
3. Reggressor_min_samples_split 2

Mean Squared Error : 4.2426

R-Squared : 0.9999999

Cross-Validation:

Cross-validation is a crucial technique in machine learning used to evaluate the performance of a model and ensure it generalizes well to unseen data. It helps prevent overfitting, where a model performs well on the training data but poorly on new data.

Suggesting 2 options:

1. k-Fold Cross-Validation

Reasoning: k-Fold Cross-Validation is a widely used technique that provides a robust estimate of model performance. It involves splitting your data into 'k' equal-sized folds. The model is trained on k-1 folds and tested on the remaining fold. This process is repeated 'k' times, with each fold serving as the test set once. The performance metric (e.g., R-squared, MAE) is averaged across all iterations to obtain a more reliable performance estimate.

Results:

Average R-squared (5-fold CV): 0.9962602746899945

R-squared on Test Set: 0.9962559564837956

2. Stratified k-Fold Cross-Validation

This technique is particularly useful when you have imbalanced classes in your target variable (although your target 'popularity' might be continuous). Stratified k-Fold ensures that each fold maintains the same proportion of target classes as the original dataset, leading to a more representative performance estimate, especially for minority classes.

Results:

Average R-squared (5-fold CV): 0.9962602746899945

R-squared on Test Set: 0.9962559564837956

Summary:

Feature	k-Fold Cross-Validation	Stratified k-Fold Cross-Validation
Data Splitting	Random	Random, but preserving class proportions
Use Cases	Regression and balanced classification	Classification, especially imbalanced datasets
Class Distribution	May not be preserved in folds	Preserved in folds
Performance Estimates	Robust for balanced datasets	More reliable for imbalanced datasets

8. Model Evaluation-*DecisionTreeRegressor*

Performance Matrics:

Model evaluating using appropriate metrics such as accuracy, precision, recall, F1-score, ROC-AUC for classification, or RMSE, MAE for regression.

Results:

Mean Squared Error (MSE): 4.901099953321971e-08

Mean Absolute Error (MAE): 4.470068284324272e-05

R-squared (R2): 0.999999742040156

Explained Variance Score: 0.9999997420709597

Max Error: 0.00551252593891105

Median Absolute Error: 2.220446049250313e-16

Root Mean Squared Logarithmic Error (RMSLE): 8.817635143957718e-05

Mean Poisson Deviance: 4.699094965690461e-08

Mean Gamma Deviance: 2.754438955068568e-06

D2 Tweedie Score (power=1): 0.9999999254747526

D2 Pinball Score (alpha=0.5): 0.9997579213843345

D2 Absolute Error Score: 0.9997579213843345