**Data Analysis and Preprocessing Report**

*1) Data Loading and Initial Analysis:*

- The dataset was loaded successfully for analysis.
- Preliminary statistical analysis was conducted, including the calculation of the mean, median, mode, data types, and standard deviation of numerical columns.

*2) Handling Missing Data:*

- The percentage of missing values ("NaNs") in each column was calculated to assess data completeness.

*3) Data Segmentation:*

- Columns were categorized based on their data types, distinguishing between numerical and categorical features.

*4) Visualization of Numerical Features:*

- The distribution of numerical features was visualized using histograms to understand their underlying patterns.

*5) Correlation Analysis:*

- A correlation matrix was generated to visualize relationships between numerical features.

*6) Visualization of Categorical Features:*

- Categorical feature distributions were visualized using histograms to gain insights into their frequencies.

*7) Data Imputation:*

- Missing values in numerical columns were imputed using the mean of each column.
- Categorical columns with missing data were imputed with the mode (most frequent value).

*8) Data Splitting:*

- The dataset was divided into training and test sets using a split ratio of 0.75 for training and 0.25 for testing.

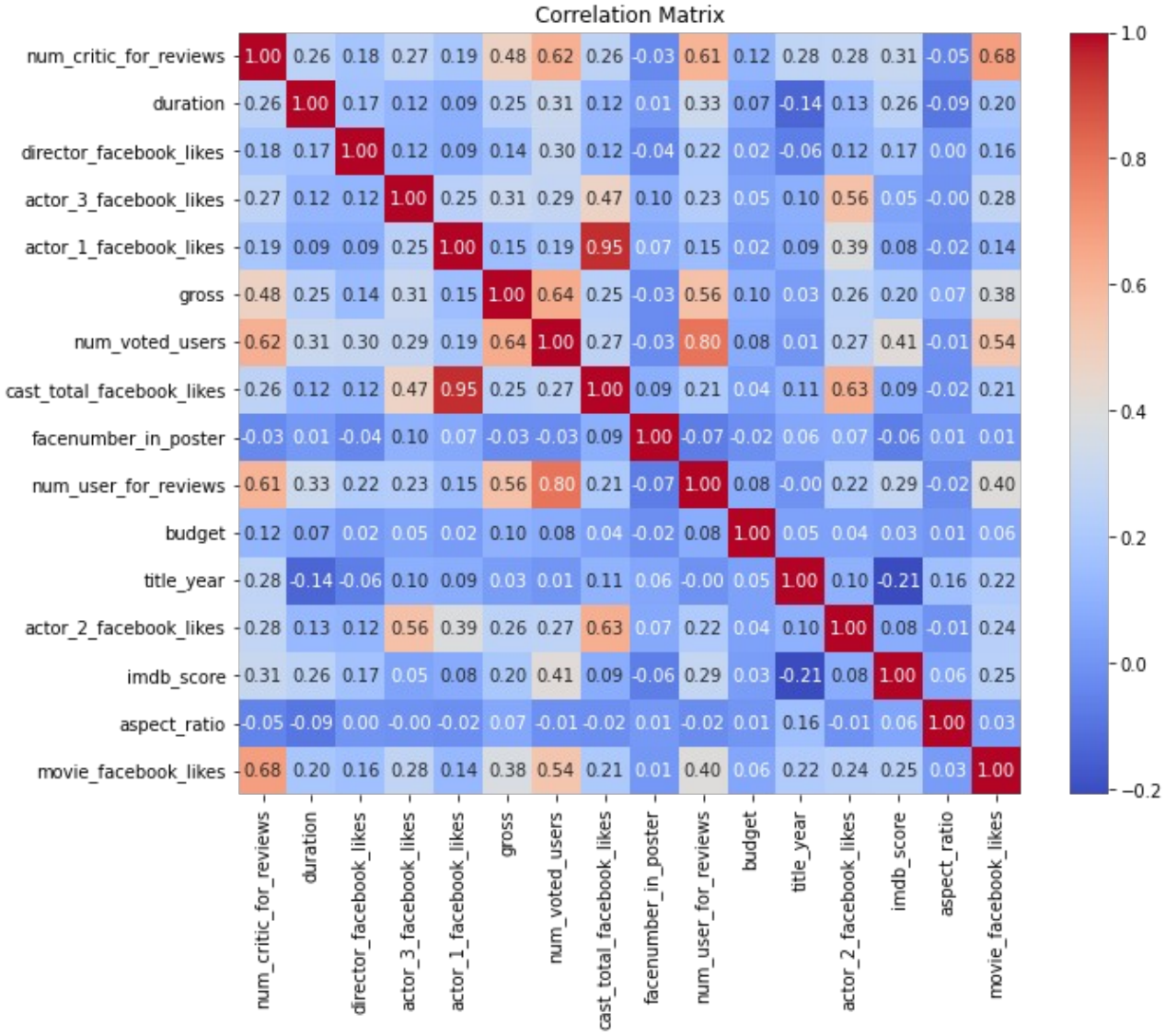
*9) Feature Encoding:*

- Categorical columns were segregated based on their cardinality.
- Low cardinality columns were encoded using label encoding.
- High cardinality columns were encoded using binary encoding to maintain information without creating excessive dimensions.

*10) Data Scaling:*

- A robust scaler was applied to scale the dataset, ignoring potential outliers to ensure robustness against extreme values.

This comprehensive data analysis and preprocessing workflow prepares the dataset for subsequent machine learning tasks, ensuring data quality, completeness, and appropriate encoding for both numerical and categorical features. The use of robust scaling helps in handling potential outliers while maintaining feature relevance.




**Result:**

| Model | RMSE |
|---|---|
| Random Forest | 0.611263 |
| GradientBoostingRegressor | 0.5834419 |
| Lasso Regressor | 1.1092548 |
| Support Vector Regressor | 0.807668 |
| Artificial Neural Network | 0.744934 |

**Final Predictions:**

I'd prefer to use GradientBoostingRegressor to be used in the production environment owing to the metrics Root mean square score. Also, GradientBoostingRegressor is flexible, extendable.