AUTOMATED REVIEW RATING SYSTEM:

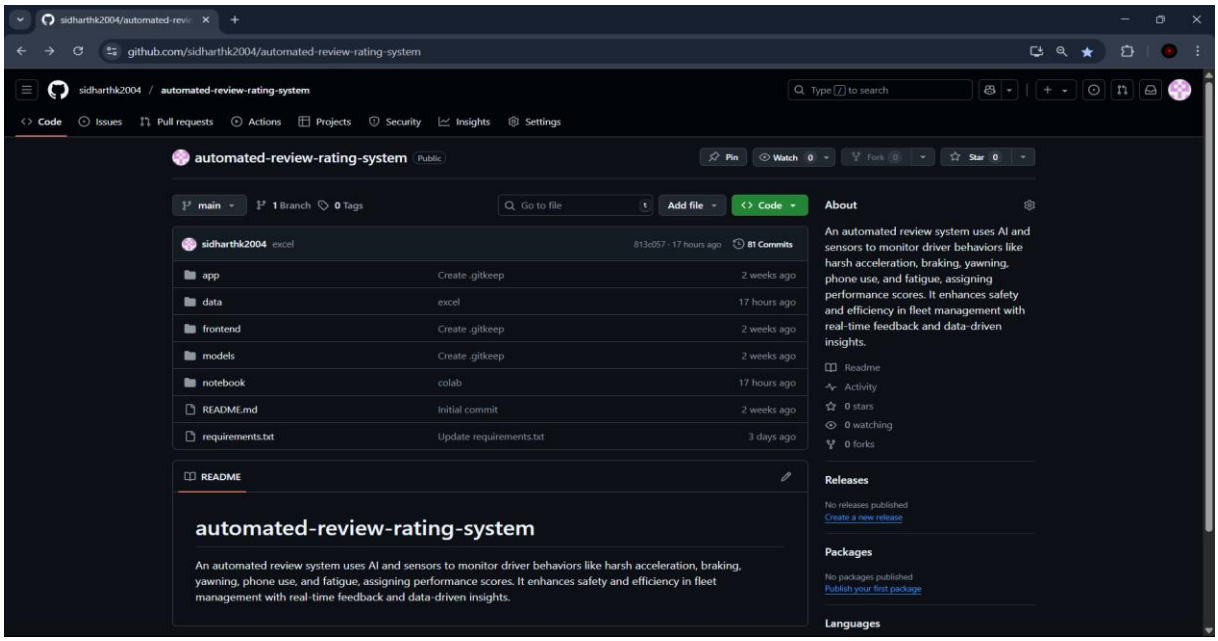
PROJECT OVERVIEW.

The Automated Review System is developed to collect, process, and analyse customer reviews using Natural Language Processing (NLP) and Sentiment Analysis. It automatically classifies user feedback into positive, negative, or neutral categories and extracts key insights from large volumes of text data. This helps businesses understand customer sentiments and improve their services or products accordingly.

ENVIRONMENT SETUP.

- Installed python with necessary libraries such as numpy, pandas, matplotlib, scikit-learn, seaborn, nltk
- Doing every code in colab

GITHUB PROJECT REPOSITORY.



DATA COLLECTION.

Every dataset was collected from Kaggle.

This project uses 9 different datasets across industries to ensure model generalization.

- Tripadvisor Hotel reviews
- Switch games reviews
- Cleaned reviews
- JAL Tripadvisor reviews
- Amazon reviews
- Musical instruments reviews
- Metacritic reviews
- Psychiatric drug reviews
- ChatGPT reviews

Each dataset contains:

- Review text
- Corresponding user rating (1-5 stars)

SN.	DATASET NAME	DOMAIN	SOURCE	SIZE	COLUMNS	LINK
1	Tripadvisor Hotel reviews	Hospitality	Kaggle	14616	8	https://github.com/sidharthk2004/automated-review-rating-system/blob/main/data/kaggle%20datasets/tripadvisor_hotel_reviews.csv
2	Switch games reviews	Games	Kaggle	5546	15	https://github.com/sidharthk2004/automated-review-rating-system/blob/main/data/kaggle%20datasets/switch-games.csv
3	Cleaned reviews	Items	Kaggle	4642	5	https://github.com/sidharthk2004/automated-review-rating-system/blob/main/data/kaggle%20datasets/cleaned.csv
4	JAL Tripadvisor reviews	E-commerce	Kaggle	5973	12	https://github.com/sidharthk2004/automated-review-rating-system/blob/main/data/kaggle%20datasets/JAL_tripadvisor_reviews.csv
5	Amazon reviews	E-commerce	Kaggle	4634	13	https://github.com/sidharthk2004/automated-review-rating-system/blob/main/data/kaggle%20datasets/amazon.csv
6	Musical instruments reviews	E-commerce	Kaggle	5950	6	https://github.com/sidharthk2004/automated-review-rating-system/blob/main/data/kaggle%20datasets/Musical_instruments_reviews.csv
7	Metacritic reviews	E-commerce	Kaggle	5246	9	https://github.com/sidharthk2004/automated-review-rating-system/blob/main/data/kaggle%20datasets/metacritic-reviews.csv
8	Psychiatric drug reviews	Drug	Kaggle	24686	18	https://github.com/sidharthk2004/automated-review-rating-system/blob/main/data/kaggle%20datasets/psychiatric_drug_webmd_reviews.csv
9	ChatGPT reviews	Information platform	Kaggle	13931	9	https://github.com/sidharthk2004/automated-review-rating-system/blob/main/data/kaggle%20datasets/clean_chatgpt_reviews.csv

All datasets are merged into one

Merged_dataset link= https://github.com/sidharthk2004/automated-review-rating-system/blob/main/notebook/merge%20dataset/merge_datasets.ipynb

DATA PREPROCESSING.

Preprocessing is the first and most important step in any machine learning or data science project. It involves cleaning, transforming, and organizing raw data into a format that your machine learning model can understand and learn from effectively.

Load and Inspect Datasets.

- Load all data sets (e.g. tripadvisor, drug reviews etc)
- Inspect: check names, data types, records
- Verify presence of review and rating columns pd.read_csv ()-Load CSV files into a dataframe pd.read excel()-For Excel files

#view first few rows

df.head()

#View last few rows

df.tail()

#view sample rows

df.sample()

#check column names

df.columns

#get dataframe shape (rows, columns)

df.shape

#check column data types

df.types #full summary

info df.info()

#summary statistics (numeric columns) df.describe()

Remove Unnecessary Columns.

- Drop columns not required for prediction, such as:

  User_id, product_id, timestamp, title, image, Url, etc.

- Keep only relevant columns: usually review_text and rating

Remove Null or Missing Values.

- Remove rows with null values in review or rating

#Check for missing values in each column

Print(df.isnull().sum())

#removing null values

Df.dropna(inplace=True)

#Remove rows where review_text or rating is missing df.dropna(subset=[‘review_text’,’rating’], inplace=Ture)

Remove Duplicates and Conflicting Reviews.

- Exact duplicates: same review and rating keep one
- Conflicting reviews: same review text, different ratings remove all

#check duplicates print(df.duplicated().sum()) #Remove all duplicate rows

Df.drop_duplicates(inplace=Ture)

Clean Text Data.

- Convert text to lowercase
- Remove:
  URL, Punctuation, Numbers and special characters, Extra spaces

Stopwords Removal.

Stopwords in a language (like 'a', 'about', 'above', 'after', 'again', 'against', 'ain', 'all', 'am', 'an', 'and', 'any', 'are', 'aren', "aren't", 'as', 'at', 'be', 'because', 'been', 'before', 'being', 'below', 'between', 'both', 'but', 'by', 'can', 'couldn', "couldn't", 'd', 'did', 'didn', "didn't", 'do', 'does', 'doesn', "doesn't", 'doing', 'don', "don't", 'down', 'during', 'each', 'few', 'for', 'from', 'further', 'had', 'hadn', "hadn't", 'has', 'hasn', "hasn't", 'have', 'haven', "haven't", 'having', 'he', "he'd", "he'll", "he's", 'her', 'here', 'hers', 'herself', 'him', 'himself', 'his', 'how', 'i', "i'd", "i'll", "i'm", "i've", 'if', 'in', 'into', 'is', 'isn', "isn't", 'it', "it'd", "it'll", "it's", 'its', 'itself', 'just', 'll', 'm', 'ma', 'me', 'mightn', "mightn't", 'more', 'most', 'mustn', "mustn't", 'my', 'myself', 'needn', "needn't", 'no', 'nor', 'not', 'now', 'o', 'of', 'off', 'on', 'once', 'only', 'or', 'other', 'our', 'ours', 'ourselves', 'out', 'over', 'own', 're', 's', 'same', 'shan', "shan't", 'she', "she'd", "she'll", "she's", 'should', "should've", 'shouldn', "shouldn't", 'so', 'some', 'such', 't', 'than', 'that', "that'll", 'the', 'their', 'theirs', 'them', 'themselves', 'then', 'there', 'these', 'they', "they'd", "they'll", "they're", "they've", 'this', 'those', 'through', 'to', 'too', 'under', 'until', 'up', 've', 'very', 'was', 'wasn't', "wasn't", 'we', "we'd", "we'll", "we're", "we've", 'were', 'weren', "weren't", 'what', 'when', 'where', 'which', 'while', 'who', 'whom', 'why', 'will', 'with', 'won', "won't", 'wouldn', "wouldn't", 'y', 'you', "you'd", "you'll", "you're", "you've", 'your', 'yours', 'yourself', 'yourselves')

These are the stopwords that do not add significant meaning to text and are usually removed during text preprocessingin NLP tasks.

Why do we Remove Stopwords?

- Improves Clarity
  Removes filler words, making bullet points crisp and easier to scan quickly.
- Focuses on Meaningful Words
  Highlights keywords or core ideas (e.g., "high accuracy model" vs "The model has high accuracy").
- Saves Space
  Keeps bullet points short and visually clean — useful in presentations, reports, or slides.
- Enhances NLP Processing
  In text analysis, removing stopwords helps algorithms focus on informative words.
- Speeds Up Search & Matching
  Reduces noise during keyword-based search, indexing, or comparison.

What is Lemmatization?

Lemmatization is the process of reducing a word to its base or dictionary form, called a *lemma*.

Ex:

- "running" → "run"
- "better" → "good"
- "studies" → "study"

Purpose**:**
To treat related words (like "runs", "ran", "running") as the same word ("run") for better text analysis.

What is stemming?

Stemming is the process of reducing a word to its root form by cutting off suffixes or prefixes, without considering the word's actual meaning.

Example:

- "running" → "run"
- "studies" → "studi"
- "faster" → "faster"
- Purpose:
  To group related words by their base form for text analysis or search indexing.

Difference from Lemmatization:
Stemming is faster but less accurate. It may produce non-dictionary words (e.g., "studies" → "studi"), unlike lemmatization which gives real words.

Why Lemmatization is better (in most ML tasks):

- Produces real words
  → Outputs valid dictionary words (e.g., "studies" → "study")
- Understands context
  → Considers part of speech (e.g., "better" → "good")
- Improves ML accuracy
  → Helps models learn from consistent, meaningful features
- Reduces feature noise
  → Avoids non-standard roots like "studie", "runn"
- Essential for deep NLP tasks
  → Used in tasks like sentiment analysis, translation, topic modelling
- More precise than stemming
  → Avoids over-simplification and retains linguistic correctness

DATA VISUALIZATION.

Data Visualization is the graphical representation of data using visual elements like charts, graphs, and maps.

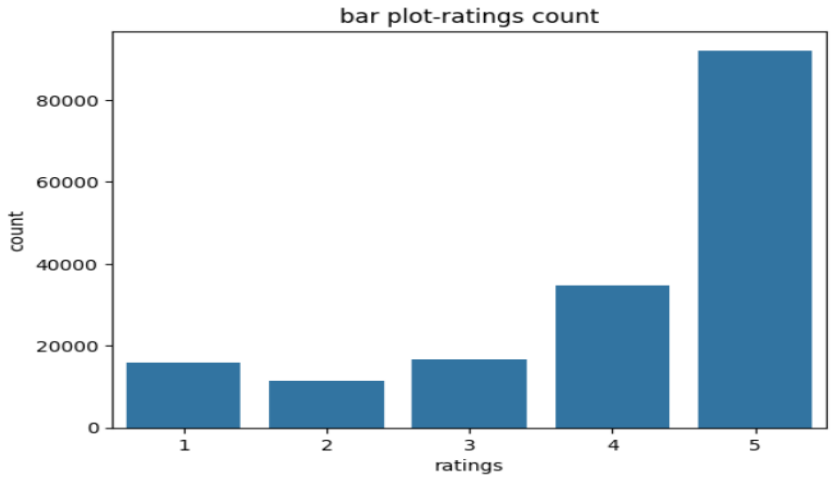
Purpose:
To make data easier to understand, identify patterns, trends, and outliers, and support data-driven decision-making.

BAR CHART.

A bar chart is a graphical representation of data using rectangular bars to show the size or frequency of different categories.

Key Features:

- Bars can be vertical or horizontal
- Length/height of bar = value of the category
- Useful for comparing discrete (categorical) data

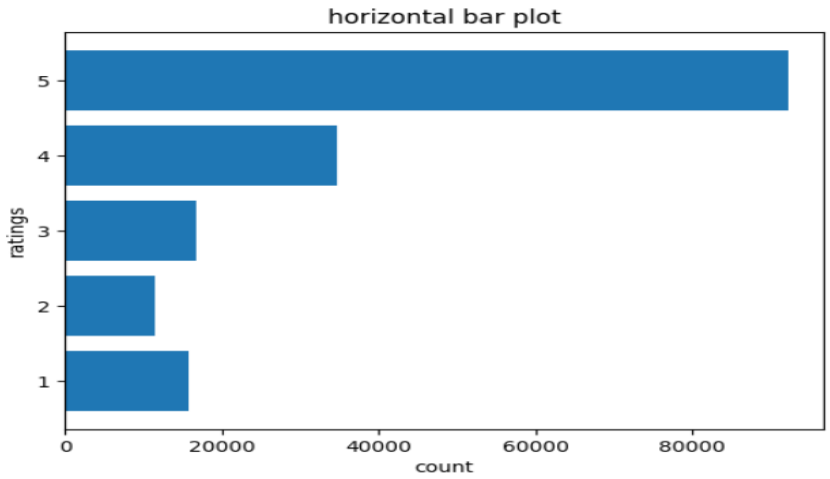


Horizontal Bar Chart (HBar).

A horizontal bar chart displays data with rectangular bars placed horizontally. The length of each bar represents the value of the category.

Key Features:

- Categories are listed on the y-axis
- Values are represented along the x-axis
- Useful when category names are long or when comparing many items

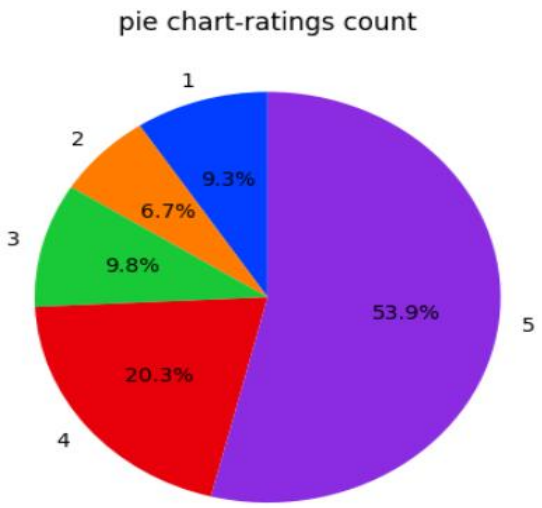


Pie Chart.

A pie chart is a circular graph divided into slices to show the proportional size of parts relative to the whole.

Key Features:

- Each slice = one category's value
- Entire circle = 100%
- Great for visualizing percentages or part-to-whole relationships



IMBALANCED DATASET.

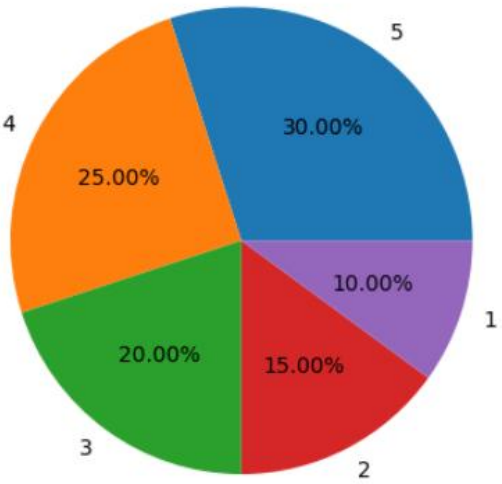
Imbalanced dataset is one where all classes (categories or labels) have roughly the same number of samples.

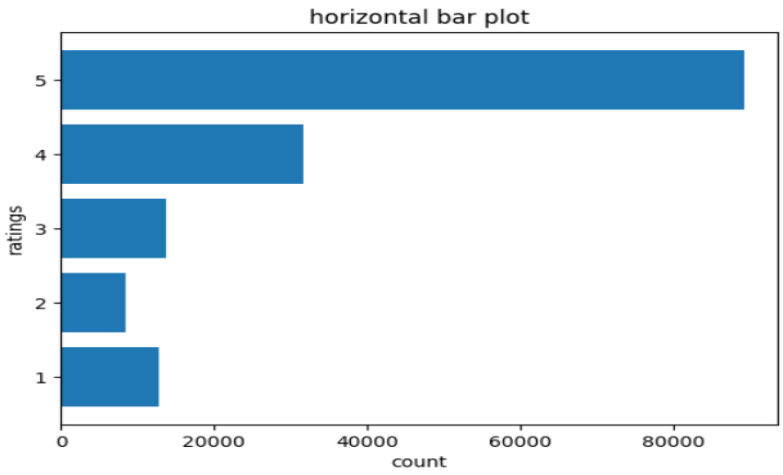
Why It's Important:

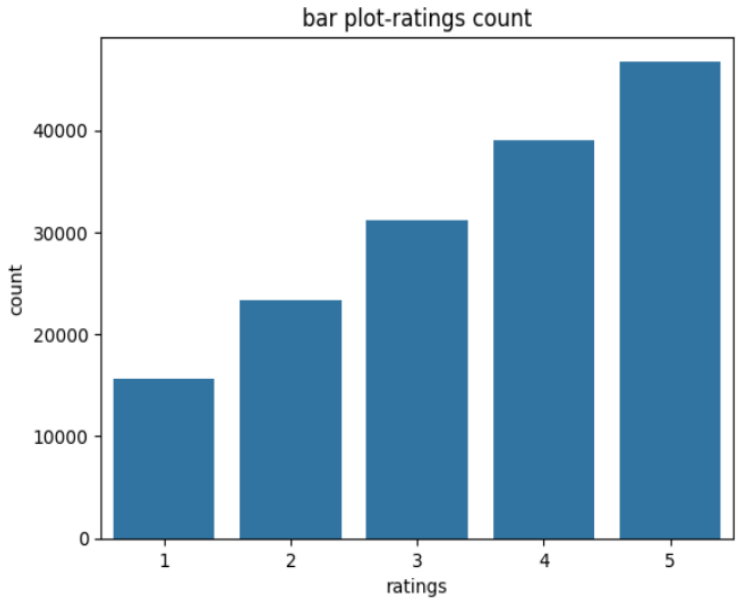
- Prevents ML models from becoming biased toward the majority class
- Helps in achieving better accuracy, recall, and precision
- Crucial for classification problems, especially in binary and multi-class tasks

Imbalanced dataset link= https://github.com/sidharthk2004/automated-review-rating-system/blob/main/data/imbalanced/imbalanced_data_visualization.zip

Imbalanced dataset with 1 star–10%, 2 star-15%, 3 star-20%, 4 star-25%, 5star-30% reviews and ratings







Train-Test Split.

What is Train-Test Split?

Train-test split is a technique used to divide a dataset into two parts:

- Training set – used to train the machine learning model
- Testing set – used to evaluate the model's performance on unseen data

Typical Ratio:

- 70%–80% for training
- 20%–30% for testing

Purpose:

- To check if the model generalizes well to new, unseen data
- Prevents overfitting and ensures realistic performance metrics

Common Tools:

- Python: train_test_split() from sklearn.model_selection

Why use stratify?

To ensure that the class distribution (i.e., the ratio of labels) is the same in both training and testing sets.

Why it's Important:

- Prevents class imbalance in the test set
- Helps model evaluation be more realistic and reliable
- Especially useful in classification tasks (e.g., spam vs. not spam)

TF-IDF Vectorization

What is TF-IDF?

TF-IDF is a numerical statistic that reflects how important a word is to a document in a collection (corpus).

Components:

- TF (Term Frequency):
  Measures how often a word appears in a document.
  TF = (Number of times word appears) / (Total words in document)

- IDF (Inverse Document Frequency):
  Measures how rare a word is across all documents.
  IDF = log (Total Documents / Documents with the word)

How it Works?

Step 1: Calculate TF (Term Frequency)

- Measures how frequently a word appears in a document
- TF = (word count in doc) / (total words in doc)

Step 2: Calculate IDF (Inverse Document Frequency)

- Measures how rare the word is across all documents
- IDF = log (Total documents / Documents containing the word)

Step 3: Multiply TF × IDF for each word

- Words that appear often in a doc but rarely elsewhere get high scores
- Common words across all docs get low scores

What is a TF-IDF Score?

- It's a numerical value assigned to each word in a document
- Shows how important that word is in that document
- Higher score = more unique & relevant to the document

How It's Calculated.

- TF = Frequency of the word in the document
- IDF = Inverse of how common the word is across documents
- TF-IDF Score = TF × IDF

LOGISTIC REGRESSION

- Definition:
  Logistic Regression is a supervised machine learning algorithm used for classification problems.
  It predicts the probability that a given input belongs to a particular class.
- Key Idea:
  - Instead of predicting continuous values (like Linear Regression), it predicts a probability between 0 and 1.
  - It uses the sigmoid (logistic) function to squash the output into this range.
- Uses:
  - Spam email detection (spam / not spam).
  - Medical diagnosis (disease / no disease).
  - Customer churn prediction (will leave / will stay).
  - Credit scoring (default / no default).

  In short: Logistic Regression is a simple yet powerful algorithm for binary and multi-class classification that outputs probabilities and makes decisions based on them.

Why choose this algorithm.

- Classification tasks – It's mainly used when the output is categorical (e.g., yes/no, spam/not spam, disease/no disease).
- Probability output – Instead of just giving class labels, it predicts probabilities between 0 and 1.
- Interpretable model – The coefficients show how each input feature affects the likelihood of the outcome.
- Simple & efficient – Requires less computation compared to complex algorithms like neural networks.
- Works well for linearly separable data – Effective when classes can be separated by a straight line (or hyperplane).
- Baseline model – Often used as the first model to try in classification problems.
- Regularization support – Can prevent overfitting using L1 (Lasso) or L2 (Ridge).
- Extends to multiple classes – With techniques like One-vs-Rest (OvR) or Softmax, it can handle multi-class problems.
- Good with smaller datasets – Performs decently even when data is not huge.
- Widely used in real-world – Popular in fields like healthcare, finance, and social sciences because of its interpretability.

Preprocessing steps.

Before training the model, the following preprocessing steps were applied:

1. **Loading and Inspecting Data** - All 10 datasets (Women's Clothing, TripAdvisor, Laptops, IMDB, Amazon Products, etc.) were loaded and merged.

2. **Removing Unnecessary** Columns - Dropped columns unrelated to review text or rating (e.g., product IDs, usernames).

3. **Removing Null Values** - Eliminated rows with missing reviews or ratings.

4. **Removing Duplicates** - Removed duplicate reviews to avoid bias.

5. **Text Cleaning** - Lowercasing, removing special characters, numbers, and extra spaces.

6. **Stop Words Removal** - Removed common words (c.g., "the", "is") that do not contribute to meaning.

7. **Lemmatization** - Reduced words to their base form for better consistency.

8. **TF-IDF Vectorization** - Converted text into numerical features using Term Frequency-Inverse Document Frequency to capture word importance.

9. **Train-Test Split with Stratification** - Ensured class distribution consistency between training and testing dataset.

Model Training Logic.

- The cleaned and vectorized dataset was split into 80% training and 20% testing sets.
- Logistic Regression was trained using Scikit-learn with the following steps:

    1. Initialize the model with logistic regression(max_iter=5000, c=1.0,solver-'lbfgs")

    2. Fit the model to the training data (model.fit(X_train, y_train)).

    3. Generate predictions for the test data (model. Predict(X_test)).

Evaluation Results and Interpretation.

The model's performance was evaluated using:

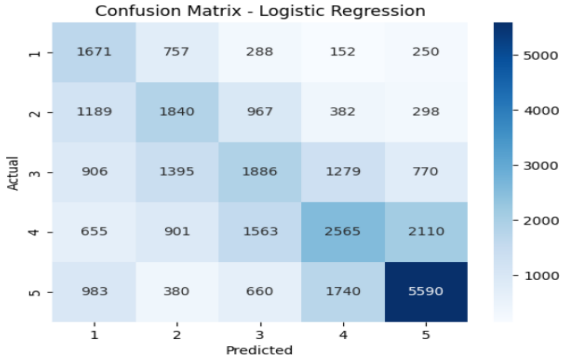
- **Accuracy Score** - Percentage of correct predictions.
- **Confusion Matrix** - To visualize correct vs. incorrect predictions per rating class.
- **Classification Report** - Showing precision, recall, and F1-score for each class.

```
Accuracy: 0.4346794111043397

Classification Report:
              precision    recall  f1-score   support

           1       0.31      0.54      0.39      3118
           2       0.35      0.39      0.37      4676
           3       0.35      0.30      0.33      6236
           4       0.42      0.33      0.37      7794
           5       0.62      0.60      0.61      9353

    accuracy                           0.43     31177
   macro avg       0.41      0.43      0.41     31177
weighted avg       0.44      0.43      0.43     31177
```



-