AUTOMATED REVIEW RATING SYSTEM

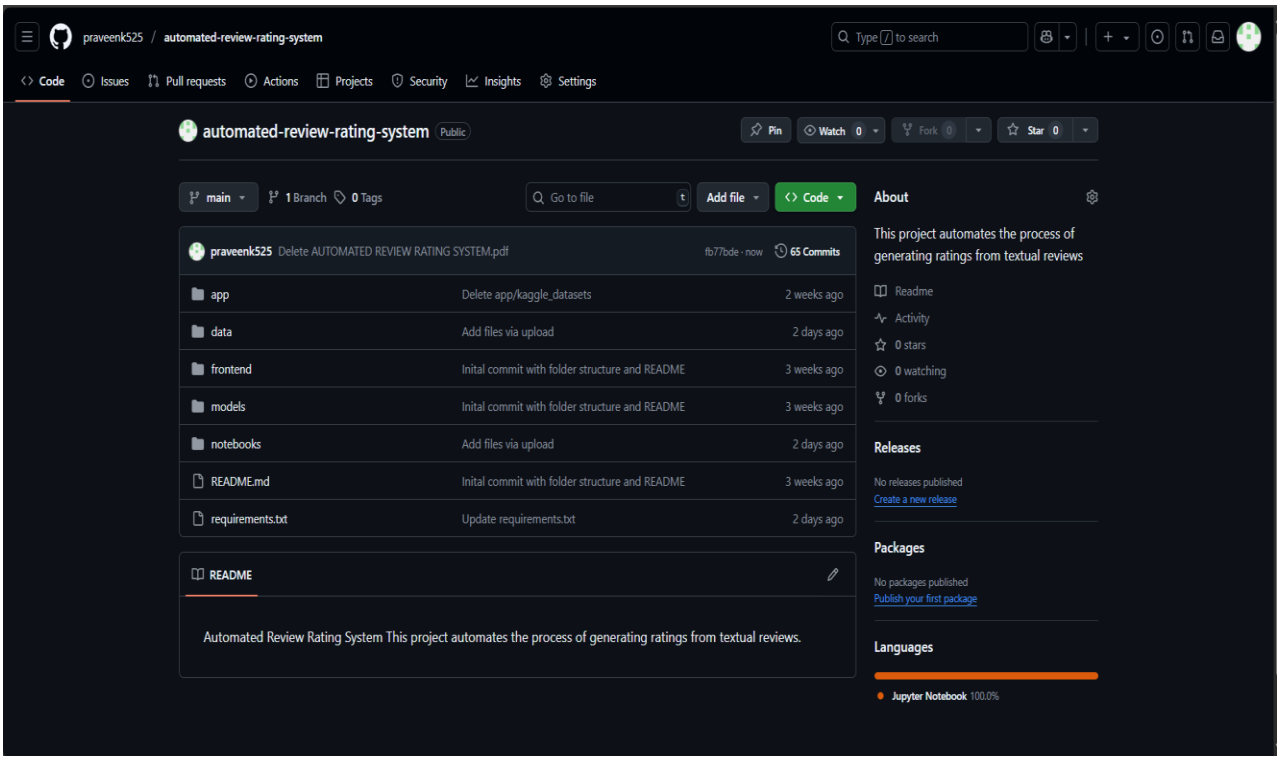
PROJECT OVERVIEW:

This project automates the process of generating ratings from textual reviews. an AI-based system that automatically predicts the numerical rating (e.g., 1 to 5 stars) of a product, service, or business based solely on user-generated textual reviews. This aims to reduce rating manipulation, standardize feedback, and assist platforms in quickly gauging sentiment at scale.

ENVIRONMENT SETUP.

- Installed python with necessary libraries such as numpy, pandas, matplotlib, scikit-learn, seaborn, nltk
- Colab

GITHUB PROJECT REPOSITORY



DATA COLLECTION.

Data was collected from Kaggle datasets:

For the Automated Review Rating System, a total of 10 datasets were collected, including reviews from Amazon in the Electronics and Movies domains. Each dataset contains two main columns: review text and star rating (1–5).

1. **Amazon Reviews**
    - Contains approximately ~ 9901889 reviews.

All datasets are merged into one

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 datasets
- Inspect structure: check column names, data types, sample 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

df1.head()

View last few rows

df1.tail()

Check column names

df1.columns

Get DataFrame shape (rows, columns)

df1.shape

Check column data

types df1.dtypes # Full

summary info df1.info()

Summary statistics (for numeric columns)

df1.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=True)

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=True)

Clean Text Data.

- Convert text to lowercase

- Remove:
    - URLs
    - Punctuation
    - Numbers and special characters
    - Extra spaces

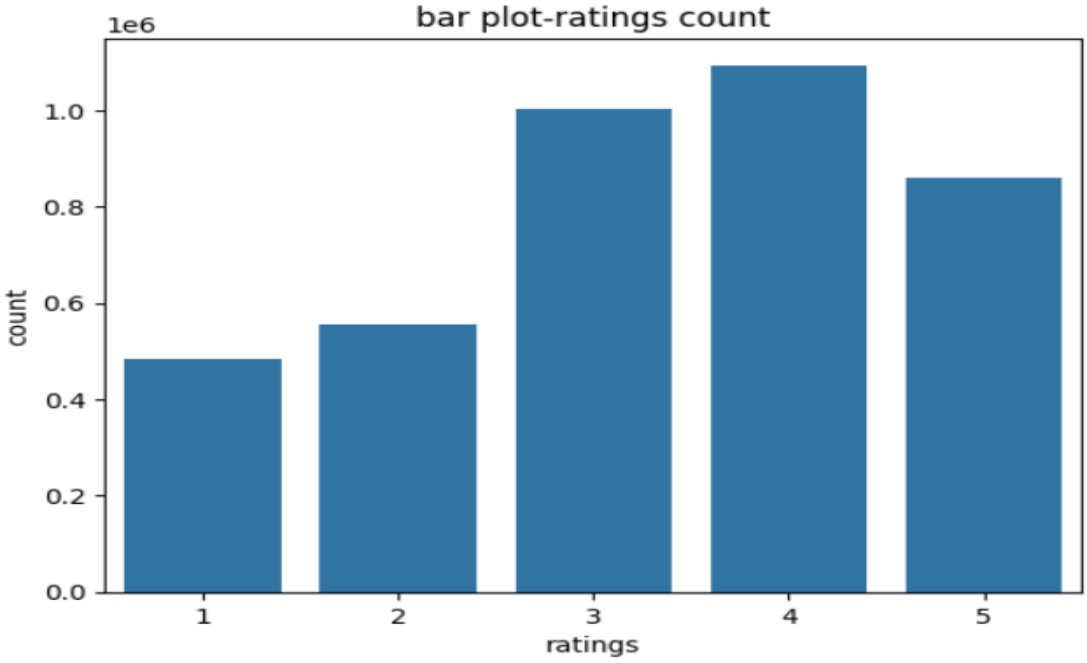
DATA VISUALIZATION:

Data Visualization is the process of representing data and information visually using charts, graphs, and other visual elements (like maps or word clouds) to help people understand patterns, trends, and insights in data.

BAR CHART.

A bar chart represents categorical data with rectangular bars. The length of each bar is proportional to the value it represents.

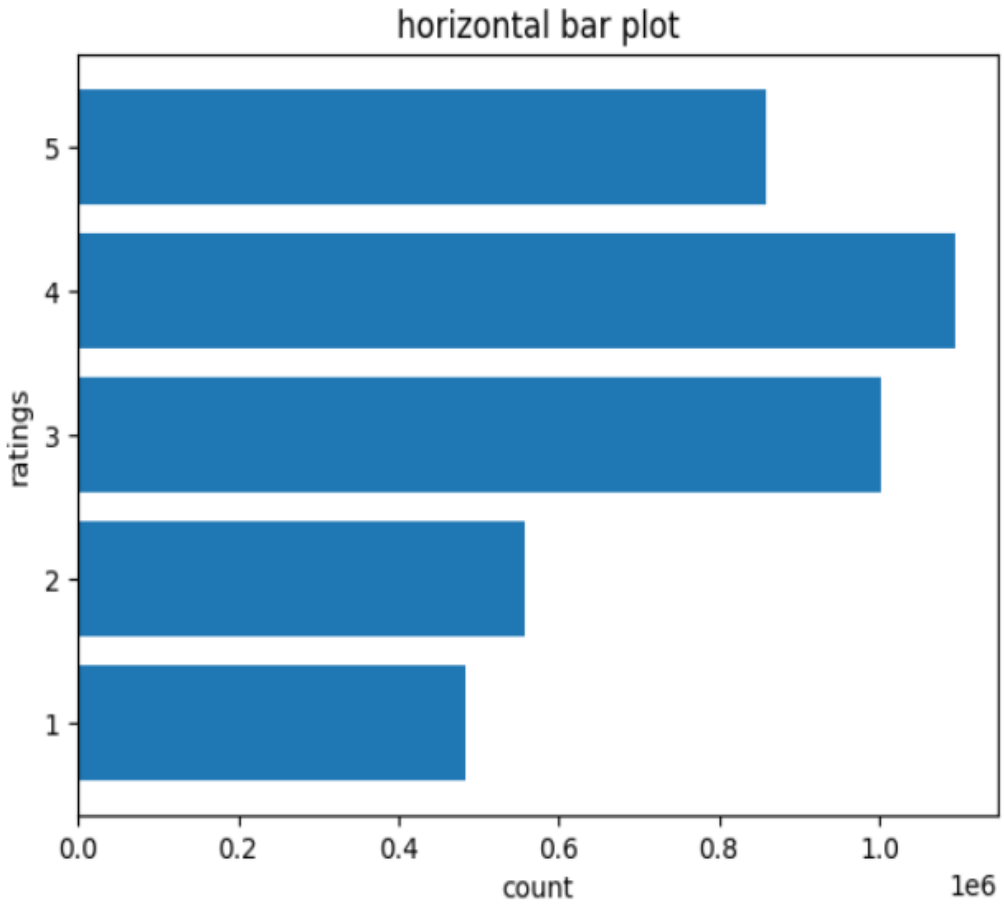
- Use: To show the number of reviews in each star rating (e.g., how many 1-star, 2-star, etc.).
- Number of reviews per rating (1 to 5).



Horizontal Bar Chart (HBar).

A horizontal bar chart is like a bar chart, but the bars extend horizontally instead of vertically.

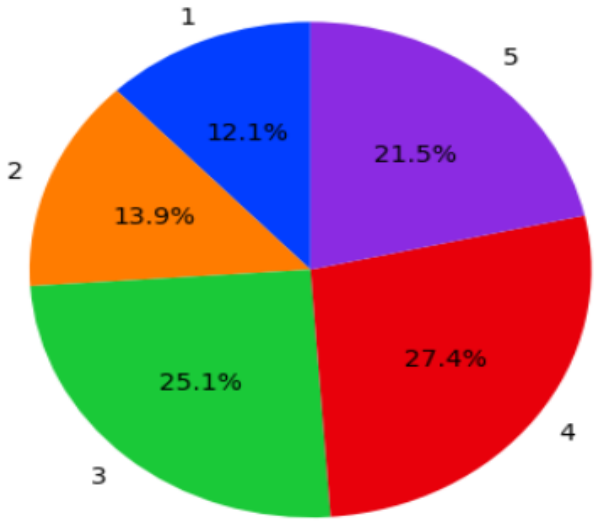
- To show average values like review length for each rating in a clean, readable format.



Pie Chart.

A pie chart is a circular chart divided into slices to illustrate proportions of categories.

- To show the percentage of each rating (how much of the data is 5-star, 4-star, etc.).

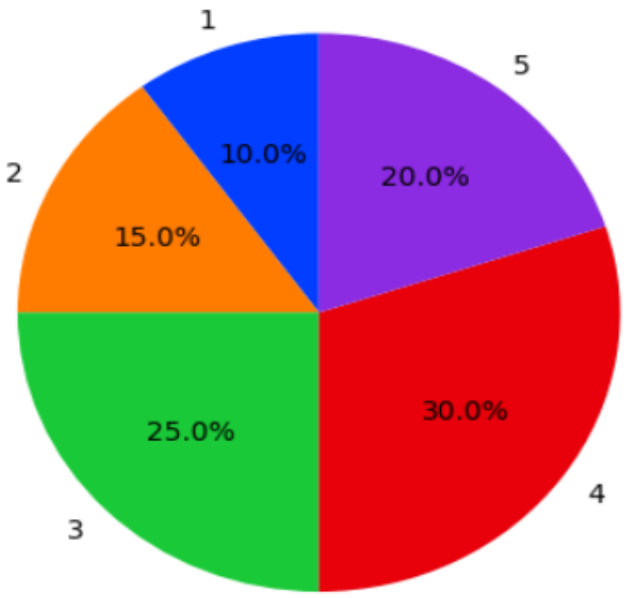


IMBALANCED DATASET:

Imbalanced data refers to a dataset in which the distribution of classes is not equal — meaning some classes (labels) have significantly more samples than others.

Balanced dataset with ratings:1-10%,2-15%,3-25%,4-2530%,5-20%

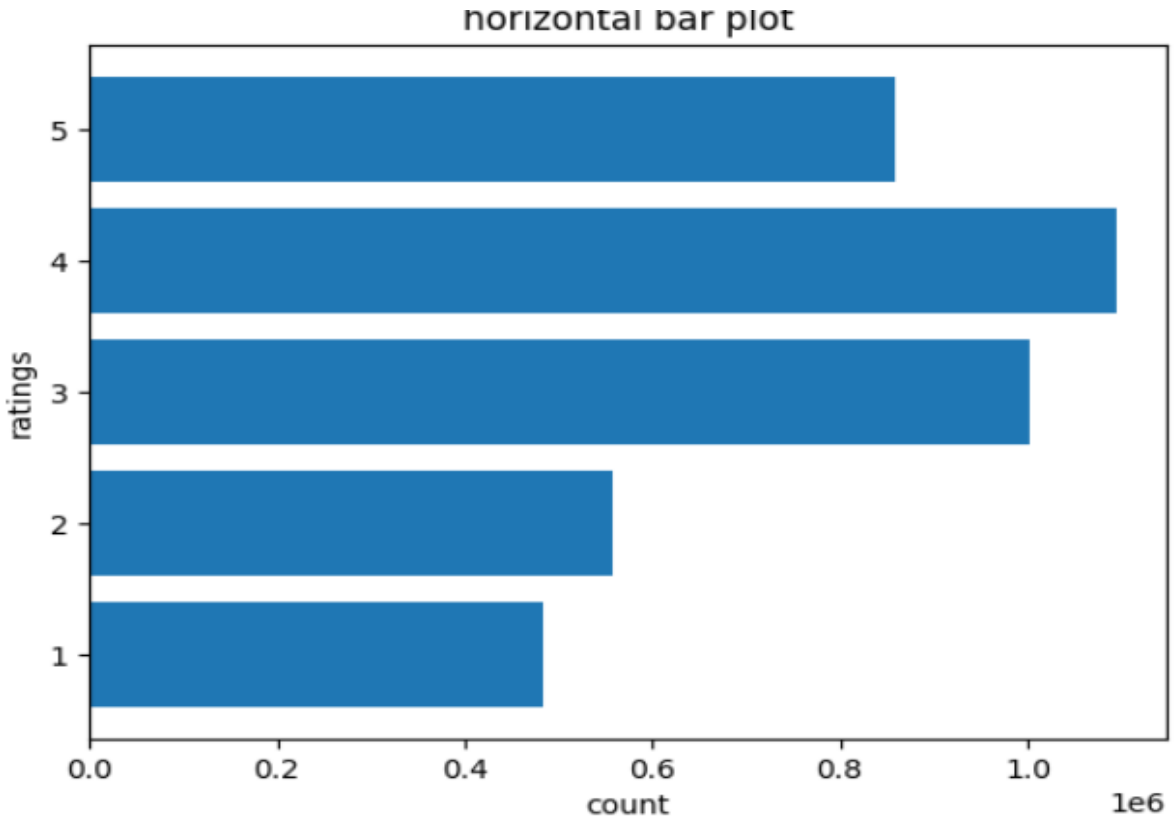
Pie Chart.


1
5
2
10.0%
20.0%
15.0%
25.0%
30.0%
3
4

BAR CHART.


bar plot-ratings count
800000
700000
600000
500000
400000
300000
200000
100000
0
count
1
2
3
4
5
ratings

Horizontal Bar Chart (HBar).



SAMPLES:

Showing 5 sample review(s) for Rating 1:

1. ingrate undermanagement
2. lack of quality leadership
3. keystone cops
4. travesty to work for
5. international brand ambassador

Showing 5 sample review(s) for Rating 2:

1. if you don’t want to be tide down don’t take the job and regret it

2. not a good place if you are looking for a rewarding job or career

3. tree of life must be preserved

4. reputation but the worst place for career advancement development

5. cultural management overhaul needed in high point

Showing 5 sample review(s) for Rating 3:

1. not sure what direction this company is going

2. go partner

3. pretty nice

4. low salaries but travel benefit perks

5. lacking leadership and direction

Showing 5 sample review(s) for Rating 4:

1. some good some okay

2. head medical affairs working on medical strategy and executing them

3. bell ringer

4. an excellent team of professionals

5. my exp at tip is good

Showing 5 sample review(s) for Rating 5:

1. Samsung sales executive

2. great company to work at only if you’re intelligent

3. i enjoy tip

4. passionate data driven development work

5. great work culture team

Train-Test Split:

What is Train-Test Split?

In machine learning, we split the dataset into:

- **Training Set** → Used to train the model (e.g., 80% of data)
- **Testing Set** → Used to evaluate the model's performance (e.g., 20% of data)

This helps ensure the model can generalize to unseen data.

Why Use stratify?

If the dataset is imbalanced (e.g., more 5-star reviews than 1-star), a random split could cause:

- Training set to have fewer examples of rare classes.
- Testing set to not represent all classes equally.

Stratification ensures:

- The class distribution (percentage of each rating) is the same in both train and test sets.

Why Bidirectional LSTM was Chosen.

The Bidirectional LSTM (BiLSTM) was chosen because:

- **Captures context in both directions:** Unlike standard LSTM, BiLSTM reads the text sequence forward and backward, which is crucial for understanding sentiment in reviews where meaning depends on surrounding words.
- **Effective for sequential data:** Customer reviews are textual sequences; LSTM handles long-term dependencies better than vanilla RNNs.
- **Proven performance:** BiLSTM models achieve higher accuracy in sentiment and review rating prediction tasks compared to traditional ML algorithms or simple neural networks.

Preprocessing Steps Used.

Before feeding data into the BiLSTM, the following preprocessing steps were applied:

1. **Text Cleaning:** Converted all text to lowercase and removed special characters and punctuation.
2. **Truncating Excessively Long Reviews:** Reviews longer than 300 words were truncated to reduce noise and ensure uniform input size.
3. **Encoding Labels:** Star ratings (1–5) were label-encoded and converted to one-hot vectors for multi-class classification.

4. **Tokenization and Padding:** Text was tokenized using Keras' Tokenizer and sequences were padded to a fixed length (max_len=200) for uniform input to the model.
5. **Optional Stopword Removal & Lemmatization:** Stopwords were removed and words were lemmatized to reduce vocabulary size and improve semantic understanding.

Model Training Logic.

The BiLSTM model was trained using the following logic:

- **Embedding Layer:** Either random or pretrained embeddings (GloVe) were used to convert words into dense vectors.
- **Bidirectional LSTM Layer:** 128 units with dropout and recurrent dropout to capture forward and backward context.
- **Dropout Layer:** 50% dropout applied after BiLSTM to prevent overfitting.
- **Dense Output Layer:** 5 neurons with softmax activation for predicting one of the five star ratings.
- **Loss Function:** categorical_crossentropy for multi-class classification.
- **Optimizer:** Adam optimizer for efficient convergence.
- **Callbacks:** EarlyStopping and ReduceLROnPlateau were used to avoid overfitting and adjust learning rate dynamically.

  Training was performed using a stratified 80-20 train-test split to ensure proportional representation of all rating classes.

Evaluation Results and Interpretation.

After training, the model was evaluated on the test set:

- **Accuracy:** Measures the proportion of correctly predicted ratings.
- **Precision, Recall, F1-Score:** Evaluated per class to see how well the model predicts each rating.
- **Confusion Matrix:** Visualized misclassifications between ratings.

Accuracy: 0.4471
Classification Report:

	precision	recall	f1-score	support
1	0.58	0.36	0.45	54232
2	0.40	0.33	0.36	81348
3	0.42	0.37	0.39	135580
4	0.42	0.63	0.51	162696
5	0.55	0.39	0.46	108464
accuracy			0.45	542320
macro avg	0.48	0.42	0.43	542320
weighted avg	0.46	0.45	0.44	542320