

AUTOATED REVIEW RATING SYSTEM

1.Project Overview

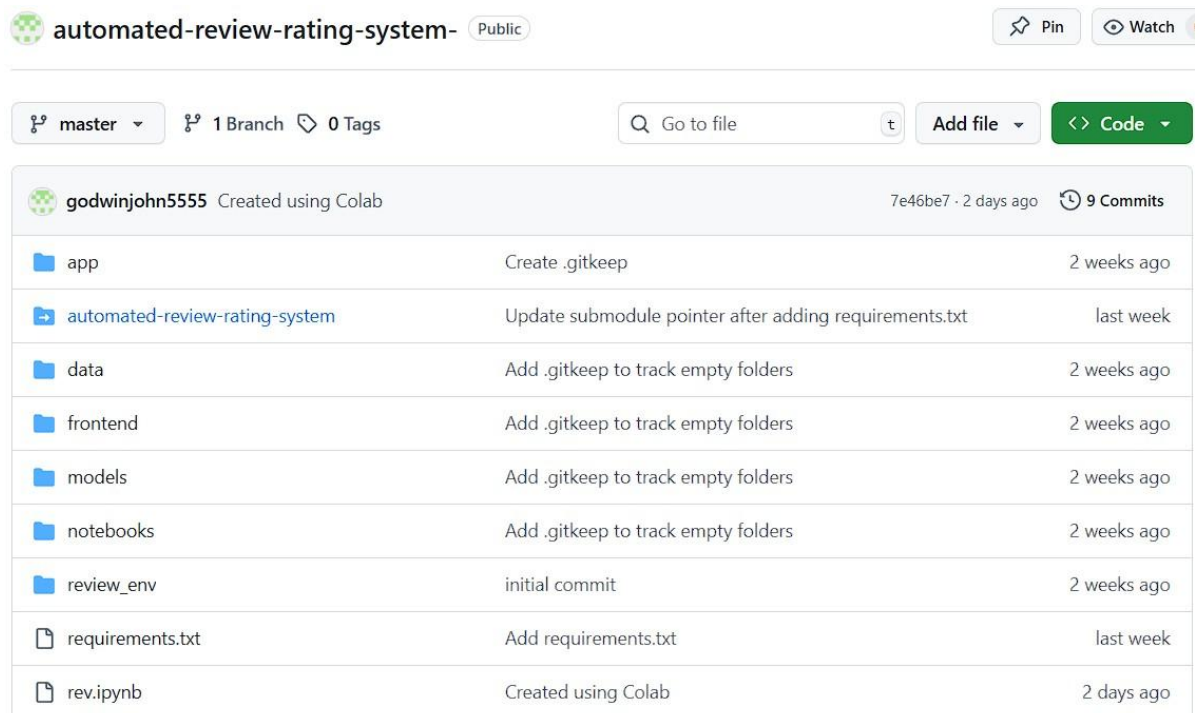
The Automated Review Rating System is a machine learning and NLP-based project that predicts star ratings (1–5) from customer reviews. It processes and analyzes review text to classify sentiment automatically, helping businesses save time, gain insights, and improve decision-making.

2.Environment Setup

- Python 3.12
- Libraries: pandas,nltk,scitkit-learn,numpy
- NLTK : Stopwords,punkt
- IDE: Google Colab,vs code

3.GitHub Project Setup

Created GitHub repository: automated-review-rating-system Structure of directory.



The screenshot shows the GitHub interface for the repository 'automated-review-rating-system'. At the top, it indicates the repository is 'Public' and has 'Pin' and 'Watch' buttons. Below this, the repository is created by 'godwinjohn5555' using Colab, with a commit hash '7e46be7' from '2 days ago' and '9 Commits'. The main section displays a list of files and folders with their commit messages and timestamps:

File/Folder	Commit Message	Timestamp
app	Create .gitkeep	2 weeks ago
automated-review-rating-system	Update submodule pointer after adding requirements.txt	last week
data	Add .gitkeep to track empty folders	2 weeks ago
frontend	Add .gitkeep to track empty folders	2 weeks ago
models	Add .gitkeep to track empty folders	2 weeks ago
notebooks	Add .gitkeep to track empty folders	2 weeks ago
review_env	initial commit	2 weeks ago
requirements.txt	Add requirements.txt	last week
rev.ipynb	Created using Colab	2 days ago

4.Data.Collection

- Data was collected from Kaggle and it related Amazon Fine Food Reviews Dataset, a publicly available dataset containing over 568,000 customer reviews of food products from Amazon.
- Dataset Size; Total Records: 568,454 reviews, Columns: 10
- Dataset link: <https://www.kaggle.com/snap/amazon-fine-foodreviews/downloads/amazon-fine-food-reviews.zip>
- Final dataset contains 2 column with Review and Rating
- 1star=52264, 2star=29743, 3star=42638, 4star=80654, 5star=363102

4.1.Imbalanced dataset

- Link for Imbalanced dataset:
<https://github.com/godwinjohn5555/automated-review-rating-system->
- Balanced dataset was created with 5000 rows each rating

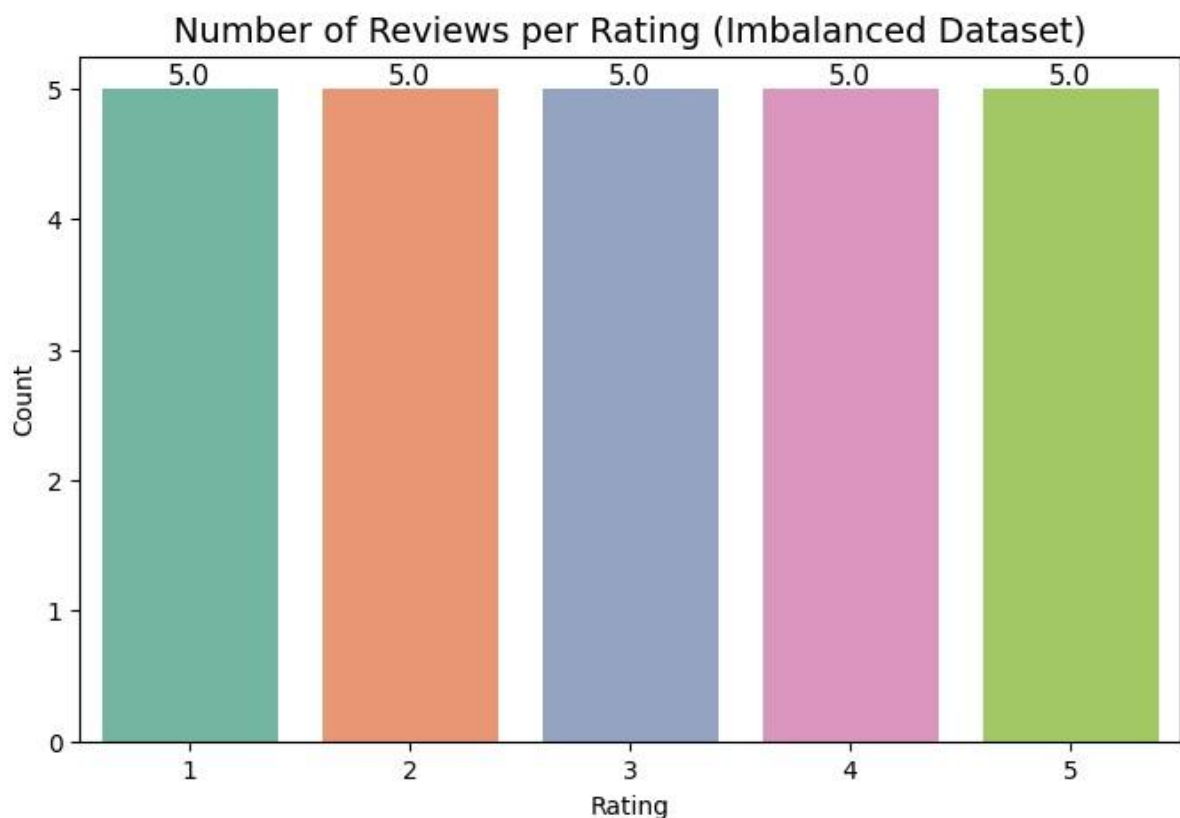


Fig count plot

Distribution of Reviews per Rating (Imbalanced Dataset)

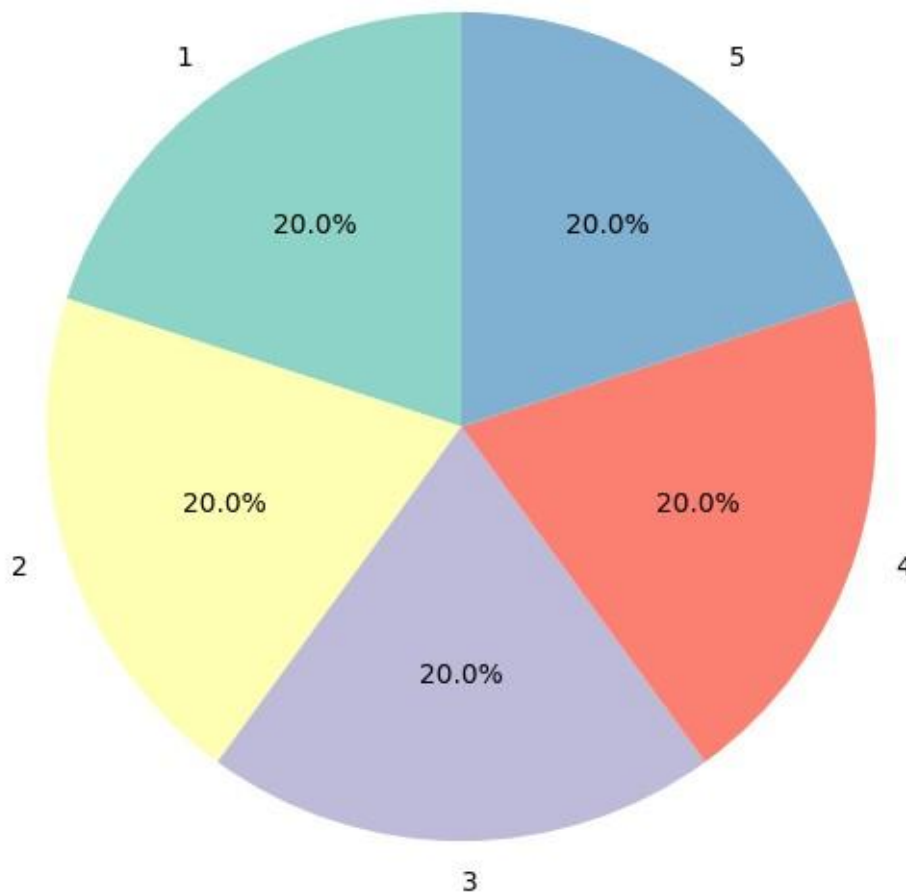


Fig pie chart

5.Data Preprocessing

Effective data preprocessing is essential to improve the performance and accuracy of machine learning models. The following techniques were applied to prepare the raw dataset for modeling.

5.1 Removing Duplicates

Duplicate rows containing the exact same review and rating were removed to prevent bias and overfitting. This ensured that the dataset only included unique observations.

Code:

```
imbalanced_cleaned=imbalanced_df.drop_duplicates(subset=['Cleaned_Review'], keep='first')
```

```
print("Shape before cleaning:", imbalanced_df.shape)
print("Shape after cleaning :", imbalanced_cleaned.shape)
```

5.2 Removing Conflicting Reviews

Some reviews had identical text but different start ratings. These inconsistencies can confuse the model. Such conflicting entries were identified and removed to maintain label clarity.

```
Code: rating_nunique =
imbalanced_df.groupby('Cleaned_Review')['Rating'].nunique() conflict_texts
= set(rating_nunique[rating_nunique > 1].index) before =
imbalanced_df.shape[0] imbalanced_df =
imbalanced_df[~imbalanced_df['Cleaned_Review'].isin(conflict_texts)].copy()
print(f"Removed conflicts (same text, different ratings): {before -
imbalanced_df.shape[0]} rows; now {imbalanced_df.shape[0]}")
```

5.3 Handling Missing Values

Rows with missing or null values particularly in the review text or rating columns were removed. This step ensured the dataset was complete and meaningful for analysts.

```
Code: imbalanced_df = imbalanced_df.dropna(subset=["Cleaned_Review",
"Rating"]) print("Shape after removing missing values:",
imbalanced_df.shape)
```

5.4 Dropping Unnecessary Columns

Non-essential columns such as review IDs, timestamps, or user identifiers were dropped. These fields did not contribute to the model and could introduce noise or privacy concerns.

5.5 Lowercasting

Text All interview text was converted to lowercase to maintain uniformity. This helps prevent duplication of tokens like “good” being treated as separate words.

5.6 Removing URLs

URLs present in the review text were removed using regular expressions.

5.7 Removing Emojis and Special Characters

It is the process of cleaning text by eliminating unnecessary symbols, emojis, and non-alphanumeric characters. This step ensures that the dataset contains only meaningful words, making it easier for NLP models to analyze and learn from the text.

5.8 Removing Punctuation

Removing punctuation means cleaning text by eliminating symbols such as ., ! ? ; () [] etc. These characters do not usually add meaning for sentiment analysis or review rating tasks, so removing them helps in simplifying the text and focusing on the actual words.

Code:

```
def remove_punctuation(text):  
    return text.translate(str.maketrans("", "", string.punctuation))  
  
imbalanced_df["Cleaned_Review"] =  
imbalanced_df["Cleaned_Review"].astype(str).apply(remove_punctuation)
```

5.9 Stopwords Removal

Stopwords are frequently used words in a language (such as “is”, “the”, “in”, “on”) that usually carry little meaning on their own.

In NLP, removing stopwords helps reduce noise and allows models to focus on the more meaningful words in the text.

We use the Natural Language Toolkit (NLTK) to access the English stopwords list. NLTK provides a predefined list of 179 English stopwords.

Stopword Statistics (Imbalanced Dataset)

- Total Stopwords in NLTK (English): 179
- Total Stopwords Found in Dataset: 449,596

Top 10 Most Frequent Stopwords

Stopword Count

the	38,202
i	32,982
a	22,398
and	22,331
it	21,288
to	19,419
of	15,757
is	13,621
this	13,017
in	10,286

Code:

```
def remove_stopwords(text):  
    words = text.split()  
    filtered = [word for word in words if word.lower() not in stop_words]  
    return " ".join(filtered)  
  
imbalanced_df["Cleaned_Review"] =  
imbalanced_df["Cleaned_Review"].astype(str).apply(remove_stopwords)
```

5.10 Lemmatization

Lemmatization is the process of reducing words to their base or dictionary form (lemma) while keeping the meaning intact. For example:

- “running” → “run”
- “better” → “good”

It helps standardize words so NLP models can treat different forms of a word as the same.

Why lemmatization is better than stemming?

- Produces valid words – Lemmatization returns proper dictionary words, while stemming may give non-words (e.g., “studies → study” vs “studies → studi”).
- Cleaner text – Creates consistent, meaningful tokens for NLP models.

5.11 Filtering by Wordcounts

Filtering by word counts means removing reviews or text entries that are too short or too long, since they may not provide useful information for analysis.

6. Data Visualisation

Box plot

A box plot (or whisker plot) is a graphical representation of a dataset's distribution and variability. It shows the minimum, first quartile (Q1), median, third quartile (Q3), and maximum values, and highlights outliers. Box plots are useful for quickly understanding the spread, skewness, and presence of extreme values in the data.

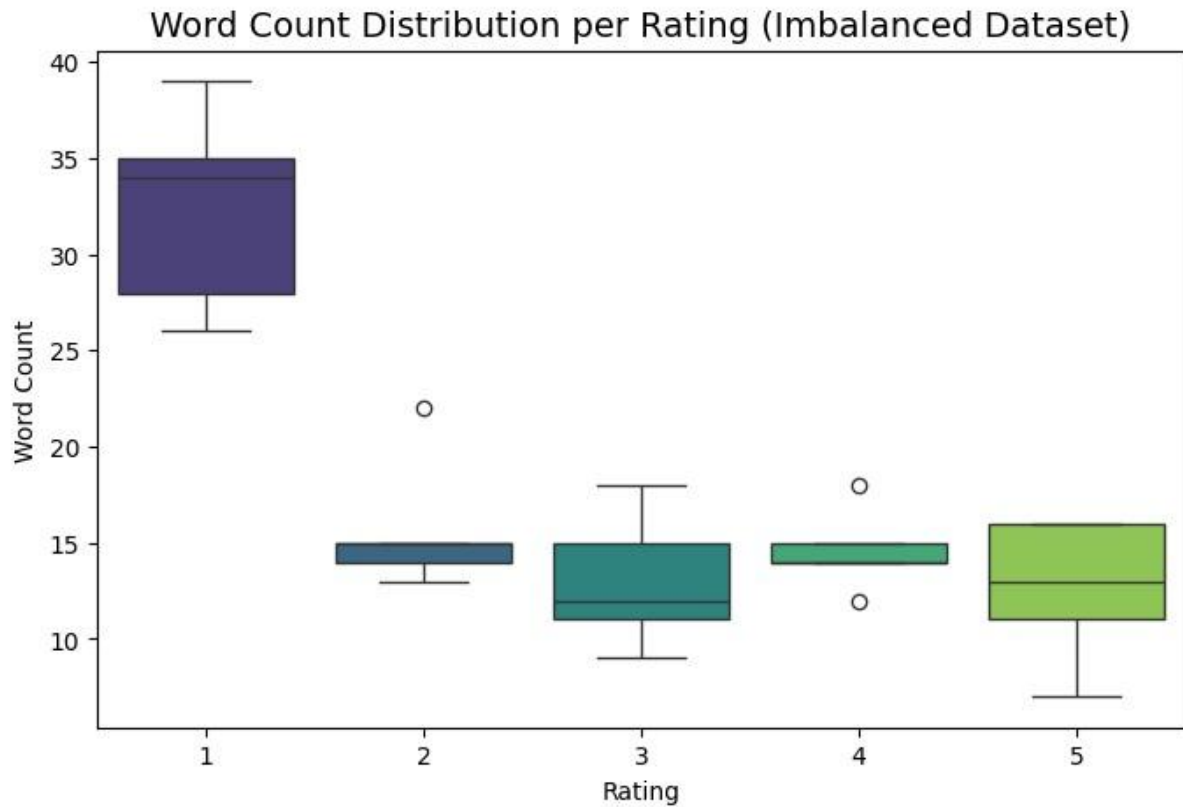
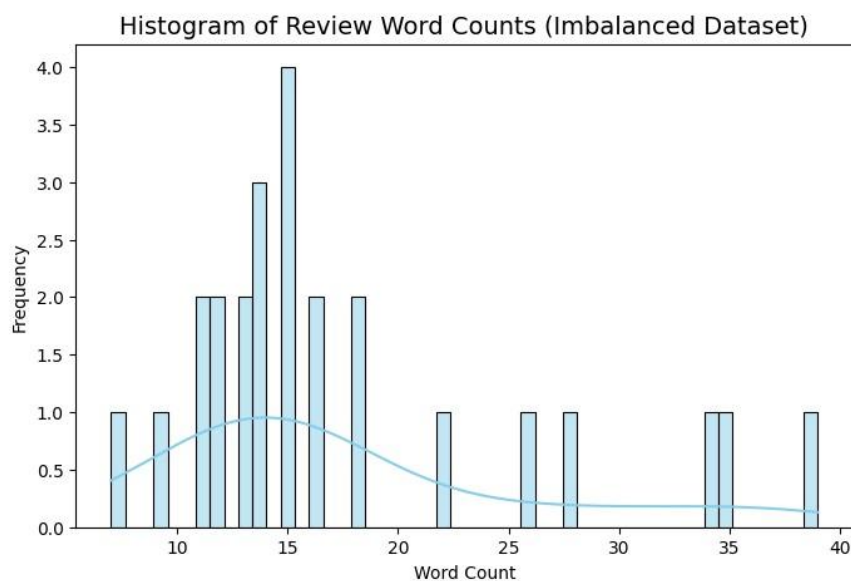


Fig box plot

Histogram

A histogram is a graphical representation of the distribution of numerical data. It divides the data into intervals (bins) and shows the frequency of values within each bin. Histograms are useful for visualizing patterns, such as skewness, spread, and the shape of the dataset.



Barplot

A bar plot is a type of chart used to represent categorical data with rectangular bars, where the length or height of each bar corresponds to the value or frequency of the category it represents. It provides a simple way to compare quantities across different groups or categories. The categories are usually placed on one axis, while the values are placed on the other, making it easy to visually compare differences between groups. Bar plots can be drawn vertically or horizontally depending on the type of comparison required.

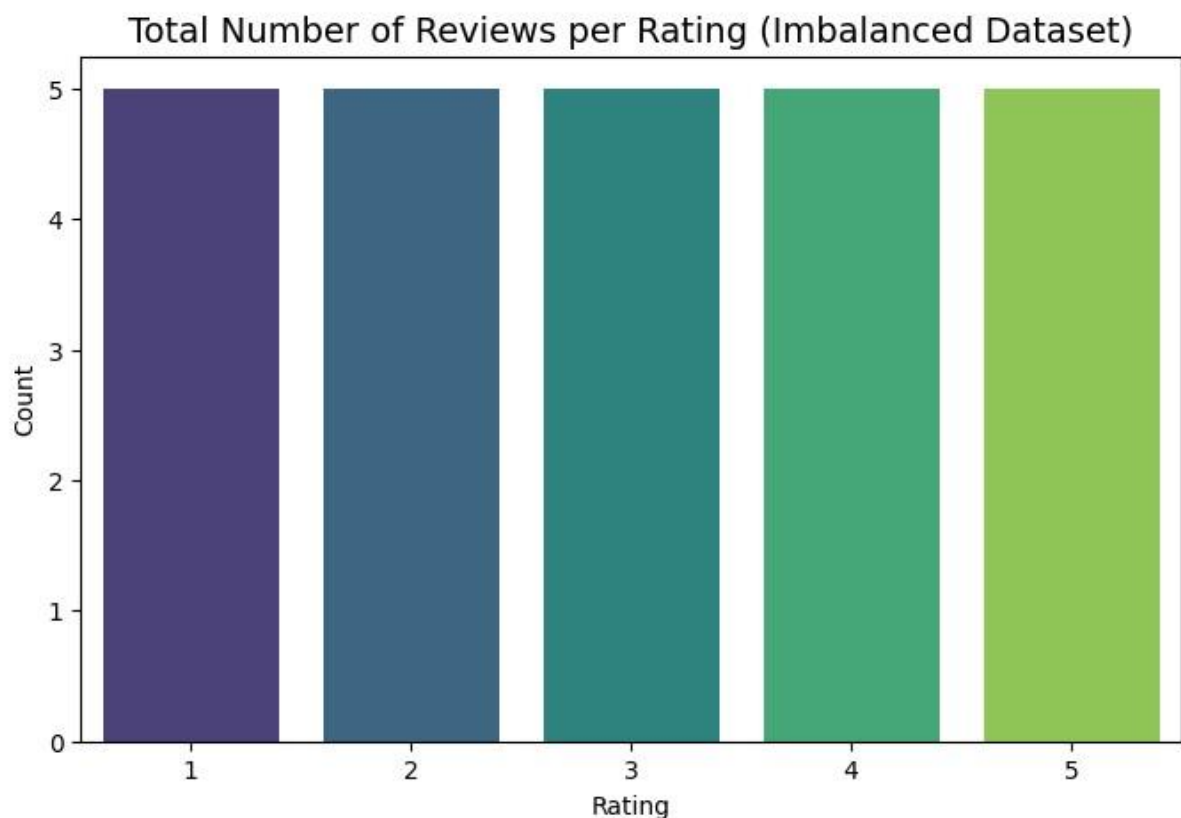


Fig barplot

Samples of 1 star rating

Showing 5 sample review for rating 1:

1. the tea i received is not caffeine free the box is different from the pictured box as well it has ok flavor but could be stronger per bag
2. got it for a friend so naturally i didn t open it to look everything was thin and cheap save yourself money get pancake mix and a real pitcher the cake ring can be found at a thrift store

3. used of the bottles over the past month and have zero effect on helping with sleep it was no different than taking nothing expensive and worthless
4. i bought this to make coconut milk for my coffee but it doesn't turn into milk even in a blender the bits are too big made me choke on my coffee will never reorder
5. arrived with one package that was loose the other one tight had tape on it so i assume it was like that when it left outside packaging was fine would not order it again.

Samples of 2 star rating

Showing 5 sample review for rating 2:

1. usually i buy freeze dried pears from whole foods disappointment doesn't even come close
2. as a note to anyone buying this product hexane extraction not really natural
3. this earl grey tea is just ok the bergamont flavor is muted and dull
4. this is an attractive machine looks good in my kitchen downside it's not very well made and broke down within months
5. while these do have a great ingredient label the texture and flavor are off putting.

Samples of 3 star rating

Showing 5 sample review for rating 3:

1. my family and i recently tasted this switch drink very sweet would not buy again
2. this is the best hot cocoa i have found so far but i hope i can find better
3. not thrilled with these lid doesn't fit well not worth it
4. absolutely love the different flavors of coffee but packaging was odd
5. this tea has a gentle laxative effect flavor chalky.

Samples of 4 star rating

Showing 5 sample review for rating 4:

1. the curry arrived well packaged and timely and most importantly was delicious
2. one of the reasons i like this product is there are no added sweeteners nice easy breakfast cereal
3. the pieces of fruit were ripe flavorful firm not water logged i will reorder
4. my dog loves these cookies box weight was listed wrong no problem with amazon
5. why oh why do studios stamp deluxe edition blu ray looks great but no extras.

Samples of 5 star rating

Showing 5 sample review for rating 5:

1. crackers are great they come in a small enough package so they don t get stale
2. these candies are not your typical kids sour candies exotic flavors
3. i bought this because my boyfriend likes green mountain coffee he loved it 4.
i bought these at a meijer s everyone was eating them saying they were the best
5. excellent coffee very rich and bold flavor.

7.Train-Test Split

Train-Test Split is a technique to divide the dataset into two separate sets:

- Training Set (typically 80%): Used to train the machine learning model.
- Test Set (typically 20%): Used to evaluate the model's performance on unseen data.

This separation ensures that the model generalizes well and is not just memorizing the training data.

Why Stratified Split?

In classification problems like review rating prediction, stratified splitting is important to maintain class balance (equal distribution of ratings) in both training and testing sets.

Without stratification, some rating classes (like 1-star or 5-star) may be underrepresented in the test set, leading to biased evaluation.

How it was done

To prepare the data for model training, the dataset was first shuffled randomly to eliminate any order bias. A stratified train-test split was then performed using `train test split()` from `sklearn.model_selection` with the `stratify=y` argument to ensure that all star ratings were proportionally represented in both training and testing sets. The dataset was split using an 80% training and 20% testing ratio. After splitting, all text preprocessing steps-including lowercasing, lemmatization, stopword removal, and cleaning were applied separately to `x_train` and `x_test` to prevent data leakage and maintain model integrity.

Code:

```
from sklearn.model_selection import train_test_split

if "Cleaned_Review" in imbalanced_df.columns:
    imbalanced_df = imbalanced_df.rename(columns={"Cleaned_Review":
"Text_clean"}) elif "Review" in
imbalanced_df.columns:
    imbalanced_df = imbalanced_df.rename(columns={"Review":
"Text_clean"}) X = imbalanced_df["Text_clean"] y =
imbalanced_df["Rating"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y
)
```

8.Text Vectorization

Machine learning models can't work directly with raw text-they require numerical input. Vectorization is the process of converting text data into numerical features.

TF-IDF (Term Frequency – Inverse Document Frequency)

TF-IDF is a statistical technique that represents how important a word is to a document relative to the entire corpus.

- TF (Term Frequency): How often a word appears in a document.
- IDF (Inverse Document Frequency): Penalizes common words and highlights rare but Important ones.

This approach helps to capture both word relevance and discriminative power, making it better than simple word counts.

Formula for TF-IDF:

TF(t,d)=Total number of terms in document d
Number of times term t appears in

$$TF(t, d) = \frac{\text{number of times } t \text{ appears in document } d}{\text{total number of terms in document } d}$$

$$IDF(t) = \log \frac{N}{df + 1}$$

$$IDF(t, d) = TF(t, d) * IDF(t)$$

Code:

```
tfidf = TfidfVectorizer(  
max_features=5000,
```

```
    ngram_range=(1, 2), # unigrams + bigrams
)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
print("TF-IDF Shape (Train):", X_train_tfidf.shape) print("TF-IDF
Shape (Test) :", X_test_tfidf.shape)
```