AUTOMATED REVIEW RATING SYSTEM

Imbalanced data set

1. PROJECT OVERVIEW

A smart machine learning system designed to analyse customer reviews and automatically predict the appropriate star rating. It leverages Natural Language Processing (NLP) techniques to extract emotions, sentiment, and meaningful insights from textual feedback. The model is trained on labelled review datasets to identify relationships between customer language and their rating behaviour.

2. Environment Setup

. **Python Version:** 3.13

. **IDE** used- VSCODE

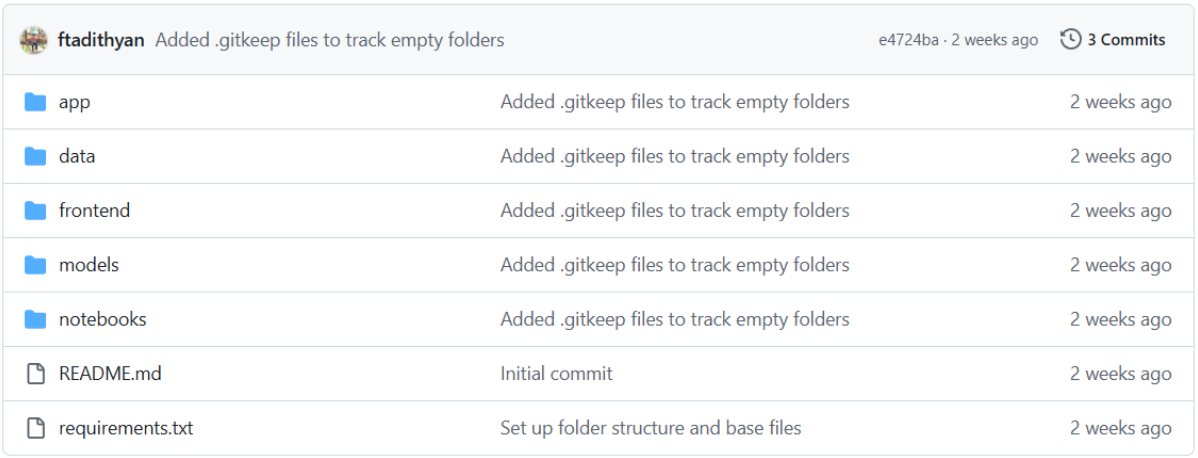
. **Libraries Used:**

- NumPy, pandas – Data handling
- seaborn, matplotlib – Visualization
- scikit-learn – Model building & evaluation
- spacy, nltk, re, string, BeautifulSoup – Text preprocessing

3. GITHUB PROJECT SET UP

Created GitHub Repository: **automated-review-rating-system**

Structure of directory



4. DATA COLLECTION

. Source: The dataset was collected from Kaggle.

🔗 **Amazon Reviews: Unlocked Mobile Phones**

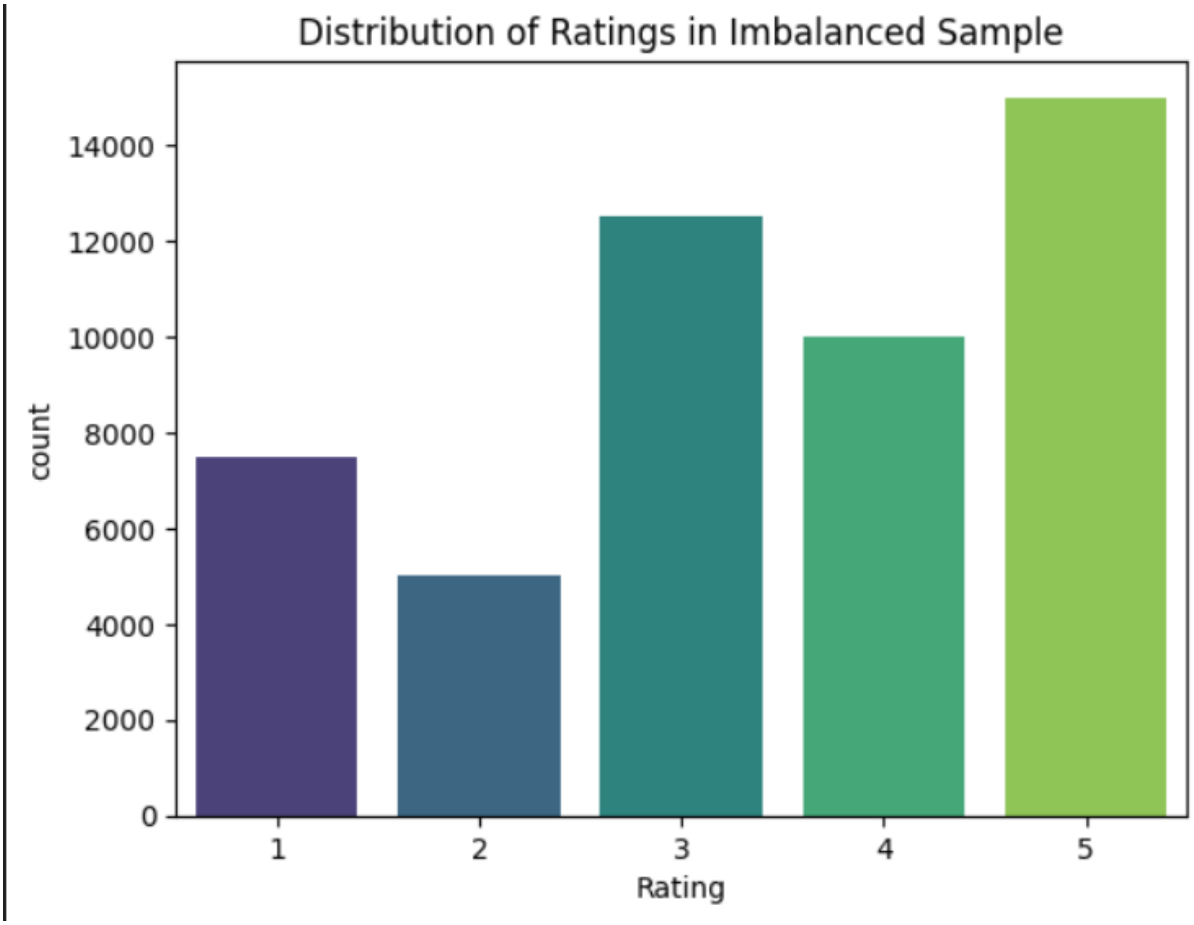
. Total Records: 413,840 customer reviews

. Total Columns: 6

. Final Data set contains 2 columns Reviews and Rating

4.1. Imbalanced Dataset

. The ratings in this imbalanced dataset of 50,000 entries are distributed in the ratio 30: 25: 20: 15: 10 across classes 5, 3, 4, 1, and 2 respectively.



5. DATA PREPROCESSING

Data preprocessing is a crucial step that involves cleaning and transforming raw data into a suitable format for model training. In this project, various text cleaning and normalization techniques were applied to prepare customer reviews for sentiment analysis. The process includes removing noise such as URLs, HTML tags, emojis, punctuation, and special characters, as well as converting text to lowercase, removing stop words, and applying lemmatization. These steps ensure that only meaningful textual information is retained, improving the accuracy and performance of the NLP model.

5.1. Handling Missing Values

. **Missing values** are empty or null entries in a dataset that can negatively affect analysis and model performance.

#checking missing values

pdf.isna().sum()

output-

Product Name - 0

Brand Name - 65171

Price - 5933

Rating - 0

Reviews - 70

Review Votes - 12296

. Dropped missing text data: Rows with missing Brand Name or Reviews were removed since these fields are essential.

. Filled numeric missing values: Missing values in Price and Review Votes were replaced with the median of each column to minimize the effect of outliers.

Result: Dataset now has no missing critical information and is ready for analysis or modeling.

Code:

```
pdf=pdf. dropna(subset=['Brand Name'])
pdf = pdf.dropna(subset=['Reviews'])
```

pdf["Price"] = pdf["Price"]. fillna(pdf["Price"]. median())

pdf["Review Votes"] = pdf['Review Votes'].fillna(pdf['Review Votes'].median())

5.2 . Removing Unwanted Columns

Non-essential columns such as Product Name, Brand Name, Price, Review Votes were dropped and assign new columns like Rating, Reviews to pd1 variable

Code :

pdf1=pdf[['Rating', 'Reviews']]
pdf1

5.3. Removing Duplicates

Duplicates rows containing the exact same review and different rating were removed to prevent bias and overfitting. This ensure that data only included unique observations

finding duplicates

pdf1['Reviews']. duplicated(). sum()

output:

np.int64(205495)

#removing duplicates

pdf1 = pdf1.drop_duplicates(subset=['Reviews'], keep='first')
pdf1

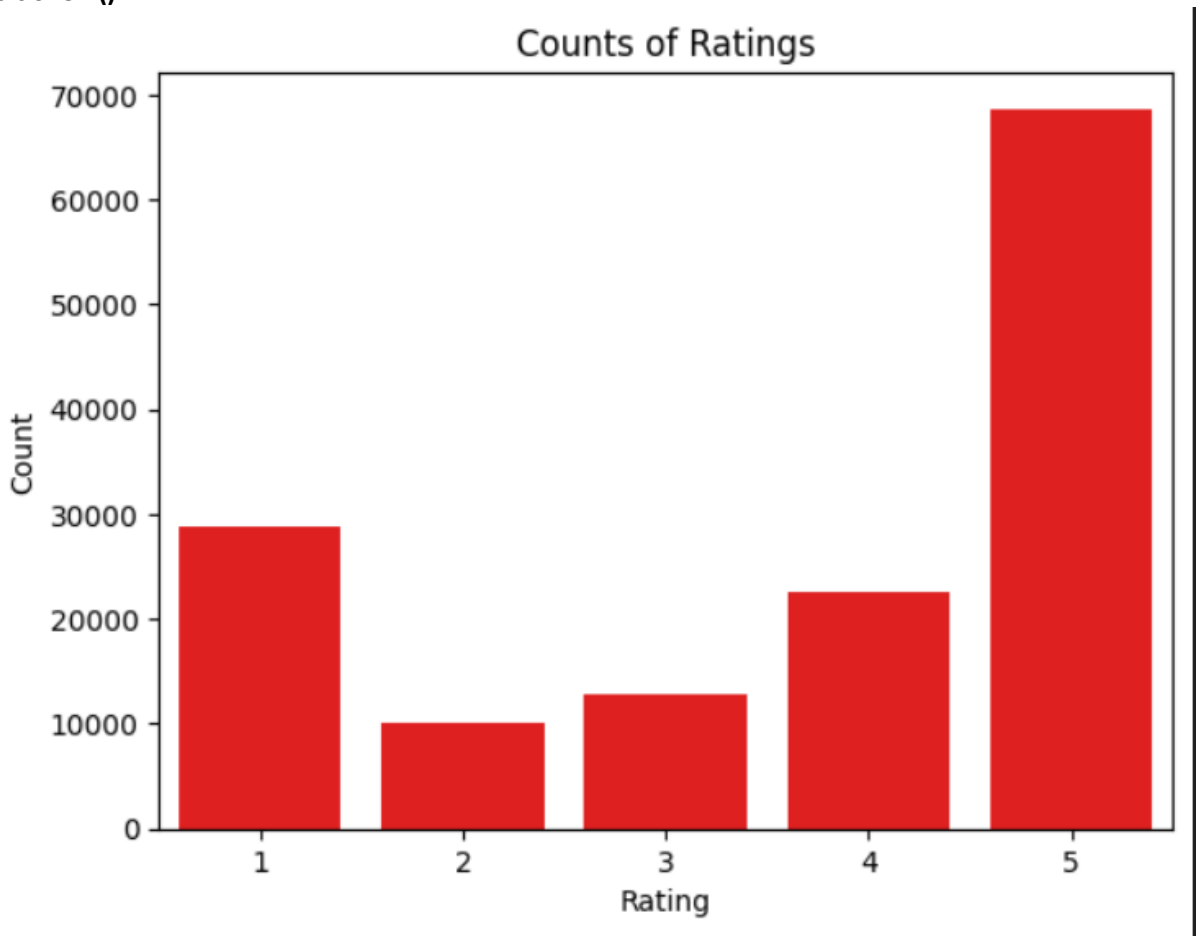
6. VISUALIZIZATION OF PREPROCESSED DATA

A bar plot was created to show the distribution of all ratings in the dataset, helping to quickly identify the frequency of each rating category

Code:

sns.barplot(x=count1.index, y=count1.values,color='red')
plt.xlabel('Rating')

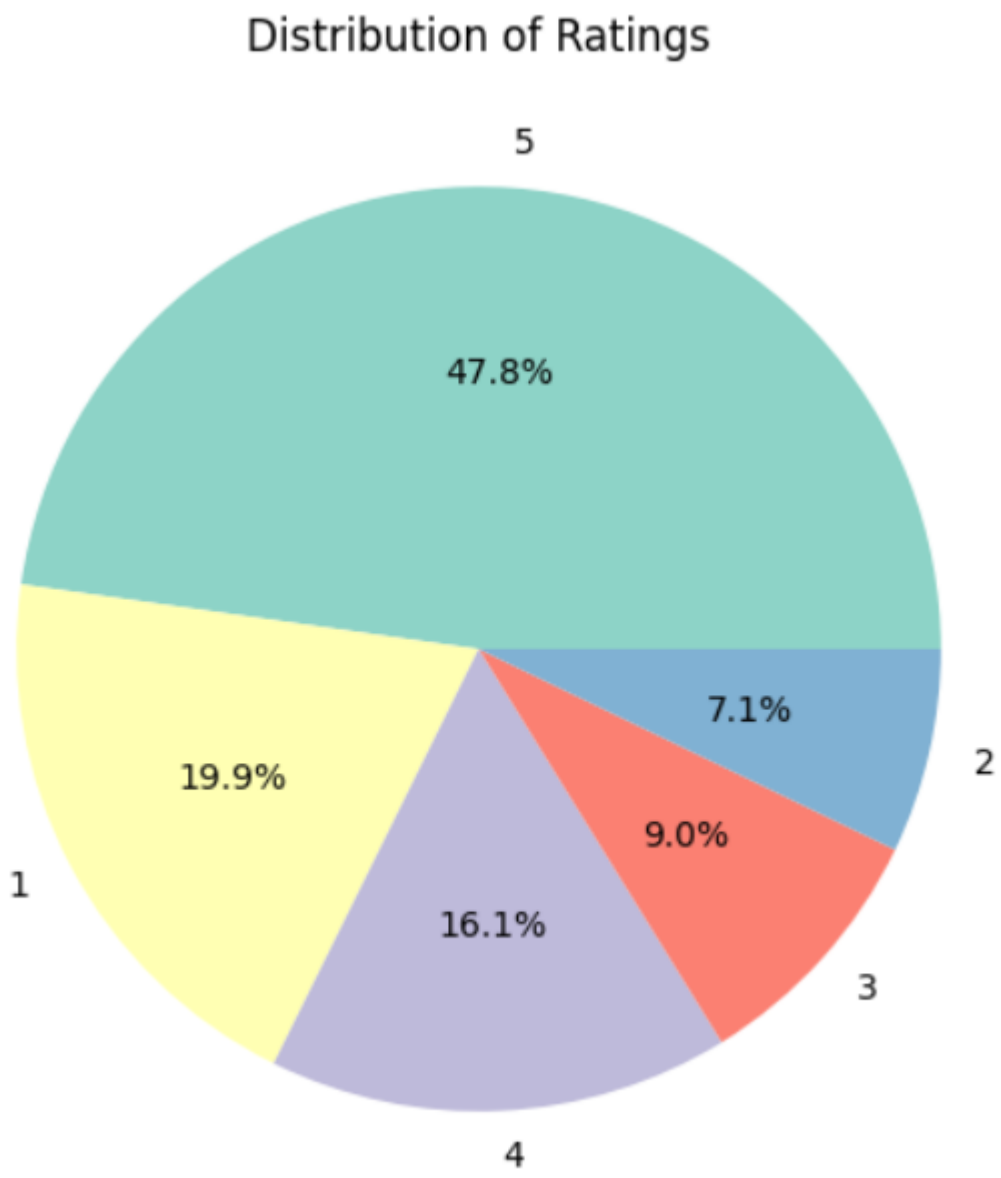
```
plt.ylabel('Count')
plt.title('Counts of Ratings')
plt.show()
```



A pie chart was created to display the distribution of ratings in the dataset. Each slice represents a rating category, and the chart shows the proportion of reviews for each rating, providing a clear overview of how ratings are spread across the dataset.

Code:

```
plt.figure(figsize=(6,6))
plt.pie(count1.values, labels=count1.index, autopct='%1.1f%%', colors=plt.cm.Set3.colors)
plt.title('Distribution of Ratings')
plt.show()
```



---

7. TEXT PREPROCESSING AND CLEANING

Reviews were cleaned by lowercasing, removing URLs, HTML tags, punctuation, stop words, and numbers, lemmatized, and filtered to keep only 3–100-word reviews.

7.1 . Lowercasing Text

Convert all text to lowercase to maintain uniformity and avoid treating the same word in different cases as different words.

Why Used: Helps reduce redundancy in text analysis

Code :

text = text.lower()

7.2. Remove URLs

Remove any links from the text that do not contribute to the review sentiment

Why Used: URLs are irrelevant for textual analysis and may add noise.

Code:

text = re.sub(r'http\S+|www\S+|https\S+', '', text)

7.3. Remove HTML Tags

Strip out any HTML tags embedded in the text**.**

Why Used: HTML tags do not carry useful information for analysis.

Code:

text = BeautifulSoup(text, "html.parser").get_text()

7.4. Remove Punctuation, Special Characters, and Numbers

Clean text by removing punctuation, symbols, and numeric characters.

Why Used: These characters are usually irrelevant for natural language processing.

Code :

text = re.sub(f"[{re.escape(string.punctuation)}]", "", text)
text = re.sub(r'\d+', '', text)

7.5 Tokenization

Split the text into individual words or tokens.

Why Used: Necessary for further processing like stopwords removal or lemmatization.

Code:

words = text.split()

7.6 Remove Stopwords

Stopwords such as “the” , ”is” and “and”, are frequently occurring words that might not add significant semantic value .We us libraries like **spaCy** to filter these words out .reducing noise and focusing on words that contribute meaning to the reviews ,there were total **326 stop words** in spaCy

a, able, about, above, abroad, according, accordingly, across, actually, adj, after, afterwards, again, against, ago, ahead, ain, all, allow, allows, almost, alone, along, already, also, although, always, am, amid, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anyway, anyways, anywhere, apart, appear, appreciate, appropriate, are, aren, aren't, around, as, aside, ask, asking, associated, at, available, away, awfully, b, back, backward, backwards, be, became, because, become, becomes, becoming, been, before, beforehand, begin, behind, being, believe, below, beside, besides, best, better, between, beyond, both, brief, but, by, c, ca, came, can, cannot, cant, can't, caption, cause, causes, certain, certainly, changes, clearly, co, com, come, comes, concerning, consequently, consider, considering, contain, containing, contains, corresponding, could, couldn, couldn't, course, currently, d, dare, daren, daren't, definitely, described, despite, did, didn, didn't, different, directly, do, does, doesn, doesn't, doing, don, don't, done, down, downwards, during, e, each, edu, eg, eight, either, else, elsewhere, end, ending, enoug

Code :

words = [w for w in words if w not in stop_words]

7.7 Lemmatization

Lemmatization is a text normalization technique in Natural Language Processing (NLP). Its main goal is to reduce words to their base or dictionary form, called a lemma. Unlike stemming, which may just cut off prefixes or suffixes, lemmatization considers the context and part of speech of a word to produce a valid word.

Reduces words to their base form, e.g., *“running”* → *“run”*.

Technology used: **WordNetLemmatizer** from **NLTK** (or similar in spaCy).

Why lemmatization is better than stemming

Lemmatization is preferred over stemming because it produces meaningful, dictionary-valid words rather than just chopping off word endings. This makes the text more accurate and contextually correct for NLP tasks like search, sentiment analysis, or machine learning.

Example:

- Stemming: "running" → "runn"
- Lemmatization: "running" → "run"

Code:

```
cleaned_words = [lemmatizer.lemmatize(w) for w in cleaned_words]
```

7.8 . Cleaned Words and Remove Extra Spaces

Rejoin the processed tokens into a single string and remove extra whitespace.

Why Used: Produces clean, standardized text suitable for modeling.

Code:

```
# Join cleaned words
cleaned_text = " ".join(cleaned_words).strip()
```

7. 9. Filter Reviews by Length:

Only reviews with **3 to 100 words** are retained to focus on meaningful content.

Code :

```
balanced_df['word_count'] = balanced_df['clean_text'].apply(lambda x: len(x.split()))

filtered_df = balanced_df[(balanced_df['word_count'] >= 3) & (balanced_df['word_count'] <= 100)]
filtered_df
```

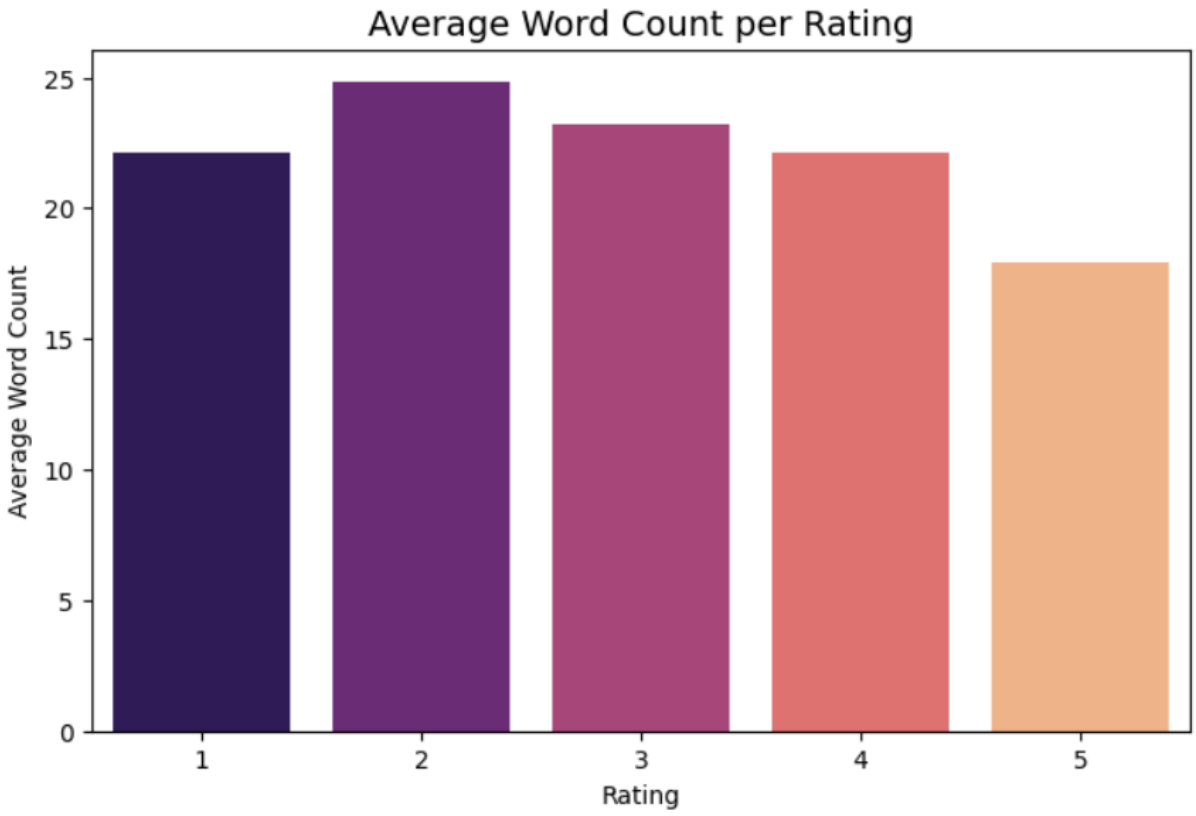
8. VISUALIZATION

8.1 AVERAGE WORD COUNT PER RATING

Code :

```
avg_words = filtered_df1.groupby('Rating')['word_count'].mean().reset_index()
```

```
plt.figure(figsize=(8,5))
sns.barplot(x='Rating', y='word_count', data=avg_words, palette='magma')
plt.title('Average Word Count per Rating', fontsize=14)
plt.xlabel('Rating')
plt.ylabel('Average Word Count')
plt.show()
```



8.2 .Samples of 5 Ratings

Show 5 random samples per rating

```
for rating in sorted(filtered_df['Rating'].unique()):
    print(f"\n===== Rating: {rating} =====")
```

```
   samples = filtered_df[filtered_df['Rating'] == rating].sample(n=5,
random_state=42)
   for i, review in enumerate(samples['clean_text'], 1):
      print(f"{i}. {review}\n")
```

output:

===== Rating: 1 =====

1. currently samsung solstice wanted replacement one ordered would accept simm card though knew refurbished phone horrified previous owner information photo still phone thankfully able returned problem

2. phone one year apple warranty

3. kept telling required sim card memory card still didnt work memory card sim card app kept freezing phone returned item received refund form amazon credit amazon account

4. product listed new showed sign wear

5. broke within two day using even expense getting new sim card fit screen pixelating like crazy amazon say late return warranty defective product ugh really disappointed

===== Rating: 2 =====

1. love razor phone th razor one signal spend time chasing signal razor got great signal one button stick spend alot time going back fix word im texting buy another razor good one aint bad

2. nokia fine needed unlocked phone phone supplier unlocked ive purchased two unlocked phone amazon problem using network sim cardi cant remember exact wording message displayed looked internet found message meant phone still locked also found unlock code offered online disclaimer code guaranteed work decided return phone

3. moment wherein suddenlly stop middle talkin turn automatically communication reason one constantly interrupted apparent reason

4. phone wasnt coming battery wasnt working phone restarts time

5. bulky stuff pocket bulldog protection

===== Rating: 3 =====

1. used temporary phone spare small phone small screen old android version putting sim hard likewise putting micro sd card cover doesnt seem close said sparetemporary phone adequate using tmobile u becomes spare phone nexus arrives wife get nexus becomes spare

2. lagged slow bad battery

3. bought mine february volume button longer work thing happened buddy phone head

4. mobile great work exc easy use problem shipping read description said phone come car kit navigation doesnt send new phone package without car kit navigation end little disappointment

5. like three star ok😊

===== Rating: 4 =====

1. love product received mine bit scratch back phone work perfectly

2. always used otterbox defender iphones sturdy last long rubber give time phone stay pristine

3. nice phone brilliant screen take great picturesupdated android

4. good know previous carrier like

5. buck shoot cant complain

===== Rating: 5 =====

1. simply put cant find better smart phone tiny sleek thin big screen real button cant stand touch screen microsoft exchange server mean receives email easily automatically need pull need erase week one define get work email let say get work email pm weekday youll actively pull genius

2. company customer satisfaction would gladly purchase additional item confidence even gave free technical advice

3. love device say blu home cant stay phablet amazing blu think next

4. second case purchased hate carrying purse wallet constantly shoving license credit card stretchy phone case wasnt secure decided try case love hold card take mirror would like hold card carry health insurance card case emergency maybe one card second one bought first one broke dont alarmed im sure fault dropped phone done million time exact angle impact caused door crack right hinge able use new one came bought new case wanted item secure cost case less pain suffering involved report lost credit card

5. got phone son replace htc g lte evo find much reliable att service area faster stronger sprint service previously used good purchase

8. 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 of bias **.A stratified train-test split was then performed using** train_test_split() from sklearn.model_selection with stratify=y argument to ensure that all the star ratings were **proportionally represented** in both training and testing sets. The data sets was split using **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 integrity

Code :

```
from sklearn.model_selection import train_test_split

# X = cleaned text, y = target (ratings)
X = filtered_df['clean_text']
y = filtered_df['Rating']

# Split 80-20
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

9. TEXT VECTORIZATION

Text Vectorization means converting text data (words or sentences) into numerical form so that machine learning models can understand and process it.
Since ML algorithms work with numbers — not words — we use vectorization to represent text as vectors (arrays of numbers).

TF–IDF VECTORIZATION(Term Frequency- Inverse Document Frequency)

TF-IDF is a text vectorization technique that transforms text into numerical features by measuring how important a word is in a document relative to the entire dataset.
It gives higher weight to words that are frequent in one document but rare across others.

TF (Term Frequency)
Meaning:
How often a word appears in a single document.
Purpose:
Measures how important a word is **within that document**.

IDF (Inverse Document Frequency)
Meaning:
How rare or unique a word is across all documents.
Purpose:
Reduces the importance of common words (like *"the"*, *"good"*) and increases the importance of rare words.

Formula / Equation:
For a word t in a document d within a set of documents D:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Where:

1. **Term Frequency (TF):**

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in } d}{\text{Total number of terms in } d}$$

2. **Inverse Document Frequency (IDF):**

$$\text{IDF}(t, D) = \log\left(\frac{N}{1 + n_t}\right)$$

   - N= Total number of documents
   - n_t= Number of documents containing the term t
   - Adding **1** prevents division by zero

Code :

from sklearn.feature_extraction.text import TfidfVectorizer

```
# Initialize vectorizer
tfidf = TfidfVectorizer(max_features=5000, ngram_range=(1,2))

# Fit on training data and transform both
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

print("TF-IDF train shape:", X_train_tfidf.shape)
print("TF-IDF test shape:", X_test_tfidf.shape)
```

output

```
: TF-IDF train shape: (34335, 5000)
TF-IDF test shape: (8584, 5000)
```