Automated Review Rating System

Intoduction:

In the digital marketplace, customer reviews significantly influence purchasing decisions. Accurately predicting the rating of a product based on its textual review has become an important task in sentiment analysis and natural language processing (NLP). This project explores a deep learning-based approach to automatically predict numerical review ratings using text data. Specifically, it compares the performance of a model with a Bidirectional LSTM (BiLSTM) with Bidirectional Gated Recurrent Unit(GRU) to evaluate whether contextual learning in both directions improves prediction accuracy. The models are trained and tested on preprocessed customer review data to investigate their effectiveness in learning meaningful patterns and sentiments from raw text.

This system automates the process of rating reviews by:

- Cleaning and preprocessing raw review text.
- Visualizing review patterns and distributions.
- Creating a balanced dataset for fair model training.
- Splitting the data appropriately for training and evaluation.
- Applying Stopword removal and Lemmatization for cleaned text
- Applying vectorization and padding for better prediction

The resulting dataset and preprocessing steps lay the groundwork for building accurate and unbiased models capable of assigning star ratings to reviews thereby enhancing user experience and platform credibility.

Environment Setup:

To ensure smooth execution of the project and compatibility across systems, the following environment setup is recommended:

Python Installation

- Use **Python 3.10+**
- **Recommended:** Install Python via **Anaconda Distribution**
  Anaconda comes preloaded with most data science libraries and Jupyter Notebook support.

Required Python Libraries

The following libraries are essential for this project:

- pandas
- numpy
- matplotlib
- seaborn
- scikit-learn
- spacy
- tensorflow

Install them using:

```
pip install pandas numpy matplotlib seaborn scikit-learn spacy tensorflow
```

Additional Setup

- Download the English NLP model for spaCy:

```
python -m spacy download en_core_web_sm
```

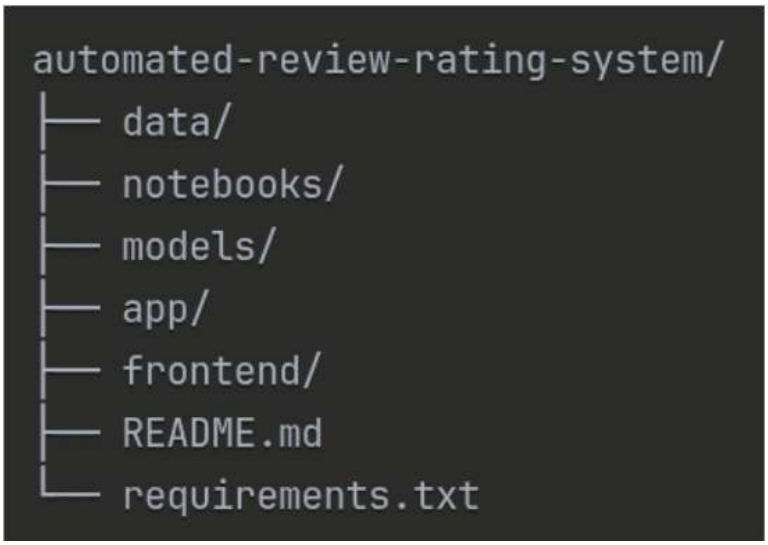
Github Project setup:

To manage version control and enable collaboration, a GitHub repository was created for the project.

Repository Name: https://github.com/hannafarsin/automatic-review-rating-system

Folder Structure:

The project is organized using a modular structure to ensure clarity, maintainability, and scalability:



Initial Commit:

- Added base folder structure.
- Created an initial `README.md` with a short description of the project.
- Included `requirements.txt` for setting up the Python environment.

1.Data Collection:

The dataset used for this project was sourced from **Kaggle**, specifically from the **Amazon Cell Phones and Accessories Reviews** dataset.

Dataset Overview:

- **Source**: Kaggle - Amazon Reviews
- **Dataset Link:** **https://github.com/hannafarsin/automatic-review-rating-system/blob/main/data/kaggle/reviews_output.csv**
- **Balanced Dataset Link:** **https://github.com/hannafarsin/automatic-review-rating-system/blob/main/data/kaggle/balanced_reviews_dataset3.csv**
- **Format**: CSV
- **Size**: 6,00,000+ reviews
- **Key Columns**:
  - `Review_text`: The main review written by the customer.
  - `Rating` (or `overall`): The star rating given by the user (ranging from 1 to 5).

Purpose of Data Collection:

The raw data serves as the foundation for training an automated system that can predict a review's rating purely from its text. Since the original dataset is

imbalanced (i.e., more 5-star reviews than 1-star), additional balancing steps were planned as part of the preprocessing.

Dataset Preview:

[2]:

	Text	Score
0	I have bought several of the Vitality canned d...	5
1	Product arrived labeled as Jumbo Salted Peanut...	1
2	This is a confection that has been around a fe...	4
3	If you are looking for the secret ingredient i...	2
4	Great taffy at a great price. There was a wid...	5
...	...	...
607443	cheap product its not a microwave safe	2
607444	it broked when the food is in on itnot one pla...	2
607445	full plate size is too small	2
607446	plates size is small bowl is too smallquality ...	2
607447	ok	2

607448 rows × 2 columns

2. Data Preprocessing:

2.1 Initial Data Inspection

Before applying text preprocessing techniques, the dataset was examined for structural issues:

Missing (null) values:

Columns like `reviewText`, `Rating` were checked.

df.isnull().any()

```
review_text     True
Rating         False
```

df.isnull().sum()

```
review_text    9
Rating         0
```

`Review_text` were removed.

```
df_no_mv=df.dropna(subset=[''reviewText'], inplace=True)
df_no_mv.isnull().sum()
```

```
review_text        0
Rating             0
```

Duplicate entries:

Exact duplicates were identified and dropped to avoid model bias.

```
df_no_mv.duplicated().sum()
```

```
0
```

Remove Conflicting Reviews(Same text,Different Ratings):

```
# Check how many reviews have conflicting ratings
conflict_counts = (df_no_mv.groupby('review_text')['Rating'].nunique()
    .reset_index().rename(columns={'Rating': 'unique_ratings'}))
# Get texts that have more than 1 unique rating
conflicting_reviews = conflict_counts[conflict_counts['unique_ratings'] > 1]['review_text']
# Filter original df for only conflicting reviews
conflict_df = df[df['review_text'].isin(conflicting_reviews)]
# Group by review_text to see all associated ratings
conflict_summary = conflict_df.groupby('review_text')['Rating'].unique().reset_index()
#  Show a few rows
conflict_summary.head(10)
```

[18]:

	review_text	Rating
0	#NAME?	[3, 2]
1	<a href="http://www.amazon.com/gp/product/B004...	[3, 4]
2	A Ming Dynasty (AD 1368 - 1644) creation belie...	[5, 4]
3	A few years ago, my cat was diagnosed with foo...	[5, 4]
4	After trying both the Chipotle as well as the ...	[5, 4]
5	Any of us who grew up during the U.S. space ag...	[4, 5]
6	Before the details: The Planters big nut bars...	[5, 4]
7	Bought this coffee in Europe when I was travel...	[4, 5]
8	Buffalo Bills Premium Snacks I real...	[3, 5]
9	Coffee drinks aren't carbonated, so it's odd t...	[2, 1]

2.2 Text Cleaning and Normalization:

In this step, the raw review text was cleaned and standardized to reduce noise and prepare it for further processing.

 # 1. Pre-split cleaning: punctuation, emojis, spaces

```
def basic_clean(text):
    text = str(text).lower()
    text = re.sub(r'[' + string.punctuation + ']', '', text)        # remove punctuation
    text = re.sub(r'[^\x00-\x7F]+', '', text)                       # remove emojis / non-ASCII
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)  # remove URLs
    text = re.sub(r'<.*?>', '', text)                               # remove HTML tags
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)                      # remove special characters
    text = re.sub(r'\s+', ' ', text).strip()                        # remove extra spaces, tabs, etc.
    return text
df_cleaned['review_text'] = df_cleaned['review_text'].apply(basic_clean)
```

Cleaning Steps Applied:

1. **Lowercasing**

```
text = str(text).lower()
```

- All text was converted to lowercase to maintain consistency.
- Example:
  `"This PHONE is Great!"` → `"this phone is great"`

2. **Removing URLs and HTML Tags**

```
text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)  # remove URLs
text = re.sub(r'<.*?>', '', text)                               # remove HTML tags
```

- Any embedded links or HTML elements were removed using regular expressions.
- Example:
  `"Check it out: <a href='...'>Link</a> "` → `"check it out"`

3. Removing Emojis and Special Characters

```
text = re.sub(r'[^\x00-\x7F]+', '', text)                       # remove emojis / non-ASCII
text = re.sub(r'[^a-zA-Z0-9\s]', '', text)                      # remove special characters
```

- Emojis, symbols, and non-ASCII characters were stripped from the text.
- Example:
  `"Great phone 😍✨!"` → `"great phone"`

4. Removing Punctuation and Numbers

```
text = re.sub(r'[' + string.punctuation + ']', '', text)          # remove punctuation
```

- Punctuation marks (e.g., `! ? , .`) and digits were removed to simplify the text.
- Example:
  `"Rated 10/10!!!"` → `"rated"`

5.Check Duplicate values after text cleaning :

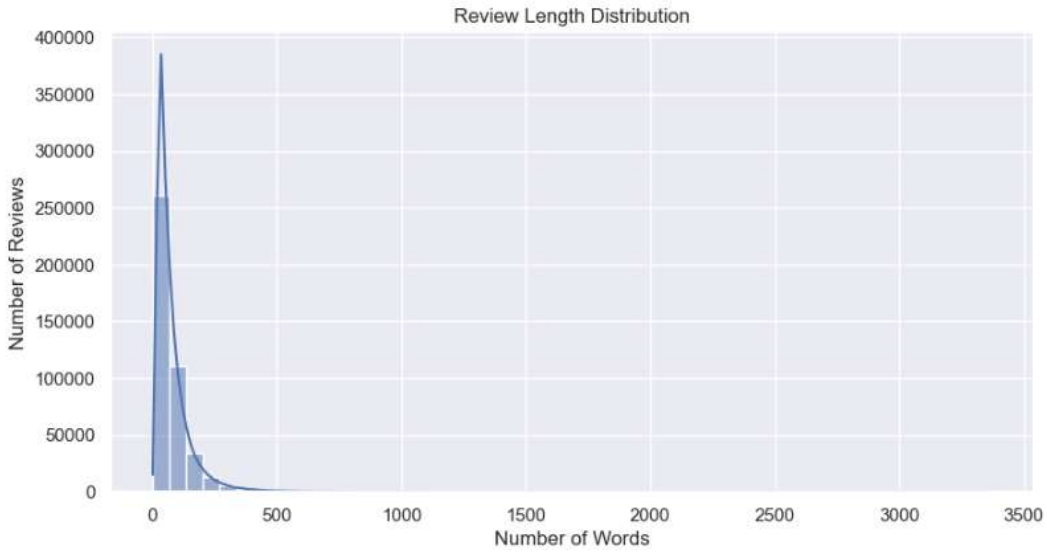
```
df_filtered.duplicated().sum()
```

```
503
```

Remove duplicate values

```
#drop duplicated now
df_filtered = df_filtered.drop_duplicates(subset='review_text', keep='first').reset_index(drop=True)
```

6. **Review Length Filtering**



- Very short reviews (fewer than 3 words) and excessively long ones (e.g., over 500 words) were removed.
- This helped eliminate outliers and non-informative entries.

Justification for Review Length Filtering

The histogram above illustrates the **distribution of review lengths** based on word count. It shows a **heavily right-skewed distribution**, where:

- The **majority of reviews contain fewer than 300 words**, with a sharp peak under 50 words.
- A **long tail of outliers** includes reviews with extremely high word counts, some exceeding **3500 words**.

To improve data quality and processing efficiency:

- **Very short reviews** (less than 3 words) were removed, as they offer limited semantic information.
- **Excessively long reviews** (e.g., over 500 words) were filtered out to eliminate outliers that may introduce noise or slow down training.

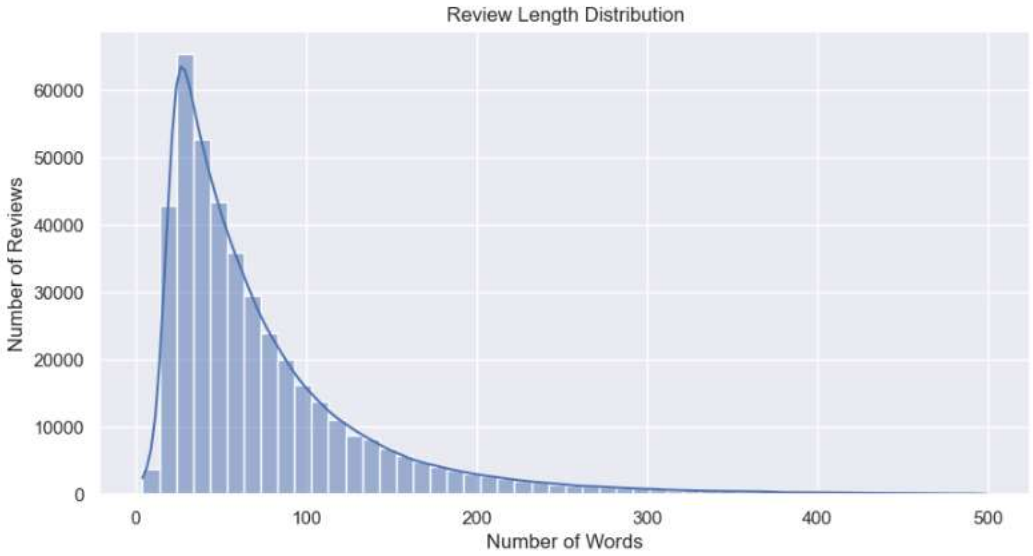
This step ensures a more **uniform distribution**, making the dataset more suitable for vectorization and model training.

After Filtering:

- Very short reviews (fewer than 3 words) and excessively long ones (e.g., over 350 words) were removed.
- This helped eliminate outliers and non-informative entries

```
df_filtered = df_cleaned[
    (df_cleaned['review_length'] > 3) &
    (df_cleaned['review_length'] <= 500)
]
print("Before filtering:", len(df_cleaned))
print("After filtering:", len(df_filtered))
```

```
Before filtering: 430456
After filtering: 427624
```



After applying the text cleaning and length-based filtering:

- Reviews with **fewer than 3 words** and those with **excessive lengths (over 500 words)** were removed.
- The remaining dataset shows a **more uniform and meaningful distribution** of review lengths, as visualized in the updated histogram above.
- Most reviews now fall between **10 to 150 words**, providing enough context while avoiding noise from overly short or verbose entries.

3. Creating a Balanced Dataset

Why Balance the Dataset?

The original dataset was **heavily imbalanced**, with a significantly higher number of 5-star reviews compared to lower ratings. This can lead to serious problems during model training, such as:

- The model becomes **biased toward majority classes** (e.g., 5-star).
- **Poor generalization** on minority classes like 1- or 2-star reviews.
- **Decreased accuracy and fairness**, especially in real-world scenarios where predicting low ratings is crucial for quality control.

Rating Distribution across 1 through 5 stars are:

```
df_filtered['Rating'].value_counts()
```

```
Rating
5    249661
4     55649
3     50567
1     36045
2     35170
```

As shown, over **60% of the reviews** is 5 stars, while the 1- and 2-star reviews together make up **less than 10%**. This imbalance would cause the model to **rarely predict low ratings**, reducing its usefulness.

Balancing Strategy:

To address this, the dataset was balanced by:

- Sampling **2,0000 reviews per class (1–5 stars)**.
- This resulted in a **balanced dataset of 100,000 reviews** with equal representation from each class.

```
#balanced dataset distribution strategy
df_balanced = pd.concat([
    df_filtered[df_filtered['Rating'] == 1].sample(20000, random_state=42),
    df_filtered[df_filtered['Rating'] == 2].sample(20000, random_state=42),
    df_filtered[df_filtered['Rating'] == 3].sample(20000, random_state=42),
    df_filtered[df_filtered['Rating'] == 4].sample(20000, random_state=42),
    df_filtered[df_filtered['Rating'] == 5].sample(20000, random_state=42),
])
#shuffle the data
 df_balanced = df_balanced.sample(frac=1, random_state=42).reset_index(drop=True)
#save the data
 df_balanced.to_csv("deep_balanced_dataset.csv", index=False)
```

Balanced Rating Distribution

The bar chart above confirms that the dataset has been successfully balanced. Each rating class (from 1 to 5 stars) now contains exactly **2,0000 reviews**, ensuring equal representation across all categories.

This balanced distribution addresses the skew present in the original dataset and helps:

- Prevent model bias toward majority classes
- Improve accuracy for underrepresented ratings
- Ensure fair and consistent performance during training and evaluation

This clean and uniform dataset is now ready for final preprocessing and modeling steps.

Display the statistics of the word count after balancing:

```
df['word_count'] = df['review_text'].apply(lambda x: len(str(x).split()))
#display the statistics of the word count
min_word_counts = df.groupby('Rating')['word_count'].describe()
print(min_word_counts)
```

```
        count      mean        std  min   25%   50%    75%    max
Rating
1       20000.0  77.59260  61.717615  6.0  36.0  60.0   97.0  499.0
2       20000.0  78.54145  72.645709  4.0  30.0  56.0  101.0  500.0
3       20000.0  88.69250  79.340089  4.0  34.0  61.0  114.0  500.0
4       20000.0  86.92260  71.600874  5.0  37.0  64.0  112.0  499.0
5       20000.0  70.49775  60.127029  7.0  31.0  51.0   88.0  500.0
```

Display Sample Review per Rating

```
#display 3 full sample reviews per rating
for rating in sorted(df['Rating'].unique()):
    print(f"\n ☆ Rating {rating}")
    reviews = df[df['Rating'] == rating].tail(3)
    for _, row in reviews.iterrows():
        print(f"- {row['review_text']}")
```

```
 ☆ Rating 1
- these are really poor skinny bully sticks i wrote a different review but it is n
ot appearing i was able to cut these w household scissors and the dog finished it
in 5 min i was very happy w previous orders of 6 sticks and opted for these as the
y were 12


- this product has no flavor besides the flavor of coffee to it no chocolate glaze
d donut no mocha nut fudge just regular coffee i was tricked by the advertising of
this product and hope to save anyone else from this level of disappointment


- this was received in pieces and my daughter said the gingerbread tasted like dog
biscuits on the positive side amazon fully refunded my money


 ☆ Rating 2
- will not work for 5 c i typed in 5c cases and this one came up i ordered it and
received it but its not a 5c case


- i was really hoping this grain free canned cat food will work for my 3 cats look
s like it has good ingredients but none of my 3 cats will eat it in fact one of th
```

em will smell it and start doing the scratching thing like he smells poop another weird thing is the whole can is kind of soft and meshed like baby food but there are some whole green peas in it not sure if that is by design or some green peas escape the meshing

- i like that you should be able to charge two iphones at the same time however it gets extremely hot and will not charge both but it works on at least one

☆ Rating 3

- this cereal is great with or without milk and it is easy to finish a box quickly amazon ships the cereal quickly and the cereal is always fresh and months away from the expiration datewhile i love this cereal the recent decision to reduce the number of boxes to 3 without adjusting the price appropriately does not differentiate this package from buying boxes in the local grocery i wish amazon would reconsider selling this as a pack of 4

- i purchased the gs3 cpen to replace my clunky mediadevil magicwand stylus the magicwand was ok but the tip was pretty big and not very precise but it worked regardless of the device i used it on i guess i just didnt do my research but i was under the impression that replacement for using fingers on your tablet or smartphone touchscreen meant i could use it on whatever touchscreen device i had but apparently that isnt the case when i first opened the box i tried it out on my new galaxy s 3 and it worked pretty good i have a zagg invisibleshield installed so the surface might not be what the pen was intended to be used on because it kinda skips around the surface so fine detail things are kinda hard to do i thought oh well it works decent enough i mainly want to use it for my tablet anyway so i pull out my tablet acer iconia a500 and try it on that and it doesnt want to work at all i tried it a dozen different times ways and it just doesnt work so i ask my wife to hand me her galaxy nexus and it doesnt work on that either same thing nothing i do works so at this point im pretty disappointed at the time of purchase it was almost 30 so i was expecting to get something that i could replace my other stylus with and then some sadly i think im going to just return it it just doesnt work that great if you have a screen protector installed and it didnt work on the other devices i had beautiful looking feeling stylus though

- the flavors are nice and that is the only reason i gave it 3 stars the dehydrated vegetables are tough and chewy even after adding the water and remaining covered the reequisite period of time i wont buy it again

☆ Rating 4

- these are a tasty bite indeedthe ingredients list is short and simple unlike a lot of other shelf stable processed foods there isnt a single ingredient in this that you couldnt buy from the organic foods marketif you are looking for anything close to restaurant quality wellgo to a restaurant for something that can be dumped out of a shelf stable pouch and microwaved its more than palatableits nowhere near as chunky as it looks on the pouch but there are still plenty of veggie and paneer bits in thereone important thing to note is that its very high in sodium so its best eaten over plain white rice at least i think its delicious that way

- i only gave these a 4star because i was disappointed in their not being saltier as you would expect when seeing seasalt on the label they just taste like the aver

age roasted salted almond they were very fresh and natural tasting however but i w
as craving an edgier salty flavor and i wont eat chips to get that this is a much
better alternative to chips for a treat as its high protein with a great helping o
f fiber almonds are one of the healthiest nuts to eat especially raw for their hig
h fiber and i believe b vitamin contentback to nature is one of the better and muc
h fresher brands so stock up while they are on special

- comes in a boxwas not sure what happened to the card listed in the item descript
iongot this for fathers day he really loved the pretzels and his girlfriend loved
the coffee turned out to be a good gift for his family

☆ Rating 5

- i have been experimenting with baking my own bread and have come to find out it
is as much an art as a science i have gotten some starters from acquaintances but
did not like the taste i found this one online and decided to give it a try it wor
ked great and tasted great just like the package said i added water and flour and
it was ready to use in a week i have the back up in the freezer but am hoping to m
ake this one last

- arrived exactly as promised and with vodka produces excellent vanilla extract at
home for cheap makes great gifts add bean to a bottle with vodka

- nonge ge stands for genetic engineering and is also referred to as gmo genetical
ly modified organism genetic engineering of food products uses gene splicing to ov
ercome natural boundaries that would otherwise prohibit the creation of a product
genes from different species can be combined to create a hardier product resistant
to certain exogenous factors as an example you might be able to add fish genes to
corn to make it resistant to pesticides although there may be instant gratificatio
n from such benefits as better taste longer shelf life and greater yield serious c
oncerns exist about the longterm ramifications of using these types of products an
d ingredients until more research backing the safety of ge products is available w
ell continue to make every effort to use only nonge ingredients and products

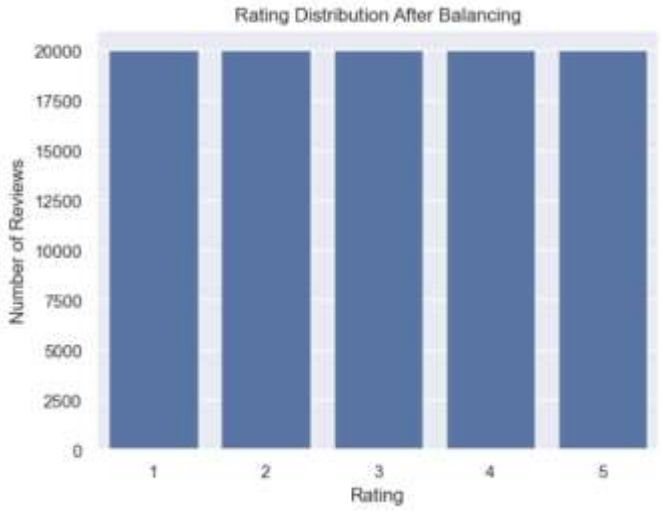
4. Data Visualization

To better understand the characteristics of the reviews and guide preprocessing decisions, several visualizations were created. These plots offer insights into rating distribution, review length, and review content.

4.1 Review Rating Distribution (Before Balancing)

Insight: The original dataset was highly imbalanced, with over 2,50,000 reviews rated 5 stars, while 1- and 2-star reviews were much less frequent. This confirmed the need for dataset balancing before training.

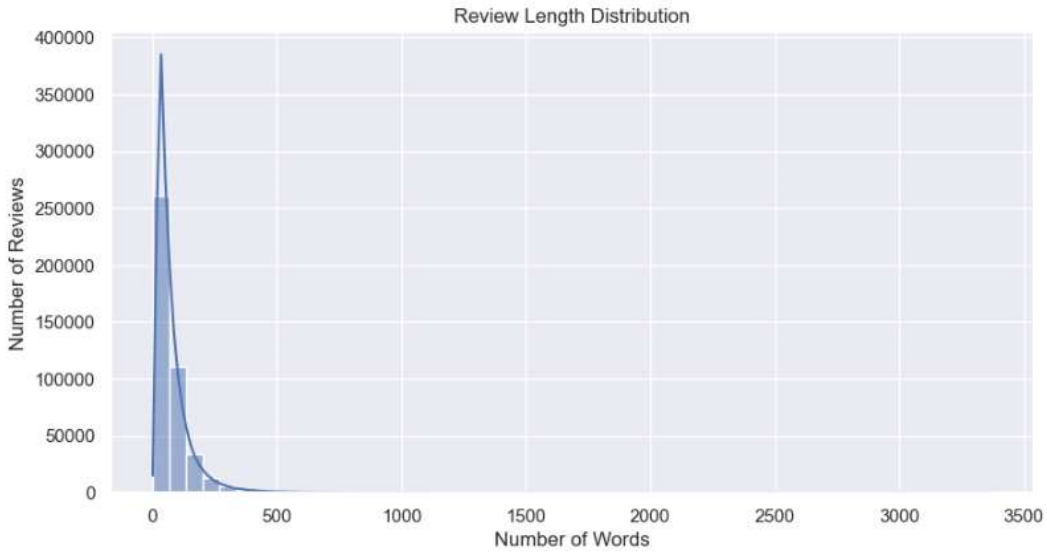
4.2 Review Rating Distribution (After Balancing)



Insight:
The distribution is now perfectly uniform, with **2,0000 reviews for each rating class (1 to 5 stars)**. This ensures that the machine learning model will be trained on an **equal representation of all sentiment levels**, which helps:

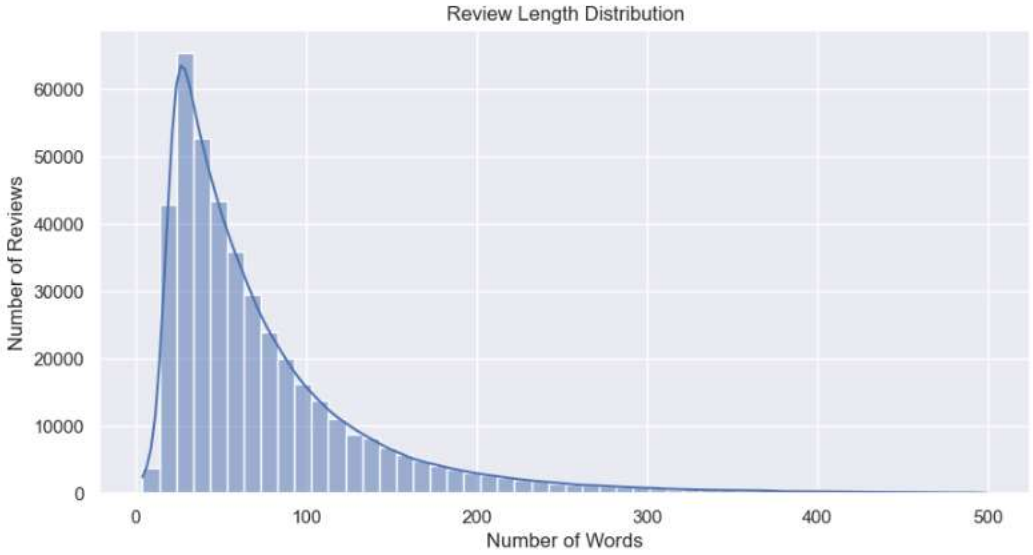
- Prevent overfitting to dominant classes (like 5-star reviews),
- Improve prediction accuracy for underrepresented ratings (1-star and 2-star),
- Enhance the **fairness and robustness** of the rating prediction system.

4.3 Review Length Distribution (Before Filtering)



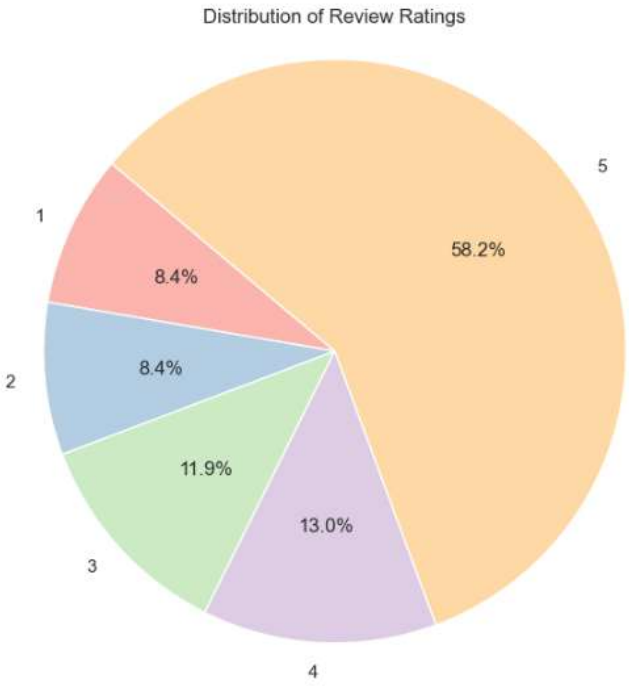
Insight: Most reviews are under 100 words, but some extend beyond 3,500 words. The long tail justified filtering out extremely long reviews as outliers.

4.3 Review Length Distribution (After Filtering)



Insight: After filtering, the majority of reviews fall between 10 and 150 words, ensuring text inputs are neither too sparse nor too verbose.

4.5 Review Rating Distribution Using Piechart (Before Balancing)

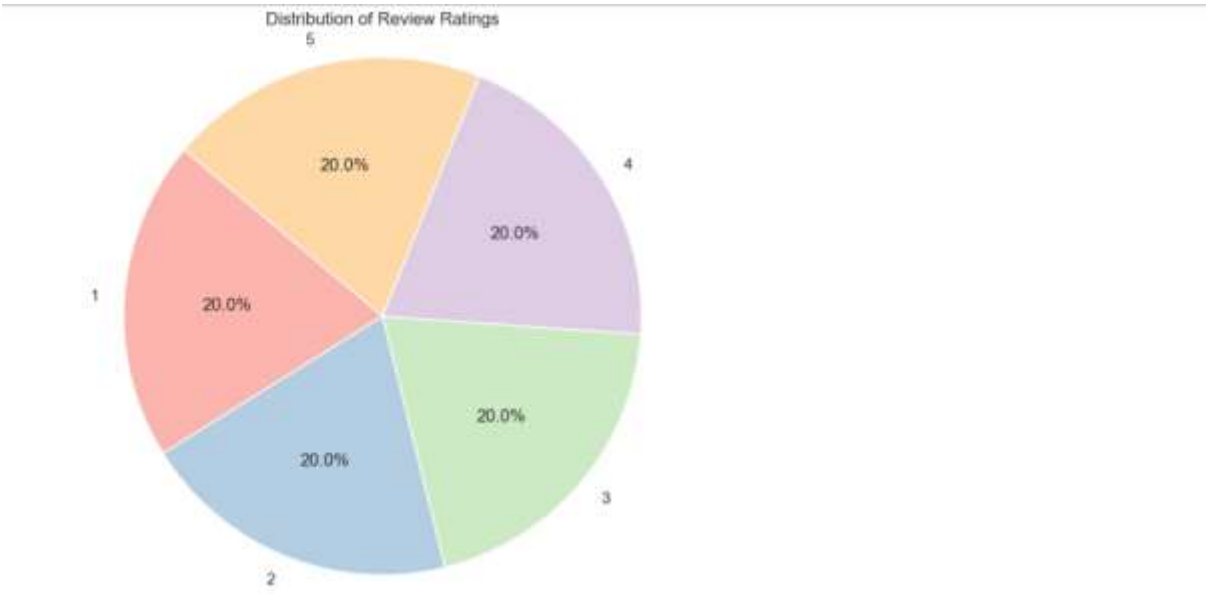


Insight:
The pie chart highlights a **highly imbalanced dataset**, with the majority of reviews skewed toward higher ratings:

- **5-star reviews** dominate at **58.2%** of the total.
- **4-star**: 13.0%
- **3-star**: 11.9%
- **2-star**: 8.4%
- **1-star**: 8.4%

This imbalance indicates that without correction, the model would likely be biased toward predicting 5-star reviews more frequently undermining performance on lower-rated reviews. Such class imbalance can lead to poor generalization and misclassification of negative feedback, which is often the most important to detect in real-world applications.

4.2 Review Rating Distribution Using Piechart (After Balancing)



Insight:
The pie chart shows that the dataset has been successfully balanced, with **each rating class (1 to 5 stars)** now contributing **exactly 20%** of the total reviews. This uniform distribution ensures:

- Equal learning opportunity for each class
- Fairer model training with **no class dominating the prediction output**
- Better performance on previously underrepresented ratings (especially 1-star and 2-star)

Balancing the dataset at this stage helps reduce bias, improve classification accuracy, and support robust model generalization across all rating levels

5. Train-Test Split

To evaluate the model's performance fairly and prevent overfitting, the cleaned and balanced dataset was split into training and testing subsets.

```
#  Split training and testing data
X_train, X_test, y_train, y_test = train_test_split(df['reviewText'],df['Rating'], test_size=0.2, random_state=42,  shuffle=True ,stratify=df['Rating'] )
```

Split Strategy:

- **Method**: Stratified Train-Test Split
- **Tool**: `train_test_split()` from Scikit-learn
- **Split Ratio**:
  - **80% Training Set**
  - **20% Testing Set**
- **Stratification**: Ensured the proportion of each rating class (1 to 5 stars) remains equal in both training and test sets.

Why Stratification?

Stratified sampling was used to **maintain class balance** in both the training and test sets. Without it, the test set might become imbalanced—even if the original dataset was balanced.

Resulting Dataset Sizes:

Subset		Number of Reviews
Training Set	-	8,000
Testing Set	-	2,000

5.1Post-Split Preprocessing & Vectorization

After splitting the dataset into training and testing sets, additional natural language processing steps were applied to prepare the review text for machine learning.

Stopword Removal & Lemmatization

To improve the semantic quality of the review text, two critical NLP operations were applied:

```
def spacy_preprocess_pipe(texts):
    processed = []
    for doc in nlp.pipe(texts, batch_size=1000, disable=["ner", "parser"]):  # disable unneeded parts
        tokens = [
            token.lemma_
            for token in doc
            if not token.is_stop and not token.is_space and token.is_alpha
        ]
        processed.append(" ".join(tokens))
    return processed
```

```
#print the list of stopwords

print(sorted(nlp.Defaults.stop_words))

# Print total number of unique stopwords in spaCy's English model

print("Number of spaCy stopwords:", len(nlp.Defaults.stop_words))
```

```
["'d", "'ll", "'m", "'re", "'s", "'ve", 'a', 'about', 'above', 'across', 'after',
'afterwards', 'again', 'against', 'all', 'almost', 'alone', 'along', 'already', 'a
lso', 'although', 'always', 'am', 'among', 'amongst', 'amount', 'an', 'and', 'anot
her', 'any', 'anyhow', 'anyone', 'anything', 'anyway', 'anywhere', 'are', 'around'
```

```
, 'as', 'at', 'back', 'be', 'became', 'because', 'become', 'becomes', 'becoming',
'been', 'before', 'beforehand', 'behind', 'being', 'below', 'beside', 'besides', '
between', 'beyond', 'both', 'bottom', 'but', 'by', 'ca', 'call', 'can', 'cannot',
'could', 'did', 'do', 'does', 'doing', 'done', 'down', 'due', 'during', 'each', 'e
ight', 'either', 'eleven', 'else', 'elsewhere', 'empty', 'enough', 'even', 'ever',
'every', 'everyone', 'everything', 'everywhere', 'except', 'few', 'fifteen', 'fift
y', 'first', 'five', 'for', 'former', 'formerly', 'forty', 'four', 'from', 'front'
, 'full', 'further', 'get', 'give', 'go', 'had', 'has', 'have', 'he', 'hence', 'he
r', 'here', 'hereafter', 'hereby', 'herein', 'hereupon', 'hers', 'herself', 'him',
'himself', 'his', 'how', 'however', 'hundred', 'i', 'if', 'in', 'indeed', 'into',
'is', 'it', 'its', 'itself', 'just', 'keep', 'last', 'latter', 'latterly', 'least'
, 'less', 'made', 'make', 'many', 'may', 'me', 'meanwhile', 'might', 'mine', 'more
', 'moreover', 'most', 'mostly', 'move', 'much', 'must', 'my', 'myself', "n't", 'n
ame', 'namely', 'neither', 'never', 'nevertheless', 'next', 'nine', 'no', 'nobody'
, 'none', 'noone', 'nor', 'not', 'nothing', 'now', 'nowhere', 'n‘t', 'n’t', 'of',
'off', 'often', 'on', 'once', 'one', 'only', 'onto', 'or', 'other', 'others', 'oth
erwise', 'our', 'ours', 'ourselves', 'out', 'over', 'own', 'part', 'per', 'perhaps
', 'please', 'put', 'quite', 'rather', 're', 'really', 'regarding', 'same', 'say',
'see', 'seem', 'seemed', 'seeming', 'seems', 'serious', 'several', 'she', 'should'
, 'show', 'side', 'since', 'six', 'sixty', 'so', 'some', 'somehow', 'someone', 'so
mething', 'sometime', 'sometimes', 'somewhere', 'still', 'such', 'take', 'ten', 't
han', 'that', 'the', 'their', 'them', 'themselves', 'then', 'thence', 'there', 'th
ereafter', 'thereby', 'therefore', 'therein', 'thereupon', 'these', 'they', 'third
', 'this', 'those', 'though', 'three', 'through', 'throughout', 'thru', 'thus', 't
o', 'together', 'too', 'top', 'toward', 'towards', 'twelve', 'twenty', 'two', 'und
er', 'unless', 'until', 'up', 'upon', 'us', 'used', 'using', 'various', 'very', 'v
ia', 'was', 'we', 'well', 'were', 'what', 'whatever', 'when', 'whence', 'whenever'
, 'where', 'whereafter', 'whereas', 'whereby', 'wherein', 'whereupon', 'wherever',
'whether', 'which', 'while', 'whither', 'who', 'whoever', 'whole', 'whom', 'whose'
, 'why', 'will', 'with', 'within', 'without', 'would', 'yet', 'you', 'your', 'your
s', 'yourself', 'yourselves', '‘d', '‘ll', '‘m', '‘re', '‘s', '‘ve', '’d', '’ll',
'’m', '’re', '’s', '’ve']

Number of spaCy stopwords: 326
```

X_train = pd.Series(spacy_preprocess_pipe(X_train))

X_test = pd.Series(spacy_preprocess_pipe(X_test))

Stopword Removal:

- **Stopwords** are common words that occur frequently in language (e.g., “the”, “is”, “and”) but carry minimal semantic meaning.
- Removing them helps reduce noise and dimensionality in the vectorized feature space.
- In this project, stopwords were removed using:
  - **spaCy’s default English stopword set**, which offers linguistic richness and flexibility.

Lemmatization:

- **Lemmatization** reduces each word to its base or **dictionary form** (called a "lemma").
- Unlike stemming (which can be more aggressive and less accurate), lemmatization ensures that words like:
  - `"running"`, `"ran"` → `"run"`
  - `"better"` → `"good"`

- This step helps group similar words and improves the model's ability to generalize across different grammatical forms.
- Lemmatization was performed using **spaCy**, a fast and powerful NLP library in Python.

About spaCy:

- SpaCy provides **tokenization**, **POS tagging**, **lemmatization**, **NER**, and **syntactic parsing**.
- The English language model used here is:

  ```
  python -m spacy download en_core_web_sm
  ```

  This model provides pre-trained weights for English NLP tasks, including lemmatization.

Note:

Both stopword removal and lemmatization were applied **after splitting** the data, and **separately on the training and testing sets**, to avoid **data leakage**. This ensures the model doesn't "peek" into the test set during training.

5.2 Tokenization

Tokenization is the foundational step in preparing text data for machine learning models. It involves breaking down raw text into smaller units called "tokens", which can be words, subwords, or even characters. This process converts human-readable text into a numerical format that computers can process, enabling further analysis and model training. Tokenization is crucial for tasks like sentiment analysis, text classification, and machine translation, as it helps models to better understand the semantic meaning and context of words within a sentence or document

```
from tensorflow.keras.preprocessing.text import Tokenizer

# Define tokenizer parameters
vocab_size = 10000
tokenizer = Tokenizer(num_words=vocab_size, oov_token="<OOV>")

# Fit tokenizer on training data
tokenizer.fit_on_texts(X_train)

# Convert text to sequences
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)
```

Text needs to be converted into a format that a deep learning model can understand (numbers). Tokenization is the process of breaking down raw text into smaller units called tokens (typically words or subwords). The `Tokenizer` class from Keras is used for this purpose.

- `texts_to_sequences` transforms each text in `x_test` into a sequence of integers. Each integer represents a specific token (word) based on the vocabulary created by the tokenizer during its initialization and fitting on training data

5.3 Padding

Padding is a technique used in sequence processing to standardize the length of tokenized text sequences. Since sentences and documents naturally vary in length, machine learning models often require uniform input dimensions. Padding addresses this by adding special tokens, typically zeros, to the beginning or end of shorter sequences until they reach a predetermined maximum length. This ensures compatibility with models expecting fixed-size inputs and allows for efficient batch processing during training, while truncation handles longer sequences by removing tokens from either the beginning or end.

```
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_len = 200

X_train_pad = pad_sequences(X_train_seq, maxlen=max_len, padding='post',
truncating='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len, padding='post',
truncating='post')
```

Neural networks often require input sequences to have a uniform length. Since sentences or text segments can vary in length, padding is applied to ensure that all sequences are of the same length by adding a predefined numeric value (usually 0) to shorter sequences.

- `pad_sequences` is a Keras function that pads or truncates sequences to a desired length (`maxlen`).

- `maxlen=max_len`: This parameter sets the maximum length for all sequences. Sequences shorter than `max_len` will be padded, while sequences longer than `max_len` will be truncated.
- `padding='post'`: This specifies that padding (adding zeros) should be done *after* the sequence.
- `truncating='post'`: This indicates that if a sequence is longer than `maxlen`, the truncation (removal of tokens) should happen *from the end* of the sequence.

Deep Learning ModelA