AUTOMATED REVIEW SYSTEM

1. Project Overview

An intelligent system that analyzes customer reviews and predicts corresponding star ratings. It uses natural language processing (NLP) to extract sentiment and key features from textual feedback. The model is trained on labeled review data to learn patterns between language and rating.

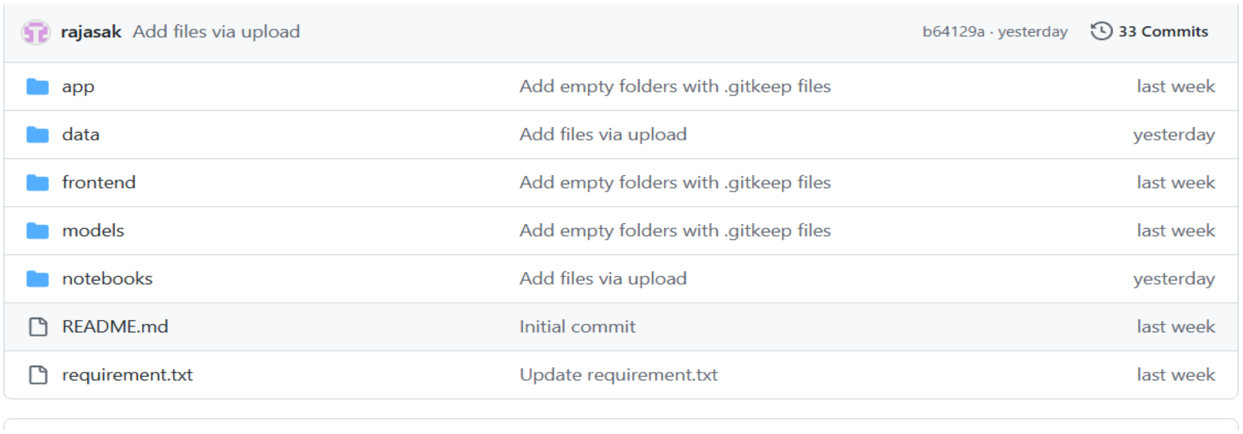
2. Environment Setup

- Installed Python 3.10+ with required libraries: pandas, numpy, matplotlib, seaborn, spacy, scikit-learn, beautifulsoup4, requests
- IDE used: VS Code
- Verified proper functioning of packages for data preprocessing, NLP, and visualization

3. GitHub Project Setup

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

Structure of directory



rajasak Add files via upload — b64129a · yesterday — **33 Commits**

app	Add empty folders with .gitkeep files	last week
data	Add files via upload	yesterday
frontend	Add empty folders with .gitkeep files	last week
models	Add empty folders with .gitkeep files	last week
notebooks	Add files via upload	yesterday
README.md	Initial commit	last week
requirement.txt	Update requirement.txt	last week

4. Data Collection

- Data was collected through web scrapping and from Kaggle
- Data collected from Kaggle had 49000 rows and it was dataset related to cloths
- Data collected from web scrapping had 1700 rows and it was dataset related to iphone 15
- Both the dataset were merged
- Dataset link= https://github.com/rajasak/automated-review-rating-system/blob/main/data/merged.csv
- Final dataset contains 2 column with Review and Rating
- 1 star=4196, 2 star= 3447, 3 star= 5695, 4 star=8316 , 5 star= 29294

4.1. Balanced dataset

- Balanced dataset was created with 2000 rows of each rating

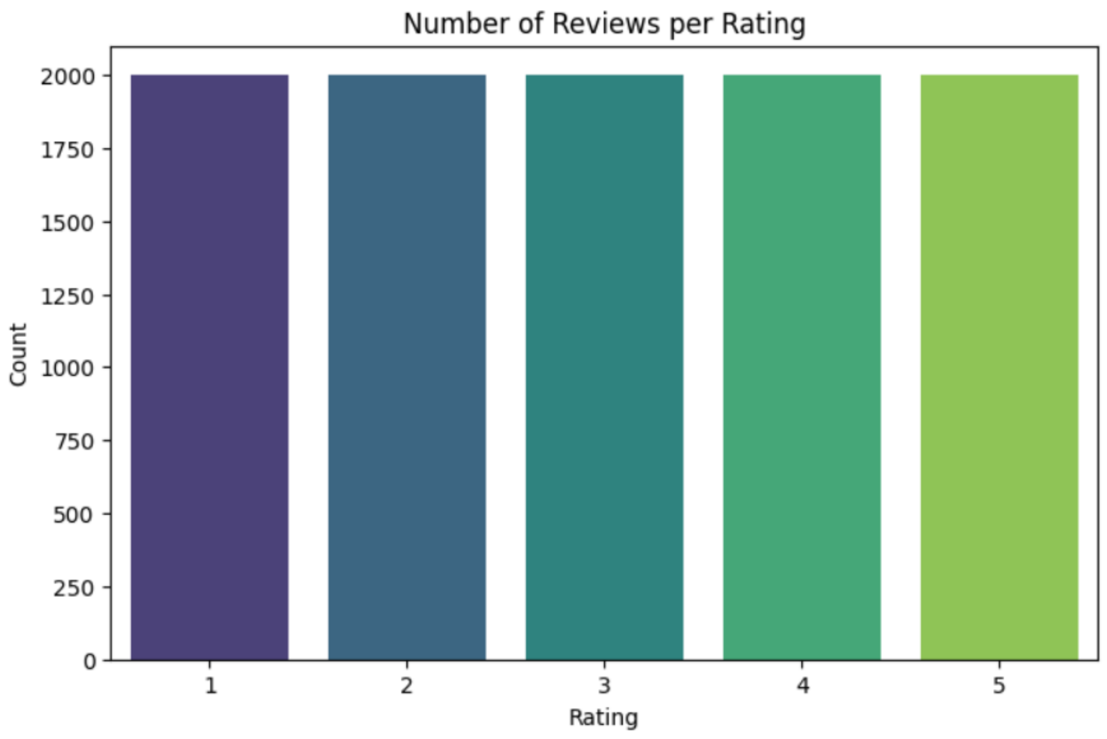


Fig count plot

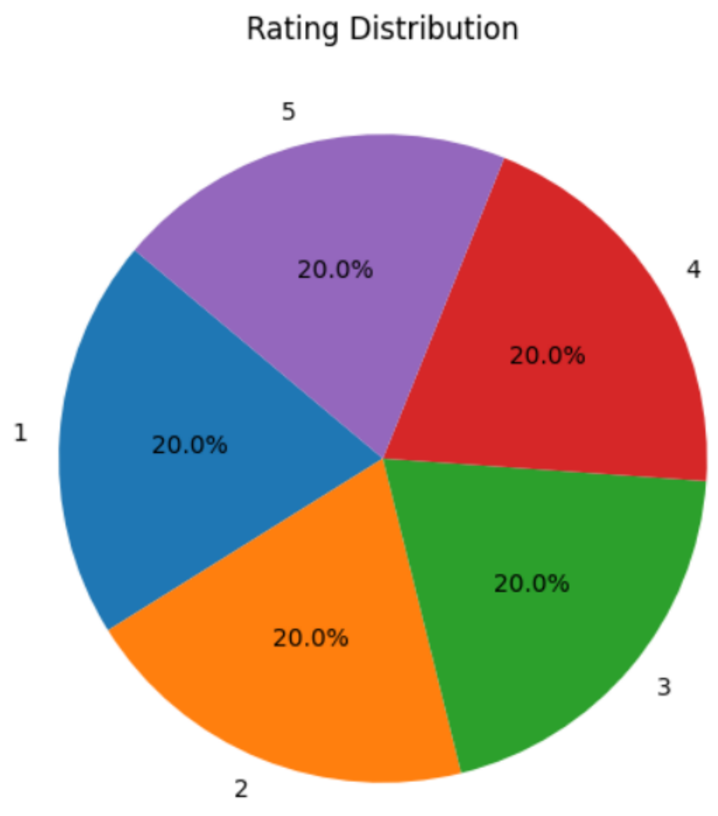


Fig pie chart

4.2. imbalanced dataset

- Imbalanced dataset was created with 1star=10%, 2 star=15%, 3 star=15%, 4 star=30%, 5 star=20%

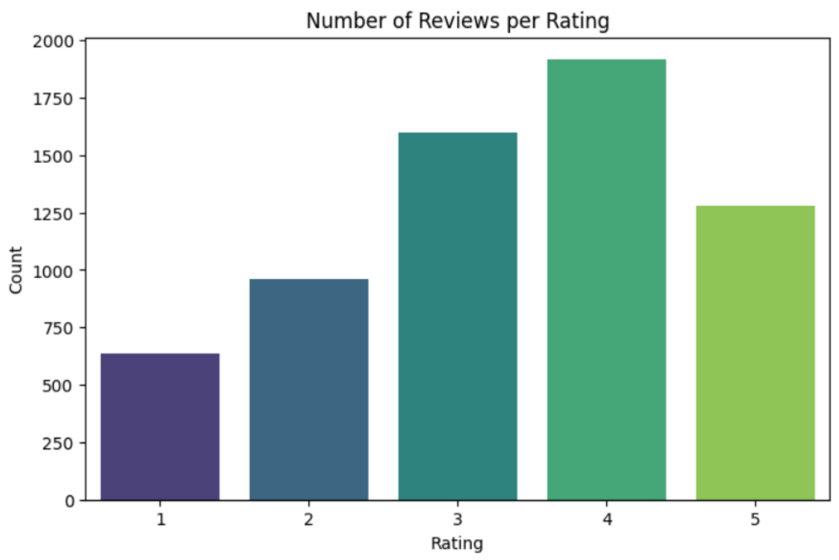


Fig count plot

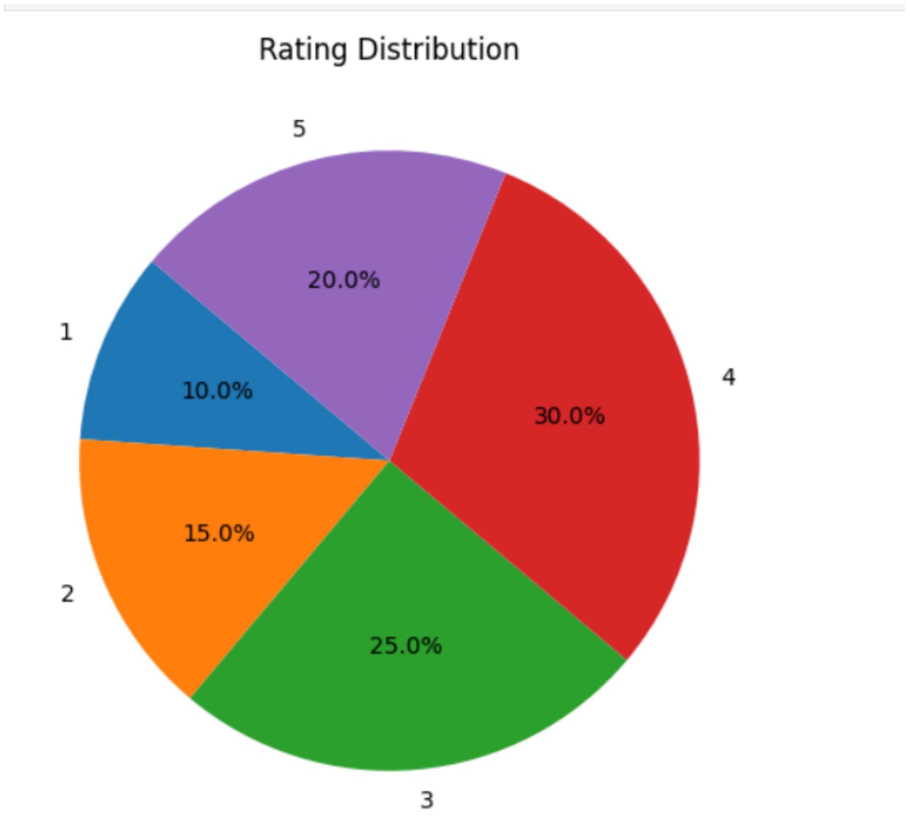


Fig pie chart

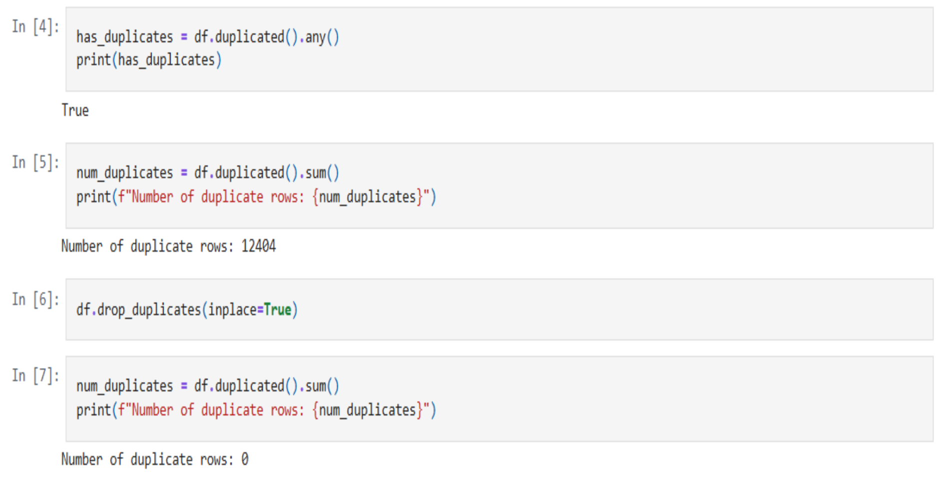
5. Data Preprocessing

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

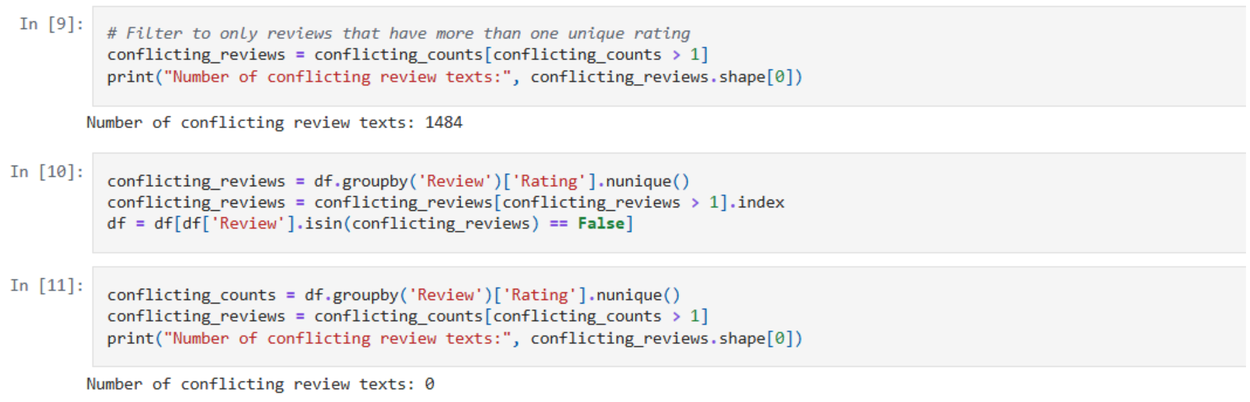


```
In [4]: has_duplicates = df.duplicated().any()
        print(has_duplicates)

        True

In [5]: num_duplicates = df.duplicated().sum()
        print(f"Number of duplicate rows: {num_duplicates}")

        Number of duplicate rows: 12404

In [6]: df.drop_duplicates(inplace=True)

In [7]: num_duplicates = df.duplicated().sum()
        print(f"Number of duplicate rows: {num_duplicates}")

        Number of duplicate rows: 0
```

5.2 Removing Conflicting Reviews

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

code:

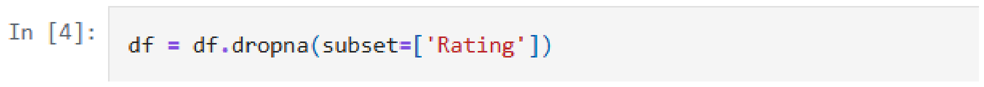


```
In [9]: # Filter to only reviews that have more than one unique rating
        conflicting_reviews = conflicting_counts[conflicting_counts > 1]
        print("Number of conflicting review texts:", conflicting_reviews.shape[0])

        Number of conflicting review texts: 1484

In [10]: conflicting_reviews = df.groupby('Review')['Rating'].nunique()
         conflicting_reviews = conflicting_reviews[conflicting_reviews > 1].index
         df = df[df['Review'].isin(conflicting_reviews) == False]

In [11]: conflicting_counts = df.groupby('Review')['Rating'].nunique()
         conflicting_reviews = conflicting_counts[conflicting_counts > 1]
         print("Number of conflicting review texts:", conflicting_reviews.shape[0])

         Number of conflicting review texts: 0
```

5.3 Handling Missing Values

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

Code:



```
In [4]:  df = df.dropna(subset=['Rating'])
```

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 Lowercasing Text

All review text was converted to lowercase to maintain uniformity. This helps prevent duplication of tokens like "Good" and "good" being treated as separate words.

5.6 Removing URLs

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

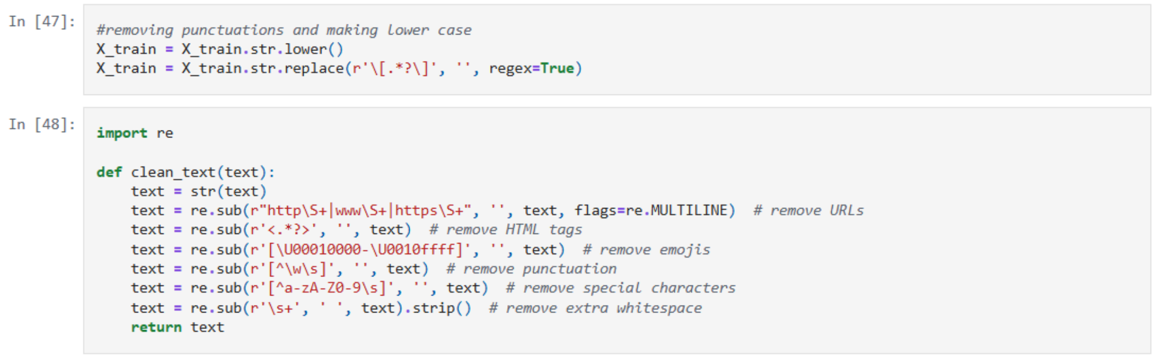
5.7 Removing Emojis and Special Characters

Emojis and special symbols may not contribute meaningful context for text analysis (unless specifically required). We remove these characters to focus on the core textual content.

5.8 Removing Punctuation

Punctuation marks and numbers can disrupt the natural language patterns of the text. By removing them, we simplify the tokenization process and prevent punctuation from being misinterpreted as separate tokens.

Code:

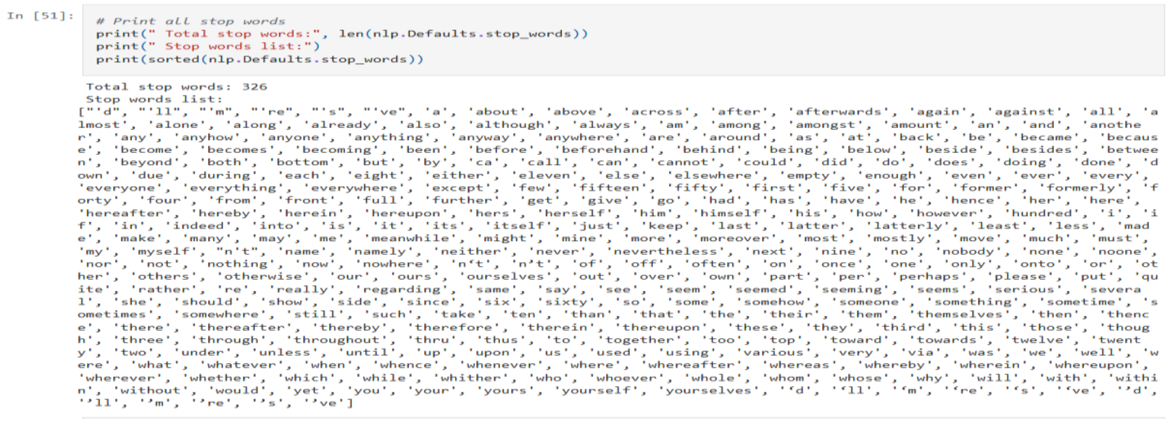


```
In [47]:
#removing punctuations and making lower case
X_train = X_train.str.lower()
X_train = X_train.str.replace(r'\[.*?\]', '', regex=True)
```

```
In [48]:
import re

def clean_text(text):
    text = str(text)
    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'[\U00010000-\U0010ffff]', '', text)  # remove emojis
    text = re.sub(r'[^\w\s]', '', text)  # remove punctuation
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)  # remove special characters
    text = re.sub(r'\s+', ' ', text).strip()  # remove extra whitespace
    return text
```

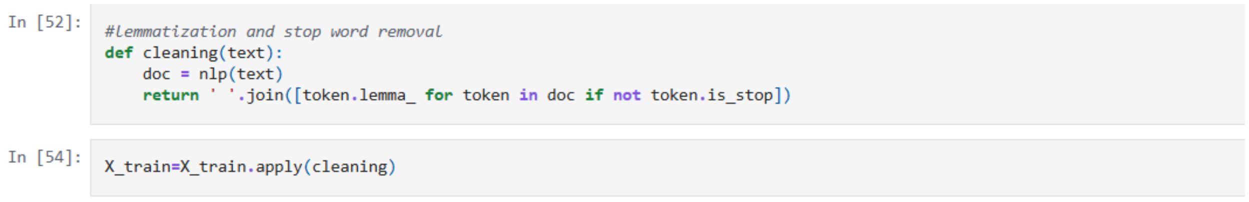
5.9 Stopwords Removal

Stopwords, such as "the", "is", and "and", are frequently occurring words that might not add significant semantic value. We use 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.



```
In [51]:
# Print all stop words
print(" Total stop words:", len(nlp.Defaults.stop_words))
print(" Stop words list:")
print(sorted(nlp.Defaults.stop_words))

 Total stop words: 326
 Stop words list:
["'d", "'ll", "'m", "'re", "'s", "'ve", 'a', 'about', 'above', 'across', 'after', 'afterwards', 'again', 'against', 'all', 'a
lmost', 'alone', 'along', 'already', 'also', 'although', 'always', 'am', 'among', 'amongst', 'amount', 'an', 'and', 'anothe
r', 'any', 'anyhow', 'anyone', 'anything', 'anyway', 'anywhere', 'are', 'around', 'as', 'at', 'back', 'be', 'became', 'becaus
e', 'become', 'becomes', 'becoming', 'been', 'before', 'beforehand', 'behind', 'being', 'below', 'beside', 'besides', 'betwee
n', 'beyond', 'both', 'bottom', 'but', 'by', 'ca', 'call', 'can', 'cannot', 'could', 'did', 'do', 'does', 'doing', 'done', 'd
own', 'due', 'during', 'each', 'eight', 'either', 'eleven', 'else', 'elsewhere', 'empty', 'enough', 'even', 'ever', 'every',
'everyone', 'everything', 'everywhere', 'except', 'few', 'fifteen', 'fifty', 'first', 'five', 'for', 'former', 'formerly', 'f
orty', 'four', 'from', 'front', 'full', 'further', 'get', 'give', 'go', 'had', 'has', 'have', 'he', 'hence', 'her', 'here',
'hereafter', 'hereby', 'herein', 'hereupon', 'hers', 'herself', 'him', 'himself', 'his', 'how', 'however', 'hundred', 'i', 'i
f', 'in', 'indeed', 'into', 'is', 'it', 'its', 'itself', 'just', 'keep', 'last', 'latter', 'latterly', 'least', 'less', 'mad
e', 'make', 'many', 'may', 'me', 'meanwhile', 'might', 'mine', 'more', 'moreover', 'most', 'mostly', 'move', 'much', 'must',
'my', 'myself', "n't", 'name', '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', 'ot
her', 'others', 'otherwise', 'our', 'ours', 'ourselves', 'out', 'over', 'own', 'part', 'per', 'perhaps', 'please', 'put', 'qu
ite', 'rather', 're', 'really', 'regarding', 'same', 'say', 'see', 'seem', 'seemed', 'seeming', 'seems', 'serious', 'severa
l', 'she', 'should', 'show', 'side', 'since', 'six', 'sixty', 'so', 'some', 'somehow', 'someone', 'something', 'sometime', 's
ometimes', 'somewhere', 'still', 'such', 'take', 'ten', 'than', 'that', 'the', 'their', 'them', 'themselves', 'then', 'thenc
e', 'there', 'thereafter', 'thereby', 'therefore', 'therein', 'thereupon', 'these', 'they', 'third', 'this', 'those', 'thoug
h', 'three', 'through', 'throughout', 'thru', 'thus', 'to', 'together', 'too', 'top', 'toward', 'towards', 'twelve', 'twent
y', 'two', 'under', 'unless', 'until', 'up', 'upon', 'us', 'used', 'using', 'various', 'very', 'via', 'was', 'we', 'well', 'w
ere', 'what', 'whatever', 'when', 'whence', 'whenever', 'where', 'whereafter', 'whereas', 'whereby', 'wherein', 'whereupon',
'wherever', 'whether', 'which', 'while', 'whither', 'who', 'whoever', 'whole', 'whom', 'whose', 'why', 'will', 'with', 'withi
n', 'without', 'would', 'yet', 'you', 'your', 'yours', 'yourself', 'yourselves', '‘d', '‘ll', '‘m', '‘re', '‘s', '‘ve', '’d',
'’ll', '’m', '’re', '’s', '’ve']
```

Code:



```
In [52]:
#lemmatization and stop word removal
def cleaning(text):
    doc = nlp(text)
    return ' '.join([token.lemma_ for token in doc if not token.is_stop])
```

```
In [54]:
X_train=X_train.apply(cleaning)
```

5.10 Lemmatization

Lemmatization is a text preprocessing technique that reduces words to their base or dictionary form, known as the **lemma**. For example, words like **"running"**, **"ran"**, and **"runs"** are all reduced to the base form **"run"**. Unlike stemming, lemmatization uses linguistic rules and vocabulary, ensuring that the resulting word is meaningful. This helps in grouping similar words and improves model performance by reducing the size of the vocabulary without losing context.

Why lemmatization is better than stemming

Lemmatization is often preferred over stemming because:

1. **Produces Real Words**:

   Lemmatization returns valid dictionary words (e.g., *"running"* → *"run"*), whereas stemming may return incomplete or non-existent words (e.g., *"running"* → *"runn"*).

2. **Context-Aware**:

   Lemmatization considers the context and part of speech (POS) of a word to find its correct root form.
   Stemming blindly trims word endings without understanding grammar.

3. **More Accurate and Clean**:

   Lemmatization provides cleaner and more accurate tokens, which improves model understanding and performance, especially in NLP tasks like sentiment analysis or classification.

5.11 Filtering by Word Count

Very short reviews might not contain enough context to be useful, and excessively long reviews could be outliers. We apply filtering to exclude:

- Reviews with fewer than 3 words.
- Reviews that exceed 100 words.
  This ensures that the dataset remains robust and relevant for model training.

6. Data visualization

Box plot

A **box plot** is a visual tool used to display the distribution of numerical data, showing the median, quartiles, and potential outliers. It helps quickly understand how data is spread and identify any extreme values. In this project, box plots were used to compare the word count distribution of reviews across different rating classes.

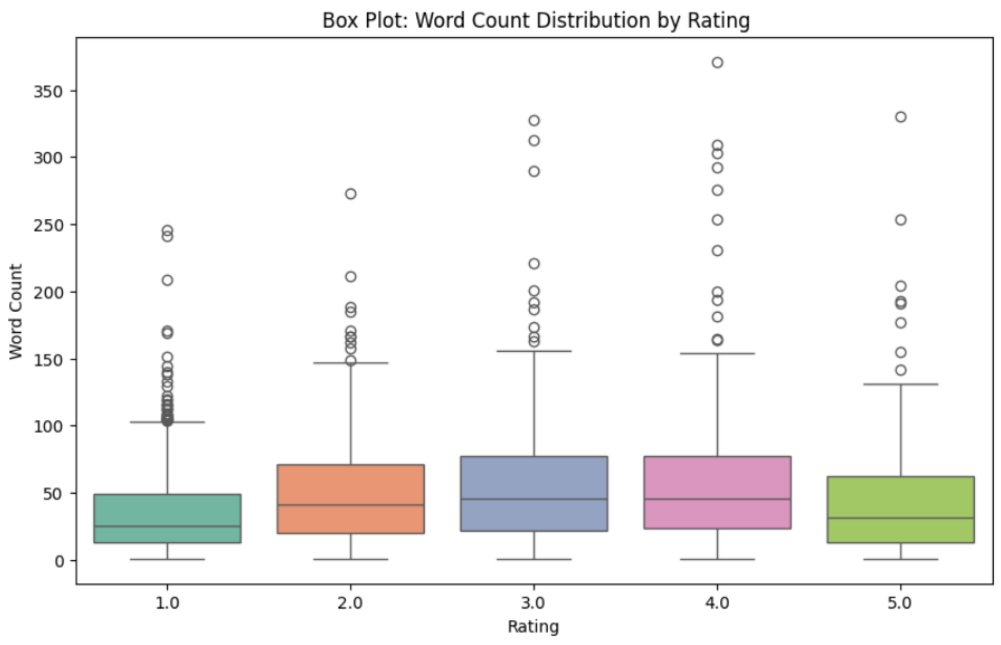


Fig box plot

Histogram

A **histogram** is a graphical representation that shows the distribution of a numeric variable by dividing the data into intervals (bins) and counting how many values fall into each bin. In this project, histograms were used to visualize the **word count distribution** of reviews, helping to identify common review lengths and spot patterns across different rating levels.

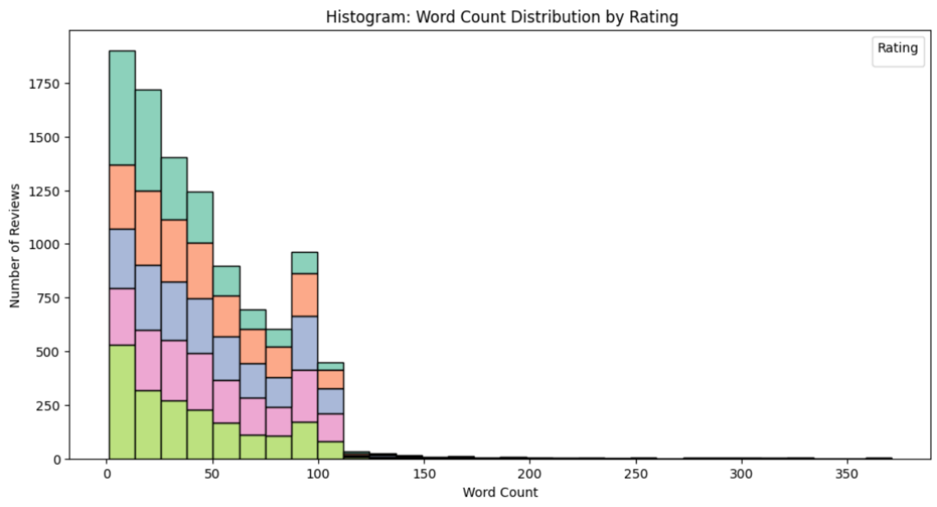


Fig histogram

Barplot

A **bar plot** is a chart that represents categorical data with rectangular bars, where the height of each bar indicates the value or frequency of that category. In this project, bar plots were used to show the number of reviews per rating class (1 to 5 stars)**,** helping to visualize class distribution and detect any imbalance in the dataset.

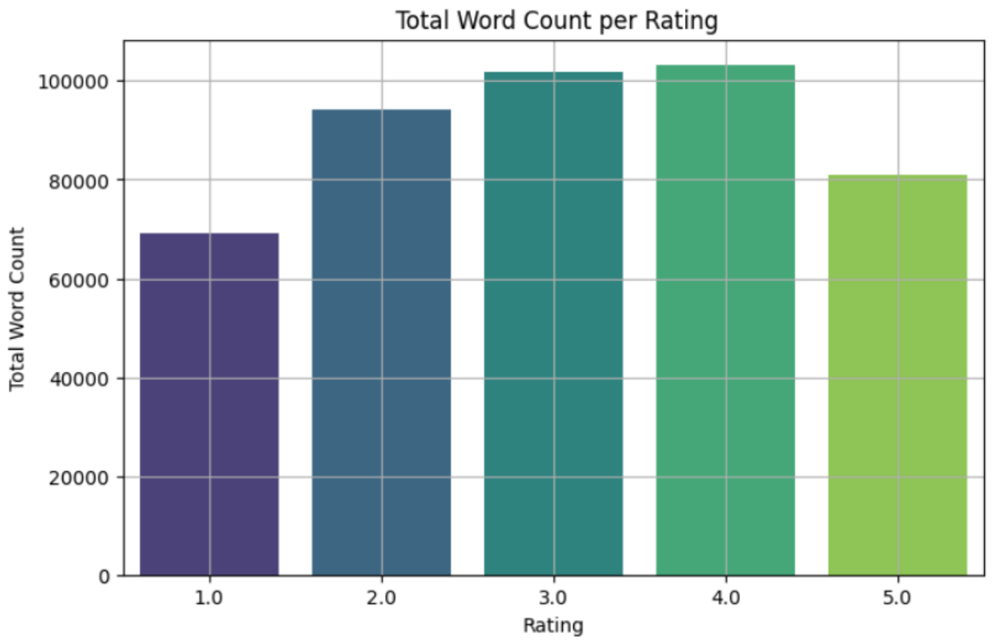
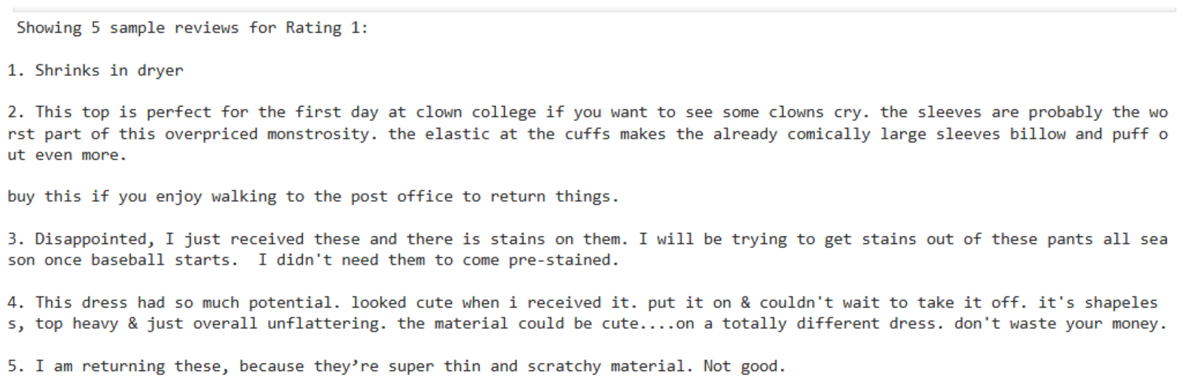


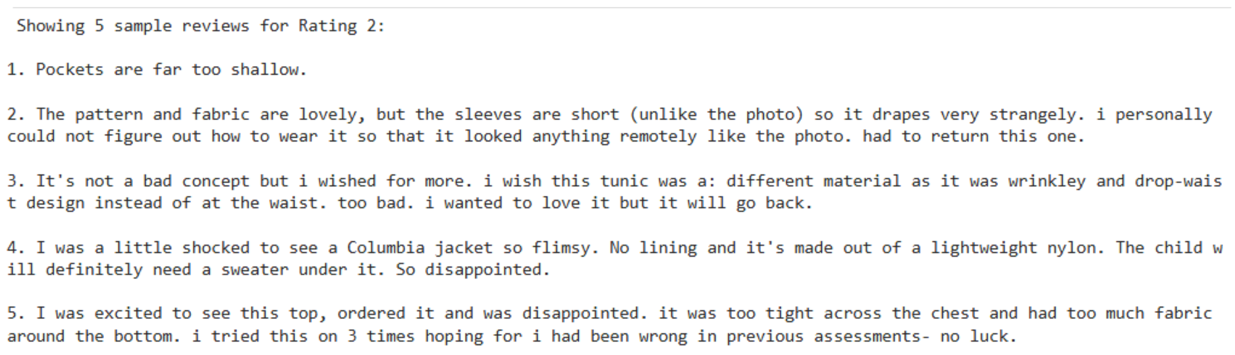
Fig bar plot

Samples of 1 star rating:



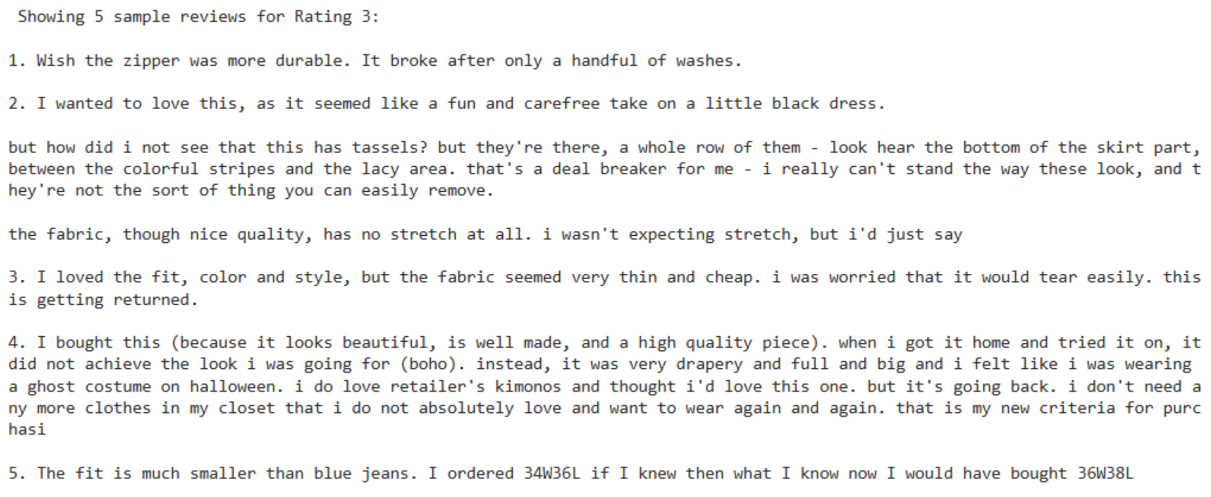
```
Showing 5 sample reviews for Rating 1:

1. Shrinks in dryer

2. This top is perfect for the first day at clown college if you want to see some clowns cry. the sleeves are probably the wo
rst part of this overpriced monstrosity. the elastic at the cuffs makes the already comically large sleeves billow and puff o
ut even more.

buy this if you enjoy walking to the post office to return things.

3. Disappointed, I just received these and there is stains on them. I will be trying to get stains out of these pants all sea
son once baseball starts.  I didn't need them to come pre-stained.

4. This dress had so much potential. looked cute when i received it. put it on & couldn't wait to take it off. it's shapeles
s, top heavy & just overall unflattering. the material could be cute....on a totally different dress. don't waste your money.

5. I am returning these, because they’re super thin and scratchy material. Not good.
```

Samples of 2 star rating



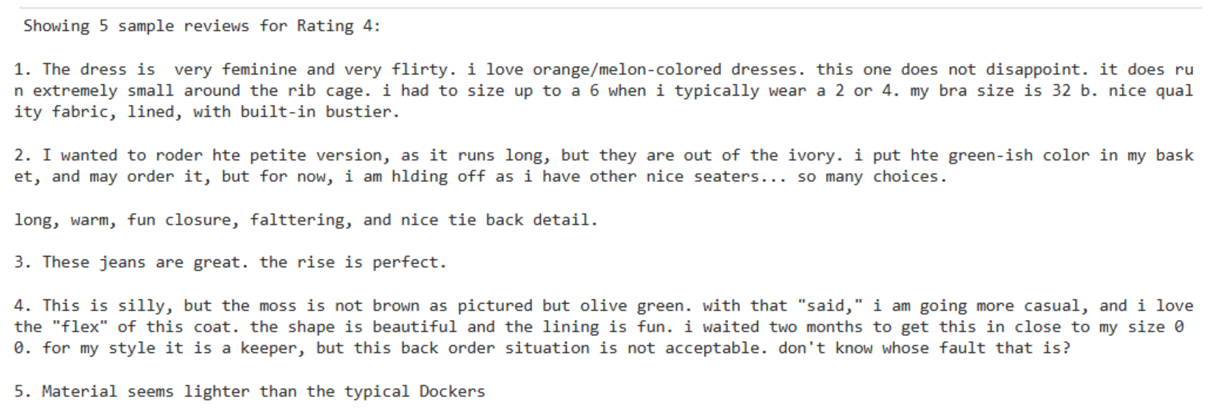
```
Showing 5 sample reviews for Rating 2:

1. Pockets are far too shallow.

2. The pattern and fabric are lovely, but the sleeves are short (unlike the photo) so it drapes very strangely. i personally
could not figure out how to wear it so that it looked anything remotely like the photo. had to return this one.

3. It's not a bad concept but i wished for more. i wish this tunic was a: different material as it was wrinkley and drop-wais
t design instead of at the waist. too bad. i wanted to love it but it will go back.

4. I was a little shocked to see a Columbia jacket so flimsy. No lining and it's made out of a lightweight nylon. The child w
ill definitely need a sweater under it. So disappointed.

5. I was excited to see this top, ordered it and was disappointed. it was too tight across the chest and had too much fabric
around the bottom. i tried this on 3 times hoping for i had been wrong in previous assessments- no luck.
```

Samples of 3 star rating



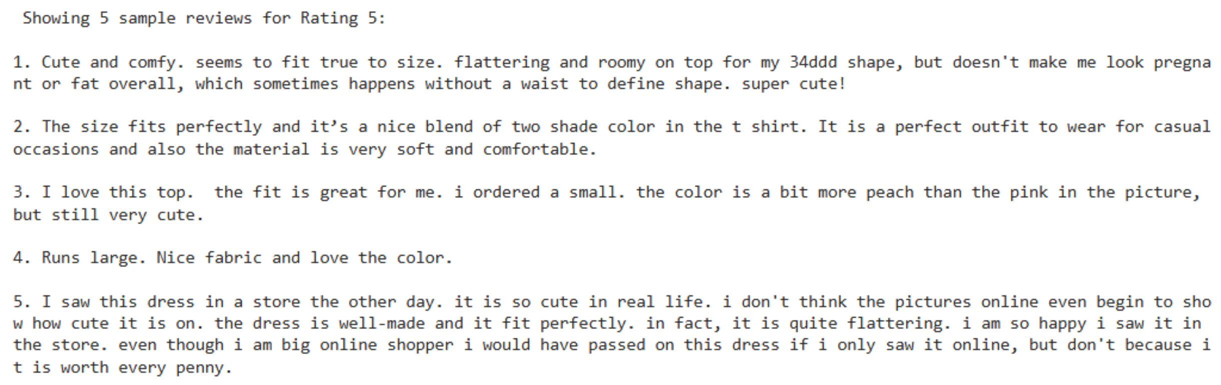
```
Showing 5 sample reviews for Rating 3:

1. Wish the zipper was more durable. It broke after only a handful of washes.

2. I wanted to love this, as it seemed like a fun and carefree take on a little black dress.

but how did i not see that this has tassels? but they're there, a whole row of them - look hear the bottom of the skirt part,
between the colorful stripes and the lacy area. that's a deal breaker for me - i really can't stand the way these look, and t
hey're not the sort of thing you can easily remove.

the fabric, though nice quality, has no stretch at all. i wasn't expecting stretch, but i'd just say

3. I loved the fit, color and style, but the fabric seemed very thin and cheap. i was worried that it would tear easily. this
is getting returned.

4. I bought this (because it looks beautiful, is well made, and a high quality piece). when i got it home and tried it on, it
did not achieve the look i was going for (boho). instead, it was very drapery and full and big and i felt like i was wearing
a ghost costume on halloween. i do love retailer's kimonos and thought i'd love this one. but it's going back. i don't need a
ny more clothes in my closet that i do not absolutely love and want to wear again and again. that is my new criteria for purc
hasi

5. The fit is much smaller than blue jeans. I ordered 34W36L if I knew then what I know now I would have bought 36W38L
```

Samples of 4 star rating



```
 Showing 5 sample reviews for Rating 4:

1. The dress is  very feminine and very flirty. i love orange/melon-colored dresses. this one does not disappoint. it does ru
n extremely small around the rib cage. i had to size up to a 6 when i typically wear a 2 or 4. my bra size is 32 b. nice qual
ity fabric, lined, with built-in bustier.

2. I wanted to roder hte petite version, as it runs long, but they are out of the ivory. i put hte green-ish color in my bask
et, and may order it, but for now, i am hlding off as i have other nice seaters... so many choices.

long, warm, fun closure, falttering, and nice tie back detail.

3. These jeans are great. the rise is perfect.

4. This is silly, but the moss is not brown as pictured but olive green. with that "said," i am going more casual, and i love
the "flex" of this coat. the shape is beautiful and the lining is fun. i waited two months to get this in close to my size 0
0. for my style it is a keeper, but this back order situation is not acceptable. don't know whose fault that is?

5. Material seems lighter than the typical Dockers
```

Samples of 5 star rating



```
 Showing 5 sample reviews for Rating 5:

1. Cute and comfy. seems to fit true to size. flattering and roomy on top for my 34ddd shape, but doesn't make me look pregna
nt or fat overall, which sometimes happens without a waist to define shape. super cute!

2. The size fits perfectly and it’s a nice blend of two shade color in the t shirt. It is a perfect outfit to wear for casual
occasions and also the material is very soft and comfortable.

3. I love this top.  the fit is great for me. i ordered a small. the color is a bit more peach than the pink in the picture,
but still very cute.

4. Runs large. Nice fabric and love the color.

5. I saw this dress in a store the other day. it is so cute in real life. i don't think the pictures online even begin to sho
w how cute it is on. the dress is well-made and it fit perfectly. in fact, it is quite flattering. i am so happy i saw it in
the store. even though i am big online shopper i would have passed on this dress if i only saw it online, but don't because i
t is worth every penny.
```

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?

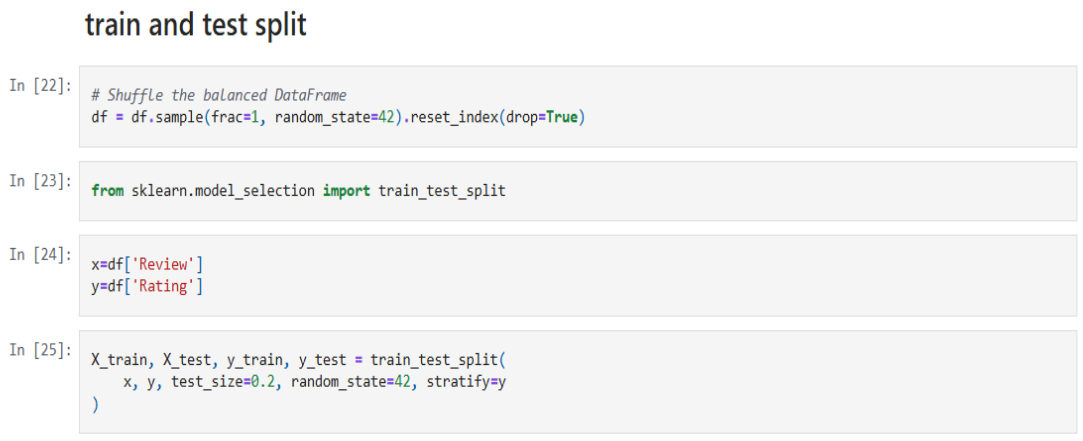
In classification problems like review rating prediction, stratified splitting is important to maintain class balance (equal distribution of ratings) in both training and testing sets.

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:


train and test split

```
In [22]:
# Shuffle the balanced DataFrame
df = df.sample(frac=1, random_state=42).reset_index(drop=True)
```

```
In [23]:
from sklearn.model_selection import train_test_split
```

```
In [24]:
x=df['Review']
y=df['Rating']
```

```
In [25]:
X_train, X_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, 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.

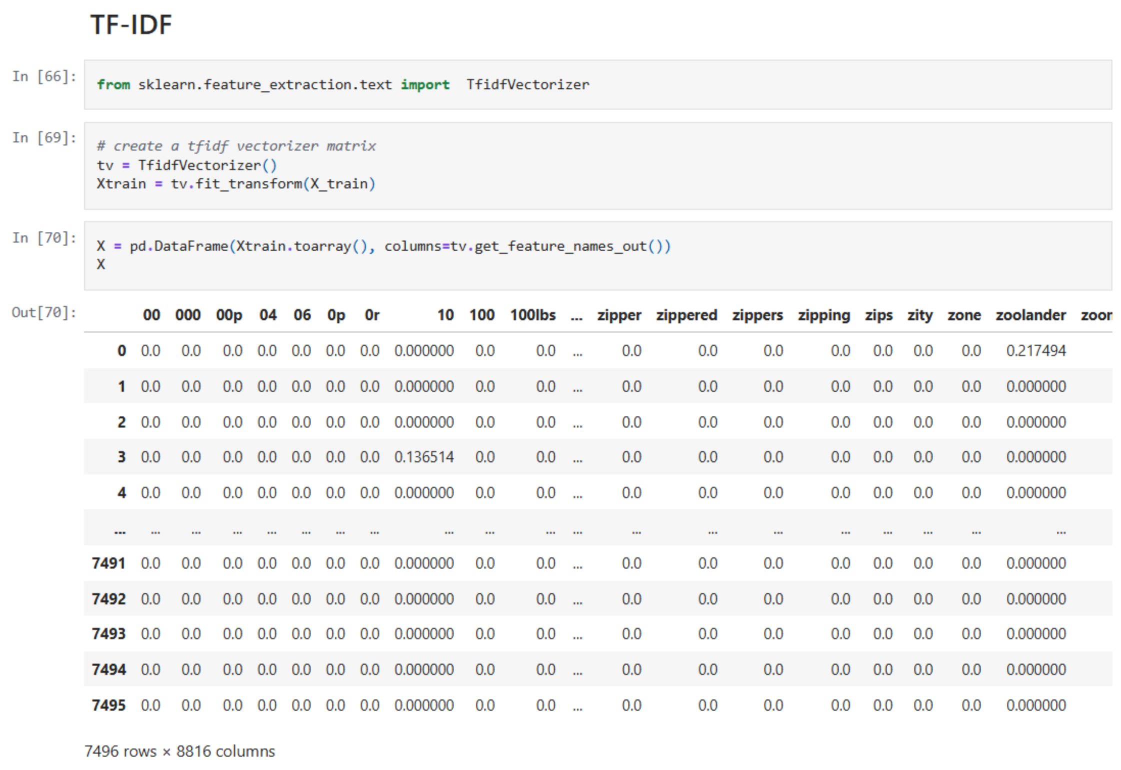
TF-IDF (Term Frequency - Inverse Document Frequency)

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.

Code:



TF-IDF

```
In [66]: from sklearn.feature_extraction.text import  TfidfVectorizer
```

```
In [69]: # create a tfidf vectorizer matrix
         tv = TfidfVectorizer()
         Xtrain = tv.fit_transform(X_train)
```

```
In [70]: X = pd.DataFrame(Xtrain.toarray(), columns=tv.get_feature_names_out())
         X
```

```
Out[70]:
        00  000  00p   04   06   0p   0r        10  100  100lbs  ...  zipper  zippered  zippers  zipping  zips  zity  zone  zoolander  zoor
   0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0     0.0  ...     0.0       0.0      0.0      0.0   0.0   0.0   0.0   0.217494
   1   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0     0.0  ...     0.0       0.0      0.0      0.0   0.0   0.0   0.0   0.000000
   2   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0     0.0  ...     0.0       0.0      0.0      0.0   0.0   0.0   0.0   0.000000
   3   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.136514  0.0     0.0  ...     0.0       0.0      0.0      0.0   0.0   0.0   0.0   0.000000
   4   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0     0.0  ...     0.0       0.0      0.0      0.0   0.0   0.0   0.0   0.000000
 ...   ...  ...  ...  ...  ...  ...  ...       ...  ...     ...  ...     ...       ...      ...      ...   ...   ...   ...        ...
7491   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0     0.0  ...     0.0       0.0      0.0      0.0   0.0   0.0   0.0   0.000000
7492   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0     0.0  ...     0.0       0.0      0.0      0.0   0.0   0.0   0.0   0.000000
7493   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0     0.0  ...     0.0       0.0      0.0      0.0   0.0   0.0   0.0   0.000000
7494   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0     0.0  ...     0.0       0.0      0.0      0.0   0.0   0.0   0.0   0.000000
7495   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.000000  0.0     0.0  ...     0.0       0.0      0.0      0.0   0.0   0.0   0.0   0.000000

7496 rows × 8816 columns
```