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.

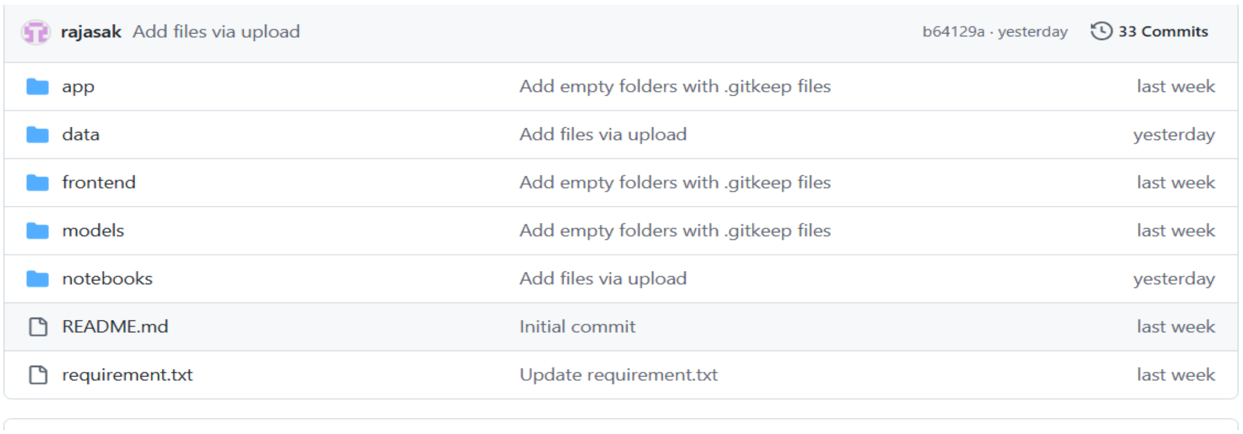
2. Environment Setup

- Installed Python 3.10+ with required libraries: pandas, numpy, matplotlib, seaborn, spacy, scikit-learn, beautifulsoup4, requests, Torch, Transformers
- 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

- Two Dataset was collected from Kaggle
- First Dataset collected has 2 lakh rows it was related to Gadgets and cloths
- Second dataset collected has 4 lakh rows it was related to food
- Both the dataset were merged
- link=https://drive.google.com/file/d/11C2vuTiq9czC7uKFdcVhCAbFHqf8OVPt/view?usp=sharing
- Final dataset contains 2 column with Review and Rating
- 1 star=72800, 2 star = 35403, 3 star= 49213, 4 star=89274 , 5 star= 355005

4.1. Imbalanced dataset

- Link for imbalanced dataset= https://github.com/rajasak/automated-review-rating-system/blob/main/notebooks/deep%20learning/imbalanced_data_for_DL.zip
- Balanced dataset was created with 1 star=30000 ,2 star=12000, 3 star=18000, 4 star=24000, 5 star=36000

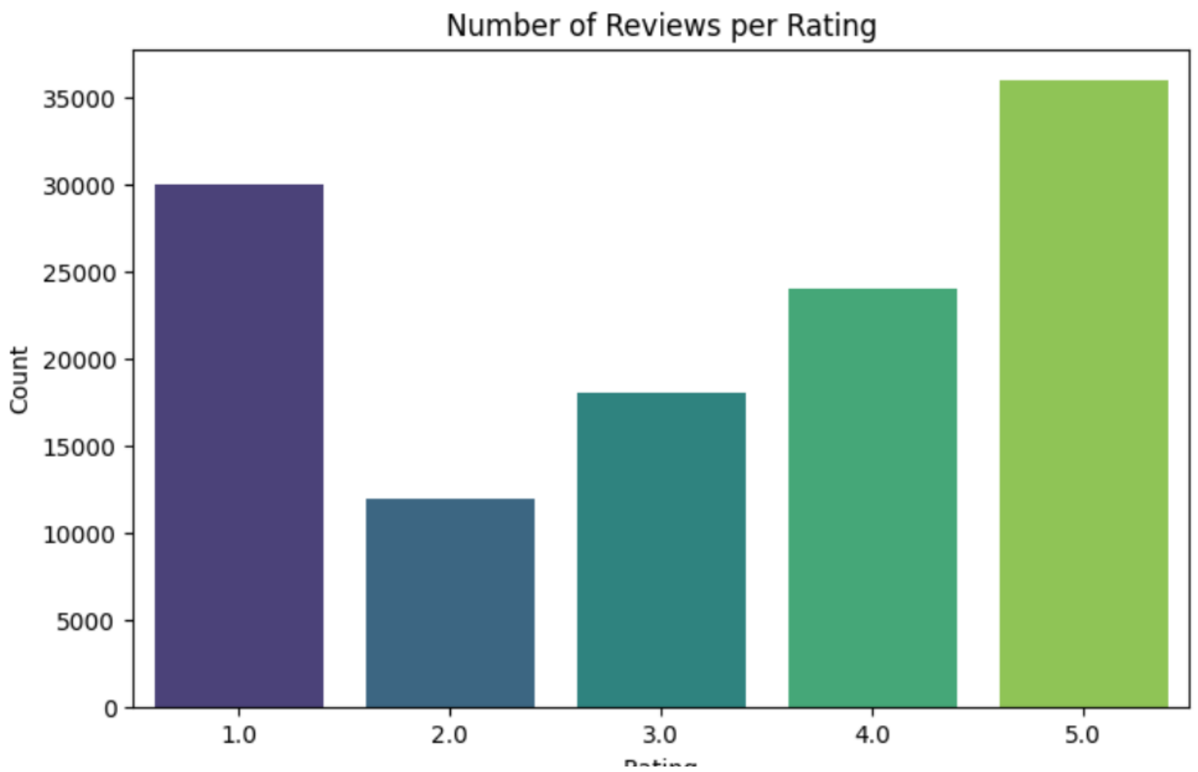


Fig count plot

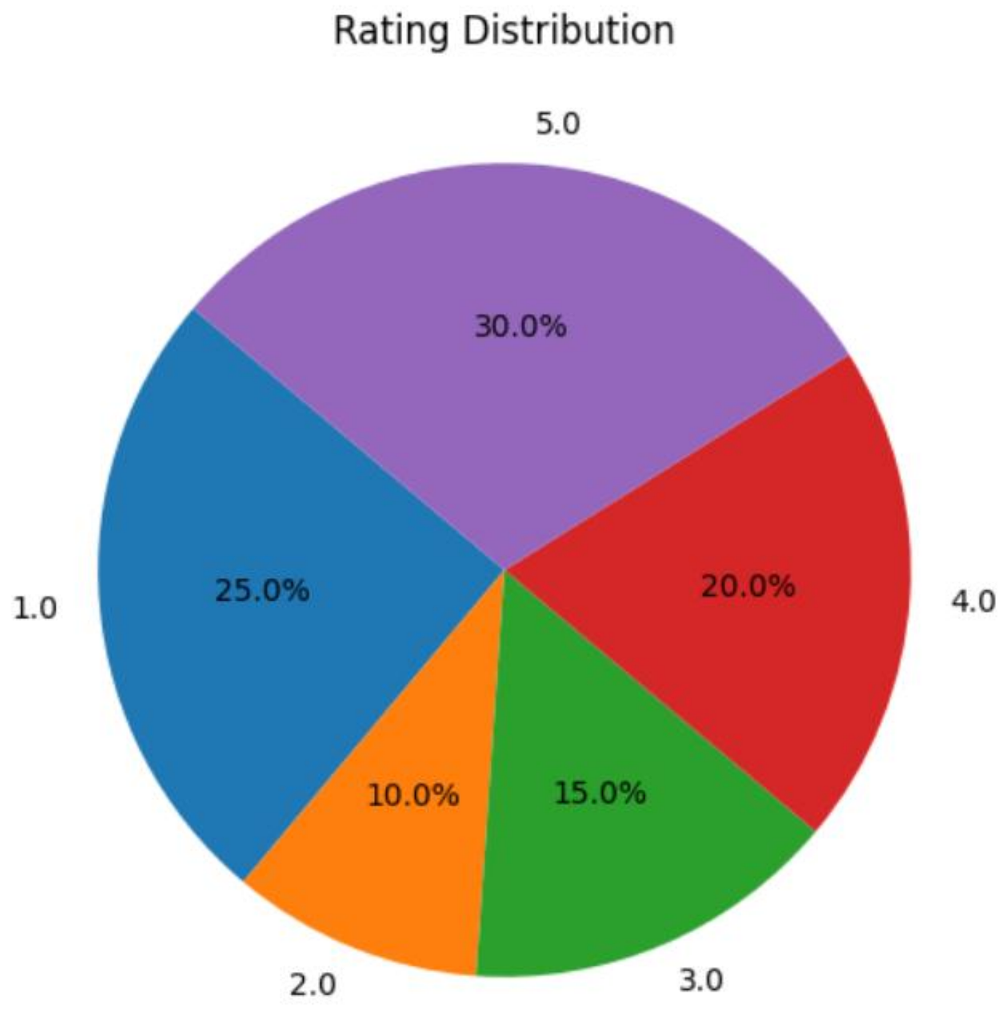


Fig pie chart

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:

```
has_duplicates = df.duplicated().any()
print(has_duplicates)

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

df.drop_duplicates(inplace=True)
```

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:

```
# 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])


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

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:

```
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:

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


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.

```
["'d", "'ll", "'m", "'re", "'s", "'ve", 'a', 'about', 'above', 'across', 'after', 'afterwards', '
again', 'against', 'all', 'almost', 'alone', 'along', 'already', 'also', 'although', 'always', 'a
m', 'among', 'amongst', 'amount', 'an', 'and', 'another', 'any', 'anyhow', 'anyone', 'anything',
'anyway', 'anywhere', 'are', 'around', 'as', 'at', 'back', 'be', 'became', 'because', 'become', '
becomes', 'becoming', 'been', 'before', 'beforehand', 'behind', 'being', 'below', 'beside', 'besi
des', 'between', 'beyond', 'both', 'bottom', 'but', 'by', 'ca', 'call', 'can', 'cannot', 'could',
'did', 'do', 'does', 'doing', 'done', 'down', 'due', 'during', 'each', 'eight', 'either', 'eleven
', 'else', 'elsewhere', 'empty', 'enough', 'even', 'ever', 'every', 'everyone', 'everything', 'ev
erywhere', 'except', 'few', 'fifteen', 'fifty', 'first', 'five', 'for', 'former', 'formerly', 'fo
rty', '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', 'if', 'in', 'indeed', 'into', 'is', 'it', 'it
s', 'itself', 'just', 'keep', 'last', 'latter', 'latterly', 'least', 'less', 'made', 'make', 'man
y', 'may', 'me', 'meanwhile', 'might', 'mine', 'more', 'moreover', 'most', 'mostly', 'move', 'muc
h', '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', 'other', 'others', 'otherwise',
'our', 'ours', 'ourselves', 'out', 'over', 'own', 'part', 'per', 'perhaps', 'please', 'put', 'qui
te', 'rather', 're', 'really', 'regarding', 'same', 'say', 'see', 'seem', 'seemed', 'seeming', 's
eems', 'serious', 'several', 'she', 'should', 'show', 'side', 'since', 'six', 'sixty', 'so', 'som
e', 'somehow', 'someone', 'something', 'sometime', 'sometimes', 'somewhere', 'still', 'such', 'ta
ke', 'ten', 'than', 'that', 'the', 'their', 'them', 'themselves', 'then', 'thence', 'there', 'the
reafter', 'thereby', 'therefore', 'therein', 'thereupon', 'these', 'they', 'third', 'this', 'thos
e', 'though', 'three', 'through', 'throughout', 'thru', 'thus', 'to', 'together', 'too', 'top', '
toward', 'towards', 'twelve', 'twenty', 'two', 'under', 'unless', 'until', 'up', 'upon', 'us', 'u
sed', 'using', 'various', 'very', 'via', 'was', 'we', 'well', 'were', 'what', 'whatever', 'when',
'whence', 'whenever', 'where', 'whereafter', 'whereas', 'whereby', 'wherein', 'whereupon', 'where
ver', 'whether', 'which', 'while', 'whither', 'who', 'whoever', 'whole', 'whom', 'whose', 'why',
'will', 'with', 'within', 'without', 'would', 'yet', 'you', 'your', 'yours', 'yourself', 'yoursel
ves', '‘d', '‘ll', '‘m', '‘re', '‘s', '‘ve', '’d', '’ll', '’m', '’re', '’s', '’ve']
```

Code:

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

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.

5.12 Label Encoding for Classification

Since BERT’s classification head expects class indices starting from 0, the original star ratings (ranging from 1 to 5) were mapped to class labels 0 to 4.

```
df['label'] = df['Rating'].astype(int) – 1
```

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.

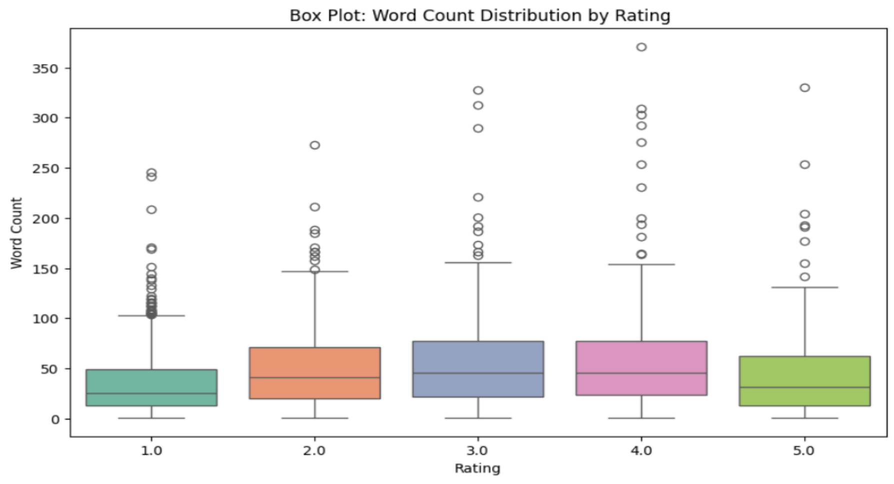


Fig box plot

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.

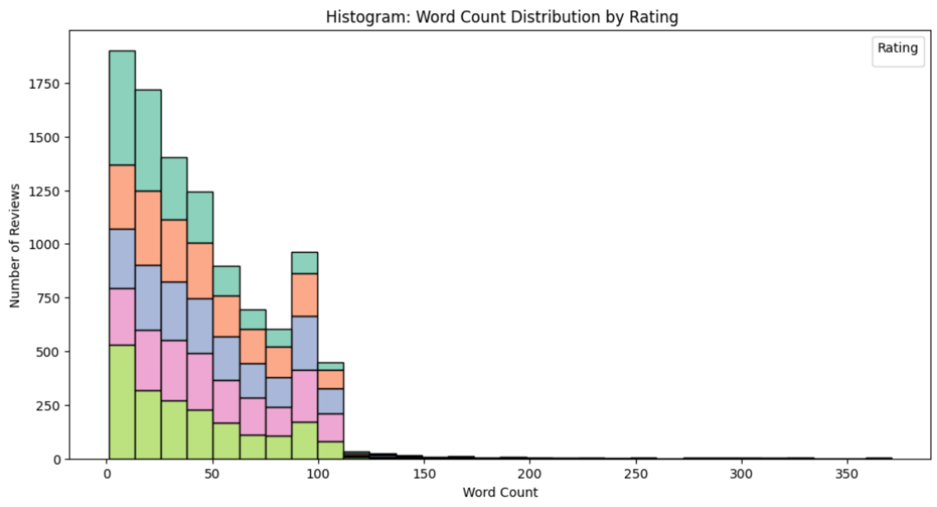


Fig histogram

Samples of 1 star rating:

Showing 5 sample reviews for Rating 1:

1. I wore these pants to work and looked great! however the first time i put them through the wash they became all discolored and are completely stained and now are unwearable. very disappointing. did not spend that much money to wear a pair of pants once.

2. This sweater is enormous. I expected it to be oversized, but I ordered a size small (5’6, 125lbs) and it fit like an XL. The sweat er also had a very bad smell to it. Returning.

3. This thing runs extremely hot, is massively oversized, and doesn’t feel soft and silky on the inside at all. Returning it.

4. Absolutely no stretch, very stiff and thick material, way too short, and shrinks quickly. Selling this item on Poshmark if anyone is interested at a discounted price @homeandcloset.

5. Retailer consistently provides unique, sophisticated and quality made clothing. with the exception of an occasional purchase at their sister companies, i shop with them exclusively. i suppose that i can't expect 100% perfection from retailer-but 99% of the ti me-that's what i get. this is that 1%...this sweater is truly awful! i cannot (or find it very difficult at least) begin to describe what t his sweater is like but it is so unlike retailer that, had it not been for the distinct pattern on the front

Samples of 2 star rating

Showing 5 sample reviews for Rating 2:

1. The color was not white it was off white and the fabric was very flimsy

2. This is such a beautiful dress. but have to return for the reasons others have noted. the chest does not fit well.

3. not perma press!has to be ironed!

4. I like the design of the dress, but the fabric makes it look cheap. for an expensive dress, i had expected better quality.

5. Must return. It looked just right on the model and an embarrassment on me. Even after wash/dry on delicate and immediate dry
er removal it was too wrinkled to wear without ironing. Reread the fabric description on Amazon...says "polyester and spandex",
but the tag on the blouse includes RAYON. This would have to be ironed before wearing. Sooo disappointing.

Samples of 3 star rating

Showing 5 sample reviews for Rating 3:

1. Couldn't wait for this sweater to arrive. i love oversized sweaters for the fall! unfortunately this sweater is extremely itchy that it has to be returned :(

2. As much as i had wanted this to work for me, it didn't. fabric is not as soft as i had expected and quality is just bad. i purchased this online (on sale at $29.95) and i had envisioned this to be comfy stretchy and swingy with a flow. i sent this back to the store as it didn't seem fit even at the low price.

3. I wore it. Probably will find its way into a give-away bag. Just too swingy; too much fabric. Not for the short gals; not for the busty gals; not for the self-conscious "hope no one is staring at my middle" gals.

4. Ordered medium as usual, way small

5. I first have to agree with other reviewers about the quality of this sweater: the stitch is pretty, but the fabric isn't nearly as soft a
s i expected. it has the look and feel of a cheaper sweater. that said, the fit is pretty flattering. i didn't have any problem getting th
e cowl over my shoulders as pictured by the model. i've been loo

Samples of 4 star rating

Showing 5 sample reviews for Rating 4:

1. I was looking for some comfortable but still stylish shorts for the weekend. these hit the mark!

2. awesome product , with lot of varieties

3. Ordered both the black and white and ordered both in 0 and p2. preferred the black over white and the p2 fit the best. however, they reminded me too much of a maternity top so they both went back.
if you order you will need to wear a cami underneath?at least i would. not a terrible top, just not for me.

4. Just note the pink is different than in the photos. much hotter/more saturated. apart from that surprise, overall i'd recommend. n ice design.

5. Like another reviewer said this was too baggy up top. the bottom length was good but arm holes were big and top was just too l arge. i'm 5'5" 110 lbs and got xs. returned.

Samples of 5 star rating

Showing 5 sample reviews for Rating 5:

1. Fit like regular wranglers but with some added stretch in the waits for those thanksgiving dinners!

2. I saw this duster and immediately ordered it in cream. i have worn it every chance i get. it goes well with dresses as well as sho rts or jeans. the length is perfect, the design is flattering, and the material is soft and comfortable. the sleeves are snug- but i like t his feature, especially for light weight summer use over a tank top or sleeveless blouse. this will be a go to item. thanks retailer- a nother gem!

3. good fabric greatfit and the price is right

4. Very nice. Material is comfortable.

5. So I knew that the pajama bottoms had squirrels on them, but no idea that those squirrels were wearing scarves, ear muffs, Santa hats and booties! I adore the pattern! I ordered the size L because I like my jammies loose. I'd say that the sizing runs a little small because I thought the top might be a little too big (I'm bigger on the bottom than the top), but it's not - it fits perfectly

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:

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


from sklearn.model_selection import train_test_split


x=df['Review']
y=df['Rating']


X_train, X_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=42, stratify=y
)
```

8. Conversion to Hugging Face Datasets

The train_df and test_df were converted to datasets.Dataset objects from Hugging Face to support seamless integration with tokenization and training workflows:

Code:

from datasets import Dataset

train_dataset = Dataset.from_pandas(train_df)

test_dataset = Dataset.from_pandas(test_df)

9. Tokenization for BERT

Used the BERT tokenizer from Hugging Face's Transformers library (bert-base-uncased):

Tokenization included:

- Padding to a fixed length (64 tokens)
- Truncating long sequences

Code:

```
from transformers import AutoTokenizer

model_name = "bert-base-uncased"

tokenizer = AutoTokenizer.from_pretrained(model_name)


def tokenize(example):

    return tokenizer(example["text"], truncation=True, padding="max_length", max_length=64)


train_dataset = train_dataset.map(tokenize, batched=True)

test_dataset = test_dataset.map(tokenize, batched=True)
```

11. Formatting for PyTorch

The tokenized datasets were formatted for PyTorch-based training

Code:

```
train_dataset.set_format(type="torch", columns=["input_ids", "attention_mask", "label"])

test_dataset.set_format(type="torch", columns=["input_ids", "attention_mask", "label"])
```

12. Training the model

Algorithm: Bert

Link= **https://github.com/rajasak/automated-review-rating-system/blob/main/notebooks/deep%20learning/bert%20imbalanced.pdf**