**AUTOMATED REVIEW RATING SYSTEM**

1. **Objective**

The goal of this project is to automatically predict or assign ratings (like 1–5 stars) to user reviews (text) using Natural Language Processing (NLP) techniques.  
This saves time compared to manual rating and can be used for e-commerce, app reviews, or any platform collecting feedback.

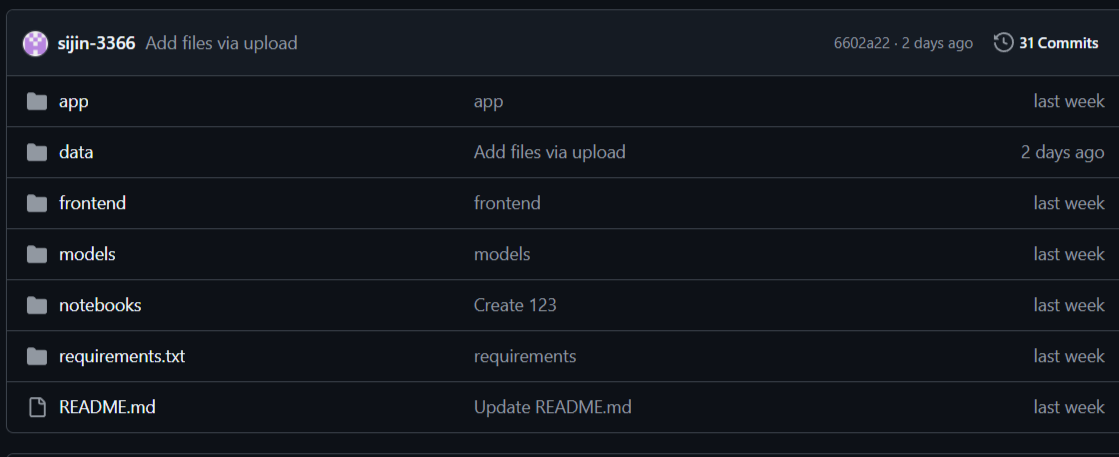
1. **Environment Setup**

* Installed python 3.13.5 with libraries: numpy,pandas ,matplotlib ,seaborn ,scikit-learn,nltk.
* IDE used:VS code

1. **GitHub Setup**

Created a repository : [Automated\_review\_rating\_system](https://github.com/sijin-3366/Automated_review_rating_system)

Structure of directory



1. **Data Collection**

* Datas are collected from Project page of amazon.
* Dataset has 25 lakh rows
* Site link: [https://amazon-reviews-2023.github.io](%20https://amazon-reviews-2023.github.io)
* Dataset link:<https://github.com/sijin-3366/Automated_review_rating_system/tree/2d985f5763234735641c32ba75565e9de834232c/data/amazon_data>

1. **Import packages + Data loading + Concatenate**

**Code:**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**import nltk**

**import warnings**

**warnings.filterwarnings('ignore')**

**df1=pd.read\_csv('product\_data.csv')**

**df2=pd.read\_csv('product\_data1.csv')**

**df=pd.concat([df1,df2],ignore\_index=True)**

**Data Preprocessing**

* **Data preprocessing** is the step of **cleaning and transforming raw data** into a usable format for analysis or model training.
* It improves **data quality**, reduces **noise**, and helps models learn more effectively.

**5.1 Basic Steps**

* Shape: (number\_of\_rows, number\_of\_columns)

df.shape

* Head: Collect first 5 rows

df.head()

* Tail: Collect Last 5 rows

df.tail()

* Columns: Find the column Labels

df.columns

* Drop Unnecessary columns

**5.2 Handling Missing Values**

* Handling missing values is one of the most important parts of data preprocessing.
* Missing values can distort your analysis or model training
* Find Missing values

df.isna().sum()

rating 0

review 1049

* Drop Missing values

df=df.dropna()

df.isna().sum()

rating 0

review 0

**5.3 Unique Rating**

* **Making row unique**

**Code:**

**df[‘rating’].unique**

**array([5, 3, 1, 4, 2])**

* 1. **Remove**
  2. **Remove duplicates and their corresponding row**

**Code:**

**duplicate=df\_clean[df\_clean.duplicated(subset=['rating','review'], keep=False)]**

**new\_data=df\_clean.drop(duplicate.index)**

* 1. **Remove Conflicting reviews**

**Code:**

**variable=df.groupby('review')['rating'].nunique().loc[lambda x: x > 1].index**

**df\_clean=df[~df['review'].isin(variable)]**

**df\_clean=df\_clean.reset\_index(drop=True)**

**print(df\_clean)**

* 1. **Visualization**

**Bar chart**

**Code:**

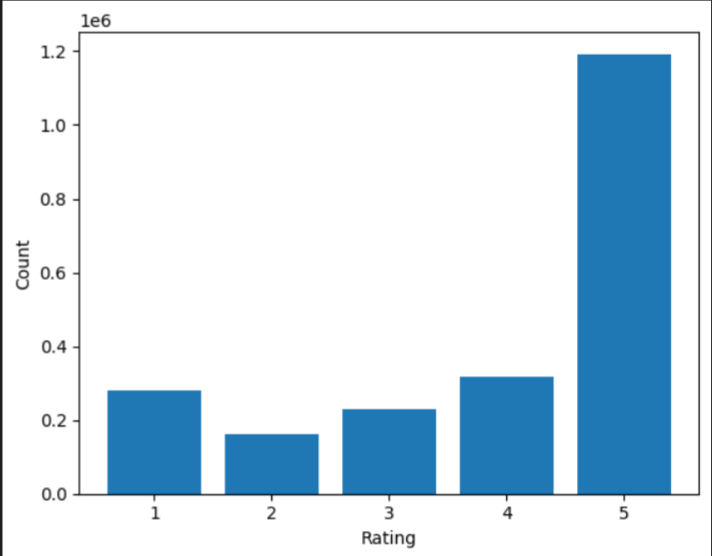
**x=new\_data['rating'].value\_counts().sort\_index()**

**plt.bar(x.index,x.values,color='red')**

**plt.xlabel("Rating")**

**plt.ylabel("Count")**

**plt.show()**

****

* 1. **Balancing**
* Data balancing is done to make sure each class is represented equally, which improves the quality and fairness of our model.
* Collect 25000 rows in each star(1,2,3,4,5)

**Code:**

size=75000

balance=(new\_data.groupby('rating', group\_keys=False)

           .apply(lambda x: x.sample(n=min(len(x),size), random\_state=42)))

balance=balance.reset\_index(drop=True)

balance['rating'].value\_counts()

rating

1 75000

2 75000

3 75000

4 75000

5 75000

Name: count, dtype: int64

**Balanced dataset link:** [**https://github.com/sijin-3366/Automated\_review\_rating\_system/tree/5359e90bd33aeab9170ebfe5d4278808c64f263e/data/balanced\_clean**](https://github.com/sijin-3366/Automated_review_rating_system/tree/5359e90bd33aeab9170ebfe5d4278808c64f263e/data/balanced_clean)

**Bar chart**

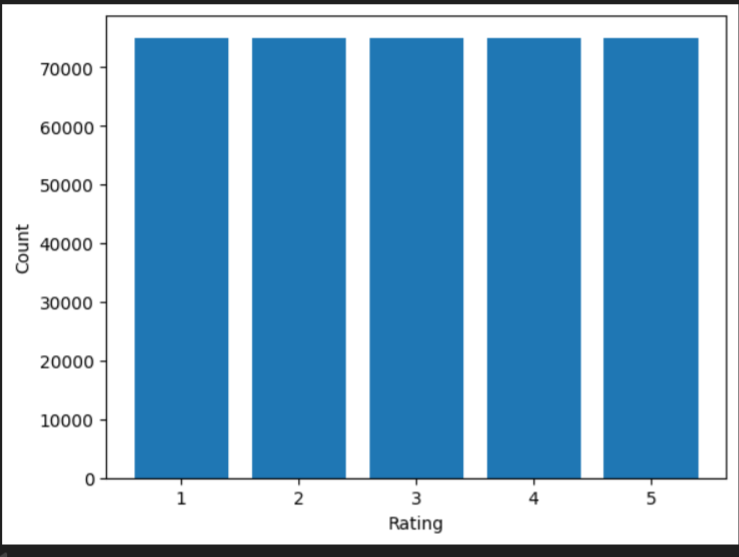
x=balance['rating'].value\_counts().sort\_index()

plt.bar(x.index,x.values)

plt.xlabel('Rating')

plt.ylabel('Count')

plt.show

****

**Pie chart**

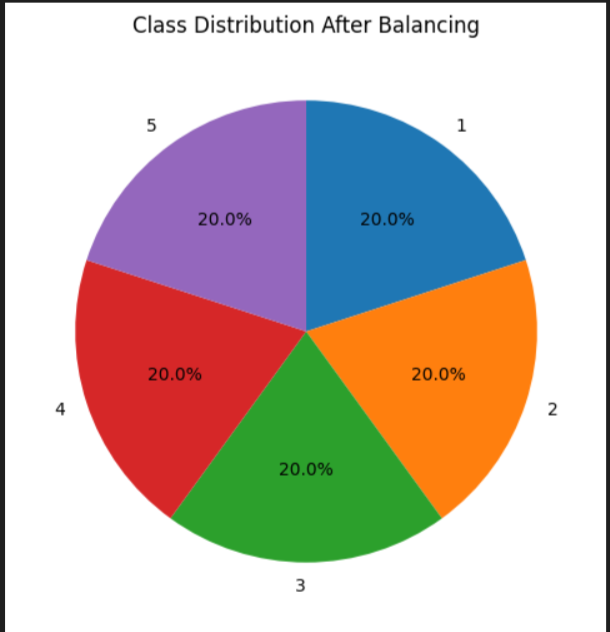
count= balance['rating'].value\_counts()

plt.figure(figsize=(6,6))

plt.pie(count, labels=count.index,autopct='%1.1f%%',startangle=90,counterclock=False)

plt.title('Class Distribution After Balancing')

plt.show()

****

1. **NATURAL LANGUAGE PROCESSING**

Natural Language Processing (NLP) is a field of Artificial Intelligence that focuses on enabling computers to understand, interpret and generate human language. In an automatic review rating system, NLP techniques are used to read customer reviews and predict their sentiment or rating automatically

**6.1 Tokenization**

Tokenization is the process of breaking a large text into smaller units called tokens. These tokens can be words, subwords, characters, or even sentences, depending on the level of analysis required.

**Code:**

**from nltk.tokenize import TweetTokenizer**

**tk=TweetTokenizer()**

**review=review.apply(lambda x:tk.tokenize(x)).apply(lambda x:' '.join(x))**

**6.2 Regular Expression**

A regular expression (often abbreviated as regex) is a special sequence of characters that defines a search pattern. It is used to match, find, or manipulate specific strings of text according to defined rules.

**Code:**

**import re**

**review=review.str.replace('[^a-zA-Z0-9]',' ',regex=True)**

**review**

What this code does:

It cleans the text by removing all special characters (punctuation, symbols) and replacing them with spaces, leaving only letters and numbers. This is a common preprocessing step in NLP to normalize text before tokenization or modeling.

**6.3 Filter text**

**Code:**

**from nltk.tokenize import TweetTokenizer**

**review=review.apply(lambda x:' '.join([w for w in tk.tokenize(x) if len(w)>=3]))**

**review**

**What this code does:**

* Tokenizes each text (especially social media-style text) into words and symbols using TweetTokenizer.
* Removes short tokens (less than 3 characters).
* Reconstructs the cleaned tokens into a single string per review.

**6.4 Lemmitization**

Lemmatization is a text-processing technique in Natural Language Processing (NLP).  
It reduces words to their lemma (dictionary form or base form) by considering the word’s meaning and part of speech (POS).The resulting form may not always be a real dictionary word but serves as a common representation for related words.

**Code:**

**from nltk.stem import WordNetLemmatizer**

**nltk.download('wordnet')**

**nltk.download('omw-1.4')**

**lemmatizer = WordNetLemmatizer()**

**review=review.apply(lambda x:[lemmatizer.lemmatize(i.lower(),pos='v') for i in tk.tokenize(x)]).apply(lambda x:' '.join(x))**

**review**

**Why use Lemmatization instead of stemming?**

**Stemming just chops off prefixes/suffixes using simple rules but Lemmatization uses a dictionary + part-of-speech to return an actual word .**

**6.5 Remove Stopwords**

* Stop words are common words (such as *“the”*, *“is”*, *“and”*, *“in”*) that usually carry little meaning by themselves.
* Removing them is a typical preprocessing step to reduce noise and focus on meaningful terms

**Purpose:**

* Reduce dimensionality of text data.
* Speed up processing and training of models.
* Improve accuracy by keeping only informative words.

**Stopwords =[**'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves',

'you', "you're", "you've", "you'll", "you'd", 'your', 'yours',

'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she',

"she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',

'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which',

'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am',

'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has',

'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the',

'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of',

'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',

'through', 'during', 'before', 'after', 'above', 'below', 'to',

'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under',

'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where',

'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most',

'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same',

'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just',

'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o',

're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't",

'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn',

"hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',

"mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan',

"shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren',

"weren't", 'won', "won't", 'wouldn', "wouldn't"]

* **English stop word list contains 179 stop words.**

**Code:**

**from nltk.corpus import stopwords**

**stop=stopwords.words('english')**

**review=review.apply(lambda x:[i for i in tk.tokenize(x) if i not in stop]).apply(lambda x:' '.join(x))**

**review**

1. **Train-Test Split**

Train–test split is the process of dividing a dataset into two parts:

* a training set used to fit (train) the model, and
* a test set used to evaluate the model’s performance on unseen data.

**Purpose:**

* Prevent overfitting: ensures the model is evaluated on data it hasn’t seen before.
* Estimate generalization: gives a realistic measure of how well the model will perform on new data.
* Provide a fair evaluation: separates data used for learning from data used for testing.

**Code:**

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.30,random\_state=42,stratify=y)**

**Parameters:**

* x,y - Features(x) and Target labels(y)
* test\_size - Fraction or number of samples to use for the test set (e.g., 0.3 = 30%)
* random\_state- Seed for reproducibility; using the same seed gives the same split.
* Stratify=y - Preserves class proportions of y in train/test sets (important for classification).

1. **TF–IDF (Term Frequency – Inverse Document Frequency)**

TF–IDF is a numerical statistic that reflects how important a word is to a document in a collection (corpus).  
It combines two measures:

* Term Frequency (TF): how frequently a word appears in a single document.
* Inverse Document Frequency (IDF): how unique or rare the word is across all documents.

**Formula:**

**TF-IDF(t,d)=TF(t,d)×IDF(t)**

* **TF(t,d) = number of times term *t* appears in document *d* ÷ total number of terms in *d*.**
* **IDF(t) =log(Total Documents/no:of Documents containing t)**
* **t=term(word)**
* **d=document**

**MODEL CREATION**

**Code:**

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

lg=LogisticRegression(max\_iter=25000)

lg.fit(x\_train,y\_train)

y\_pred=lg.predict(x\_test)

accuracy=accuracy\_score(y\_test,y\_pred)

print("Accuracy:",accuracy)

**Model explaination:**[**https://github.com/sijin-3366/Automated\_review\_rating\_system/blob/5359e90bd33aeab9170ebfe5d4278808c64f263e/models/Logistic%20Regression%20Model.docx**](https://github.com/sijin-3366/Automated_review_rating_system/blob/5359e90bd33aeab9170ebfe5d4278808c64f263e/models/Logistic%20Regression%20Model.docx)

**Samples of 1 star**

* **Waste no use...**
* **Cheap, don't even waste your time.**
* **Fabric is really cheap and the shape is not right at the top. Falls off shoulders even when it is otherwise the right size.**
* **This dress was made very cheap and it did not meet the expectations of a $35 Dollar purchase. You couldn't tell the front from the back. The dress didn't have any tags to determine the front from back. Very poor item.**
* **They both had broken handle.....The first one I received had broken handle, so I asked to replace it. But, the second one also had the same problem.....OMG"**
* **It lasted less than 2 months before the light button fell off. Don’t plan on buying these again!**

**Samples of 2 star**

* **This is not good in price**
* **Nice price but very light weight and cheaply made.**
* **The fit was not right for me**
* **I returned these, they were too heavy to even give the yellow tint a chance. Were not comfortable to have on my glasses.**
* **I purchased this yesterday and I didn't get the 110 pieces, from what I counted I got half  And not only that I didn't get the ones from this ad shows, I got the fuzzy balls  ....but it's great quality and a great price.....sadly I'll think twice about buying from amazon because some have the no return policy....think about it when it's looks good to be true...then something wrong .just saying"**
* **The beads are not the same color in the photo and they fit a little snug**

**Samples of 3 star**

* **I really did not like these cookies..**
* **Way too small despite the size label of 2-4t. My 3 year old is small for her size and these socks didn't cover her entire foot. Too bad because she otherwise loved them but I had to return them.**
* **nice look way to small**
* **Much larger than it appears. It appears that the heart was opaque butbis actually clear glass. Good quality**
* **Thinner material than expected... also smaller than anticipated... normally wear a size XL in US sizes...**
* **Not sure how to contact the seller however we really enjoyed our beautiful purse except there was an ink stain on one of the handles of the purse. Unfortunately my wife was a little displeased about it. I looked everywhere for the sellers email address I couldn't find anything. Maybe they'll read the review and contact us. Otherwise the purse is exactly as described and it's extremely beautiful with lots of pockets and plenty of room. I would post some photos of The purse but don't know how to do it for this review. Thank you.**

**Samples of 4 star**

* **Good fit, not crazy about the look of the material but they are comfortable**
* **I loved it. The only issue I had was that the top doesn't fit completely well. The bottom band folded into itself and it's hard to keep straighten. If you can get past that it's super cute. and that's a minor issue for the price and compliments.**
* **Very cute but had to return it. Very short and I'm 5'4&#34;. Needs tights**
* **Super comfy, great color... the only reason why I'm giving this 4 stars is because it sheds.**
* **Cute, but darker than I wanted....I wanted real light blue quartz...."**
* **Very nice fitting top but the fabric does curl on the curve of the bodice where the decorative buttons are.  I know how to sew and can stitch it down.  So, I kept it.**

**Samples of 5 star**

* **I wear them everyday at work I simply just love them! Yes, I will buy them again. They are not too large, so you look like the 85 year old driving down the street, pretty stylish for over the glasses shades."**
* **It is a great leotard.  I do wish it had snaps underneath for the bathroom.**
* **My kid ordered this without asking. While I think it looks dumb... I have checked thoroughly to make sure this won’t hurt her teeth in the long run and let her wear them for stupid pictures. It’s lasted awhile with no show of wear. So if this is your kinda thing I guess I’m giving it 5 stars. I still find it weird but to each their own.**
* **Great purse, took on a cruise. Was lightweight, has lots of secure zip pockets. Love the outside pockets which securely held my glasses.  Strap could go on shoulder or cross body, adjusts easily. The various shades of pink are very pretty. Reasonably priced. One of my best purchases!**
* **Fits well. Flatters. Pockets are reasonably sized. Very soft and comfy. Not too thin, slightly stretchy t-shirt knit. Neckline may be a little low if you're looking for something for work /church, but a great little dress for every day errands, lunch, etc. I probably going to buy it in another color.**
* **They are very vibrant colors and the length were perfect for my braids!! They kept my braids fresh for 3 weeks straight!.**