# AUTOMATED REVIEW RATING SYSTEM

## 1. Project Overview

The Automated Review Rating System is natural language processing (NLP) project aimed at predicting the rating of a review based on its textual content. This project automates the process by cleaning and preprocessing text data, extracting meaningful features and preparing the dataset for machine learning models that can learn the relation between review text and its associated rating.

## 2. Environment Setup

- Programming Language used : Python 3.12.1
- Libraries Used:
  - Pandas & Numpy → for data manipulation
  - Seaborn & Matplotlib → for data visualization
  - Nltk & BeautifulSoup → for text normalization
  - Emoji → for filtering out emojis
  - Scikit-Learn → for sampling, train-test and Vectorization
- IDE/Notebooks : Jupyter Notebook

## 3. GitHub Project Setup

Created a GitHub repository for the project: **automated-review-rating-system**

Structure of directory:

- App: Contains the **application logic** (e.g. Django backend code) that serves your trained model.
- Data: Stores the **raw or preprocessed datasets** used for training, validation, and testing.
- Frontend: Contains the **UI code**.
- Models: Holds the **trained machine learning/NLP models**.

- Notebooks: Contains **Jupyter notebooks** used for exploration, preprocessing, visualization, and model experimentation.
- Readme: The **documentation entry point** of the project.
- Requirements: A text file listing all the **Dependencies**.

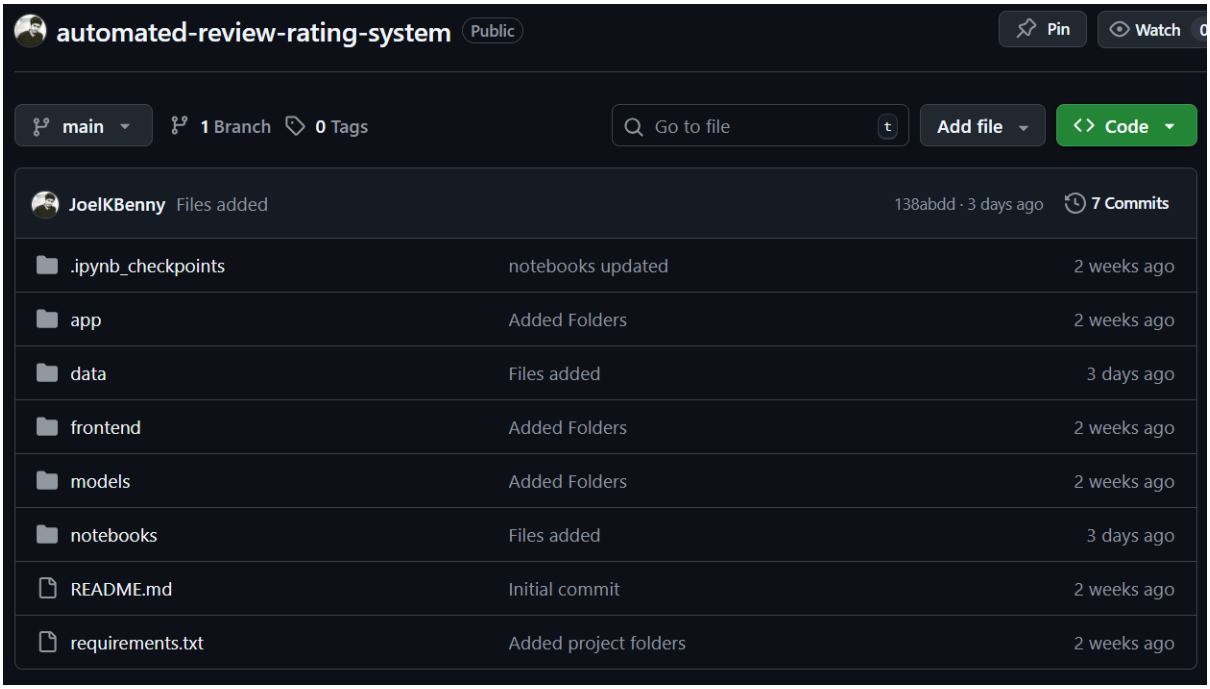



*Figure 1: GitHub Repository Structure*

## 4. Data Collection

- Data is collected from Kaggle Datasets
- Dataset Link: https://github.com/JoelKBenny/automated-review-rating-system/blob/main/data/Kaggle%20Datasets/Disneyland_Reviews.csv
- Dataset Contains 42656 samples along with 6 attributes:
  - Review_ID : Numerical Identification number.
  - Rating : 1 to 5, 1 being lowest and 5 being highest.
  - Year_Month : Year and month which review is posted.
  - Reviewer_Location : Location of the reviewer.
  - Review_Text : Textual content of the review.
  - Branch : Disneyland branch (location).

- The count of sample of each rating:
  - **1 star** : 1,499
  - **2 star** : 2,127
  - **3 star** : 5,109
  - **4 star** : 10,775
  - **5 star** : 23,146

### 4.1. Balanced Dataset

- Balanced dataset link: https://github.com/JoelKBenny/automated-review-rating-system/blob/main/data/Kaggle%20Datasets/balanced_disneyland_reviews_data.csv
- Balanced Dataset contains 2000 samples of each rating.

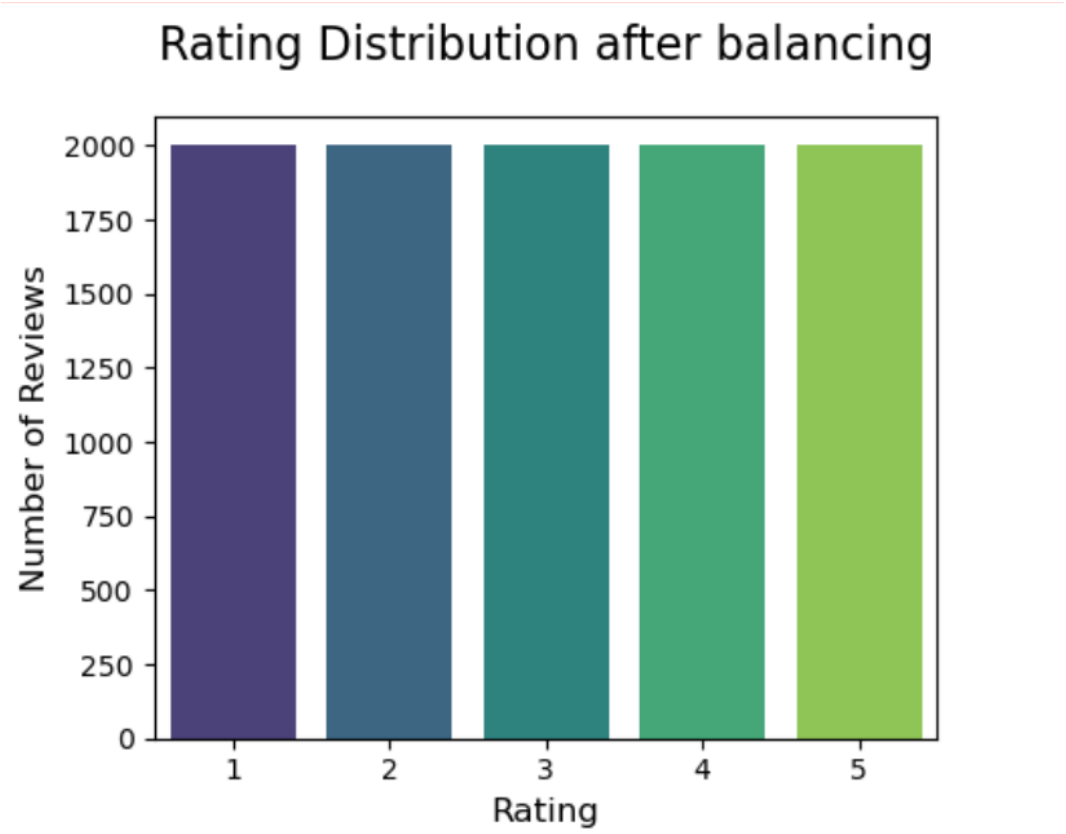



*Figure 2: Bar Chart of Rating*

# 5. Data Preprocessing

Preprocessing is the stage in machine learning where raw data is cleaned and transformed into a structured, usable format. It involves steps like removing noise, handling missing values, reducing irrelevant information from the data.

Preprocessing ensures the data fed into machine learning models is consistent, meaningful and free from errors. This improves model accuracy, reduces bias, and speeds up training by removing unnecessary complexity.

## 5.1. Removing Duplicates

The exact same reviews and rating were removed to avoid redundancy and ensure that each sample contributes unique information to the model.

**Code:**

```
df.duplicated().sum()

df = df.drop_duplicates(subset=['Review_Text', 'Rating'])
```

## 5.2. Removing Conflicting Reviews

The same review text appeared with different ratings, were removed to maintain consistency between the content and its assigned rating. Such conflicts can confuse the model during training and reduce its ability to learn meaningful patterns.

**Code:**

```
#filtering the review that have more than 1 unique review

conflicts = df.groupby('Review_Text')['Rating'].nunique()

conflicting_reviews = conflicts[conflicts > 1].index

print("Number of reviews with conflicting ratings:",
len(conflicting_reviews))

df = df[~df['Review_Text'].isin(conflicting_reviews)]
```

## 5.3. Handling Missing Values

No missing values found in the data when checked.

**Code:**

```
df.isnull().sum()
```

```
Review_ID            0
Rating               0
Year_Month           0
Reviewer_Location    0
Review_Text          0
Branch               0
```

## 5.4. Removing Unnecessary Columns

Irrelevant columns like Review_ID, Reviewer_Location, Branch, Year_Month were dropped because these attributes does not contribute to the model.

**Code:**

```
df = df.drop(['Year_Month', 'Review_ID', 'Branch',
'Reviewer_Location'], axis=1)
```

## 5.5. Lowercasing Text

All review text was converted to lowercase to ensure uniformity and prevent the model from treating words like 'Disney' and 'disney' as different tokens.

**Code:**

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

## 5.6. Removing HTML Tags, URL's & Emojis

HTML tags, URLs, and emojis were removed from the reviews to eliminate irrelevant elements that do not contribute to the semantic meaning of the text.

**Code:**

```
#html tags
text = BeautifulSoup(text, "html.parser").get_text()
```

```
#URL
text = re.sub(r"http\S+|www\S+|https\S+", "", text)

#Emoji
text = text.encode("ascii","ignore").decode("ascii")
```

## 5.7. Removing Punctuations and Special Characters

Punctuation and special characters were removed to reduce noise and focus the analysis on meaningful textual content.

**Code:**

```
text = text.translate(str.maketrans("", "",
string.punctuation))
```

## 5.8. Tokenizing the Reviews

Tokenization was applied to split each review into individual words or tokens, allowing the model to analyse and process the text at the word level.

**Code:**

```
tokens = text.split()
```

## 5.9. Removing Stopwords

Stopwords (common words like 'the', 'is', 'and') were removed to reduce noise and focus the analysis on words that carry meaningful information. NLTK library is used to filter the stopwords; there were total 179 stopwords in NLTK.

A total of 151 unique stopwords found and removed from the whole reviews.

{'why', 'above', 'from', "we've", 'ain', "he'll", 'with', "we'd", 'her', "she's", 'each', 'be', 'isn',
"i'd", 'where', 'weren', 'having', 'when', 'him', 'on', 'then', 'its', 'very', 'few', "don't", 'has', "have
n't", 'wasn', 'didn', 's', 'into', "aren't", "i've", "he's", "wouldn't", 'just', 'd', 'than', 'should', 'sh
ouldn', "shouldn't", "it'd", 've', 'hasn', "it's", 'wouldn', 'i', 'doing', 'during', "it'll", 'we', 'any',
'your', 'both', 'have', "she'll", "she'd", 'mustn', "i'll", "should've", 'ours', 'that', "isn't", 'yourselv
es', 'before', 'not', 'after', 'by', 'same', 'what', 'is', 'ourselves', 'their', "weren't", "i'm", 'yoursel
f', "didn't", "they're", 'off', 'nor', 'herself', 'the', "hadn't", 'am', 'now', 'a', "shan't", 'll', 'you',
'do', 'was', 'did', 't', 'no', 'other', 'up', 'because', 'needn', 'against', 'these', 'theirs', "you've",
'more', 'hadn', 'it', 'me', 'are', 'such', 'about', "they'll", 'again', 'were', "hasn't", 'through', 'her
s', 'under', 'all', 'own', "we'll", 'won', 'only', 'once', 'can', 'most', 'in', 'yours', "couldn't", 'bein
g', 'his', 'had', 'shan', 'to', "won't", 'who', 'at', 'but', 'he', 'or', "you're", 'y', 'myself', 'himsel
f', "mightn't", 'some', 'itself', 'here', 'there', 'as', "needn't", "they'd", 'aren', "that'll", 'm', 'to
o', 'an', 'for', 'while', 'until', 're', 'themselves', 'and', "mustn't", "we're", 'couldn', 'o', 'down', 'd
on', 'so', 'those', 'this', 'she', 'been', 'which', 'of', 'will', "wasn't", 'ma', 'haven', 'further', 'the
y've", 'if', 'over', 'below', 'my', 'whom', 'does', "he'd", 'them', 'mightn', 'they', "you'd", 'out', "yo
u'll", "doesn't", 'doesn', 'between', 'our', 'how'}

*Figure3: NLTK Stopwords*

**Code:**

```
stop_words = set(stopwords.words('english'))
tokens = [word for word in tokens if word not in stop_words]
```

## 5.10. Lemmatization

Lemmatization is the process of reducing words to their base or dictionary form, called a lemma. For example, 'better' is reduced to 'good', and 'studying' is reduced to 'study'. This helps the model treat different forms of the same word as one, improving consistency and reducing the feature space.

**Why Lemmatization was preferred than Stemming?**

Stemming is a crude method that chops off word endings to reduce words to their root form, which can sometimes create non-words (e.g., 'studying' → 'studi', 'happier' → 'happi'). Lemmatization, on the other hand, reduces words to their proper dictionary form (lemma) while preserving meaning (e.g., 'studying' → 'study', 'happier' → 'happy'), making it more suitable for text analysis and sentiment modelling.

**Code:**

```
tokens = [lemmatizer.lemmatize(word) for word in tokens]
```

### 5.11. Filtering Reviews Using Word Count

Reviews with fewer than 3 words or more than 100 words were removed to eliminate overly short or excessively long texts that could introduce outliers and reduce model effectiveness.

```
count    42617.000000
mean        66.162658
std         74.995422
min          3.000000
25%         23.000000
50%         42.000000
75%         80.000000
max        970.000000
Name: word_count, dtype: float64
```

*Figure 4: Word Count Statistics*

Most of the reviews falls under of 100 words. So filtering thresholds are set to: 3 for minimum threshold and 100 for maximum threshold, Other reviews are considered as Outliers and were removed.

**Code:**

```
if len(tokens) < 3 or len(tokens) > 100:
        return None
```

## 6. Data Visualization

### Box Plot

A boxplot is a graphical representation of the distribution of a dataset that shows the median, quartiles, and potential outliers. In this project, it is used to visualize the distribution of review word counts across different ratings, highlighting unusually short or long reviews.

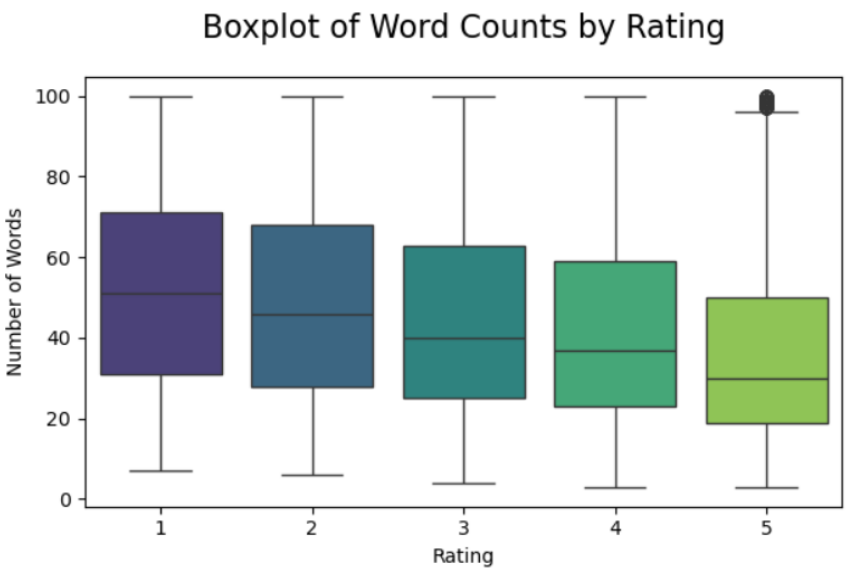



*Figure 5: Boxplot of word count*

## Histogram

A histogram is a graphical representation of the distribution of numerical data using bars to show the frequency of data points within specified intervals (bins). In this project, a histogram is used to visualize the distribution of review word counts across different ratings, allowing us to see how review lengths vary by rating.

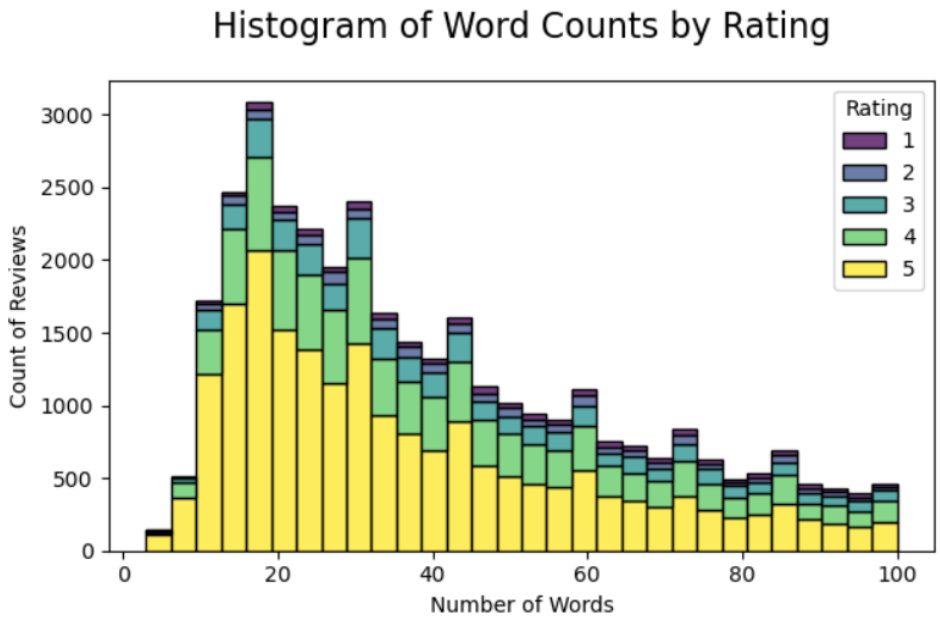



*Figure 6: Histogram for word count*

## Bar Chart

A bar chart is a graphical representation of categorical data using rectangular bars to show the frequency or count of each category. In this project, bar chart is used to visualize the distribution of review ratings, showing how many reviews fall into each rating from 1 to 5.

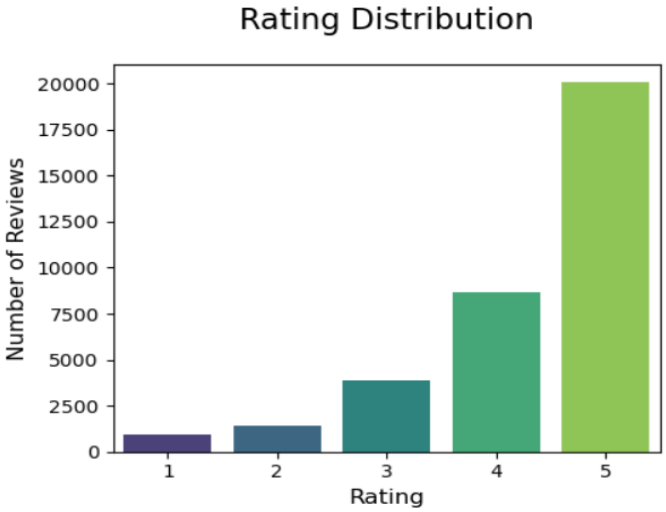



*Figure 7: Bar Chart for Rating*

## Sample of 1 star Rating:

5 sample reviews of 1 star rating:

1. partner recently stayed teacher school trip spoke highly service attraction whilst stay bought mikey minnie photo frame gift bring home though got home discovered shocking quality product material peeling back stand paint work look like done toddler crayon general quality product disgracefuli would advise future review quality product sale resort shop avoid disappointment shabby photo frame ruined supposed romantic gesture well done disney

2. visiting disneyland world around globe feel qualified write review particular oneit definitely worst longest line ever seen ride open many staff member around lot people problem using fastpass machine one sight help happened one occasioni definitely agree another review saw visitor park bad queue etiquette pushing back pushing standing way close argh yes smokingdisneyland tokyo far best disneyland go one make oneif come paris dont waste day 6 hour park got 1 ride theatre showstand line louvre eiffel tower much better reward

3. ridiculus wait 30 minute queu ride 1 minute would prefer come tivoli much better nicer place recommended people enjoying waiting utku

4. price found park ride disappointing travelled feb 12 half term queue long several ride closed due technical fault buzz light year small world etc food within park astronomically priced 3 euro tiny bag chip

5. whole place feel like homage 1990s parade basic ride broke frequently already shut refurbs waited 35 minute allegedly fastpass queue allowing people regardless timing join big thunder mountain big ride day open disney studio better tiny ride dont fast pass waited 2 hour watch ride break front u restaurant either shut full fried food mickey cafe basic food booking queued 30 minute valentine waitress rude forgot thing mixed order completely hotel santa fe basic queue breakfast one pay attention time slot best thing planet hollywood thats saying something whole place need investment magic putting back

## Sample of 2 star Rating:

5 sample reviews of 2 star rating:

1. park small great ride closed limited number ride biggest thing missing park friendly staff missing expected disney smile courtesy ride operator folk restaurant recommended big fan happiest place earth way functioning u disney park take literally

2. disney happy ever afters smile magic laughternot disneyland parisstaff ignorant rude havent clue smile visited park two day queue would expect long fast pas joke queue fast pas ticket queue ride didnt bother end queued anyway food ok bit expensive rest paris areano character walking round parade good short missing something disney magic like visited disney florida dont go paris comparison disappointed

3. disneyland favorite vacation way busy october friend trip said busier summer visit lot ride closed temporary repair waited line 1 hr frustrating

4. cafe orleans nice lobster salad gone choice getting worselets review dining death actually good disney like spend money good foodlobster saladcafe orleansfrito misto alfresco terracecinnomon rollsdisneyland bakerycinnomon roll french toastcarnation cafemahi mahi fish tacostangaroa terrace cheap small shrimp tacosnice breakfastriver belle terrace expensive mediocre restarauntshort rib sandwhich carthay circlechili fry sour creamsmokejumpers grillblt flatbreadnow red rose tavernefried green tomatoeshungry bear get 5x price bbim sick disneyland taking away good thing replacing obviously cheaper dining option want u spend dollar park improve dining

5. large party 8x adult 6x child travelled disneyland paris 26th december 2015 spent two day theme park purchased ticket ahead time uk cost 95 per person is more

## Sample of 3 star Rating:

5 sample reviews of 3 star rating:

1. 4 lovely day park october seems great time go quite quiet line werent long especially week big complaint staff never seen many bored disinterested staff meant happiest place earth smile eh daughter didnt care notice anoyed u alot perhaps disney start customer service training quick

2. sure disney label totally lack florida magicmore like amusement parkpossibly expensive outlet ever experienced 1 plain hotdog 1 cheese hotdog 1 chicken nugget 2 large fry 1 white coffee 1 sprite 25

3. gate price seem keep going cant really afford go every couple year park always crowded line take forever even charge fast pas lane thought used free anyway still happiest place earth except raining pay 20 poncho really come

4. disney expected crowded overpriced use fastpass whenever possible dont eat park food awful overpriced found tortilla joes taqueria downtown disney huge burrito shared two people kid loved placeand better cheaper eating park

5. ride old tired couldnt believe theyre still showing michael jackson captain eo unfortunately get away long kid still want go disney place earth disappointing child 9 charged adult fare

## Sample of 4 star Rating:

5 sample reviews of 4 star rating:

1. always good time disneyland even hiccup year 25 hour wait ride hot sun shaded queue perhaps consider water ice cream stand line super popular ride california adventure side wasnt happy ride closure breakdown though

2. ive disney hongkong californiathey fantastic beautiful scenery show still kind similar disney world except one orlando orlando biggest one world

3. husband medical emergency first day park staff fantastic u ambulance time unfortunately cut stay short didnt get experience park

4. didnt stay resort went day 4 day honeymoon paris parisvisite travel pas get 20 entrance fee 1 day 2 park pas cost u 50 euro get went pretty much ride excluding little kid stuff space mountain best went 3 time tower terror also good buzz lightyear good fun even 28 year old avoided eating taking baguette u glad menu park looked expensive half good little parisian restaurant train centre paris take 40 min literally take front doorhave disney florida couple time far better still good fun

5. enjoyed experience disneyland hong kong found catered 5 year old teenager seems suited younger child rather eurodisney found better older kid adult definitely worth

checking day spare havent disneyland firework park close lovely packed id suggest want buy thing store earlier day crowded crazy end

### Sample of 5 star Rating:

5 sample reviews of 5 star rating:

1. great ever really enjoyable age sure extra magic hour worthwhile nothing much open early still great day two three

2. lucky enough come los angeles family holiday disneyland focus dad keen queueing crowd noise imagine excited rest family disneyland awesome great time year cast member call ghost town kiwi found food good price used high priced food back home crowd fine really buy fast past far best money spent cast member staff standing place spotless watched rubbish would ground matter minute pick went four day two disneyland two disney adventure great felt saw offer age kids1110 8if come hope much fun didcheerskiwi los angeles

3. long line expensive kid love place along adult act like kid fun whole family

4. still nice classic park third visit although first 9 year materhorn updated better set special effect inside space mountain smoothest way best side side seating indiana jones ride unique closest thing disney dinosaur animal kingdom florida one best thing electric light parade brought back year classic original form found fun nostalgic even 19 year old son thought great doesnt really connection wife aladins oasis lunch tie gave u priority viewing near small world thought worth

5. one spend 2 3 day see allthe best avoid kid holiday day weekend holiday possablei found sign direction park poor even holding map ask directionsthere enough personal walk around assist peoplethe line long express line help enoughin overall great intertaiment

## 7. Balancing the Dataset

Balancing a dataset involves ensuring that each class has an approximately equal number of samples to prevent model bias toward majority classes.

In this project, the Disneyland review dataset was imbalanced across ratings. Classes with fewer than 2,000 reviews (ratings 1 and 2) were oversampled with replacement, while classes with more than 2,000 reviews (ratings 3, 4, and 5) were downsampled without replacement, resulting in a balanced dataset of 2,000 reviews per rating.

After resampling, the dataset was shuffled randomly to mix reviews from different ratings, ensuring that the order of samples does not introduce unintended patterns when splitting into training and test sets.

After Balancing :

| Rating | Samples |
|---|---|
| 1 | 2000 |
| 2 | 2000 |
| 3 | 2000 |
| 4 | 2000 |
| 5 | 2000 |

## 8. Train-Test Split

Train-test splitting is the process of dividing a dataset into two subsets: one for training the model and one for evaluating its performance, ensuring that the model is tested on unseen data.

In this project, the balanced Disneyland review dataset was split into training and test sets using an 80:20 ratio. Stratified splitting was applied to maintain class balance in both sets, so each rating (1–5) is proportionally represented in the training and test subsets.

**Why Stratified Splitting is used?**

Stratified splitting was used to ensure that each rating class is proportionally represented in both the training and test sets, preventing the model from being biased toward majority classes and ensuring fair evaluation.

**Code:**

```
from sklearn.model_selection import train_test_split
x = balanced_df['cleaned_review']
y = balanced_df['Rating']

x_train, x_test, y_train, y_test = train_test_split(x,
y, test_size=0.2, stratify=y, random_state=42)
```

**Working**

- The dataset is divided into **training (80%)** and **test (20%)** sets.
- stratify=y ensures each rating (1–5) is proportionally represented in both sets.
- random_state=42 makes the split **deterministic**, so running the code again produces the same split.
- x_train and y_train are used to **train the model**, while x_test and y_test are used to **evaluate model performance** on unseen data.

## 9. Text Vectorization

Text vectorization is the process of converting textual data into numerical representations so that machine learning models can process and analyse the text.

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

**TF-IDF** is a vectorization technique that assigns weights to words based on how important they are in a document relative to the entire corpus, giving higher importance to words that are frequent in a document but rare across other documents.

**Formula for TF-IDF**

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF - IDF(t,d) = TF(t,d) * IDF(t)$$

*Figure 7: TF-IDF Formula*

**Code:**

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=5000)

# fitting on training data
x_train_vec = vectorizer.fit_transform(x_train_clean)

# Transforming the test data
x_test_vec = vectorizer.transform(x_test_clean)
```

## 10. Model Building

Model building is the process of selecting an appropriate machine learning algorithm, training it on preprocessed data, and enabling it to learn patterns or relationships within the dataset. The trained model can then make predictions on new or unseen data.

In this project, various classification models such as Logistic Regression, Random Forest, Naïve Bayes, and SVM were trained on the processed review text to predict the corresponding rating based on the content of the review.

## 10.1. Logistic Regression

Logistic Regression is a supervised machine learning algorithm used for classification tasks. It predicts the probability that an instance belongs to a particular category by using a logistic (sigmoid) function, which maps any real-valued number into a range between 0 and 1.

**Methods:**

- **Binary Logistic Regression:** Used when the target variable has two possible classes (e.g., yes/no, true/false).
- **Multinomial Logistic Regression:** Used when there are more than two classes, applying the softmax function to compute probabilities for each class.
- **Ordinal Logistic Regression:** Used when the dependent variable has ordered categories (e.g., ratings from 1 to 5).

**Code:**

```
model = LogisticRegression(multi_class = "multinomial",
                           solver = "lbfgs", #softmax function
                           max_iter = 500,
                           random_state = 42)
model.fit(x_train_vec, y_train)

y_pred = model.predict(x_test_vec)
```

```
Accuracy of model :  0.581
Precision         :  0.5681542639581506
Recall            :  0.581
F1 Score          :  0.5724634808291365
```

```
Confusion Matrix :

 [[339  30  16  13   2]
 [ 66 263  58   6   7]
 [ 44  92 153  72  39]
 [ 11  27  87 156 119]
 [ 11   9  24 105 251]]
```

**Fine-Tuning the model**

Fine-tuning is the process of optimizing a machine learning model or its preprocessing parameters to improve performance. It involves adjusting hyper parameters to achieve better accuracy, generalization, and overall model efficiency without overfitting the data.

Initially, the Logistic Regression model achieved an accuracy of **58%** using the default TF-IDF vectorization settings.

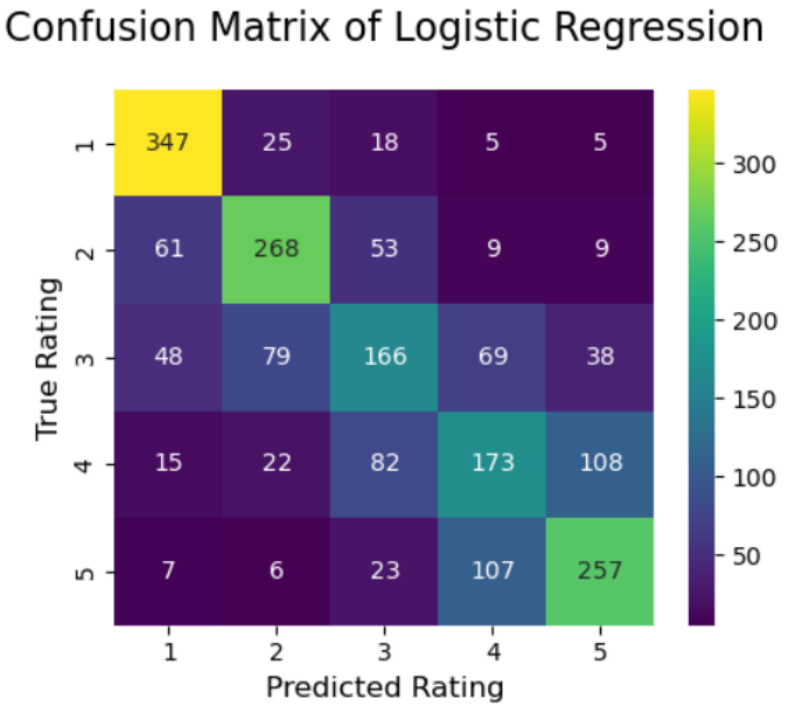
To improve performance, the **TF-IDF vectorizer parameters** were fine-tuned by adjusting:

- **max_features** → Increased vocabulary size to 7000 to capture more relevant words.
- **ngram_range** → Changed from (1,1) to (1,2) to include both single words and pairs of words (bigrams).
- **min_df** and **max_df** → Adjusted to remove extremely rare and overly common words, reducing noise.

After Tuning:

```
Accuracy of model :  0.611
Precision         :  0.6011059315850262
Recall            :  0.611
F1 Score          :  0.6045153746627362

Confusion Matrix :
 [[348  25  17   5   5]
 [ 55 277  54   9   5]
 [ 47  73 172  73  35]
 [ 15  19  90 170 106]
 [  6   6  27 106 255]]
```




**Inferences :**

- 58% of the reviews were correctly classified.
- Rating 1, 2 and 5 are best predicted ratings

Rating 1 : 339/400 is correct (mostly predicted correctly)
Rating 2 : 263/400 is correct
Rating 5 : 251/400 is correct

- Rating 3 and 4 is more misclassified
- Most of the misclassifications are 1 rating away. (for example rating 3 is predicted as 2 or 4).

## 10.2. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression. It works by finding the optimal boundary (called a hyperplane) that best separates data points of different classes. In text classification, it performs well

because it can handle high-dimensional data effectively using kernel functions.

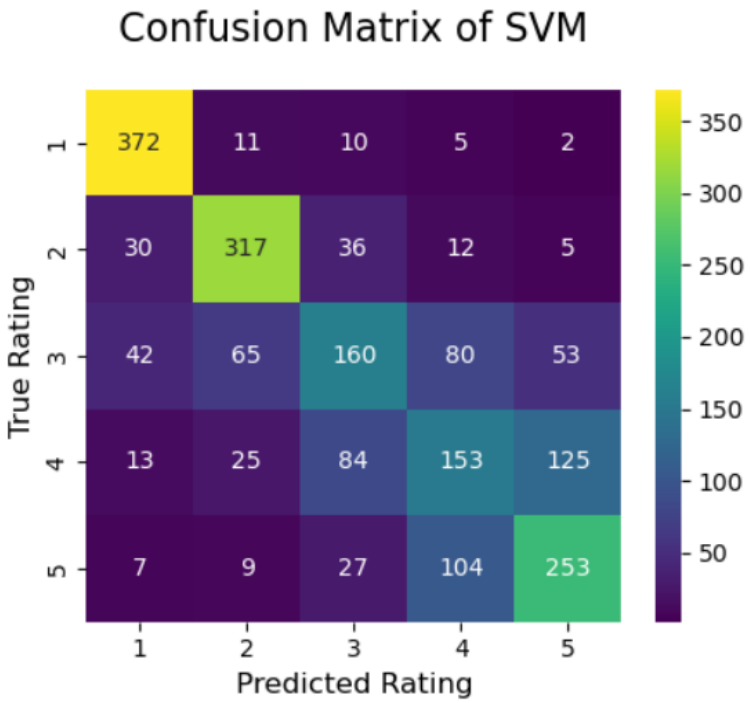
**Code:**

```
from sklearn.svm import LinearSVC
svm = LinearSVC(random_state=42)
svm.fit(x_train_vec, y_train)
y_pred_svm = svm.predict(x_test_vec)
```

**Accuracy of model :  0.6275**
**Precision  :  0.6117347437037477**
**Recall  :  0.6275**
**F1 Score  :  0.6167391351196915**

**Confusion Matrix :**

```
 [[372  11  10   5   2]
 [ 30 317  36  12   5]
 [ 42  65 160  80  53]
 [ 13  25  84 153 125]
 [  7   9  27 104 253]]
```




**Inferences:**

- The SVM model performs well on extreme ratings (1 and 5), where the sentiment polarity is clear and strong.
- The middle ratings (2, 3, and 4) show relatively lower accuracy due to overlapping linguistic patterns.

## Model Training Logic

The training process begins with splitting the preprocessed dataset into training and testing sets to evaluate the model's performance on unseen data. The text data is converted into numerical features using TF-IDF vectorization, capturing the importance of words in each review. The Logistic Regression model and SVM model is then fitted on the training data, learning the relationship between the input features and the target ratings. Once trained, the models can predict ratings for new reviews, and its performance is assessed using metrics such as Accuracy, Precision, Recall, F1-Score, and Confusion Matrix.

## Model Comparison

Two supervised machine learning models → Logistic Regression and Support Vector Machine (SVM) → were implemented and evaluated on the balanced dataset to determine the best-performing algorithm for the automated review rating prediction task.

Both models were trained using **TF-IDF vectorized text data** with the following configuration:

- max_features = 7000
- ngram_range = (1, 2) (unigrams and bigrams)
- min_df = 5, max_df = 0.9

## Performance Metrics:

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Logistic Regression | 0.611 | 0.601 | 0.611 | 0.605 |
| SVM | 0.6275 | 0.612 | 0.6275 | 0.617 |

## Analysis and Observations

- The SVM model outperformed Logistic Regression in all major metrics, showing a 1.6% improvement in overall accuracy and higher F1 score.

- In the confusion matrix, SVM displayed better prediction consistency across all rating classes, especially for the middle ratings (2, 3, and 4) where Logistic Regression tended to misclassify samples more frequently.
- Both models performed well on extreme ratings (1 and 5), but SVM showed better generalization by maintaining slightly higher recall for minority classes.
- The difference in averaging method (macro vs. weighted) highlights that SVM handled class-level variations more effectively without bias toward any specific rating class.

## Conclusion

Between the two models, SVM was selected as the preferred model for the balanced dataset because of its higher overall accuracy, better class-wise prediction stability, and robustness in handling textual data represented by TF-IDF features. Logistic Regression, while simpler and faster, exhibited slightly lower performance and weaker handling of middle rating classes.