CASE STUDY

Yelp Reviews

Rating Classification

Author: Sameer Hate
Date:11-07-2025

Table of Contents

Introduction

General

In today's digital ecosystem, user-generated reviews play a critical role in shaping consumer perception and business reputation. Yelp, one of the most widely used platforms for local business reviews, hosts millions of textual reviews ranging from highly positive to extremely negative sentiments. Accurately classifying these reviews into categories such as positive or negative sentiment is essential for businesses aiming to analyze customer feedback at scale and respond effectively.

This case study involves the development of a machine learning-based classification model that can automatically determine the sentiment of Yelp reviews based on their textual content. By training on a labeled dataset of reviews, the model learns to identify linguistic patterns and keywords associated with various sentiments. This enables efficient sentiment analysis, which is valuable for brand monitoring, customer experience optimization, and strategic decision-making.

Through natural language processing (NLP) techniques and classification algorithms, this project demonstrates how unstructured textual data can be transformed into actionable insights, ultimately empowering businesses to better understand and serve their customers.

Author's Note

This case study presented in this work has been independently completed by me. This project serves as a medium to deepen my understanding of Data Science concepts and enhance my practical skills in applying machine learning and data analysis techniques to real-world problems.It may be prone to errors but I try my best to eliminate them as much as I can. In case you identify any such error or have a more optimal technique to solve this problem, your feedback and suggestions are always welcome as this will eventually help me strengthen my concepts.

Problem Statement & Expected Results

Objective

The primary objective of this machine learning model is to accurately classify Yelp reviews based on their textual content into sentiment categories—such as positive(5 Stars) or negative(1 Star). This classification enables automated sentiment analysis, allowing businesses and analysts to efficiently process large volumes of user-generated reviews. By identifying the overall tone and sentiment of customer feedback, the model aims to support data-driven decision-making, enhance customer satisfaction, and improve brand reputation management.

Goals

- To preprocess and clean the raw Yelp review texts for effective feature extraction.
- To convert textual data into numerical representations using vectorization techniques.
- To build and train a machine learning classification model capable of distinguishing between positive and negative reviews.
- To evaluate the model's performance using key metrics such as accuracy, precision, recall, and F1-score

Expected Outcomes

The expected outcome of this project is a robust and reliable machine learning model that can accurately classify Yelp reviews as either positive or negative based on their textual content. The model should demonstrate high performance across standard evaluation metrics such as accuracy, precision, recall, and F1-score, indicating its effectiveness in real-world scenarios. Ultimately, the model will enable automated sentiment analysis of large-scale Yelp review datasets, providing valuable insights into customer satisfaction and supporting data-driven strategies for business improvement.

Dataset

Description

The dataset used for this study is a collection of Yelp reviews containing both textual content and associated metadata. Each record in the dataset represents a single review submitted by a user on the Yelp platform. The dataset primarily contains the following columns:

- Text: The raw review written by the user. This is the core feature used for sentiment classification.
- stars: The rating given by the user, typically ranging from 1 to 5 stars. This serves as the label for sentiment—lower ratings (e.g., 1 or 2) are typically considered negative, while higher ratings (e.g., 4 or 5) are considered positive.

The dataset includes 10,000 review samples, offering a balanced representation of both positive and negative messages. This labeled dataset is well-suited for supervised machine learning tasks, particularly classification, as it allows the model to learn patterns and characteristics associated with spam content based on historical data.

Dataset Link = Link

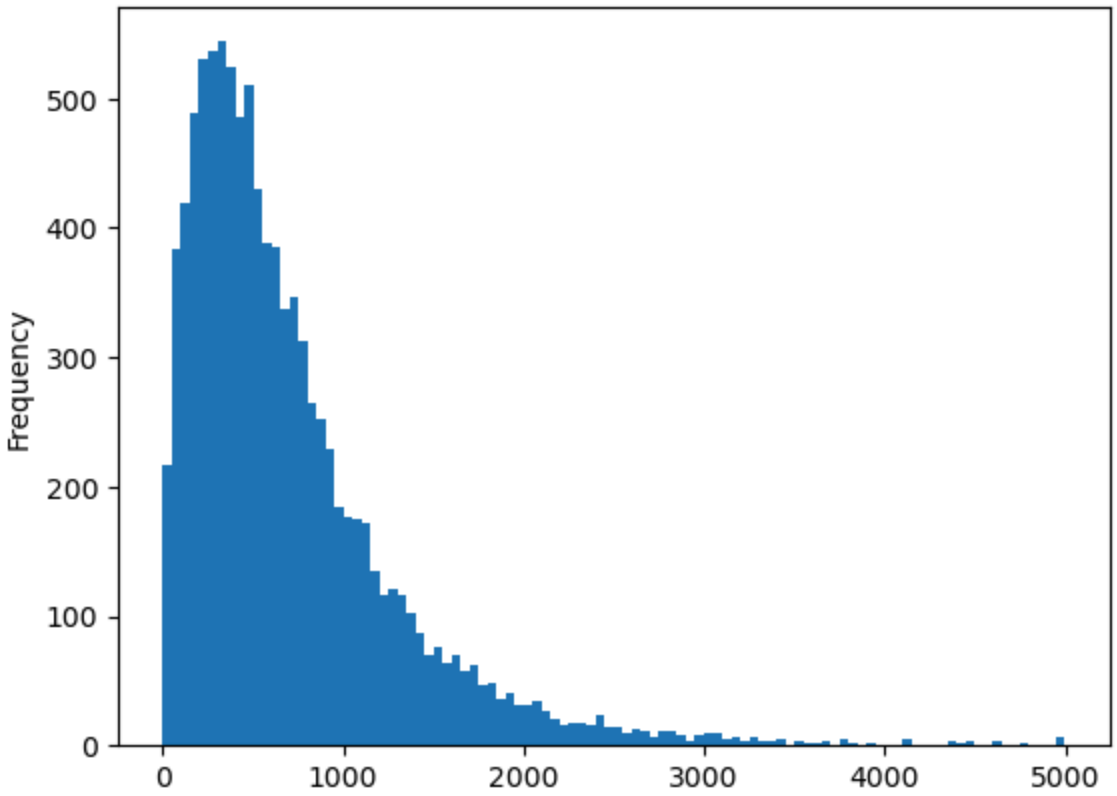
Approach

- #IMPORTING LIBRARIES: This code cell imports the fundamental libraries required for data analysis and visualization. The pandas library is used for handling and manipulating structured data in the form of dataframes, while numpy provides support for numerical operations. matplotlib.pyplot and seaborn are used for creating visualizations; matplotlib enables basic plotting, and seaborn builds on it to provide more attractive and informative statistical graphics. These libraries form the backbone of any data science workflow in Python.

- #IMPORTING THE DATASET: In this code cell, the Yelp reviews dataset is imported using the pandas library and stored in a DataFrame named yelp_df. The dataset is read from a CSV file titled "yelp.csv" using the read_csv() function. To gain an initial understanding of the data structure and its contents, the head(17) method is used, which displays the first 17 rows of the dataset. This helps in visually inspecting key features such as the review text, star ratings, and other relevant attributes, allowing for a clearer picture of how the data is organized before proceeding with further analysis or model development.

- #GETTING INFORMATION ABOUT THE DATASET: In this code cell, an overview of the Yelp dataset is obtained using two important pandas functions: describe() and info(). The describe() function generates a statistical summary of the numerical columns, including measures such as mean, standard deviation, minimum, maximum, and various percentiles. This helps in understanding the distribution and spread of the numerical data, particularly the stars column which is central to the review classification task. The info() function provides essential structural details about the dataset, including the number of entries, column names, data types, and the count of non-null values. This is useful for identifying missing data and confirming the types of each column before any preprocessing or analysis is carried out. Together, these functions offer a clear understanding of both the statistical and structural aspects of the dataset.

- #VISUALIZING THE DATASET: In this code cell, the dataset is being prepared for visualization by analyzing the length of each review. A new column named length is created by applying the len() function to the text column, which contains the review content. This column captures the number of characters in each review. Once the lengths are calculated, a histogram is plotted using the plot() function with 100 bins to visualize the distribution of review lengths across the dataset. This visualization helps in identifying whether longer or shorter reviews are more common, which can provide insights into user behavior and guide preprocessing or feature engineering decisions in the classification model.

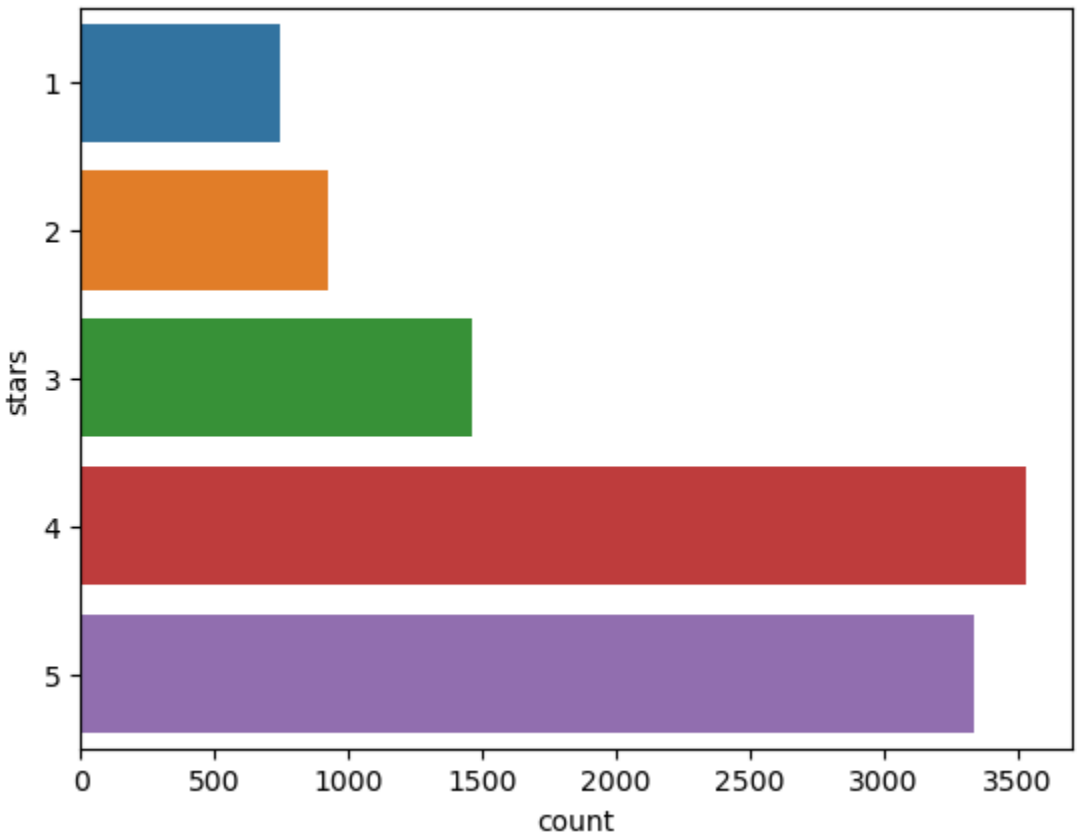


Frequency vs Length of Reviews

- #GETTING INFO ON THE LENGTH OF REVIEWS: In this code cell, the describe() function is used on the length column of the dataset to generate a summary of statistical information related to the length of Yelp reviews. This includes key metrics such as the total number of reviews, the mean (average) length, the standard deviation, and the minimum and maximum review lengths. Additionally, it provides the 25th, 50th (median), and 75th percentiles, which help in understanding how the review lengths are distributed across the dataset. This analysis is useful in gaining insights into the variability and nature of the textual data, and can guide preprocessing steps like truncating or padding text before feeding it into a machine learning model.

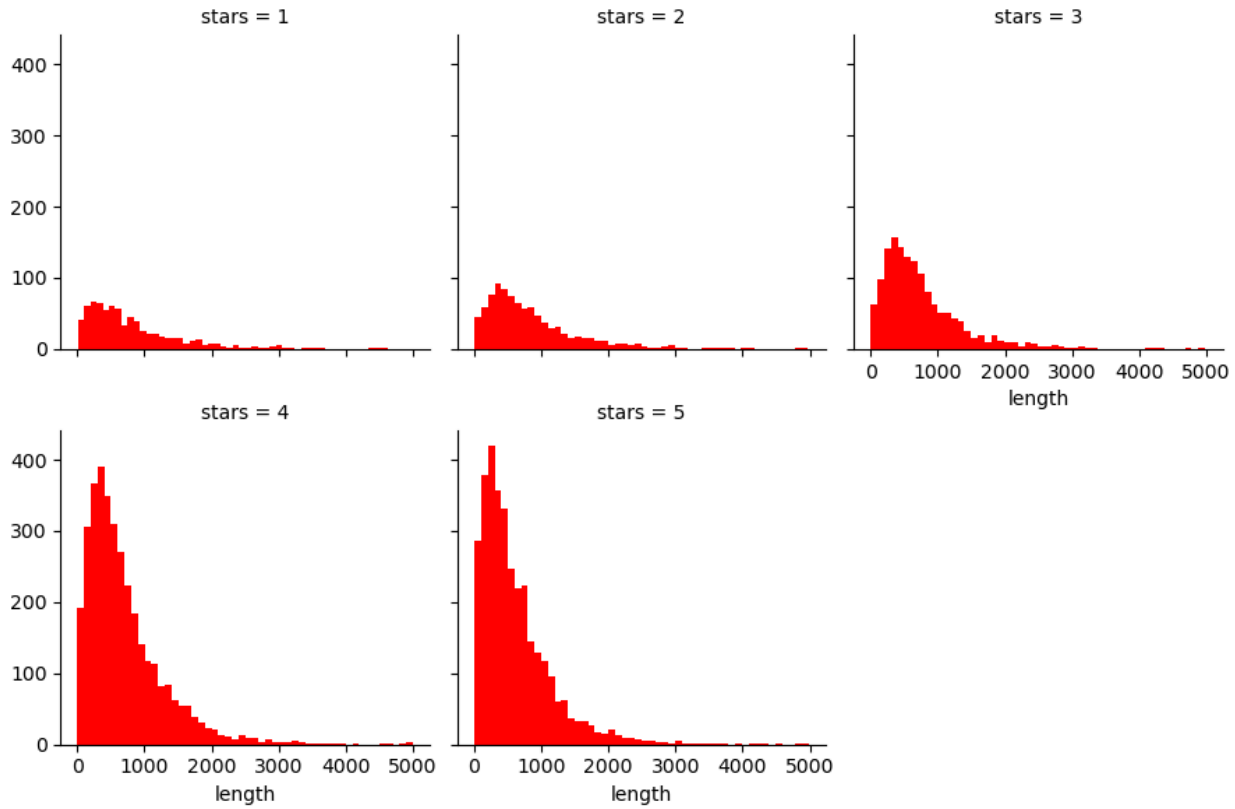
- count = 10000: There are 10,000 reviews in the dataset.
- mean = 710.74: On average, a Yelp review contains about 711 characters.
- std = 617.40: The standard deviation is 617, indicating a wide spread in review lengths — some are very short while others are quite long.
- min = 1: The shortest review is just 1 character long.
- 25% = 294: 25% of the reviews have 294 characters or fewer. This is the first quartile.
- 50% = 541.5: Half of the reviews have 541.5 characters or fewer. This is the median.
- 75% = 930: 75% of the reviews are 930 characters or fewer.
- max = 4997: The longest review in the dataset is 4,997 characters long.

- #VISUALIZING THE DATASET BASIS STARS RATING: This plot helps you understand the distribution of reviews across different star ratings (e.g., how many 1-star, 2-star, up to 5-star reviews are present). It's useful for checking class imbalance in your dataset — for example, if there are too many 5-star reviews and very few 1-star reviews, you may need to address this imbalance during model training.



Stars vs Count of Reviews

- #PLOTTING THE LENGTH OF REVIEWS BASIS STARS RATING: This code cell generates a series of histograms to visually explore how the length of Yelp reviews varies across different star ratings. Using Seaborn's FacetGrid, it creates separate subplots for each star category (from 1 to 5 stars), allowing for a clear side-by-side comparison. Each subplot displays a histogram that represents the distribution of review lengths for that specific star rating. The col_wrap=3 argument ensures that the plots are arranged neatly in rows of three. By mapping the length column (which was previously calculated as the number of characters in each review) to these plots, we can observe patterns such as whether longer or shorter reviews are more common within certain ratings. This visualization helps in understanding how detailed users tend to be when giving low or high ratings, which can further inform how review text might be used in classification models.



Individual Stars Rating vs Length of Reviews

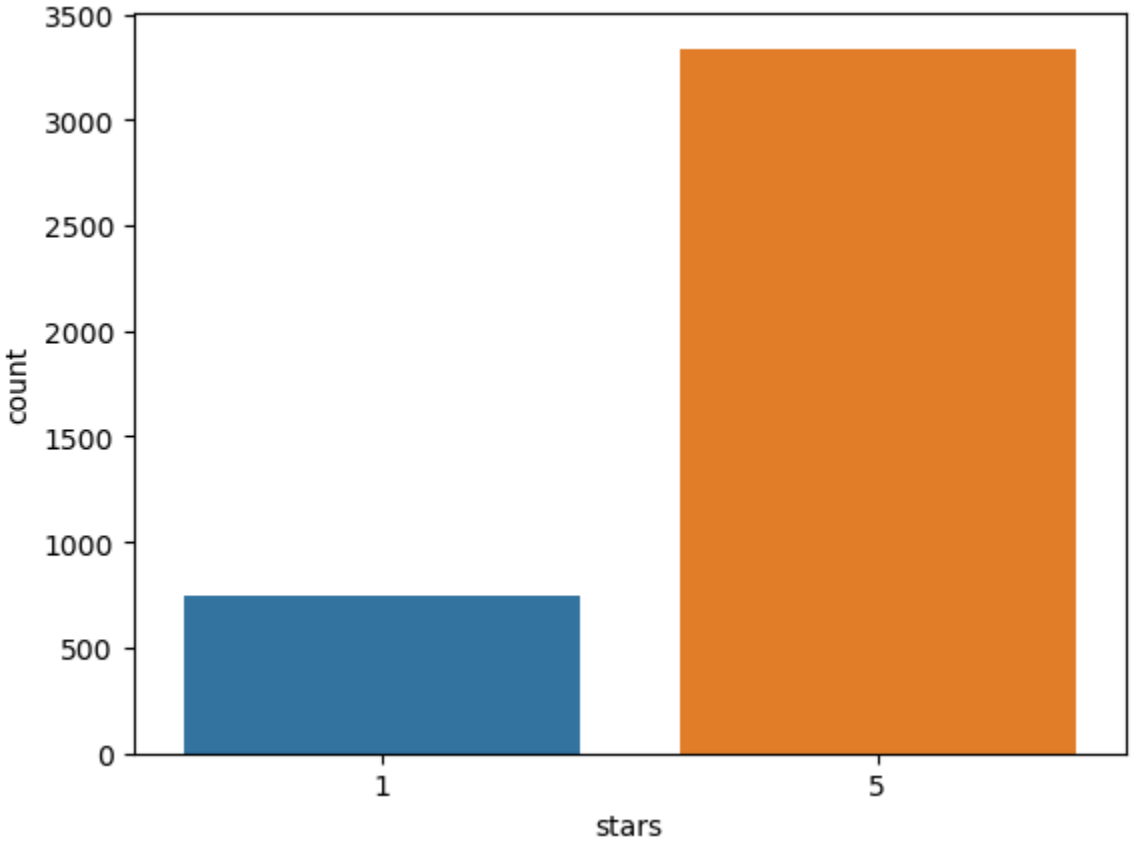
- #DIVIDING THE DATASET BASED ON STARS RATING: This code cell filters the original Yelp dataset to create two new subsets: one containing only the reviews with a 1-star rating (yelp_df_1) and another containing only the reviews with a 5-star rating (yelp_df_5). These two extremes are commonly chosen in sentiment analysis tasks, as 1-star reviews generally reflect highly negative sentiment, while 5-star reviews indicate highly positive sentiment. By isolating these reviews, the task of building a binary classification model (e.g., positive vs. negative sentiment) becomes more straightforward and reliable, as the sentiment polarity is clearer and more distinct at the extremes.

- #MERGING THE 1 STAR AND 5 STAR DATASET: This code cell combines the previously separated 1-star and 5-star review datasets into a single dataset named yelp_df_1_5 using the pd.concat() function. By merging these two subsets, the resulting dataset is prepared for binary sentiment classification, where 1-star reviews represent negative sentiment and 5-star reviews represent positive sentiment. This simplification allows for a clearer distinction in training a classification model, making it easier to teach the algorithm to differentiate between strongly negative and strongly positive reviews.

- #GETTING INFO ON THE MERGED DATASET: In this code cell, the .info() function is used to obtain a concise summary of the merged dataset yelp_df_1_5, which contains reviews with 1-star and 5-star ratings. The output provides essential information including the total number of entries, the number and names of columns, data types for each column, the count of non-null values in each column, and the memory usage of the dataset. This step is crucial for confirming that the dataset has been correctly merged and contains no missing values, thereby ensuring it is ready for further preprocessing and model development.

- #PERCENTAGE OF REVIEWS WITH 1 STAR AND 5 STAR: This code calculates and prints the percentage distribution of 1-star and 5-star reviews within the merged dataset yelp_df_1_5. It does so by dividing the number of 1-star reviews (len(yelp_df_1)) and 5-star reviews (len(yelp_df_5)) by the total number of reviews in the combined dataset (len(yelp_df_1_5)), and multiplying by 100 to express the result as a percentage. This step helps assess whether the dataset is balanced or imbalanced, which is important for ensuring fair and accurate model training and evaluation.

- #VISUALIZING THE MERGED DATASET: This code uses Seaborn's countplot function to visually represent the distribution of reviews by star rating within the merged dataset yelp_df_1_5, which contains only 1-star and 5-star reviews. The x='stars' parameter specifies that the x-axis will show the star ratings (1 and 5), while the y-axis will show the count of reviews for each rating. This bar chart helps you quickly assess if the dataset is balanced or skewed toward either 1-star or 5-star reviews—an important insight before training a binary classification model.



Count of Rating vs Stars
for Merged Dataset

- #CLEANING THE TEXT: This code cell defines a text cleaning function called message_cleaning, which is a crucial step in preparing textual data for Natural Language Processing (NLP) tasks like sentiment classification.
    - Here's what the function does in simple terms:
        - Removes Punctuation: It iterates through each character in the review text and filters out punctuation using Python's string.punctuation. This helps reduce noise in the data.
        - Removes Stopwords: After punctuation is removed, the remaining text is split into words. Common, non-informative words (like "the", "and", "is", etc.)—called stopwords—are removed using the NLTK library's English stopword list. This improves the quality of features by keeping only the most meaningful words.
        - Returns a Cleaned List of Words: The final cleaned text is returned as a list of words, ready for further processing (e.g., vectorization).
        - Although the last line that applies the function to the 'text' column is commented out, if executed, it would generate a cleaned version of the reviews in the yelp_df_1_5 dataset.

- #APPLYING COUNT VECTORIZER: In this code cell, Count Vectorization is applied to the text data in order to convert the cleaned Yelp reviews into a numerical format suitable for machine learning models. First, the CountVectorizer class from sklearn.feature_extraction.text is imported, which transforms a collection of text documents into a matrix of token (word) counts. A custom text-cleaning function, message_cleaning, is passed as the analyzer to ensure that punctuation and common stopwords are removed before vectorization. The vectorizer is then fitted and applied to the 1-star and 5-star reviews in the yelp_df_1_5 DataFrame. This process results in a sparse matrix (yelp_count_vectorizer) where each row corresponds to a review and each column corresponds to a word from the vocabulary, with the values representing the frequency of the words. This matrix becomes the input feature set for training a classification model.

- #IMPORTANT WORDS: In this code cell, key information about the text data after vectorization is retrieved and displayed. First, vectorizer.get_feature_names_out() prints the complete list of vocabulary words (features) identified from the dataset—essentially the unique words present in the cleaned text reviews. Next, yelp_count_vectorizer.toarray() converts the sparse matrix into a dense NumPy array, showing the actual frequency counts of these words across all reviews, where each row represents a review and each column represents a word from the vocabulary. Finally, yelp_count_vectorizer.shape prints the dimensions of this matrix, indicating how many reviews and unique words are present in the processed dataset. This step helps confirm that the text data has been successfully transformed into a numerical format ready for model training.

- #DIVIDING THE DATASET INTO TRAIN AND TEST: In this code cell, the dataset is split into training and testing sets to prepare for model building. Using the train_test_split function from sklearn.model_selection, the vectorized review data (yelp_count_vectorizer) is divided into X_train and X_test, while the corresponding star ratings (yelp_df_1_5['stars']) are split into y_train and y_test. The test_size=0.2 parameter indicates that 20% of the data will be used for testing, and the remaining 80% will be used to train the model. This ensures that the model is trained on one portion of the data and evaluated on unseen data to assess its performance objectively.

- #TRAINING THE MODEL: In this code cell, a Naive Bayes classification model is trained on the training dataset. Specifically, the MultinomialNB algorithm from sklearn.naive_bayes is used, which is well-suited for text classification tasks involving discrete features such as word counts. The model is instantiated with nb = MultinomialNB() and then trained using the fit() method on X_train (features) and y_train (labels). This allows the model to learn patterns in the word distributions that are characteristic of 1-star and 5-star Yelp reviews.

- #EVALUATING THE MODEL: This code cell evaluates the performance of the trained Naive Bayes model using standard classification metrics. Predictions are first made for both the training and test datasets using the predict() method. The classification_report() function from sklearn.metrics is then used to print detailed evaluation summaries, which include precision, recall, F1-score, and support for each class (1-star and 5-star reviews). The results are printed separately for the training and test sets, allowing comparison between how well the model performs on seen vs. unseen data. This helps assess potential overfitting or underfitting in the model.

- #PLOTTING THE CONFUSION MATRIX - TRAINING/TEST SET: This code evaluates the performance of the trained Naive Bayes classification model on the training dataset by calculating its accuracy and visualizing the confusion matrix.The accuracy_score function computes the overall accuracy — i.e., the percentage of reviews correctly predicted by the model. This is printed as a simple numeric value. Following that, the confusion_matrix is visualized using a Seaborn heatmap, where the rows represent the actual star ratings (1 or 5), and the columns represent the predicted star ratings.Each cell in the matrix shows the number of reviews falling into each category. The diagonal values (top-left and bottom-right) indicate correct predictions, while any values off the diagonal would signify misclassifications. By annotating the heatmap with actual counts (annot=True) and formatting them as integers (fmt='d'), this visualization offers a clear snapshot of how well the model distinguishes between 1-star and 5-star reviews in the training data.

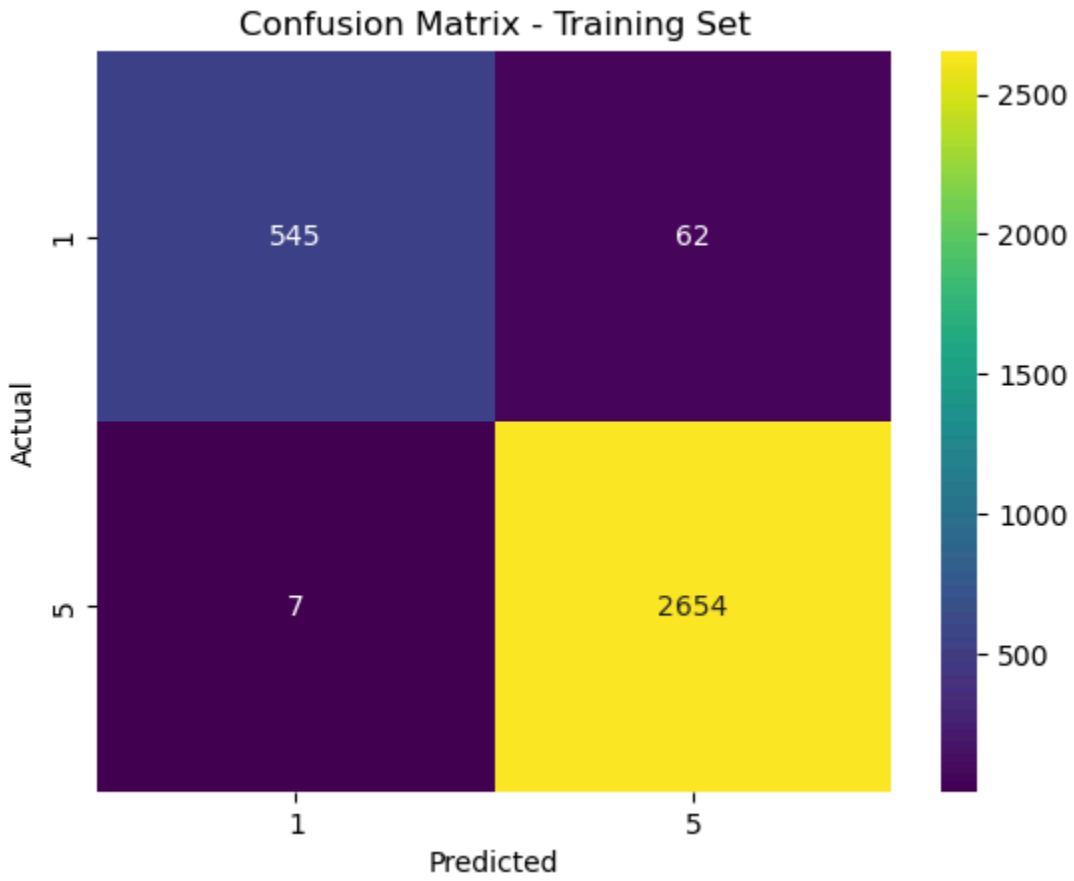
- #PREDICTING A NEW REVIEW: The code provided is used to test the sentiment classification model on a new user-defined review. In this case, the review text is “This place is really disgusting. Never go here.” which is clearly negative in tone. The review is first transformed into a numerical feature vector using the same CountVectorizer that was fitted on the training data. This ensures that the new input is processed in the same format as the data used to train the model. The vectorized input is then passed to the trained Naive Bayes classifier (nb), which predicts whether the review is a 1-star (negative) or 5-star (positive) rating. Based on the model’s output, a conditional statement prints the classification result in a user-friendly format. This approach effectively demonstrates how the model can be used in real-world scenarios to classify unseen customer reviews into predefined sentiment categories.

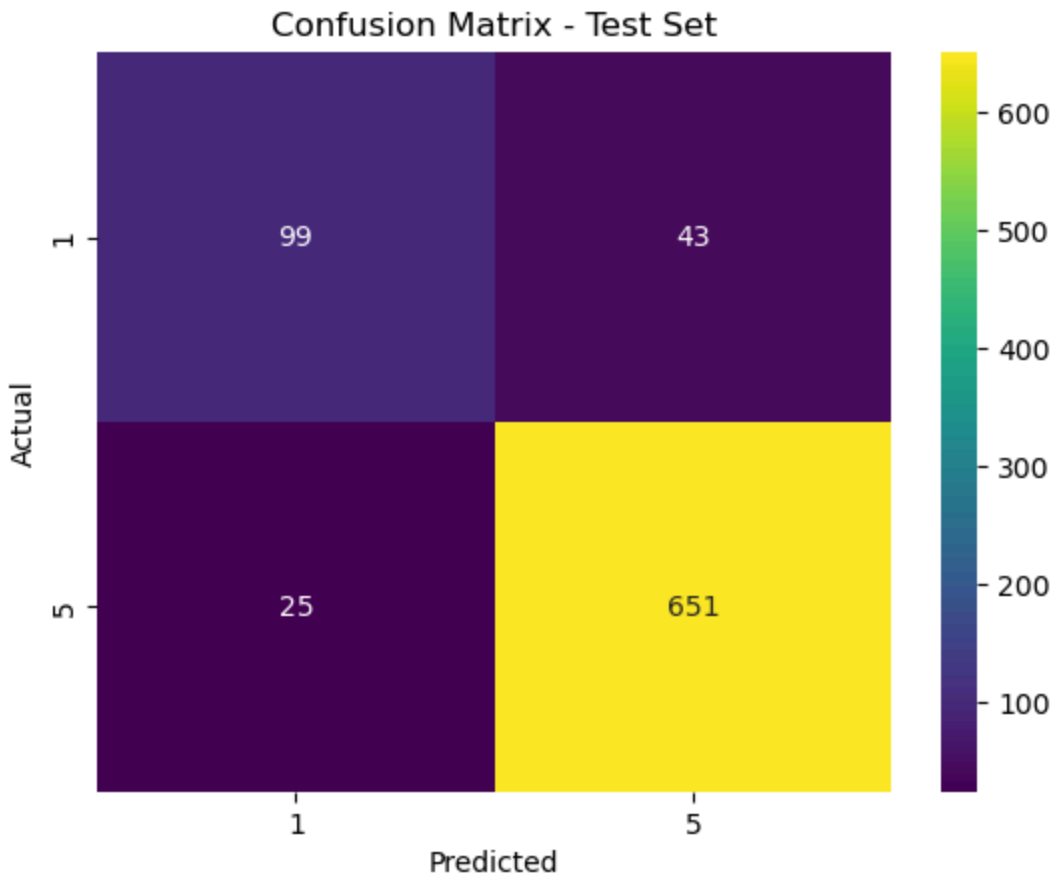
Results

Overall

The Naive Bayes classification model achieved an accuracy of approximately **91.69%** on the test dataset, indicating strong generalization performance on unseen data. This means that the model was able to correctly classify about 9 out of 10 reviews as either 1-star (negative) or 5-star (positive) based on their text content.

Confusion Matrix





Prediction

Review: ‘This place is really disgusting. Never go here.’
Output: The review is classified as: 1-Star (Negative Review)

References & Glossary

- GitHub Repository - Click Here
- Email Spam Detection(Similar to Yelp Review Classification) = Click Here
- Bins = In a histogram they are like buckets that group similar values together so we can count how often each group appears.
    - Imagine you are sorting socks by their lengths:
        - You create buckets like: 0–10 cm, 11–20 cm, 21–30 cm, and so on.
        - You then count how many socks fall into each bucket.
        - This helps you understand which sock length is most common.
- Classification Report Terminologies:
    - Class 1 (1-star reviews):
        - Precision (0.99): Out of all the reviews the model said were 1-star, 99% were actually 1-star.Analogy: Like a food critic who rarely mislabels a good restaurant as bad.
        - Recall (0.90): Out of all the actual 1-star reviews, the model correctly identified 90% of them.Analogy: But the same critic misses 10% of truly bad restaurants.
        - F1-score (0.94): This balances precision and recall. A score of 1.0 is perfect. So, 94% is a strong result.
        - Support (607): There were 607 actual 1-star reviews in the training data.
        - Macro avg: Treats both classes equally. Average of precision, recall, F1-score for both 1-star and 5-star.
        - Weighted avg: Takes class imbalance into account (more 5-star reviews than 1-star), and gives more weight to larger classes when averaging.
-

****