



# Programming for Data Analytics 1

POE

## **Pipeline and Text Data**

PDAN8411

Mukeba Mbunda Eliezer

ST10486340

PDDA0801 VCWCCR Term1 GRO1

Varsity College Cape Town Newlands

LECTURER NAME:

Dr Rudolf Holzhausen

DUE:

June 27, 2025

## Table of Contents

|  |           |
|--|-----------|
| <b>1. Introduction.....</b>                      | <b>4</b>  |
| <b>2. Exploratory Data Analysis (EDA) .....</b>  | <b>8</b>  |
| <b>3. Data cleaning &amp; Preprocessing.....</b> | <b>13</b> |
| <b>4. Feature selection.....</b>                 | <b>18</b> |
| <b>5. Train model: Logistic regression .....</b> | <b>20</b> |
| <b>6. Evaluate model.....</b>                    | <b>21</b> |
| <b>7. Conclusion .....</b>                       | <b>24</b> |

# Table of figures

|  |    |
|--|----|
| Figure 1 : Import libraries.....                                     | 8  |
| Figure 2: Load dataset .....   | 8  |
| Figure 3: Information dataset.....                                   | 9  |
| Figure 4 : Dimension of the dataset .....                            | 9  |
| Figure 5: Remove unnecessary column .....                            | 9  |
| Figure 6: Check for missing values .....                             | 9  |
| Figure 7: Check and remove duplicate values .....                    | 10 |
| Figure 8 : Bar plot distribution(Positive/ Negative).....            | 10 |
| Figure 9: Pie chart distribution (Positive, Negative) .....          | 11 |
| Figure 10: Histogram distribution of review lengths .....            | 11 |
| Figure 11: Boxplot distribution of review length by sentiment .....  | 12 |
| Figure 12: Reviews column.....                                       | 13 |
| Figure 13: Cleaning functions .....                                  | 13 |
| Figure 14: Preprocess function .....                                 | 14 |
| Figure 15: Reviews column after preprocessing .....                  | 14 |
| Figure 16: Function to count 15 most common words .....              | 14 |
| Figure 17: Function to find the 15 most common negative words .....  | 15 |
| Figure 18: Word cloud for negative reviews .....                     | 15 |
| Figure 19: Treemap of 15 most common words in negative reviews ..... | 16 |
| Figure 20: Convert tokens into single string.....                    | 18 |
| Figure 21: Create pipeline with TF-IDF and apply chi-square .....    | 18 |
| Figure 22: Find the best hyperparameters .....                       | 20 |
| Figure 23: Train model using selected hyperparameters.....           | 20 |
| Figure 24: Accuracy .....  | 21 |
| Figure 25: Classification report .....                               | 21 |
| Figure 26: Roc curve .....   | 23 |

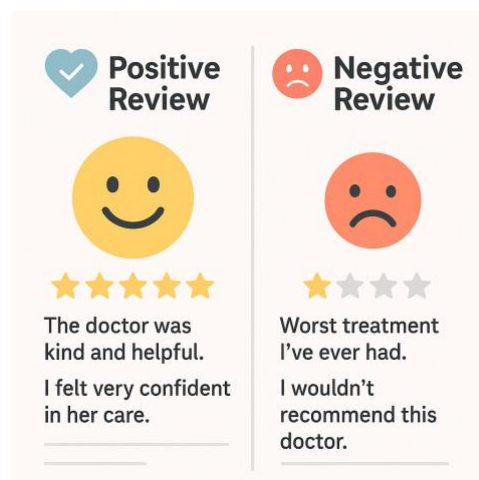
# 1. Introduction

We have been asked by a medical aid scheme in South Africa to conduct a text analysis to help them identify the areas of concern mentioned in the few negative reviews they are receiving on Hello Peter, and to gain a general sentiment of how their customer base is currently experiencing their services. First, we need to understand what sentiment analysis is. According to Nzonga (2023), sentiment analysis is a subfield of natural language processing (NLP) that involves categorizing text data into positive, negative, or neutral sentiment. Similarly, Tonkin (2016) explains that sentiment analysis is the classification of texts according to the emotions they convey, such as positive, negative, or neutral. As explained by Pang and Lee (2004), the neutral class in sentiment analysis is usually ignored because it is considered less informative. That is why, in this project, we are going to focus on binary classification using a dataset that contains only positive and negative sentiment.

We have two major points to focus on during this project:

- a. Identify the areas of concern in the negative reviews
- b. analyzing customer sentiments.

To address this, we have chosen a doctor reviews dataset. We were unable to find a dataset that specifically focuses on reviews about medical aid services. However, considering that medical aid providers work closely with doctors (momentum,[s.a]), we decided to focus on patient feedback related to doctors.



The doctor reviews dataset was chosen because it provides textual feedback from patients regarding their medical consultations and offers rich insight into how patients perceive their care, making this dataset suitable for sentiment analysis.

#### a. Suitability

The dataset contains the following columns:

- **Unnamed 0** :This is an integer feature that is unique for each entry in the dataset. It does not carry any meaning related to the review content or sentiment.
- **reviews** : This is an unstructured text data feature (object type) that contains written feedback from patients regarding their medical consultations. These text reviews reflect their opinions, feelings, attitude, emotions, and experiences([Parr, 2024]). They play a key role in sentiment analysis, as they help determine whether the overall sentiment expressed by the patient is positive, negative or neutral ([Gillis, 2024]).
- **labels**: This is an integer feature that indicates whether a text review is 0 (negative) or 1 (positive). It serves as the target variable when predicting the sentiment of new reviews. Labeled data is crucial for supervised machine learning models ([Kanade, 2022]).
- **tag** : This is a categorical feature that provides a human-readable form of the sentiment in a review, indicating whether it is positive or negative.

Based on the structure of the dataset, it is well-suited for this case study, which requires conducting a text analysis as it contains important features such as reviews and labels.

#### b. Quality

The dataset contains only 143 entries. It is well-structured, clean, and clearly labeled, which makes it particularly suitable for text analysis. Each entry includes a patient review and a corresponding sentiment label, presented in both binary and human-readable form, allowing for the effective application of supervised machine learning techniques. Although the dataset is relatively small, its quality makes up for the size. There are no major

inconsistencies or missing values, which reduces the need for extensive preprocessing. Its organized structure, with almost an equal number of negative and positive reviews, along with its labeled format, makes it ideal for this project.

Link to the dataset : <https://www.kaggle.com/datasets/avasaralasaipavan/doctor-review-dataset-has-reviews-on-doctors>

To conduct the analysis, the TF-IDF + Logistic Regression model was selected for text analysis, first identifying broad areas of concern and then analyzing customer sentiments.

TF-IDF was selected for its ability to convert textual data into numerical representations. Since machine learning algorithms cannot understand raw text, we needed a method to transform the text data into a numeric format that could be processed effectively. TF-IDF (Term Frequency–Inverse Document Frequency) is a technique used to convert text data into numerical representations (feature vectors). The Term Frequency (TF) reflects how often a term appears in a document, while the Inverse Document Frequency (IDF) indicates how rare the term is across the entire corpus (Simha, 2021). By combining the two, TF-IDF calculates the numerical weight of each word based on its frequency in a given document and adjusts for how often it appears in all documents. As a result, it serves as an effective feature extraction method for transforming text data into vectors that can be used as input for machine learning models for classification, such as logistic regression (CrawlSpider, 2024).

Unlike Bag of Words, which simply counts the frequency of words in a document without considering inverse document frequency (IDF), or Word2Vec, which uses a shallow two-layer neural network to generate word vectors but often requires additional steps to produce a single vector representation, TF-IDF is simpler, computationally cheap, and more efficient (CrawlSpider, 2024). In contrast, advanced models like BERT, which use transformer-based machine learning techniques to capture semantic meaning and context, are much more computationally expensive. This is due to their complex architecture and large number of parameters, making them less suitable for small datasets or resource-constrained environments (CrawlSpider, 2024).

Logistic regression was chosen as the classification algorithm due to its proven effectiveness on small labeled datasets and its suitability for binary classification tasks, such as classifying positive, neutral and negative sentiments (GeeksforGeeks, 2025). Its simplicity ensures faster training and evaluation, while reducing the risk of overfitting, which can be problematic with more complex models based on the small dataset we have. More importantly, logistic regression offers simple interpretation and does not require a lot of data to produce accurate results (Ashmed, 2024;Indeed Team , 2025).

Naive Bayes is another popular text classification algorithm that performs well in many cases. However, it relies on the strong assumption that all features are independent of each other, which is rarely true in natural language (Turning, 2022). This assumption can result in less accurate performance than logistic regression, which does not rely on this assumption (Kavya, 2023). SVMs are also strong classifiers and are often used in text classification. However, they require more computational resources and are more difficult to understand and interpret than Logistic Regression, particularly with large feature spaces like those created by TF-IDF (Bassey, 2019).

In conclusion, the TF-IDF + Logistic Regression model was the ideal choice for this case study because it is fast, works well with small data, and offers clear and explainable results. While deep learning, Naive Bayes, and SVMs are useful in many NLP tasks, they are less suitable for small datasets.

In this project, we perform exploratory data analysis (EDA), clean and preprocess the data, apply feature selection, train and test a logistic regression model, and finally evaluate its performance.

## 2. Exploratory Data Analysis (EDA)

In this section, we go through several important steps to prepare the data for analysis. First, we import the necessary libraries and load the dataset. We then inspect its dimensions, data types, and structure. Unnecessary columns are removed to simplify the dataset, followed by checks for missing values and duplicated entries to ensure data quality. Finally, we explore the class distribution using various visualizations such as bar plots, pie charts, histograms, and boxplots.

**Step 1:** import the necessary libraries and load the dataset

```
import numpy as np # Numerical operations
import pandas as pd # Data manipulation
import matplotlib.pyplot as plt # Data visualisation
import seaborn as sns # Advanced visualisations
import plotly.express as px # High-level API for creating figures
%matplotlib inline
import warnings # Set plots to appear under the code cell
warnings.filterwarnings('ignore') # Ignore warnings messages for cleaning output
from sklearn.model_selection import train_test_split # Split dataset into training and testing sets
from sklearn.pipeline import Pipeline # Create a machine learning pipeline
#from sklearn.decomposition import PCA # Reduce dimensionality of data using Principal Component Analysis
from sklearn.linear_model import LogisticRegression # Perform classification using logistic regression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix # Metrics to evaluate model performance
import re # Regular expressions for text processing
import nltk # Natural Language Toolkit for text processing
from nltk.corpus import stopwords # Stopwords for removing common words
from nltk.stem import WordNetLemmatizer # Lemmatizer for reducing words to their base form
from collections import Counter # Import Counter from collections module to count occurrences of words
nltk.download('stopwords') # Download stopwords from NLTK
nltk.download('wordnet') # Download WordNet for lemmatization
nltk.download('punkt') # Download Punkt tokenizer for sentence splitting
nltk.download('punkt_tab') # Download Punkt tokenizer for sentence splitting (tab version)
from wordcloud import WordCloud # Import WordCloud for generating word clouds
import squarify # Import squarify for creating treemaps
from sklearn.feature_extraction.text import TfidfVectorizer # Convert a collection of raw documents to a matrix of TF-IDF features
from sklearn.feature_selection import SelectKBest, chi2 # Feature selection using chi-squared test
from sklearn.model_selection import GridSearchCV # Grid search for hyperparameter tuning
from sklearn.metrics import roc_curve, auc # Import roc_curve and auc for ROC curve and AUC calculation
```

Figure 1 : Import libraries

```
# Load data and display the first 5 rows.
# Encoding used to read the CSV file, which is necessary for handling special characters.
df = pd.read_csv('doctorReviews.csv', encoding='ISO-8859-1') # Read the CSV file into a DataFrame
pd.set_option('display.max_colwidth', 200) # Set maximum column width for display
df.head(5) # Display the first 5 rows of the DataFrame
```

✓ 2.2s Python

|   | Unnamed: 0 |  | reviews | labels | tag      |
|---|------------|--|---------|--------|----------|
| 0 | 93         | he explained initially that it takes 4-5 sittings and its total treatment cost is is about 10000 rupees but the total treatment he charged 22250 rupees including medicines the estimation of treatment... |         | 0      | negative |
| 1 | 33         | great dr definitely recommend he recommends less medicine and explain everything and provide solutions   |         | 1      | positive |
| 2 | 129        | doctor came and spent 9 seconds and recommended for nose throat endoscopy & some hearing test it was night 10 and hearing test doctor is not available does it make sense? charges for endoscopy is 2...   |         | 0      | negative |
|   |            | I am completely satisfied with the consultation I have been having acute severe bronchitis and acute laryngitis for the last 20 days which I was not   |         |        |          |
|   |            | on post course of days I'm recommend people term   |         |        |          |

Figure 2: Load dataset



## Step 2: Inspect the dimensions, data types, and general structure of the dataset

```
# Display the shape of the DataFrame, which gives the number of rows and columns.
df.shape # Output the shape of the DataFrame

✓ 0.0s

(143, 4)
```

Figure 4 : Dimension of the dataset

```
# Display information about the DataFrame, including data types and non-null counts.
df.info() # Output the information of the DataFrame

✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 143 entries, 0 to 142
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype  
---  -
0   Unnamed: 0    143 non-null   int64  
1   reviews      143 non-null   object  
2   labels        143 non-null   int64  
3   tag           143 non-null   object  
dtypes: int64(2), object(2)
memory usage: 4.6+ KB
```

Figure 3: Information dataset

## Step 3: Remove unnecessary columns, check for missing values, and duplicate entries

```
df_cleaned = df.drop(['Unnamed: 0', 'tag'], axis=1) # Drop unnecessary columns from the DataFrame

✓ 0.0s
```

Figure 5: Remove unnecessary column

```
# Count the missing values in the dataset
df_cleaned.isnull().sum() # Output the count of missing values in each column

✓ 0.0s

reviews    0
labels     0
dtype: int64
```

Figure 6: Check for missing values

```
# Count the duplicated row
df_cleaned.duplicated().sum()
✓ 0.0s

1

One duplicated value was found. Let's remove that row

df_cleaned = df_cleaned.drop_duplicates() # Remove duplicate rows from the DataFrame
✓ 0.0s
```

Figure 7: Check and remove duplicate values

**Step 4:** Class distribution using various visualizations such as bar plots, pie charts, histograms, and boxplots.

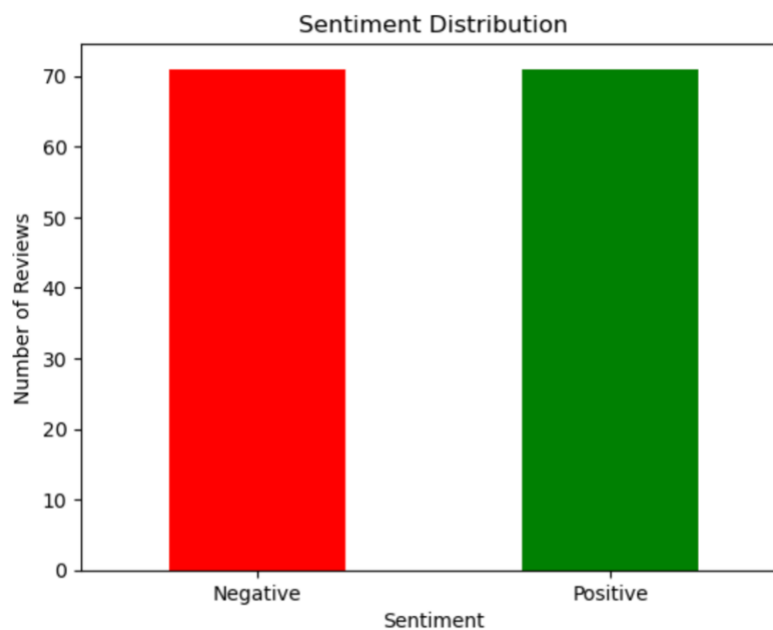


Figure 8 : Bar plot distribution(Positive/ Negative)

This bar plot distribution shows that the number of negative and positive reviews in the dataset is almost the same.

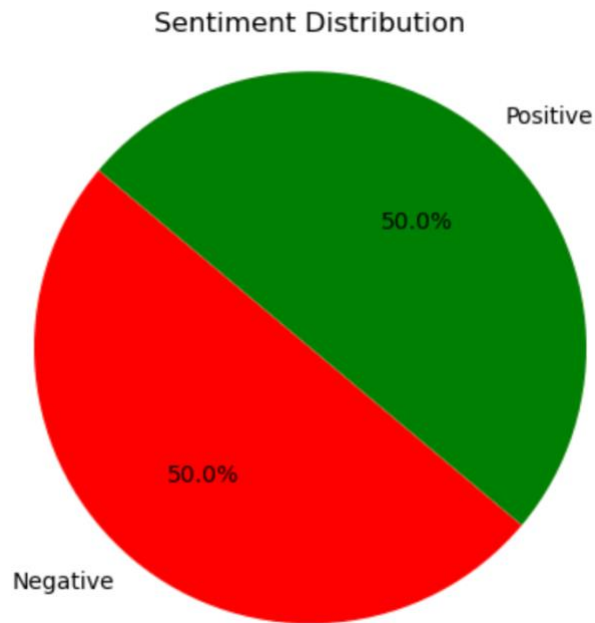


Figure 9: Pie chart distribution (Positive, Negative)

This pie chart shows that both negative and positive reviews, each representing approximately 50% of the total, indicate a balanced distribution of sentiments in the dataset. This balance is important for training a sentiment analysis model, as it helps prevent bias toward one class.

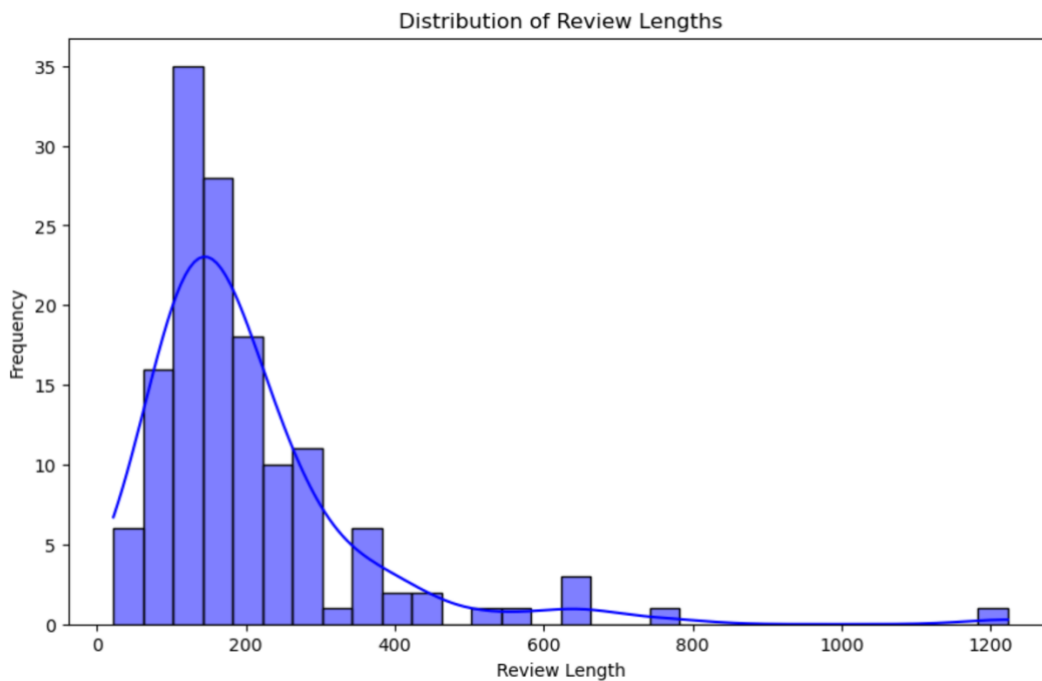


Figure 10: Histogram distribution of review lengths

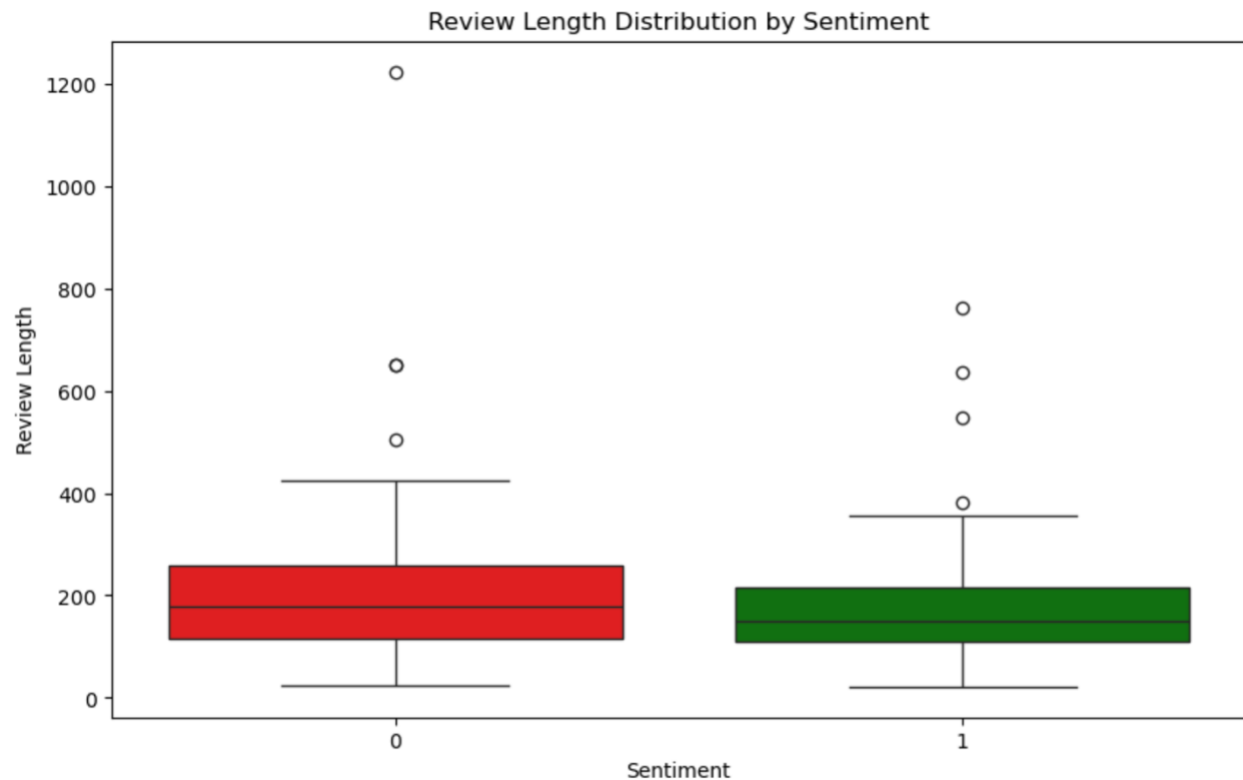


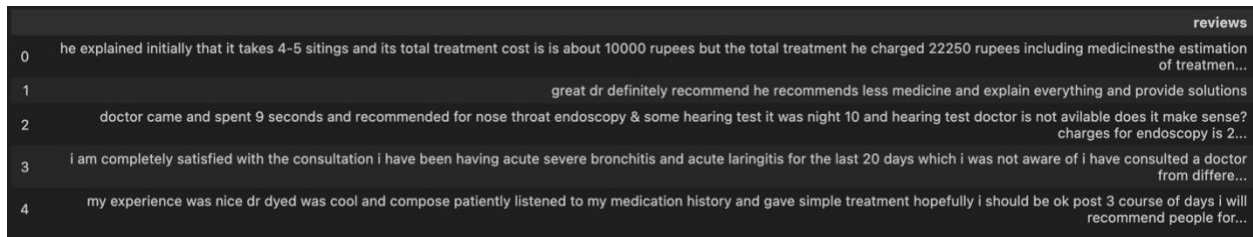
Figure 11: Boxplot distribution of review length by sentiment

The distribution of review lengths is right-skewed, indicating that most reviews are relatively short, with a few significantly longer ones. At this stage, we will not attempt to normalize the distribution. Both negative and positive reviews contain outliers in terms of length, which can be addressed later during the data preprocessing stage. Interestingly, we also observe that negative sentiment reviews tend to be longer than positive ones, possibly reflecting more detailed expressions of dissatisfaction.

### 3. Data cleaning & Preprocessing

After visualizing our dataset and gaining initial insights, we now proceed with text preprocessing as part of the Natural Language Processing (NLP) pipeline. To prepare the text data for analysis, we implement a series of cleaning functions. These functions convert all text to lowercase, remove special characters, and tokenize the text into individual words. We also remove stopwords, filter out very short and excessively long words, and perform lemmatization to reduce words to their root forms. This preprocessing step is crucial for ensuring that the data is clean, consistent, and ready for effective feature extraction and model training.

#### Step 1: Look at the reviews column



|   | reviews   |
|---|---|
| 0 | he explained initially that it takes 4-5 sittings and its total treatment cost is is about 10000 rupees but the total treatment he charged 22250 rupees including medicine the estimation of treatment... |
| 1 | great dr definitely recommend he recommends less medicine and explain everything and provide solutions  |
| 2 | doctor came and spent 9 seconds and recommended for nose throat endoscopy & some hearing test it was night 10 and hearing test doctor is not available does it make sense? charges for endoscopy is 2...  |
| 3 | i am completely satisfied with the consultation i have been having acute severe bronchitis and acute laringitis for the last 20 days which i was not aware of i have consulted a doctor from differe...   |
| 4 | my experience was nice dr dyed was cool and compose patiently listened to my medication history and gave simple treatment hopefully i should be ok post 3 course of days i will recommend people for...   |

Figure 12: Reviews column

#### Step 2: Cleaning functions

```
# create function to convert reviews column to lowercase
> def to_lowercase(text): --

# Function to remove special characters and numbers from the reviews column
> def remove_special_characters(text): --

# Function to tokenize the reviews column into words
> def tokenize_text(text): --

# Function to remove stopwords from the reviews column
> def remove_stopwords(words): --

# Function to remove short words (length <= 2) from the reviews column
> def remove_short_words(words): --

# Function to remove long words (length >= 20) from the reviews column
> def remove_long_words(words): --

# Function to lemmatize the words in the reviews column
> def lemmatize_words(words): --
```

Figure 13: Cleaning functions

### Step 3: Clean the reviews column and display the first five rows

```
# Function to preprocess the reviews column
def preprocess_reviews(text):
    text = to_lowercase(text) # Convert reviews to lowercase
    text = remove_special_characters(text) # Remove special characters and numbers
    text = tokenize_text(text) # Tokenize the reviews into words
    text = remove_stopwords(text) # Remove stopwords
    text = remove_short_words(text) # Remove short words
    text = remove_long_words(text) # Remove long words
    text = lemmatize_words(text) # Lemmatize the words
    return text # Return the preprocessed text

# Preprocess the text reviews in the DataFrame
df_cleaned['reviews'] = df_cleaned['reviews'].apply(preprocess_reviews)
```

Figure 14: Preprocess function

|   | reviews  | labels | review_length |
|---|--|--------|---------------|
| 0 | [explained, initially, take, sitings, total, treatment, cost, rupee, total, treatment, charged, rupee, including, medicines, the, estimation, treatment, false]  | 0      | 206           |
| 1 | [great, definitely, recommend, recommends, less, medicine, explain, everything, provide, solution]   | 1      | 102           |
| 2 | [doctor, came, spent, second, recommended, nose, throat, endoscopy, hearing, test, night, hearing, test, doctor, avilable, make, sense, charge, endoscopy, doctor, going, perform, request, hospital...] | 0      | 651           |
| 3 | [completely, satisfied, consultation, acute, severe, bronchitis, acute, laringitis, last, day, aware, consulted, doctor, different, hospital, earlier, wasnt, advised, test, xray, doctor, didnt, ga...] | 1      | 637           |
| 4 | [experience, nice, dyed, cool, compose, patiently, listened, medication, history, gave, simple, treatment, hopefully, post, course, day, recommend, people, medication]                                  | 1      | 211           |

Figure 15: Reviews column after preprocessing

Now, based on the cleaned data, we can begin to address the first question: identifying the main areas of concern in the negative reviews. While a similar analysis could be done for positive reviews, our focus here will remain on the negative ones.

**Step 1:** Create a function to count the 15 most common words in the negative and positive reviews.

```
# Function to count the most common words in the reviews for a given sentiment
def find_15_most_common_words(df, sentiment):
    words = [] # Initialize an empty list to store words
    for review in df[df['labels'] == sentiment]['reviews']:
        words.extend(review) # Extend the list with words from each review
    return Counter(words).most_common(15) # Return the most common words
```

Figure 16: Function to count 15 most common words

**Step 2:** Call the function with a negative sentiment

```
# Find the 15 most common words in negative reviews
most_negative_words = find_15_most_common_words(df_cleaned, 0)
```

Figure 17: Function to find the 15 most common negative words

Let us visualize the 15 most common negative words using a word cloud, treemap, and pie chart. The same visualizations are also applied to the 15 most common positive words (please refer to the Jupyter Notebook for details).



Figure 18: Word cloud for negative reviews

According to Vevox (2025), a word cloud helps gain insight into the most popular concepts and can be used to reveal sentiment. It provides instant analysis and visualization of word data and feedback, making it a valuable tool in text analysis.

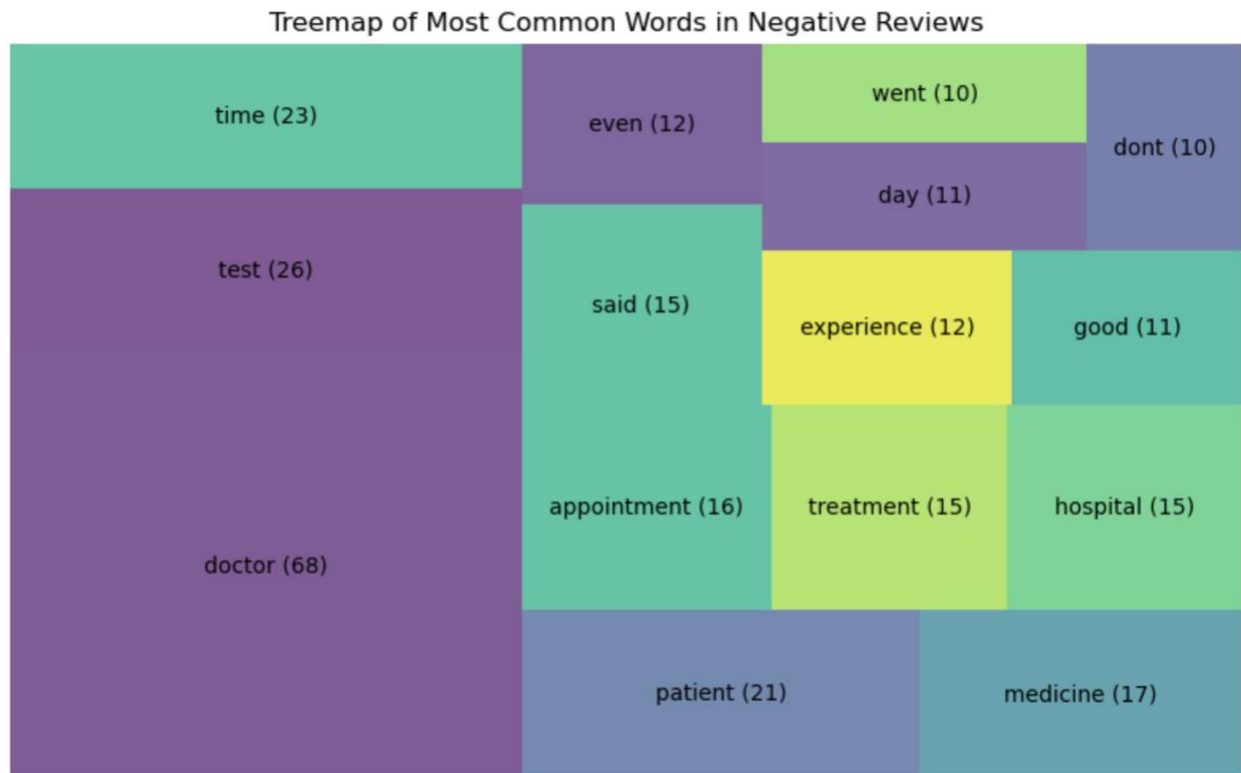
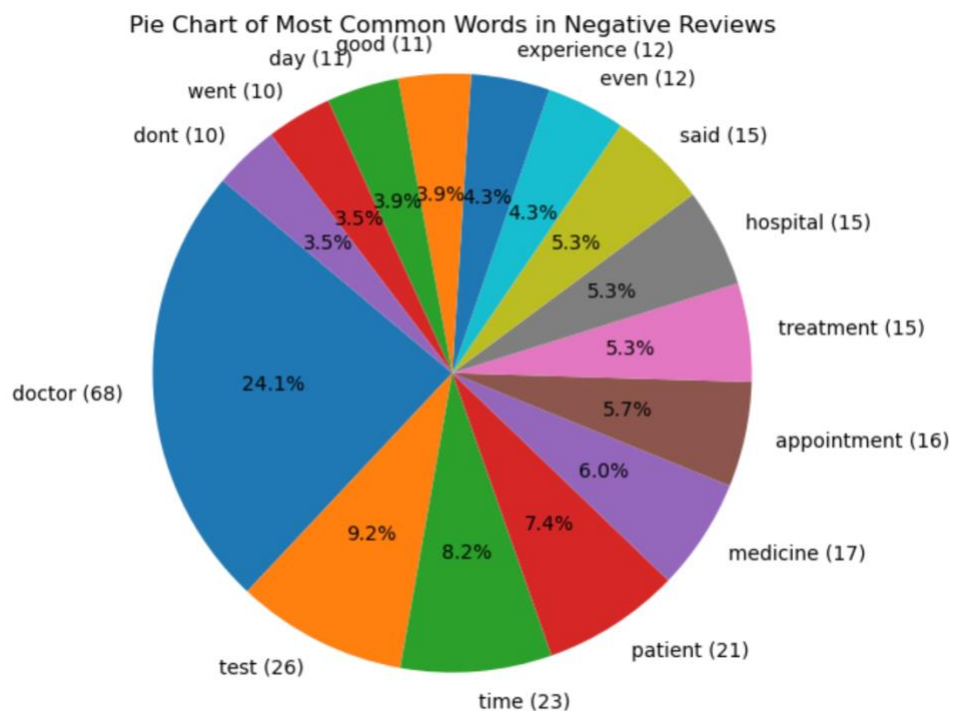


Figure 19: Treemap of 15 most common words in negative reviews

According to Tableau ([s.a]), treemaps are primarily used to display data that is grouped and nested within a hierarchical, tree-based structure. They provide an accessible way for viewers to interpret data at a glance, making complex information easier to understand (Tableau, [s.a]).





To identify the main areas of concern in negative reviews, we analyze the 15 most common words. We can then group them into key categories to better understand the underlying issues.

|  |
|--|
| <b>Main Area 1:</b> Interaction Between Doctor and Patient   |
| <b>Common words:</b> doctor (68), patient (21), said (15), experience (12), dont (10)                  |
| <b>Interpretation:</b> Indicates issues with communication, negative experiences, or poor interactions |

|  |
|--|
| <b>Main Area 2:</b> Medical Tests and Treatment  |
| <b>Common words:</b> test (26), treatment (15), medicine (17)                                |
| <b>Interpretation:</b> Shows dissatisfaction with tests, medication, or treatment procedures |

|   |
|---|
| <b>Main Area 3:</b> Time and Appointment Management   |
| <b>Common words:</b> time (23), appointment (16), day (11), went (10)                         |
| <b>Interpretation:</b> Highlights complaints about long waits, delays, or scheduling problems |

|  |
|--|
| <b>Main Area 4: Healthcare</b>                                       |
| <b>Common words:</b> hospital (15)                                   |
| <b>Interpretation:</b> Points to concerns related to hospital issues |

## 4. Feature selection

Feature selection is a crucial step in selecting the best features to train our model effectively. In this project, we use the Chi-square feature selection method, which evaluates the relationship between each feature and the target variable, helping us identify the features most relevant for predicting sentiment (Sami'un, 2024). To implement this, we first convert the list of tokens back into a single string. We then create a pipeline that transforms the review column into TF-IDF vectors and applies the Chi-square test to each feature. The top K features most correlated with the sentiment label are selected and displayed along with their scores, ensuring that our model is trained on the most informative data.

### Step 1: Convert the list of tokens back into a string

```
# Convert the list of tokens back into a single string for each review
def join_words(words):
    return ' '.join(words) # Join the list of words into a single string
# Apply the join_words function to the 'reviews' column
df_cleaned['reviews'] = df_cleaned['reviews'].apply(join_words) # Convert the list of words back into a single string for each review
df_cleaned.head(5) # Output the first 5 rows of the DataFrame after joining words
```

✓ 0.0s Python

|   | reviews  | labels | review_length |
|---|--|--------|---------------|
| 0 | explained initially take sitings total treatment cost rupee total treatment charged rupee including medicines the estimation treatment false   | 0      | 206           |
| 1 | great definitely recommend recommends less medicine explain everything provide solution  | 1      | 102           |
| 2 | doctor came spent second recommended nose throat endoscopy hearing test night hearing test doctor available make sense charge endoscopy doctor going perform request hospital humanity ground spend t... | 0      | 651           |
| 3 | completely satisfied consultation acute severe bronchitis acute laringitis last day aware consulted doctor different hospital earlier wasnt advised test xray doctor didnt gave proper treatment con...  | 1      | 637           |
| 4 | experience nice dyed cool compose patiently listened medication history gave simple treatment hopefully post course day recommend people medication  | 1      | 211           |

Figure 20: Convert tokens into single string

### Step 2: Create a pipeline that transforms the review column into TF-IDF vectors and applies the Chi-square test to each feature to find the best k.

```
# Create pipeline with TF-IDF vectorization, feature selection, and logistic regression
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()), # Convert text data to TF-IDF features
    ('select', SelectKBest(chi2)), # Select the best features using chi-squared test
    ('classifier', LogisticRegression(max_iter=1000)) # Classifier using logistic regression
])

param_grid_search = {
    'select__k': [10, 20, 30, 40, 50] # Hyperparameter grid for the number of features to select
}

grid_search_value = GridSearchCV(pipeline, param_grid_search, cv=5, scoring='accuracy') # Perform grid search with cross-validation to find
grid_search_value.fit(df['reviews'], df['labels']) # Fit the grid search model to the data
best_k = grid_search_value.best_params_['select__k'] # Get the best number of features found by grid search
print("Best k value found:", grid_search_value.best_params_['select__k']) # Output the best number of features found by grid search
```

Figure 21: Create pipeline with TF-IDF and apply chi-square

**Step 4:** Display the best 30 features with their score (please refer to the notebook for the full list)

|    | Feature        | Score    |
|----|----------------|----------|
| 10 | friendly       | 3.521928 |
| 19 | nice           | 2.410642 |
| 7  | even           | 1.911773 |
| 2  | available      | 1.905569 |
| 29 | well           | 1.673217 |
| 0  | appointment    | 1.635385 |
| 5  | dont           | 1.627607 |
| 8  | experienced    | 1.601882 |
| 12 | happy          | 1.521790 |
| 17 | medication     | 1.471620 |
| 4  | doesnt         | 1.468286 |
| 14 | hour           | 1.417560 |
| 9  | explains       | 1.381970 |
| 16 | listens        | 1.359121 |
| 1  | approach       | 1.344979 |
| 27 | test           | 1.289179 |
| 6  | emergency      | 1.270139 |
| 3  | come           | 1.225783 |
| 13 | hoped          | 1.213902 |
| 11 | great          | 1.183837 |
| 20 | patience       | 1.170502 |
| 22 | report         | 1.125908 |
| 21 | quite          | 1.121854 |
| 28 | treated        | 1.092512 |
| 23 | said           | 1.090933 |
| 25 | strongly       | 1.063620 |
| 24 | spent          | 1.061216 |
| 26 | tablet         | 1.050188 |
| 15 | issuetreatment | 1.021192 |

## 5. Train model: Logistic regression

In this section, we focus on training the model by first finding the best hyperparameters to optimize its performance. We use techniques such as grid search or cross-validation to identify the optimal combination of parameters. Once the best hyperparameters are selected, we proceed to train the model using these values on the training dataset. This ensures that the model learns patterns from the data effectively and performs well during evaluation.

### Step 1: Find the best hyperparameters

```
# Define the hyperparameter grid for logistic regression
param_grid_plan = {
    'classifier__C': [0.01, 0.1, 1, 10, 100], # Regularization parameter
    'classifier__penalty': ['l1', 'l2', 'elasticnet', 'none'], # Penalty type
    'classifier__solver': ['lbfgs', 'liblinear', 'saga'] # Solvers to use
}

# Create a pipeline with TF-IDF vectorization, feature selection, and logistic regression
pipeline_plan = Pipeline([
    ('tfidf', TfidfVectorizer()), # Convert text data to TF-IDF features
    ('select', SelectKBest(chi2, k=best_k)), # Select the best features using chi-squared test
    ('classifier', LogisticRegression(max_iter=1000)) # Classifier using logistic regression
])

# Create a GridSearchCV object for hyperparameter tuning
grid_search = GridSearchCV(pipeline_plan, param_grid_plan, cv=5, scoring='accuracy') # Perform grid search with cross-validation
# Fit the grid search model to the data
grid_search.fit(df_cleaned['reviews'], df_cleaned['labels']) # Fit the grid search model to the reviews and labels
# Output the best hyperparameters
print("Best hyperparameters found:", grid_search.best_params_) # Output the best hyperparameters
```

Figure 22: Find the best hyperparameters

### Step 2: Train the model using the selected hyperparameters

```
# Create pipeline with TF-IDF vectorization, feature selection, and logistic regression
pipeline_train = Pipeline([
    ('tfidf', TfidfVectorizer()), # Convert text data to TF-IDF features
    ('select', SelectKBest(chi2, k=best_k)), # Select the best features using chi-squared test
    ('classifier', LogisticRegression(max_iter=1000, solver='liblinear', C=1.0, penalty='l2')) # Classifier using logistic regression
])

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df_cleaned['reviews'], df_cleaned['labels'], test_size=0.2, random_state=42)
# Fit the pipeline to the training data
pipeline_train.fit(X_train, y_train) # Fit the pipeline to the training data
```

✓ 0.0s




Figure 23: Train model using selected hyperparameters

## 6. Evaluate model

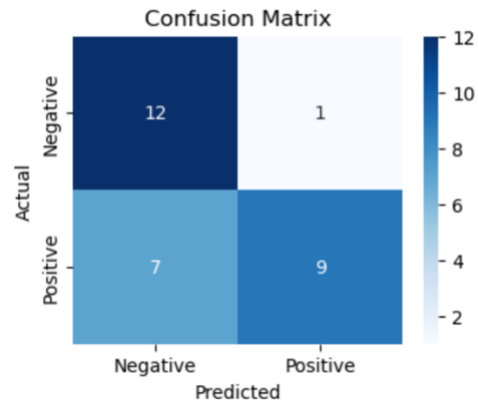
To evaluate our model's performance, we begin by examining its accuracy, which indicates the percentage of correct predictions overall. However, accuracy alone doesn't provide the full picture, so we also use a confusion matrix to break down correct and incorrect predictions for each class. This helps us identify where the model may be making mistakes. Additionally, we analyze precision, which measures how many of the reviews predicted as positive were truly positive, and recall, which indicates how effectively the model identified all actual positive reviews. The F1 score combines both precision and recall to offer a more balanced assessment of performance. To deepen our understanding, we plot the ROC curve and compute the AUC (Area Under the Curve) to evaluate how well the model distinguishes between positive and negative reviews at various thresholds. Collectively, these metrics provide a comprehensive and clear view of our model's reliability.

Accuracy of the model on the testing data: 0.7241379310344828

Figure 24: Accuracy

| Classification Report: |           |        |          |         |
|------------------------|-----------|--------|----------|---------|
|                        | precision | recall | f1-score | support |
| 0                      | 0.63      | 0.92   | 0.75     | 13      |
| 1                      | 0.90      | 0.56   | 0.69     | 16      |
| accuracy               |           |        | 0.72     | 29      |
| macro avg              | 0.77      | 0.74   | 0.72     | 29      |
| weighted avg           | 0.78      | 0.72   | 0.72     | 29      |

Figure 25: Classification report



The logistic regression model, trained on the selected TF-IDF features, achieved an accuracy of approximately 72% on the test set. According to CloudFactory ([s.a]), we should consider accuracy  $> 0.7$  as a good one.

We can use the confusion matrix, which provides a comprehensive overview of how well the model's predictions align with the actual outcomes, to calculate other metrics such as accuracy, precision, recall, and F1 score.

|  |
|--|
| TP (True Positive): Model correctly predicts positive sentiment.                             |
| TN (True Negative): Model correctly predicts negative sentiment.                             |
| FP (False Positive): Model incorrectly predicts positive when it's negative (Type I error).  |
| FN (False Negative): Model incorrectly predicts negative when it's positive (Type II error). |

The confusion matrix reveals that the model correctly identified 9 positive reviews and 12 negative reviews, with only a few misclassifications, 1 false positive and 7 false negatives. This indicates that the model performs slightly better at detecting negative reviews than positive ones.

We can use this matrix to calculate:

|           |   |   |  |
|-----------|---|---|--|
| -Accuracy | $(TP + TN) / (TP + TN + FP + FN)$                           | $(9 + 12) / (9 + 12 + 1 + 7)$                           | 0.724  |
| Precision | $TP / (TP + FP)$  | $9 / (9 + 1) = 0.90$                                    | (label 1), measures the accuracy of the positive predictions made by the model.  |
| Recall    | $TP / (TP + FN)$  | $9 / (9 + 7) = 0.562$                                   | (label 1), measures the model's ability to correctly identify all the actual positive cases in the dataset. It tells us how many of the real positive instances the model was able to find positive. |
| F1 Score  | $2 \times (Precision \times Recall) / (Precision + Recall)$ | $2 \times (0.90 \times 0.562) / (0.90 + 0.562) = 0.692$ | (label 1), the harmonic mean of precision and recall. It provides a balanced measure that considers both how many predictions were correct and how many  |

|  |  |  |                             |
|--|--|--|-----------------------------|
|  |  |  | actual positives were found |
|--|--|--|-----------------------------|

The precision for positive reviews is quite high at 0.90, meaning that when the model predicts a review as positive, it's correct 90% of the time. However, the recall is lower at 0.56, which shows that the model misses several actual positive reviews. The F1-score of 0.69 for positive reviews reflects a balance between precision and recall.

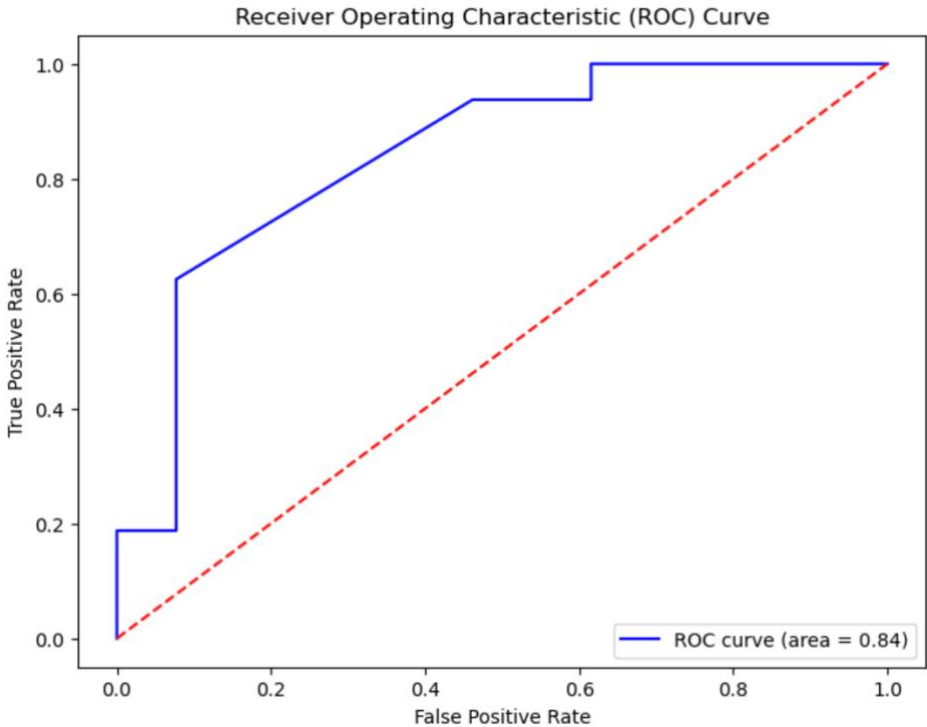


Figure 26: Roc curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation of a model's diagnostic performance (Evidently,2025). It is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds (Recforge, [s.a]) This visualizes the model's ability to distinguish between two classes (Evidently,2025).

From the figure above , the curve is positioned near the top-left corner of the graph, indicating that the model performs well at distinguishing between positive and negative reviews. The area under the curve (AUC) is 0.84, which reflects a strong ability to differentiate between the two classes. This high AUC value confirms the model's effectiveness in correctly identifying sentiment in the text data.

## 7. Conclusion

We have successfully addressed our two main objectives: identifying the key areas of concern in the negative reviews and analyzing overall customer sentiment. Given the small size of the dataset, the model performs well. It is able to effectively capture and distinguish between positive and negative sentiments in patient reviews. However, there are some limitations with the use of TF-IDF, as it does not consider the context in which words appear. By using more advanced techniques, such as BERT, which use transformer-based machine learning techniques to capture semantic meaning and context, we could potentially improve the model's performance. Additionally, increasing the dataset size would likely enhance the model's ability to generalize and deliver more accurate results.



## References

Ashmed. 2024. *ML Series 9: Understanding Logistic Regression: A Practical Guide*. [Online]. Available at: <https://medium.com/@sahin.samia/understanding-logistic-regression-a-practical-guide-7aa94ca62c80> (Accessed: 25 June 2025).

Bassey. 2019. *Logistic Regression in Machine Learning*. [Online]. Available at: <https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/> (Accessed: 25 June 2025).

CrawlSpider. 2024. *Can TF-IDF be used for text Classification, How?*. [Online]. Available at: <https://www.crawlspider.com/can-tf-idf-be-used-for-text-classification-how/> (Accessed: 25 June 2025).

Evidently. 2025. *How to explain the ROC curve and ROC AUC score?*. [Online]. Available at: <https://www.evidentlyai.com/classification-metrics/explain-roc-curve> (Accessed: 25 June 2025).

GeeksforGeeks. 2025. *Logistic Regression in Machine Learning* [Online]. Available at: <https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/> (Accessed: 25 June 2025).

Gillis. 2024. *\*What is sentiment analysis* [Online]. Available at: <https://www.techtarget.com/searchbusinessanalytics/definition/opinion-mining-sentiment-mining> (Accessed: 26 June 2025).

Indeed Team. 2025. *When to Use Logistic Regression (With Definition and Types)\** [Online]. Available at: <https://ca.indeed.com/career-advice/career-development/when-to-use-logisticregression#:~:text=Logistic%20regression%20is%20a%20popular,data%20to%20p,roduce%20accurate%20results>. (Accessed: 26 June 2025).

Kanade. 2022. *What Is Logistic Regression? Equation, Assumptions, Types, and Best Practices* [Online]. Available at: <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/> (Accessed: 26 June 2025).

Kavya. 2023. *Naive Bayes Classifier Explained: Assumptions, Types, and Uses* [Online]. Available at: <https://medium.com/@kavyasrirelangi100/naive-bayes-classifier-explained-assumptions-types-and-uses-bef767a758a3/> (Accessed: 27 June 2025).

Parr. 2024. *Sentiment Analysis in Healthcare: Transforming Patient Feedback into Actionable Insights* [Online]. Available at: <https://www.repugen.com/blog/sentiment-analysis-in-healthcare> (Accessed: 27 June 2025).

Recforge. 2025 \**Sentiment analysis* [Online]. Available at: <https://academy.recforge.com/#/course/sentiment-analysis-216> (Accessed: 27 June 2025).

Simha. 2021. *Understanding TF-IDF for Machine Learning* [Online]. Available at: <https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/> (Accessed: 27 June 2025).

Turning. 2022. *How to Create Naive Bayes Document Classification in Python?* [Online]. Available at: <https://www.turing.com/kb/document-classification-using-naive-bayes> (Accessed: 27 June 2025).