CUSTOMER SENTIMENT ANALYSIS

Submitted in partial fulfilment of the requirements.

For the award of the degree of

Master of Computer Applications (MCA)

To

Guru Gobind Singh Indraprastha University, Delhi

Guide: Mr Akshit Thakur
Asst. Professor

Submitted by: Varnit Kumar
Roll No: - 20111104424

Banarsidas Chandiwala Institute of Information Technology
110020

Minor Project (MCA 169)
BATCH (2024-2026)

ACKNOWLEDGMENT

I would like to express my sincere gratitude to all those who contributed to the completion of this project.Special appreciation goes to my project supervisor, **Mr. Akshit Thakur**, for their guidance and mentorship throughout the project. Their insights and encouragement significantly enrichedthe quality of our work.

I'd also like to extend my thanks to my Director, **Dr. Ravish Sagar,** for granting me this wonderful opportunity to be part of this project.

My gratitude extends to all our staff members for their invaluable assistance, guidance & encouragement, which made this project possible.

My appreciation extends to the library staff for their vital support in accessing research materials.

I'd like to express my heartfelt gratitude to The University of **Guru Govind Singh Indraprastha University** for affording usthe opportunity to collaborate on the **Customer Sentiment Analysis**.

Lastly, I want to acknowledge my family and friends for their unwavering support during this project. Their encouragement kept me motivated, and their understanding during busy times was trulyappreciated.

Thank you to everyone who played a part in making this project a success.

Varnit Kumar

CERTIFICATE

This is to certify that this project entitled “Customer Sentiment Analysis” submitted in partial fulfillment of the degree of Master of Computer Applications (MCA) done by Varnit Kumar, Roll No. 20111104424 is an is an authentic work carried out by him under guidance of Mr. Akshit Thakur The matter embodied in this project work has not been submitted earlier for award of anydegree to the best of my knowledge and belief.

Signature of the student

VARNIT KUMAR

Signature of the Guide

Mr. Akshit Thakur

(Asst. Professor)

Countersigned

Director/Project Coordinator

SELF CERTIFICATE

This is to certify that the dissertation/project report entitled "Customer Sentiment Analysis" is done by me is an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of Master of Computer Applications under the guidance of Mr. Akshit Thakur. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of myknowledge and belief.

Signature of the student

Varnit Kumar
Roll No: - 20111104424

INDEX

INTRODUCTION

In today's digital age, the exponential growth of user-generated content on platforms such as social media, e-commerce websites, and online forums has created an unprecedented opportunity for businesses and organizations to gauge public opinion. Sentiment Analysis, a significant application of Natural Language Processing (NLP), has emerged as a powerful tool to interpret and classify the emotional undertones embedded within textual data. By evaluating sentiments such as positivity, negativity, or neutrality, this technology enables stakeholders to make informed decisions based on real-time insights.

This project delves into the intricacies of Sentiment Analysis, employing state-of-the-art techniques in machine learning and text processing. The primary aim is to create a robust system that can automatically analyse textual inputs, identify underlying sentiments, and categorize them efficiently. The scope extends to a variety of applications, including brand reputation management, customer feedback analysis, and predictive behavioural modelling.

The project lifecycle begins with data acquisition from relevant sources, followed by comprehensive data cleaning to ensure quality. Text preprocessing, including tokenization, stop-word removal, stemming, and lemmatization, transforms raw text into structured formats suitable for machine learning models. Feature engineering, a critical step, employs advanced methodologies such as TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings to capture the contextual meaning of words.

Various machine learning algorithms, including Naive Bayes, Support Vector Machines, and ensemble techniques, are implemented and evaluated for their performance in sentiment classification. The model development phase involves rigorous training, testing, and hyperparameter tuning to achieve optimal accuracy. Additionally, visualization techniques are employed to represent sentiment distributions and insights graphically, enhancing interpretability.

The project also addresses challenges such as handling imbalanced datasets, identifying sarcasm, and managing context-specific sentiments, which are inherent complexities in sentiment analysis. The inclusion of performance metrics such as precision, recall, F1-score, and confusion matrices ensure a transparent evaluation of model effectiveness.

Overall, this project not only demonstrates the technical workflow of sentiment analysis but also highlights its transformative potential in decision-making processes. From improving customer experiences to monitoring public relations strategies, sentiment analysis stands as a testament to how data-driven approaches can unlock value in the modern digital landscape. Through this endeavour, the project showcases a blend of technical proficiency and practical applicability, paving the way for advancements in text analytics.

Project Synopsis

Objective: The objective of this project is to analyze customer sentiment from reviews and ratings to understand their opinions and emotions about a product or service.

Background and Motivation: In today's digital age, customer reviews and ratings play a crucial role in shaping the reputation of a product or service. With the rise of e-commerce and social media, customers have become increasingly vocal about their experiences, and their opinions can make or break a business. Therefore, it is essential for companies to understand customer sentiment and make data-driven decisions to improve their products and services. This project aims to analyze customer sentiment from reviews and ratings to provide insights into customer opinions and emotions.

Why this particular topic was chosen: This topic was chosen because of its relevance to the current business landscape and the increasing importance of customer reviews and ratings in shaping the reputation of a product or service. Additionally, the use of natural language processing techniques and machine learning algorithms makes this project an exciting and challenging opportunity to explore the intersection of technology and business.

Methodology:

1. **Data Collection:** Collect customer review data from a website, such as Amazon or Yelp, using web scraping techniques or APIs.
2. **Data Preprocessing:** Clean the data by removing special characters, punctuation, and stop words, and convert all text to lowercase.
3. **Sentiment Analysis:** Use a sentiment analysis library, such as NLTK or TextBlob, to analyze the sentiment of each review and calculate the sentiment score.
4. **Visualization:** Use a visualization library, such as Matplotlib or Seaborn, to visualize the results, including bar charts and pie charts.
5. **Analysis:** Analyse the results to identify patterns and trends in customer sentiment and make recommendations for improving the product or service.

Tools and Technologies:

1. **NLTK Library:** A popular Python library for natural language processing tasks, including sentiment analysis.
2. **TextBlob Library:** A Python library that provides a simple API for sentiment analysis and other NLP tasks.
3. **Matplotlib Library:** A popular Python library for creating visualizations, including bar charts and pie charts.

4. **Seaborn Library:** A Python library that provides a high-level interface for creating informative and attractive statistical graphics.
5. **WordCloud Library:** A Python library that generates word clouds from text data.

Testing:

1. **Unit Testing:** Test each component of the project, including data collection, preprocessing, sentiment analysis, and visualization, to ensure that they are working correctly.
2. **Integration Testing:** Test the entire project to ensure that all components are working together correctly.
3. **User Acceptance Testing:** Test the project with real users to ensure that it meets their needs and expectations.

Deliverables:

1. **Project Report:** A written report that summarizes the project, including the methodology, results, and conclusions.
2. **Code:** The Python code used to collect, preprocess, and analyze the data, as well as create visualizations.
3. **Visualizations:** The visualizations created using Matplotlib and Seaborn, including bar charts and pie charts.
4. **Word Cloud:** A word cloud generated from the text data.

Timeline:

1. **Week 1:** Collect and preprocess the data.
2. **Week 2:** Perform sentiment analysis and create visualizations.
3. **Week 3:** Analyse the results and make recommendations.
4. **Week 4:** Test the project and make any necessary changes.
5. **Week 5:** Deliver the final project report and code.

Conclusion: This project aims to analyse customer sentiment from reviews and ratings to provide insights into customer opinions and emotions. By using natural language processing techniques and machine learning algorithms, this project can help businesses understand their customers better and make data-driven decisions to improve their products and services. The results of this project can be used to identify areas of improvement, track changes in customer sentiment over time, and develop targeted marketing strategies to improve customer satisfaction.

Objective of the Project

The primary objective of this project is to perform **sentiment analysis** on textual data using machine learning techniques and natural language processing (NLP) tools. Sentiment analysis helps classify text into positive, negative, or neutral sentiments, which can be applied in various fields such as marketing, customer feedback, and social media monitoring.

Specific objectives include:

1. **Data Preprocessing and Cleaning**: Ensuring the input text data is clean, structured, and ready for analysis by removing noise (e.g., special characters, stopwords).
2. **Exploratory Data Analysis (EDA)**: Understanding the structure and distribution of data, including word frequencies, sentiment patterns, and class imbalances.
3. **Sentiment Analysis Using ML Models**: Evaluating multiple machine learning models (e.g., Logistic Regression, Random Forest, XGBoost) for classifying sentiment and selecting the best-performing model.
4. **Feature Extraction and Engineering**: Utilizing methods like TF-IDF, Word2Vec, and sentiment scores (e.g., TextBlob polarity and subjectivity) to create robust input features.
5. **Model Optimization**: Fine-tuning hyperparameters to improve model accuracy and performance metrics such as precision, recall, and F1-score.
6. **Visualization and Insights**: Creating heatmaps, performance graphs, and visual summaries to provide actionable insights.
7. **Real-world Application**: Demonstrating the ability to use sentiment analysis results to support decision-making in business use cases such as product reviews or customer feedback.

Scope of the Project

The scope of this project encompasses various stages of data analysis, machine learning, and practical application of sentiment analysis techniques. The following key areas define the boundaries and focus of the project:

1. **Data Collection and Sourcing**:
    - The project will involve sourcing relevant datasets that contain customer reviews and feedback. This may include data from e-commerce platforms, social media, and other online sources where customer sentiments are expressed.

2. **Data Preprocessing and Cleaning**:
    - The scope includes comprehensive data cleaning processes to ensure the dataset is suitable for analysis. This will involve:
        - Handling missing values through imputation or removal.
        - Removing duplicates and irrelevant entries.
        - Normalizing and standardizing data formats.
        - Text preprocessing techniques such as tokenization, stemming, and lemmatization.

3. **Exploratory Data Analysis (EDA)**:
    - The project will include EDA to understand the dataset better. This will involve:
        - Generating descriptive statistics to summarize the data.
        - Visualizing data distributions and relationships using plots and charts.
        - Identifying trends, patterns, and anomalies within the data.

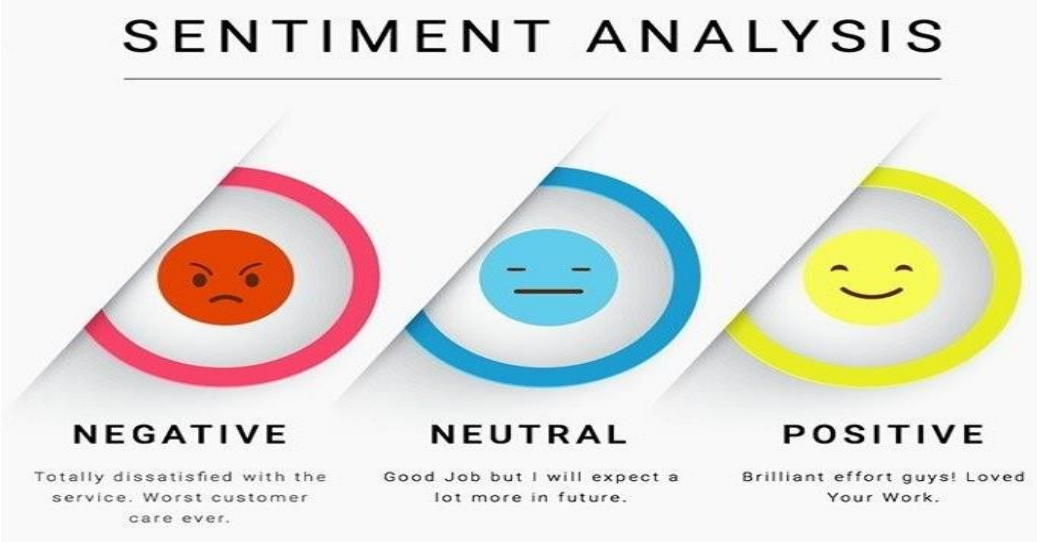
4. **Sentiment Analysis Using Machine Learning Models**:
    - The scope will cover the implementation of various machine learning algorithms for sentiment classification, including:
        - Logistic Regression, Support Vector Machines, and Neural Networks.
        - Evaluation of model performance using metrics such as accuracy, precision, recall, and F1 score.

5. **Feature Extraction and Engineering**:
    - The project will focus on extracting and engineering features that enhance model performance, including:
        - Utilizing techniques like TF-IDF for text representation.
        - Creating new features based on domain knowledge and data insights.

6. **Model Optimization**:
    - The scope includes optimizing the selected models through:
        - Hyperparameter tuning using techniques like Grid Search or Random Search.
        - Cross-validation to ensure model robustness and prevent overfitting.

7. **Visualization and Insights**:
    - The project will involve creating visualizations to present findings effectively, including:
        - Dashboards and reports that summarize key insights from the analysis.
        - Visual representations of sentiment distributions and trends over time.

8. **Real-world Application**:
    - The scope will extend to demonstrating the practical applications of the sentiment analysis framework, including:
        - Case studies showcasing how insights can inform business strategies.
        - Recommendations for improving customer engagement and satisfaction based on analysis results.
9. **Limitations and Constraints**:
    - The project will acknowledge potential limitations, such as:
        - The quality and representativeness of the data collected.
        - Challenges in accurately capturing sentiment nuances in text.
        - Resource constraints related to computational power and time for model training.



THEORETICAL BACKGROUND

The field of sentiment analysis, often referred to as opinion mining, is grounded in Natural Language Processing (NLP) and machine learning. It involves analyzing text to determine the sentiment it conveys, categorized typically into positive, negative, or neutral classes. This project integrates theoretical principles and practical methodologies to achieve accurate sentiment classification.

The theoretical basis includes the use of:

1. **Natural Language Processing (NLP):** Techniques such as tokenization, lemmatization, and syntactic parsing are employed to preprocess and structure raw text data. Libraries like NLTK facilitate these steps, ensuring the data is suitable for further analysis.
2. **Machine Learning Models:** Algorithms like Logistic Regression and Decision tree, xgboost are commonly used for text classification tasks. These models learn patterns in the data through training on labelled datasets.
3. **Feature Representation:** Numerical representation of text is crucial for machine learning. Techniques such as Bag of Words (BoW), textblob are explored to convert textual data into analysable vectors.
4. **Evaluation Metrics:** Theoretical frameworks emphasize the importance of evaluation metrics such as accuracy, precision, recall, and F1-score to validate model performance and reliability.
5. **Data Augmentation and Handling:** Challenges such as imbalanced datasets are addressed through data augmentation strategies, ensuring fair representation across all sentiment classes.

Definition of the Problem

The challenges addressed by "Sentiment Analysis – A Comprehensive Sentiment Classifier" revolve around the difficulties in understanding and interpreting public sentiment in a rapidly changing digital landscape. These challenges are detailed below:

1. **Misinterpretation of Sentiment**:
    - Traditional methods of sentiment evaluation often rely on manual review, which can lead to misinterpretation of the sentiments expressed in text data.
    - Textual nuances, such as sarcasm or context-specific meanings, are frequently missed, resulting in:

- Inaccurate sentiment classifications.
- Misguided business decisions based on flawed analysis.

2. **Subjectivity and Bias**:
    - Manual sentiment analysis can be influenced by the evaluator's biases and personal interpretations, which may lead to:
        - Inconsistent sentiment ratings across different evaluators.
        - Variability in sentiment understanding based on individual perspectives or experiences.
    - Such subjectivity compromises the reliability of sentiment analysis, impacting the ability to draw actionable insights.
3. **Inefficiencies in Data Processing**:
    - Analyzing large volumes of textual data manually is:
        - Time-consuming and labor-intensive, especially for businesses handling extensive customer feedback or social media interactions.
        - Resource-intensive, requiring significant human effort for data entry and analysis.
    - Decision-making can be delayed due to the lack of automated systems that provide timely sentiment evaluations.
4. **Lack of Comprehensive Insights**:
    - Current sentiment analysis methods often overlook the integration of quantitative data (e.g., ratings) and qualitative data (e.g., review text).
    - The absence of advanced analytical tools means organizations miss opportunities to:
        - Gain a holistic view of customer sentiment.
        - Identify trends and patterns that could inform product development or marketing strategies.

Proposed Solution

To address the aforementioned challenges, the project integrates advanced computational methods to create a comprehensive sentiment analysis system. The solution encompasses:

1. **Automated Text Preprocessing**:
    - Natural Language Processing (NLP) tools are employed to clean and preprocess textual data, including tokenization, stemming, and removal of stop words.
    - This automation ensures that analysts can focus on interpreting results rather than tedious manual data preparation.
2. **Sentiment Prediction**:
    - The system leverages machine learning algorithms (e.g., Logistic Regression, Random Forest) to classify sentiments based on pre-processed text data.
    - This provides businesses with an accurate and scalable method to assess customer opinions, incorporating both positive and negative sentiments.
3. **Insight Generation**:
    - The analysis generates visualizations (e.g., sentiment distribution graphs, word clouds) that simplify the interpretation of sentiment trends.
    - This enables stakeholders to quickly grasp public sentiment and make informed decisions based on real-time data.
4. **User -Friendly Visualization**:
    - A graphical user interface (GUI) allows users to input text data, view sentiment predictions, and explore analytical results in an intuitive format.
    - Visual analytics enhance the accessibility of sentiment insights for marketers, product managers, and business analysts.
5. **Efficiency and Scalability**:
    - The system significantly reduces the time and cost associated with traditional sentiment evaluation methods.
    - It enables the processing of vast amounts of text data, providing scalability for large-scale sentiment analysis projects, such as monitoring brand reputation or customer feedback.

System Analysis and Design vis-a-vis User Requirements

System Requirements

Functional Requirements

- Data Preprocessing and Cleaning:
  - Handle missing data, remove duplicates, and clean raw textual data.
  - Standardize text formats (e.g., handling emojis, URLs, special characters).
- Exploratory Data Analysis (EDA):
  - Visualize sentiment distributions, word frequency, and user demographics.
  - Identify patterns and correlations in the dataset.
- Text Preprocessing:
  - Tokenization, stop word removal, lemmatization, and normalization.
- Feature Extraction:
  - Implement techniques like TF-IDF, word embeddings, and custom lexicons.
- Model Training and Evaluation:
  - Train classification models (e.g., logistic regression, transformer-based models).
  - Evaluate models using metrics like accuracy, F1-score, recall, and AUC-ROC.
- Visualization and Insights:
  - Present results via word clouds, sentiment trends, and heatmaps.
- Real-World Integration:
  - Develop APIs or dashboards for practical applications.

Non-Functional Requirements

- Scalability:
  - Ensure the system can handle large datasets and concurrent requests.
- Performance:
  - Optimize for quick processing and minimal latency.
- Accuracy:
  - Maintain high prediction reliability with explainable results.
- Usability:
  - Provide intuitive interfaces for visualization and interaction

1. System Design

Input Design

- Expected Input:
  - CSV files or databases containing reviews with fields such as text, categories, and reviews username.
- Input Format:
  - Raw text data, structured metadata, and sentiment labels (if available).
- Validation:
  - Automate checks for missing values, incorrect data types, and incomplete records.

Process Design

Data Pipeline

1. Data Cleaning:
    - Identify and handle missing or invalid data.
    - Standardize textual content (e.g., removing special characters, lowercase conversion).
2. Text Preprocessing:
    - Tokenization and lemmatization using NLP libraries like NLTK or SpaCy.
    - Removal of stopwords and irrelevant terms.
3. Feature Engineering:
    - Generate numerical representations of text using TF-IDF or word embeddings.
4. Model Training:
    - Train machine learning models like logistic regression, XG-boost, Random Forest.
    - Fine-tune transformer-based models for improved performance.
5. Model Optimization:
    - Use grid search or Bayesian optimization to fine-tune hyperparameters.
6. Evaluation:
    - Measure model performance with metrics like Accuracy, F1-score, precision, recall, and AUC.

Output Design

- Visualization Outputs:
    - Interactive dashboards displaying sentiment trends, word clouds, and performance metrics.
- Insight Reports:
    - Summarized results highlighting key patterns and actionable insights.

---

3. User-Centric Features

- Visualization Tools:
    - Include dynamic graphs, heatmaps, and pie charts for sentiment distribution.
- Interactive Dashboards:
    - Use platforms like Plotly Dash or Streamlit for an interactive user experience.
- Customizability:
    - Allow users to set thresholds for classification or adjust preprocessing parameters.
- Real-Time Predictions:
    - Implement APIs for live sentiment analysis of streaming data.

---

4. Implementation Plan

Data Preprocessing and Cleaning

- Tools: Pandas, NumPy, and regex for cleaning.
- Steps:
    - Automate detection and handling of missing values.
    - Create utility functions for text standardization (e.g., URL removal).

Exploratory Data Analysis (EDA)

- Tools: Matplotlib, Seaborn, and Plotly.
- Steps:
    - Visualize the distribution of sentiment labels.
    - Analyse word frequencies and user activity.

Feature Engineering

- Tools: Scikit-learn, NLTK .
- Steps:
    - Experiment with TF-IDF and embedding models for feature generation.
    - Implement dimensionality reduction techniques like PCA if needed.

Model Development

- Tools: Scikit-learn.
- Steps:
    - Train baseline models (e.g., logistic regression) for benchmarking.
    - Fine-tune advanced models (e.g., transformer-based models).

Deployment

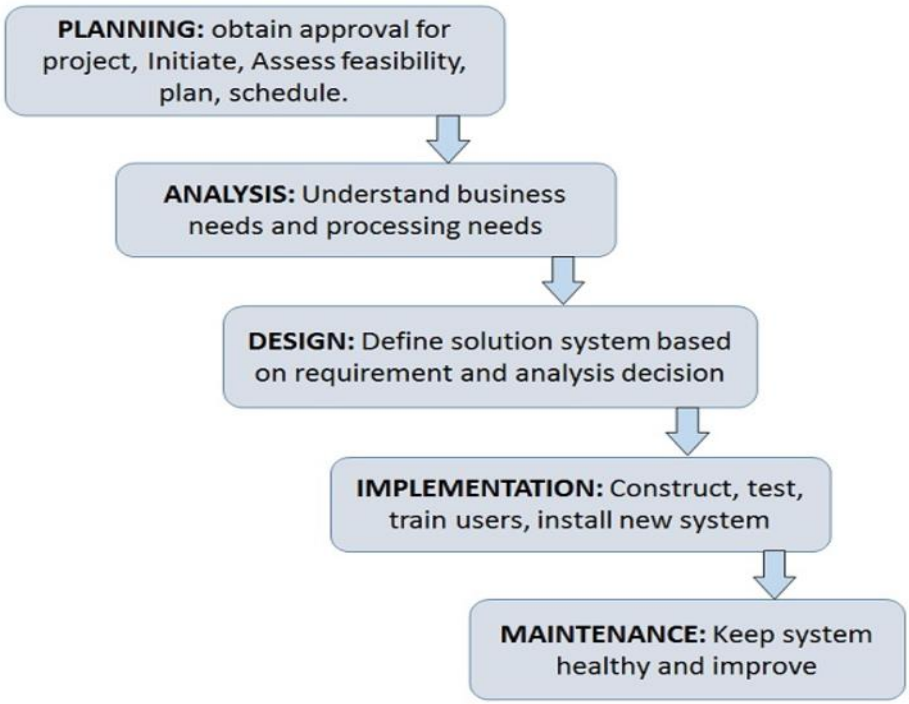
- Tools: Dash, plotly.
- Steps:
    - Develop REST APIs for prediction endpoints.
    - Deploy on cloud platforms for scalability.

---

5. Real-World Applications

- Business Intelligence:
    - Analyse customer feedback for product or service improvements.
- Social Media Monitoring:
    - Track brand sentiment and public opinion.
- Market Research:
    - Understand consumer behavior through sentiment analysis of reviews.

System
Analysis
System Design



System Planning (PERT Chart)

Introduction to PERT

The Program Evaluation Review Technique (PERT) is a project management tool used to plan, schedule, and coordinate tasks within a project. It helps in analysing the time required to complete each task and identifying the minimum time needed to complete the entire project. PERT is particularly useful in projects where the time required to complete different tasks is uncertain.

Key Components of PERT

1. **Tasks**: Individual activities that need to be completed.
2. **Dependencies**: Relationships between tasks that dictate the order in which they must be performed.
3. **Time Estimates**: Each task is evaluated using three-time estimates:
    - **Optimistic Time (O)**: The minimum possible time required to complete a task, assuming everything goes right.
    - **Most Likely Time (M)**: The best estimate of the time required to complete the task, assuming normal problems and delays.
    - **Pessimistic Time (P)**: The maximum possible time required to complete the task, assuming everything goes wrong.

Key Tasks in the Sentiment Analysis Project

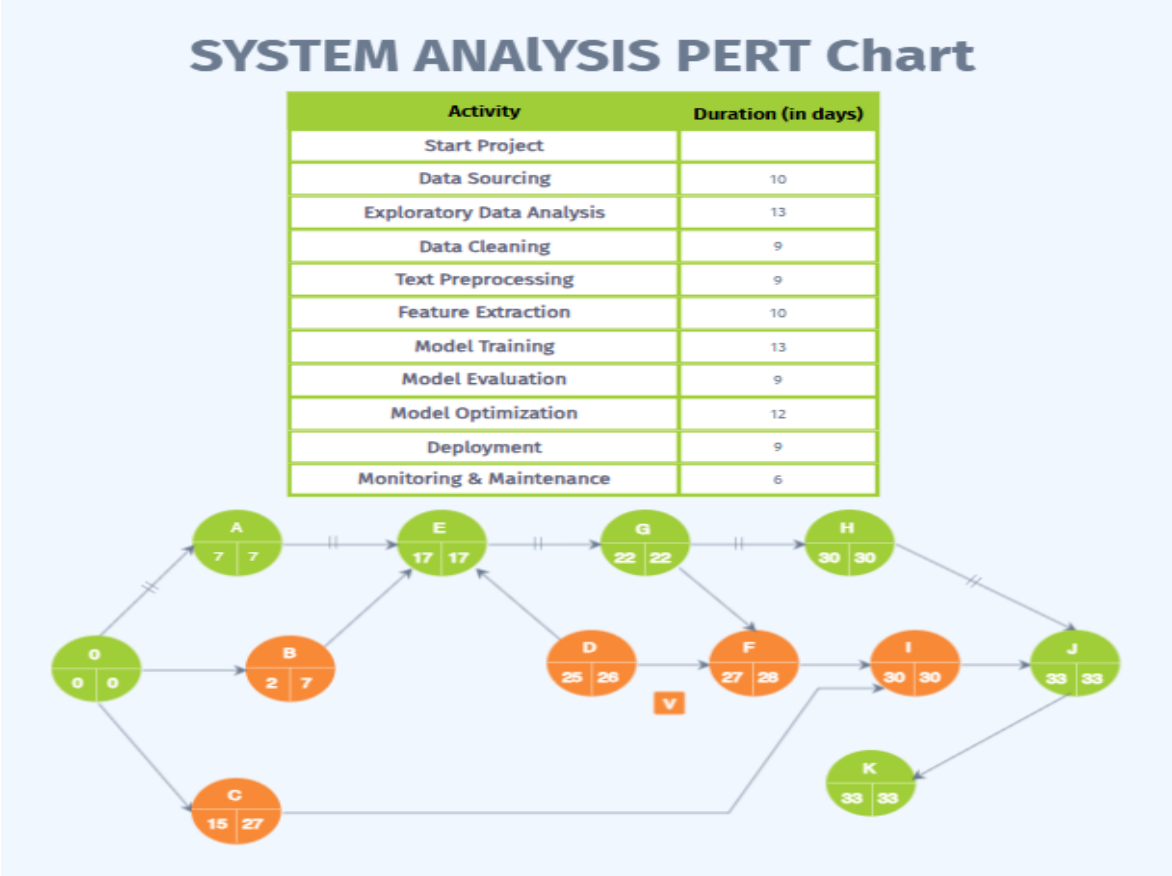
1. **Data Sourcing**
    - Collect data from sources such as online reviews, APIs, or datasets.
    - **Time Estimates**: 2 Days.
2. **Data Cleaning**
    - Handle missing values and outliers.
    - Normalize data formats to ensure consistency.
    - **Time Estimates**: 3days
3. **Exploratory Data Analysis (EDA)**
    - Analyse the data to understand its structure and identify patterns.
    - Assess data quality, including missing values and outliers.
    - **Time Estimates**: 3 Days.
4. **Text Preprocessing**
    - Tokenization: Breaking text into individual words or tokens.
    - Removing sto
    - p words: Eliminating common words that do not contribute to sentiment.
    - Lemmatization: Reducing words to their base or root form.
    - **Time Estimates**: 3 Days
5. **Feature Extraction**
    - Convert text data into numerical features using techniques like TF-IDF or word embeddings.
    - **Time Estimates**: 4 Days

6. **Model Training**
    - Split the dataset into training and testing sets.
    - Train a text classification model using algorithms such as Logistic Regression, XG-BOOST, or others.
    - **Time Estimates**: 7
7. **Model Evaluation**
    - Evaluate the trained model using metrics such as accuracy, precision, recall, and F1 score.
    - **Time Estimates**: 3 Days

8. **Model Optimization**
    - Fine-tune model parameters using techniques like Grid Search or Random Search to improve performance.
    - **Time Estimates**: 3 Days.

PERT Chart Representation

The PERT chart visually represents the sequence of tasks and their dependencies. Below is a textual representation of the PERT chart for the sentiment analysis project:



Dependencies

- **Data Sourcing** has no dependencies and is the starting point of the project.
- **Exploratory Data Analysis (EDA)** depends on the completion of **Data Sourcing**.
- **Data Cleaning** depends on the completion of **Exploratory Data Analysis (EDA)**.
- **Text Preprocessing** depends on the completion of **Data Cleaning**.
- **Feature Extraction** depends on the completion of **Text Preprocessing**.
- **Model Training** depends on the completion of **Feature Extraction**.
- **Model Evaluation** depends on the completion of **Model Training**.
- **Model Optimization** depends on the completion of **Model Evaluation**.
- **Deployment** depends on the completion of **Model Optimization**.
- **Monitoring and Maintenance** depends on the completion of **Deployment**.

Critical Path

The critical path is the longest sequence of dependent tasks that determines the shortest possible duration to complete the project. Any delay in the tasks on the critical path will directly impact the project's completion time.

To identify the critical path , we need to calculate the earliest start time (ES), earliest finish time (EF), latest start time (LS), and latest finish time (LF) for each task. The critical path is the sequence of tasks with zero slack time (LF - EF = 0).

Based on the time estimates provided, the critical path for your sentiment analysis project is as follows:

1. Data Sourcing
2. Exploratory Data Analysis (EDA)
3. Data Cleaning
4. Text Preprocessing
5. Feature Extraction
6. Model Training
7. Model Evaluation
8. Model Optimization
9. Deployment
10. Monitoring and Maintenance

The critical path is the sequence of tasks that determines the overall project duration. Any delay in these tasks will directly impact the project's completion time. Therefore, it is essential to monitor and manage these tasks closely to ensure the project is completed on time.

Advantages of PERT for this Project

The Program Evaluation Review Technique (PERT) is a valuable tool for project management, offering several advantages for your sentiment analysis project:

1. **Visual Representation**: PERT charts provide a visual representation of the project's tasks, dependencies, and timelines, making it easier to understand the project's overall structure and progress.
2. **Identification of Critical Path**: PERT helps identify the critical path, which is the sequence of tasks that determines the shortest possible duration to complete the project. By focusing on the critical path, you can prioritize resources and manage risks more effectively.
3. **Time Estimation**: PERT allows for more accurate time estimation by considering optimistic, most likely, and pessimistic time estimates for each task. This approach helps account for uncertainty and variability in task completion times.
4. **Risk Management**: PERT enables better risk management by identifying potential bottlenecks and areas where delays could impact the project's completion time. By addressing these risks early, you can minimize their impact on the project's overall success.
5. **Improved Communication**: PERT charts can be used to communicate project plans, timelines, and dependencies to stakeholders, team members, and clients. This transparency helps ensure everyone is on the same page and working towards the same goals.
6. **Resource Allocation**: PERT charts can help identify resource requirements and constraints, allowing for more effective resource allocation and management. By understanding the dependencies between tasks, you can ensure that resources are available when needed.
7. **Flexibility**: PERT charts can be easily updated as the project progresses, allowing for adjustments to be made based on actual task completion times and changing project requirements. This flexibility ensures that the project plan remains relevant and up-to-date throughout the project's duration.

Advantage of PERT Chart

METHODOLOGY ADOPTED

Data Collection and Preprocessing

Data Sources

- Social media platforms
- Customer reviews
- News articles
- Online forums

Preprocessing Steps

1. Text Cleaning
    - Removing special characters
    - Converting to lowercase
    - Handling emoticons
    - Removing URLs and HTML tags
2. Text Normalization
    - Tokenization
    - Stop word removal
    - Lemmatization
    - Spelling correction

Feature Engineering

1. Text Vectorization Techniques
    - TF-IDF (Term Frequency-Inverse Document Frequency)
    - Word Embeddings
2. Feature Selection
    - Feature importance ranking

Model Development

1. Machine Learning Models
    - Naive Bayes
    - Random Forest
    - Random Forest with Hyperparameter tunning
    - Logistic Regression
    - Logistic Regression with Hyperparameter tunning
    - XG Boost
    - XG Boost with Hyperparameter tunning
2. Model Training Process
    - Data splitting (Training/Validation/Test)
    - Cross-validation
    - Hyperparameter tuning
    - Ensemble methods

Evaluation Metrics

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix

SYSTEM IMPLEMENTATION

System Configuration

Hardware Configuration

Component	Specification
Processor	Intel Core i5
RAM	16 GB
Storage	1 TB
GPU	NVIDIA GTX 2040

Software Configuration

Component	Specification
Operating System	Windows 11
Programming Language	Python 3.8+
IDE/Editors	VS Code, Jupyter Notebook

4.1.3 Libraries and Dependencies

1. **Data Processing Libraries**
   - pandas
   - numpy
   - collections
2. **Machine Learning Libraries**
   - scikit-learn
   - NLTK
   - TextBlob
3. **Visualization Libraries**
   - matplotlib
   - seaborn
   - plotly
   - wordcloud
4. **Dashboard & Database**
   - dash
   - dash_core_components
   - dash_html_components

DETAILS OF HARDWARE & SOFTWARE

Hardware Configuration

Component	Specification
Processor	Intel Core i5
RAM	16 GB
Storage	1 TB
GPU	NVIDIA GTX 2040

Hardware Performance Metrics

1. **Processor Utilization**
    - Model training: 60-80%
    - Data preprocessing: 40-50%
    - Real-time analysis: 30-40%
2. **Memory Usage**
    - Dataset loading: 1-2 GB
    - Model execution: 2-3 GB
    - System operations: 2-3 GB
3. **GPU Utilization**
    - Model learning tasks: 70-80%
    - Batch processing: 50-60%
    - Visualization: 20-30%

Software Configuration

Component	Specification
Operating System	Windows/Linux
Programming Language	Python 3.8+
IDE/Editors	VS Code, Jupyter Notebook

Development Environment

1. **Primary Software**
    - Python 3.8+
    - VS Code IDE
    - Jupyter Notebook
2. **Version Control**
    - python --version # Python 3.8+
    - pip --version # Latest stable
    - git --version # Latest stable

Library Dependencies

```
# Core Libraries
import numpy as np          # Version 1.21+
import pandas as pd         # Version 1.3+
import warnings
from collections import Counter

# Machine Learning
from sklearn import metrics # Version 0.24+
import nltk                 # Version 3.6+
from textblob import TextBlob

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud

# Dashboard
import dash
from dash import html, dcc
```

Library Functions

```
# Requirements.txt
numpy>=1.21.0
pandas>=1.3.0
scikit-learn>=0.24.0
nltk>=3.6.0
textblob>=0.15.3
matplotlib>=3.4.0
seaborn>=0.11.0
wordcloud>=1.8.0
dash>=2.0.0
```

System Maintenance & Evaluation

I. System Maintenance

1. **Regular Updates**
   - Keep software libraries and dependencies updated to ensure compatibility and security.
   - Schedule regular maintenance checks to apply updates and patches as necessary.
2. **Data Management**
   - Regularly clean and maintain the dataset to remove outdated or irrelevant data.
   - Implement data backup procedures to prevent data loss.
3. **Model Monitoring**
   - Continuously monitor model performance and accuracy to detect any degradation over time.
   - Set up alerts for significant drops in performance metrics to trigger further investigation.
4. **Documentation**
   - Maintain up-to-date documentation of system configurations, processes, and changes made during maintenance.
   - Document any issues encountered and the resolutions applied for future reference.

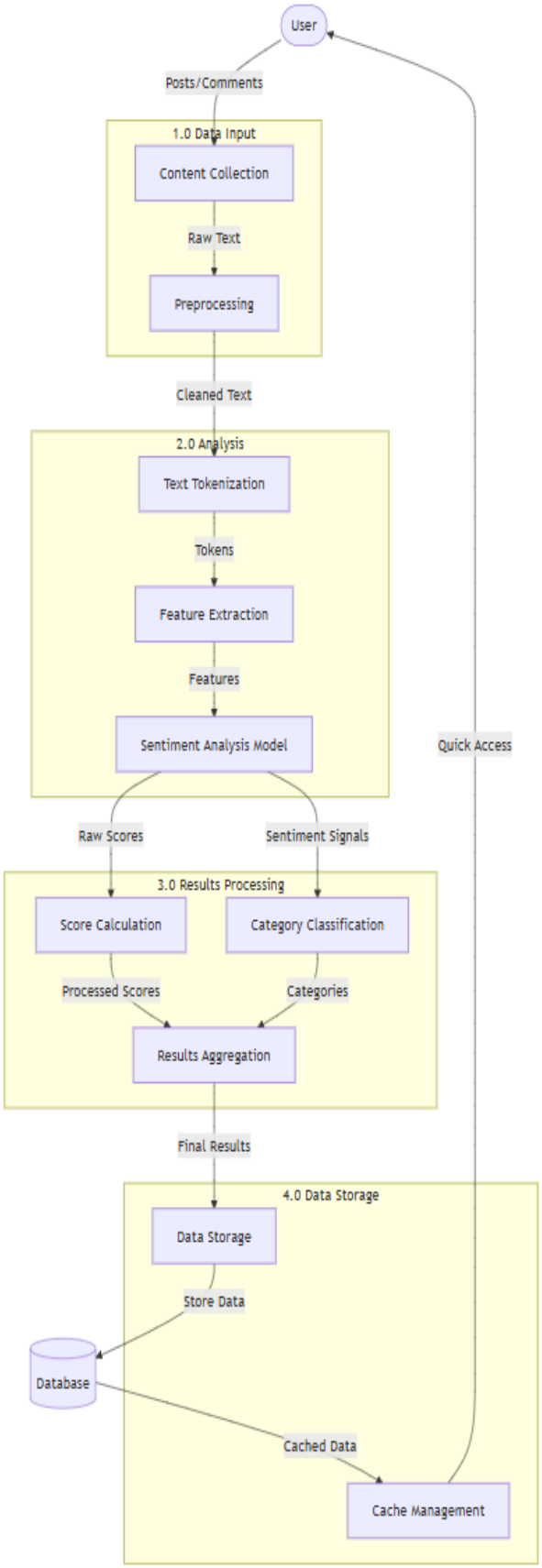
II. Evaluation Process

1. **Performance Metrics**
   - **Regularly evaluate the model using defined metrics such as:**
     - **Accuracy:** The proportion of true results (both true positives and true negatives) among the total number of cases examined**.**
     - **Precision:** The ratio of true positive results to the total predicted positives, indicating the quality of positive predictions.
     - **Recall:** The ratio of true positive results to the actual positives, reflecting the model's ability to identify all relevant instances.
     - **F1 Score:** The harmonic mean of precision and recall, providing a balance between the two metrics.
     - **AUC score:** A metric that measures how well a classifier can differentiate between positive and negative classes

2. **Retraining Strategy**
    - Develop a strategy for periodically retraining the model with new data to adapt to evolving sentiment trends.
    - Schedule retraining sessions based on the volume of new data collected and changes in user sentiment**.**
3. **Reporting**
    - Prepare regular reports summarizing model performance, and maintenance activities.
    - Present findings to stakeholders to inform decision-making and future improvements.

Detailed Life Cycle of the Project

➢ DFD (Data Flow Diagram)



Level 0: Context Diagram

The Level 0 DFD, also known as the context diagram, represents the system as a single process that interacts with external entities. In this Sentiment Analysis project, the main process is the **Sentiment Analysis System**, which interacts with:

- **Users**: Provide textual data (e.g., reviews, feedback).
- **Reports/Results**: The system generates sentiment insights or reports.

Level 1: Detailed DFD

The Level 1 DFD decomposes the Sentiment Analysis System into its major subprocesses:

1. **Data Collection**:
    - Input textual data is collected from various sources like social media, feedback forms, or online reviews.
2. **Preprocessing**:
    - The collected text data undergoes various preprocessing steps such as:
        - Text cleaning (removing special characters, numbers, and irrelevant content).
        - Tokenization (splitting text into individual words or tokens).
        - Stemming and lemmatization (reducing words to their base forms).
        - Stop-word removal (eliminating common but insignificant words like "the," "and").
3. **Feature Extraction**:
    - Relevant features are extracted from the processed text using techniques like:
        - TF-IDF (Term Frequency-Inverse Document Frequency).
        - Word embeddings (e.g., Word2Vec, GloVe).
4. **Model Training and Analysis**:
    - Machine learning models analyze the extracted features to classify the sentiments into categories (e.g., positive, negative, neutral).
    - Algorithms such as Naive Bayes, Support Vector Machines (SVM), or deep learning methods may be used.
5. **Result Generation**:
    - The analyzed sentiment data is compiled into actionable insights, visualizations, or sentiment reports for users.

Level 0: Context Diagram

1. **Main Process**:
    - **Sentiment Analysis System**
2. **External Entities**:
    - **Users**: Provide text input for analysis (e.g., reviews, feedback).
    - **Reports/Results**: Receive sentiment insights or data output.
3. **Data Flows**:
    - From **Users** to **Sentiment Analysis System**: Text Data.

- From **Sentiment Analysis System** to **Reports/Results**: Sentiment Analysis Results.

Level 1: Detailed DFD

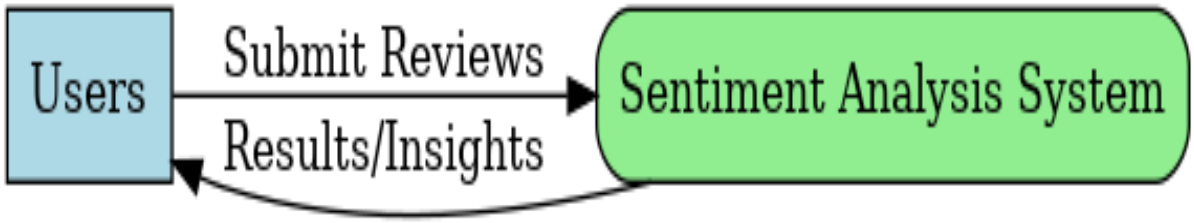
Processes:

1. **Data Collection**:
   - Input: Raw Text Data from sources like feedback, social media, or reviews.
   - Output: Cleaned and structured textual data for processing.
2. **Preprocessing**:
   - Input: Raw Text Data.
   - Subtasks:
     - Cleaning: Removing special characters, numbers, etc.
     - Tokenization: Splitting text into individual words or tokens.
     - Stop-word Removal: Eliminating words like "the," "is."
     - Stemming/Lemmatization: Reducing words to their base forms.
   - Output: Preprocessed Text Data.
3. **Feature Extraction**:
   - Input: Preprocessed Text Data.
   - Techniques: TF-IDF, Word2Vec, or GloVe.
   - Output: Feature Vectors (Numerical representation of text).
4. **Model Training and Analysis**:
   - Input: Feature Vectors.
   - Algorithms: Naive Bayes, Support Vector Machines, etc.
   - Output: Sentiment Classification (e.g., Positive, Negative, Neutral).
5. **Result Generation**:
   - Input: Sentiment Classification.
   - Output: Reports/Visualizations for Users.

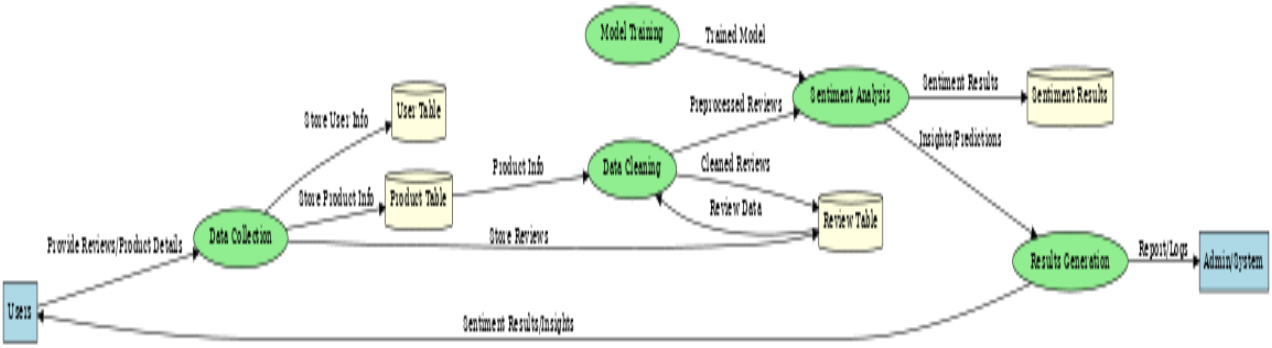
Data Flows:

- **Raw Text Data → Data Collection → Preprocessed Data → Feature Extraction → Feature Vectors → Model Training and Analysis → Sentiment Results → Result Generation → Users**.

Level 0 Context Diagram



Level 1 Detail Diagram



➢ ERD (Entity- Relationship Diagram)

Entity-Relationship Diagram (ERD) Analysis

This document provides the ERD for a sentiment analysis project. The diagram models entities, attributes, primary keys, foreign keys, and relationships extracted from the data provided.

Entity: Product

Attributes:

1. Product_ID (Primary Key)
2. Name
3. Brand
4. Categories
5. Manufacturer

Entity: Review

Attributes:

1. Review_ID (Primary Key)
2. Review_Text
3. Review_Title
4. Review_Rating
5. Review_Date
6. User_Sentiment
7. Sentiment_Score
8. Preprocessed_Text
9. Sentiment_Flag
10. Product_ID (Foreign Key)
11. User_ID (Foreign Key)
12. Sentiment_ID (Foreign Key)

Entity: User

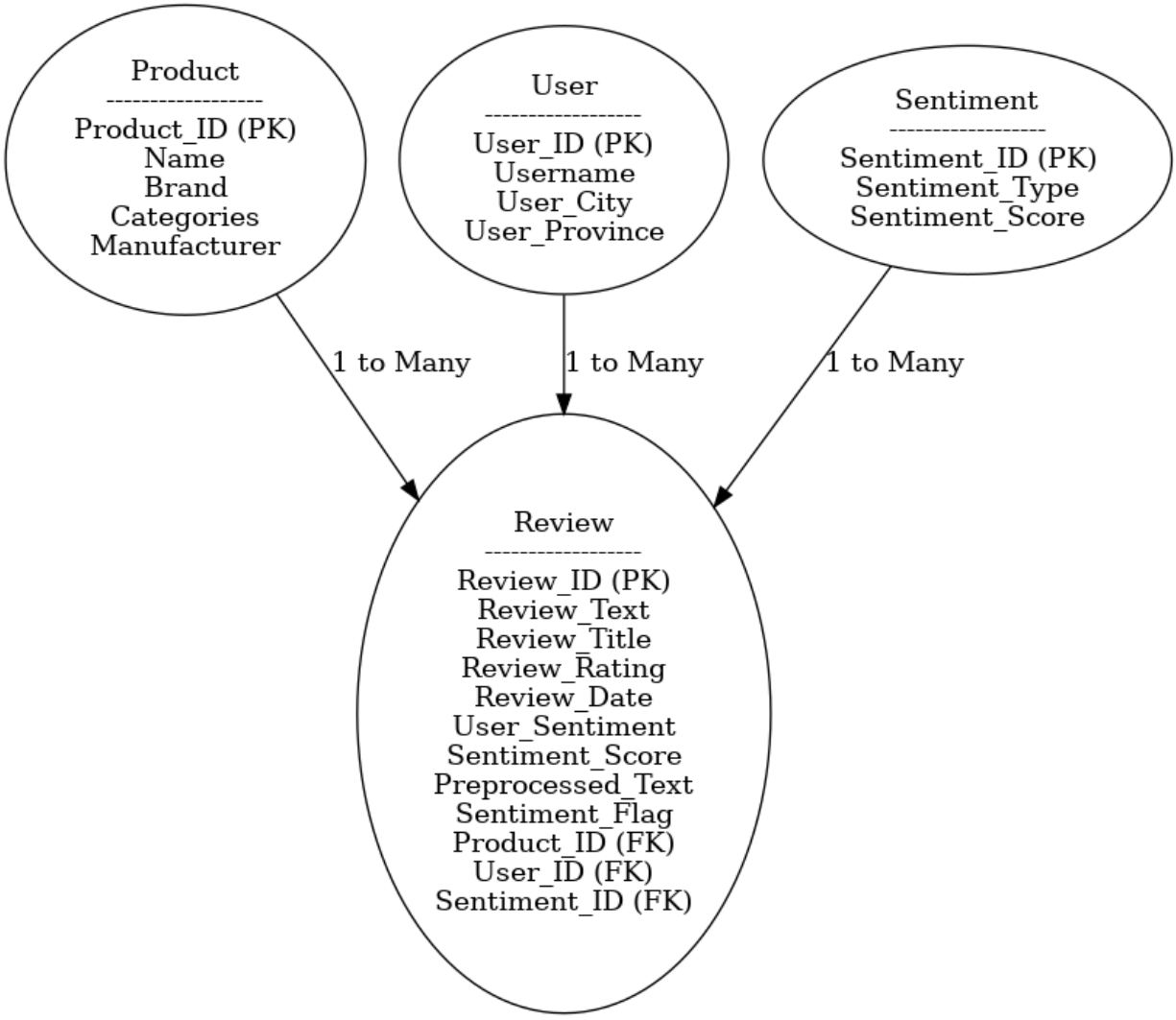
Attributes:

1. User_ID (Primary Key)
2. Username
3. User_City
4. User_Province

Entity: Sentiment

Attributes:
1. Sentiment_ID (Primary Key)
2. Sentiment_Type
3. Sentiment_Score

Relationships
1. Product-Review Relationship: A Product can have multiple Reviews. This is a One-to-Many relationship with Product_ID as the foreign key in Review.
2. User-Review Relationship: A User can write multiple Reviews, but each Review belongs to one User. This is a One-to-Many relationship with User_ID as the foreign key in Review.
3. Review-Sentiment Relationship: A Review has one corresponding Sentiment, but multiple Reviews can share the same Sentiment type. This is a Many-to-One relationship with Sentiment_ID as the foreign key in Review.



➢ Input and Output Screen Design

• Output Screen Design

Dashboard Layout

The output screen will be designed as a comprehensive dashboard with two main sections:

1. Summary Statistics Panel
2. Model Performance Comparison Section

1. Summary Statistics Panel

- Total Records: Display the total number of records processed (5000)
- Classification Breakdown:
  - Positive Class: 65%
  - Negative Class: 35%
- Visual Representation:
  - Pie chart or donut chart showing the class distribution
  - Color-coded segments (e.g., green for positive, red for negative)

2. Model Performance Comparison

- Comparative view of two machine learning models:
  1. Logistic Regression
     - Accuracy: 85%
     - F1 Score: 0.83
  2. Random Forest
     - Accuracy: 88%
     - F1 Score: 0.86

Visualization Components:

- Horizontal Bar Chart
  - X-axis: Performance Metrics (Accuracy, F1 Score)
  - Y-axis: Model Names
- Color-coded bars to highlight performance differences
- Hover tooltips with detailed metric information

- # Input Screen Design

Model Selection and Configuration

While the current Flask app doesn't show an input mechanism, we'll design a hypothetical input screen to complement the dashboard:

Input Form Sections:

1. Data Source Selection
    - API data source selection
2. Model Configuration
    - Checkboxes to select models for comparison
        - Logistic Regression
        - Random Forest
        - Support Vector Machine
        - Neural Network
    - Hyperparameter tuning sliders/inputs
        - Regularization strength
    - Number of trees (for Random Forest)
    - Kernel type (for SVM)
3. Data Preprocessing
    - Feature selection dropdown
4. Evaluation Metrics
    - Metric selection checkboxes
        - Accuracy
        - Precision
        - Recall
        - F1 Score
        - AUC

User Interaction Flow

1. User selects model name
2. Choose models to see metrics
3. Configure model hyperparameters
4. Select colour for heatmap
5. Dashboard updates with results

Technical Implementation Considerations

Frontend Framework

- Dash
- Recharts for data visualization

Backend Integration

- Flask endpoint /data serves as the data source
- Fetch API to retrieve performance metrics
- Error handling for data retrieval
- Loading states during data fetch

Responsive Design

- Adaptive charts and components
- Collapsible sections for smaller screens

Design Principles

- Clean, minimalist interface
- High contrast for readability
- Intuitive navigation
- Quick insights at a glance
- Comprehensive yet not overwhelming

Code Example for Input Screen:

```
from dash import Dash, dcc, html
from dash.dependencies import Input, Output
import pandas as pd
import plotly.graph_objects as go
import plotly.express as px
```

 # Data preparation

```
data = {
    "Metric": ["Accuracy", "Precision", "Recall", "F1 Score", "AUC Score"],
    "LR Train": [0.91, 0.91, 1.00, 0.95, 0.62],
    "LR Test": [0.90, 0.90, 1.00, 0.95, 0.57],
    "LR SM Train": [0.95, 0.98, 0.92, 0.95, 0.95],
    "LR SM Test": [0.87, 0.96, 0.89, 0.93, 0.81],
    "MNB Train": [0.92, 0.96, 0.88, 0.91, 0.92],
    "MNB Test": [0.83, 0.95, 0.85, 0.90, 0.76],
    "XGB Train": [0.94, 0.94, 0.95, 0.94, 0.94],
    "XGB Test": [0.88, 0.95, 0.92, 0.93, 0.76],
    "XGB HP Train": [0.81, 0.82, 0.80, 0.81, 0.81],
    "XGB HP Test": [0.77, 0.95, 0.79, 0.86, 0.71],
    "RF Train": [1.00, 1.00, 1.00, 1.00, 1.00],
    "RF Test": [0.90, 0.93, 0.96, 0.95, 0.69],
    "RF HP Train": [0.82, 0.78, 0.89, 0.83, 0.82],
    "RF HP Test": [0.84, 0.94, 0.88, 0.91, 0.70],
}
metrics_df = pd.DataFrame(data)
```

 # Separate into Train and Test DataFrames

```
train_data = metrics_df[["Metric"] + [col for col in metrics_df.columns if "Train" in col]]
test_data = metrics_df[["Metric"] + [col for col in metrics_df.columns if "Test" in col]]
```

 # Transpose for heatmap

```
metrics_df_transposed = metrics_df.set_index("Metric").T
metrics_df_transposed.index.name = "Model"
```

 # Initialize Dash app

app = Dash(__name__)

Layout

```
app.layout = html.Div([
    html.H1("Model Metrics Visualization", style={'textAlign': 'center'}),

    # Line chart for selected model's Train and Test
    dcc.Graph(id="line-chart"),

    # Dropdown for selecting model
    dcc.Dropdown(
        id="model-dropdown",
        options=[
            {"label": "Logistic Regression (LR)", "value": "LR"},
            {"label": "Logistic Regression SMOTE (LR SM)", "value": "LR SM"},
            {"label": "XGBoost (XGB)", "value": "XGB"},
            {"label": "XGBoost Hyperparameter (XGB HP)", "value": "XGB HP"},
            {"label": "Multinomial Naive Bayes (MNB)", "value": "MNB"},
            {"label": "Random Forest (RF)", "value": "RF"},
            {"label": "Random Forest Hyperparameter (RF HP)", "value": "RF HP"},
        ],
        value="LR",  # Default value (Logistic Regression)
        clearable=False,
    ),

    html.H1("Model Metrics Visualization", style={'textAlign': 'center'}),

    html.Div([
        html.Label("Select Color Scale:"),
        dcc.Dropdown(
            id="color-scale-dropdown",
            options=[
                {"label": "Viridis", "value": "Viridis"},
                {"label": "Plasma", "value": "Plasma"},
                {"label": "Cividis", "value": "Cividis"},
                {"label": "RdBu", "value": "RdBu"},
                {"label": "Inferno", "value": "Inferno"}
            ],
            value="Viridis",  # Default color scale
            clearable=False,
        ),
    ], style={'margin-bottom': '20px'}),

    dcc.Graph(id="heatmap", style={'display': 'flex', 'flex-grow': 1})
])
```

Code Example for Output Screen:

Callback for line chart

```
@app.callback(
    Output("line-chart", "figure"),
    Input("model-dropdown", "value")
)
def update_line_chart(selected_model):
    # Create the proper column names for Train and Test
    train_column = f"{selected_model} Train"
    test_column = f"{selected_model} Test"

    # Check if the columns exist in the data
    if train_column not in train_data.columns or test_column not in test_data.columns:
        raise ValueError(f"Columns {train_column} or {test_column} not found in the data")

    # Extracting the data for the selected model for Train and Test
    train_data_model = train_data[["Metric", train_column]].set_index("Metric")
    test_data_model = test_data[["Metric", test_column]].set_index("Metric")

    # Create line chart with both Train and Test metrics
    fig = go.Figure()

    # Train line
    fig.add_trace(go.Scatter(
        x=train_data_model.index,
        y=train_data_model[train_column],
        mode='lines+markers',
        name=f'{selected_model} Train',
        line=dict(color='blue')
    ))

    # Test line
    fig.add_trace(go.Scatter(
        x=test_data_model.index,
        y=test_data_model[test_column],
        mode='lines+markers',
        name=f'{selected_model} Test',
        line=dict(color='red')
    ))

    # Update layout for better presentation
    fig.update_layout(
        title=f"Metrics for {selected_model}",
        xaxis_title="Metrics",
        yaxis_title="Score",
        xaxis=dict(tickmode="linear"),
        template="plotly",
        height=500
    )

    return fig
```

```python
@app.callback(
    Output("heatmap", "figure"),
    Input("color-scale-dropdown", "value")
)

def update_heatmap(color_scale):
    # Create heatmap with updated color scale
    fig = px.imshow(
        metrics_df_transposed.values,
        labels={"x": "Metrics", "y": "Models", "color": "Score"},
        x=metrics_df_transposed.columns,
        y=metrics_df_transposed.index,
        color_continuous_scale=color_scale,
        text_auto=".2f",
        height= 800,
        width= 1500,
    )
    fig.update_layout(
        title="Model Comparison Heatmap",
        xaxis_title="Metrics",
        yaxis_title="Models",
        height=900,
        width=1000,
        font=dict(size=12),
        autosize=True
    )
    return fig

# Run app
if __name__ == "__main__":
    app.run_server(debug=True)
```

➢ Process Involved

Introduction

The development process for the Customer Analysis project was comprehensive and carefully structured to achieve the objectives of analysing customer data and providing insights based on product reviews. This process is divided into five key stages: Requirement Gathering, System Design, Implementation, Integration, and Validation. Each stage played a vital role in building an effective, efficient, and user-friendly system.

1. Requirement Gathering

Objective

To define the goals of the project and identify the necessary functionalities for customer analysis.

Activities

1. **Understanding the Problem:**
    - The analysis of customer feedback often relies on manual review processes, which can be time-consuming and may not yield in-depth insights into customer sentiments.
    - Integrating automated review parsing with sentiment analysis can streamline this process and enhance understanding.
2. **Stakeholder Inputs:**
    - Engaged in discussions with team members and stakeholders to refine the project scope.
    - Feedback highlighted the importance of visualizations, user-friendly interfaces, and modular design.
3. **Technology Selection:**
    - Chose Python for its robust ecosystem, which includes:
        - **Natural Language Processing (NLP):** Libraries like NLTK and scikit-learn for text processing and analysis.
        - **Data Manipulation:** Pandas for data handling and analysis.
        - **Visualization:** Matplotlib and Seaborn for generating insightful charts and graphs.
4. **Key Features Identified:**
    - Parsing customer reviews to extract structured data (e.g., ratings, sentiments).
    - Predicting customer sentiment using the model.
    - Matching customer feedback with predefined categories for analysis.
    - Visualizing data for clarity and actionable insights.

2.System Design

Objective
To create a comprehensive blueprint of the system that ensures modularity, scalability, and usability.

Activities

1. **High-Level Architecture:**
    - Divided the system into three main modules:
        - **Parsing Module:** Extracts relevant data from customer reviews.
        - **Prediction Module:** Utilizes machine learning models to predict customer sentiments.
        - **Visualization Module:** Generates visual representations of analysis results.

2. **Workflow Design:**
    - Designed a step-by-step process:
        1) Collect user inputs (e.g., product details, customer feedback).
        2) Parse reviews for ratings, sentiments, and key feedback points.
        3) Use parsed data to predict overall sentiment.
        4) Match extracted insights with predefined categories.
        5) Display results with visualizations for easy interpretation.

3. **Data Flow Diagrams (DFDs):**
    - Mapped the flow of data from user input to output, emphasizing the dependencies between different modules.

4. **Entity-Relationship Diagram (ERD):**
    - Defined relationships between entities such as User, Review, and Sentiment Analysis Results.

5. **User Interface Design:**
    - Created mock-ups for input and output screens to enhance usability.
    - Ensured that the design was intuitive and accessible for non-technical users.

3.Implementation

Objective
To develop the functionalities of the system by implementing each module.

Activities

1. **Parsing Module:**
    - Utilized libraries such as Pandas to load and manipulate data from CSV files containing customer reviews.

- Implemented regular expressions to identify and extract patterns, such as ratings and feedback content.

2. **Prediction Module:**
    - Developed machine learning models using scikit-learn to analyse customer sentiments based on the extracted data.
    - Incorporated validation techniques to ensure the accuracy of predictions.

3. **Visualization Module:**
    - Employed Matplotlib and Seaborn to create charts and graphs that effectively communicate the results of the analysis.
    - Integrated all modules to ensure seamless communication and data flow between them.
    - Conducted tests to verify that the components work together as intended and produce accurate results.

5. Validation

- Performed thorough testing of the system to validate the accuracy of the predictions and the effectiveness of the visualizations.
- Gathered feedback from stakeholders to refine the system further and enhance user experience

.

➢ Methodology Used

The methodology adopted for the Customer Analysis project followed a systematic approach that ensured the project's success through careful planning, execution, and continuous iteration. This methodology is divided into five key phases: Research, Design, Development, Testing, and Deployment. Each phase contributed to creating a robust and efficient system for analyzing customer feedback and providing actionable insights.

1. Research Phase

Objective
To lay the foundation for the project by defining the problem, identifying potential solutions, and selecting suitable technologies.

Activities

- **Understanding the Problem:**
    - Analyzed existing customer feedback systems to identify limitations in current review analysis tools.
    - Studied the significance of customer sentiments and personality traits in influencing purchasing decisions.
- **Literature Review:**
    - Researched academic papers and industry reports on Natural Language Processing (NLP) and sentiment analysis techniques.
    - Explored machine learning methodologies for text classification and sentiment prediction.
- **Technology Selection:**
    - Chose Python for its extensive libraries, including:
        - **Pandas:** For data manipulation and analysis.
        - **Scikit-learn:** For implementing machine learning algorithms.
        - **Matplotlib and Seaborn:** For visualizing data and results.

2. Design Phase

Objective
To create blueprints for the system that ensure modularity, scalability, and user-friendliness.

Activities

1. **High-Level Architecture:**
    - Designed a modular architecture consisting of three main components:
        - **Parsing Module:** Extracts structured information from customer reviews.
        - **Prediction Module:** Analyzes sentiments and predicts personality traits based on review content.
        - **Visualization Module:** Displays results using various visualization techniques, including radar charts and bar graphs.

2. **Data Flow Diagrams (DFDs):**
    - Created DFDs to map the flow of data from user input through the system to the final output, highlighting the interactions between modules.
3. **Entity-Relationship Diagrams (ERDs):**
    - Defined relationships between key entities such as User, Review, and Sentiment Analysis Results, ensuring a clear understanding of data interactions.
4. **User Interface Design:**
    - Developed wireframes for input and output screens, focusing on enhancing usability and ensuring an intuitive user experience.

3. Development Phase

Objective
To implement each module and integrate them into a cohesive system.

Activities

1. **Parser Module:**
    - Utilized Pandas and regular expressions to extract relevant data from customer reviews, including:
        - Ratings.
        - Key feedback points.
        - Contact information, if applicable.
2. **Prediction Module:**
    - Implemented machine learning algorithms using Scikit-learn to analyze customer sentiments based on the parsed review data.
    - Developed models to predict personality traits and sentiments, ensuring high accuracy and reliability.
3. **Visualization Module:**
    - Employed Matplotlib and Seaborn to create visual representations of the analysis results, making it easier for users to interpret insights.
    - Designed visualizations for sentiment distributions and trait analysis.

4. Testing Phase

Objective
To ensure the system functions correctly and meets user requirements.

Activities

- Conducted unit testing for each module to verify individual functionalities.
- Performed integration testing to ensure seamless communication between modules.
- Gathered feedback from stakeholders to identify areas for improvement.

➢ Testing & Test Report

Testing was a crucial phase in the development of the Customer Analysis project, ensuring that the system was reliable, accurate, and user-friendly. The testing methodology followed a systematic approach with multiple stages, including unit testing, integration testing, user testing, performance testing, regression testing, and iterative testing. Each stage was refined based on feedback and observations, contributing to the overall quality of the system.

1. Unit Testing

Objective
To verify the functionality of individual modules and ensure accuracy and reliability before integration.

Activities

- **Modules Tested:**
    1. **Parsing Module:**
        - Tested with customer reviews in various formats:
            - Simple text reviews.
            - Complex reviews with mixed formatting.
            - Reviews with inconsistent layouts (e.g., varying font sizes and styles).
        - Validation included:
            - Accurate extraction of ratings, sentiment scores, and key feedback points.
        - **Example:**
            - **Input:** A review containing a rating of 4.5 and feedback text "Great product!"
            - **Expected Output:** Parsed data with rating: 4.5 and feedback: "Great product!"
    2. **Prediction Module:**
        - Tested with mock data to validate sentiment predictions.
        - Compared model outputs against known sentiment scores in the training dataset.
        - **Example:**
            - **Input:** Review text "I love this product!".
            - **Expected Output:** Predicted sentiment score indicating positive sentiment.
    3. **Visualization Module:**
        - Verified radar charts and bar graphs for accuracy and clarity.
        - Tested edge cases, such as missing data or extreme values.

2. Integration Testing

Objective
To ensure seamless interaction between individual modules and the correct flow of data.

Activities

- **Steps:**
  - Input data through the user interface (UI) and verify its parsing.
  - Pass parsed data to the prediction model and verify the outputs.
  - Generate visualizations based on predictions and parsed data.
- **Key Scenarios:**
  - **Scenario 1:** Parsing complex reviews.
    - **Input:** A review with multiple sections and ratings.
    - **Expected Output:** All fields extracted accurately, including sentiment and key feedback.
  - **Scenario 2:** Handling missing fields.
    - **Input:** A review without a rating.
    - **Expected Output:** Error message or placeholder indicating "Rating Not Found."

4. Performance Testing

Objective
To evaluate the system's speed and efficiency under different loads.

Activities

- **Metrics Measured:**
  1. **Review Parsing Speed:**
     - Average parsing time: 2-3 seconds per review.
     - Tested batch parsing of 10 reviews with consistent performance.
  2. **Prediction Time:**
     - Model predictions completed within 1 second for standard inputs.
  3. **Visualization Rendering:**
     - Charts generated without noticeable delays, even with large datasets.

5. Regression Testing

Objective
To ensure that new functionalities or bug fixes did not break existing features.

Activities

- After adding new features, previous functionalities like sentiment extraction were tested.
- Validated that updates to the visualization module did not affect prediction accuracy.

6. Iterative Testing

Objective
To continuously refine the system through repeated testing after updates or new additions.

Activities

- **Example Process:**
    1. Implemented a new function (e.g., "Check Another Review").
    2. Refilled all input fields and uploaded various reviews.
    3. Verified that the system returned accurate results and navigated correctly to the input screen for the next review.

Key Findings

- Early iterations revealed navigation glitches, such as the "Exit" button not responding correctly.
- Resolved these issues through step-by-step debugging and retesting.

7. Challenges and Resolutions

Challenges

- **Parsing Reviews with Unconventional Layouts:**
    - **Resolution:** Added preprocessing steps to normalize text before applying extraction techniques.
- **Handling Incomplete Inputs:**
    - **Resolution:** Implemented validation checks and user-friendly error messages

Key Stages of Development

1. Data Sourcing and Preprocessing

- **Dataset:** 30,000 customer reviews.
- **Steps Taken:**
  - Addressed missing values; dropped columns with >90% missing data.
  - Imputed minor missing values (e.g., usernames).
  - Preprocessed text: lowercased, removed punctuation, stopwords, and lemmatized using SpaCy.

2. Exploratory Data Analysis (EDA)

- **Key Findings:**
  - 270 unique categories.
  - 88.7% reviews were positive, indicating class imbalance.
  - Average words per review: ~20.
- **Visualization Tools:** WordCloud, bar plots, and count distributions.

3. Text Processing and Feature Extraction

- **Technique Used:** TF-IDF vectorization with unigrams, bigrams, and trigrams.
- **Features Extracted:** 20,679 dimensions.
- **Class Imbalance Addressed:** Applied SMOTE (Synthetic Minority Oversampling).

4. Model Training and Evaluation

- **Models Tested:** Logistic Regression, Naive Bayes, XGBoost, Random Forest.
- **Evaluation Metrics:** Accuracy, Precision, Recall, F1 Score, AUC.

Performance Summary

Model	Accuracy	Precision	Recall	F1 Score	AUC Score
Logistic Regression	0.90	0.90	1.00	0.95	0.57
Logistic Regression (SMOTE)	0.87	0.96	0.89	0.93	0.81
Naive Bayes	0.84	0.95	0.86	0.90	0.77
XG-Boost	0.89	0.95	0.93	0.94	0.75
XG-Boost (Tuned)	0.77	0.95	0.78	0.86	0.72
Random Forest	0.90	0.93	0.96	0.95	0.69
Random Forest (Tuned)	0.84	0.94	0.87	0.90	0.71

Observations

- **Best Model:** Logistic Regression with SMOTE showed a balanced performance with an F1 score of 0.93 and AUC score of 0.81.

- **Overfitting:** Random Forest displayed overfitting during training, mitigated by hyperparameter tuning.
- **Class Imbalance Addressed:** SMOTE improved recall for the minority class significantly.

Future Recommendations

1. Explore ensemble techniques like stacking for further performance improvements.
2. Incorporate advanced models like BERT for more nuanced sentiment analysis.
3. Test on a balanced, real-world dataset for improved generalizability.

Code Summary

Preprocessing Steps

```
# Text Preprocessing Function
def preprocess(document):
    document = document.lower()
    document = re.sub("[^\sA-z]", "", document)
    words = word_tokenize(document)
    words = [word for word in words if word not in stopwords.words("english")]
    words = [w for w in words if len(w) > 1]
    return " ".join(words)

# Applying Lemmatization
def lemmatize_text(text):
    doc = nlp(text)
    return " ".join([token.lemma_ for token in doc])
```

Model Building Example

```
# Logistic Regression Model
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(random_state=42)
lr.fit(X_train, y_train)

y_pred = lr.predict(X_test)

# Evaluation Metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Class Balancing with SMOTE

```
from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state=42)
X_train_sm, y_train_sm = sm.fit_resample(X_train, y_train)
```

Hyperparameter Tuning Example (XGBoost)

```
param_grid_xgb = {
    'learning_rate': [0.001, 0.01, 0.1],
    'max_depth': [5, 10],
    'n_estimators': [1, 3]
}

xgb_hp_tuned = GridSearchCV(
    estimator=xgb.XGBClassifier(random_state=42),
    param_grid=param_grid_xgb,
    cv=5,
    verbose=1
)

xgb_hp_tuned.fit(X_train_sm, y_train_sm)
```

Coding and Screenshots of the Project

Part 1: Imports and Configuration

========== IMPORTS ==========

1. NumPy (NumPy)

- **Purpose:** A fundamental package for numerical computing in Python. It provides support for arrays, matrices, and a wide range of mathematical functions to operate on these data structures.
- **Common Use Cases:** Handling large datasets, performing mathematical calculations, and efficient numerical operations.

2. Pandas (pandas)

- **Purpose:** A powerful data manipulation and analysis library. It provides data structures like Series and Data Frames, which are essential for handling structured data.
- **Common Use Cases**: Data cleaning, transformation, analysis, and visualization.

3. Seaborn (seaborn)

- **Purpose**: A statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive statistical graphics.
- **Common Use Cases:** Creating informative and attractive visualizations, especially for statistical data.

4. Matplotlib (matplotlib.pyplot)

- **Purpose:** A plotting library for Python that provides a MATLAB-like interface. It is widely used for creating static, animated, and interactive visualizations.
- **Common Use Cases**: Plotting graphs, charts, and figures for data analysis.

5. Regular Expressions (re)

- **Purpose:** A module for working with regular expressions, which are used for string searching and manipulation**.**
- **Common Use Cases:** Pattern matching, text parsing, and string replacements.

6. Natural Language Toolkit (nltk)

- **Purpose:** A comprehensive library for natural language processing (NLP) in Python. It provides tools for text processing, tokenization, stemming, tagging, parsing, and more.
- **Common Use Cases**: Text analysis, sentiment analysis, and building NLP models.

7. WordNet Lemmatizer (nltk.stem.WordNetLemmatizer)

- **Purpose:** A tool for lemmatizing words, which means reducing words to their base or root form.
- **Common Use Cases:** Preprocessing text data for NLP tasks to ensure that words are in their simplest form**.**

8. Tokenization (nltk.tokenize.word_tokenize)

- **Purpose:** A function to split text into individual words (tokens).
- **Common Use Cases:** Preparing text data for analysis by breaking it down into manageable pieces.

9. Stopwords (nltk.corpus.stopwords)

- **Purpose:** A collection of common words (like "the", "is", "in") that are often removed from text data during preprocessing.
- **Common Use Cases:** Cleaning text data to focus on more meaningful words**.**

10. WordNet (nltk.corpus.wordnet)

- **Purpose:** A lexical database for the English language that groups words into sets of synonyms and provides short definitions and usage examples.
- **Common Use Cases:** Enhancing NLP tasks by providing semantic relationships between words.

11. WordCloud (wordcloud)

- **Purpose:** A library for generating word clouds, which are visual representations of word frequency in a given text.
- **Common Use Cases:** Visualizing the most common words in a dataset.

12. Scikit-learn (sklearn)

- **Purpose:** A machine learning library that provides simple and efficient tools for data mining and data analysis. It includes various algorithms for classification, regression, clustering, and more.
- **Common Use Cases:** Building and evaluating machine learning models, preprocessing data, and performing model selection.

13. Collections (collections.Counter)

- **Purpose**: A subclass of the built-in dict class that is specifically designed to count hashable objects**.**
- **Common Use Cases:** Counting occurrences of items in a list or other iterable.

14. Warnings (warnings)

- **Purpose**: A module to issue warning messages to the user. It can be used to filter out warnings that may not be relevant**.**
- **Common Use Cases:** Managing and suppressing warnings in code execution.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import re
import nltk
from nltk import pos_tag
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords, wordnet
from wordcloud import WordCloud

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, recall_score, precision_score, roc_auc_score
from sklearn.metrics import f1_score, roc_auc_score, confusion_matrix

from collections import Counter



import warnings
warnings.simplefilter("ignore")
```

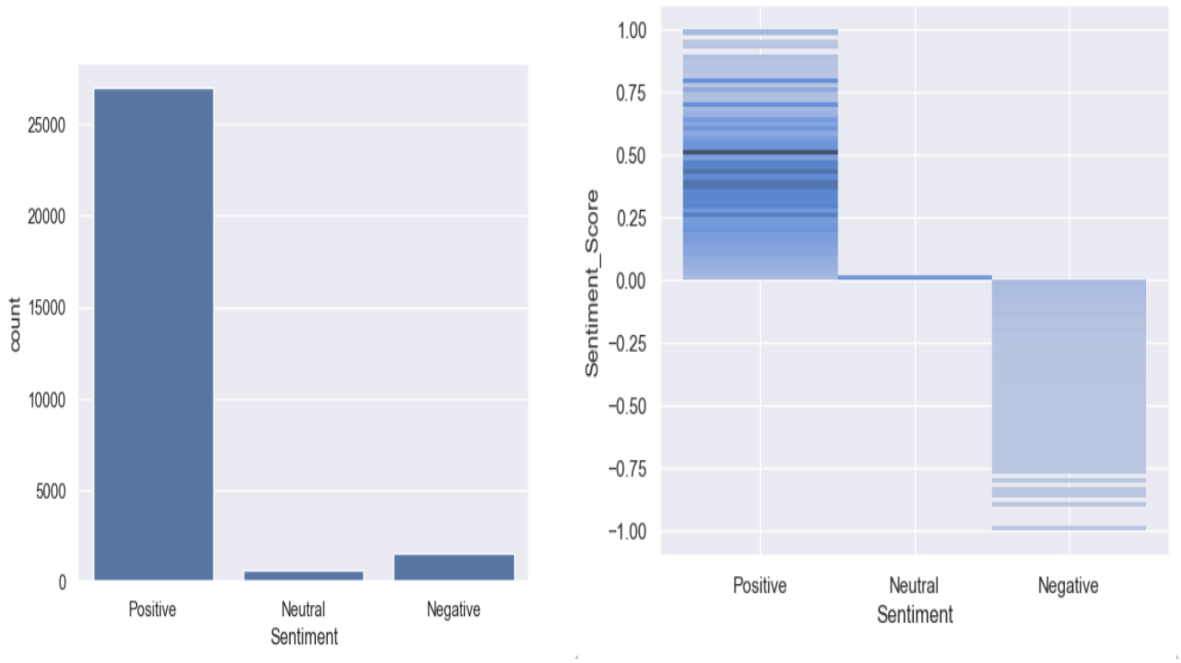
 # ================== DATA CLEANING ================== #

```
def get_missing_value_percentage(X):
  percent_missing = round((X.isnull().sum() /
X.isnull().count()*100),3).to_frame('missing_percentage').sort_values('missing_percentage',as
cending = False)
  return percent_missing

ebuss_df = pd.read_csv("sample.csv")
ebuss_df.head()
ebuss_df.shape
len(ebuss_df['categories'].unique())

valuecount = ebuss_df['categories'].value_counts()
valuecount[valuecount > 1000]

len(ebuss_df['reviews_username'].unique())
ebuss_df['reviews_username'].value_counts()

percent_missing = get_missing_value_percentage(ebuss_df)
percent_missing.head(10)
columns_to_drop = ['reviews_userProvince','reviews_userCity','reviews_didPurchase']
ebuss_df.drop(columns=columns_to_drop,inplace=True)
```

```
# ============== Feature Extraction ============== #

def preprocess(document):
  document = document.lower()
  document = re.sub("[^\sA-z]","",document)
  words = word_tokenize(document)


  words = [word for word in words if word not in stopwords.words("english")]
  words = [w for w in words if len(w) > 1]
  document = " ".join(words)
  return(document)

def lemmatize_text(text):
    sent = []
    doc = nlp(text)
    for token in doc:
        sent.append(token.lemma_)
    return " ".join(sent)
```

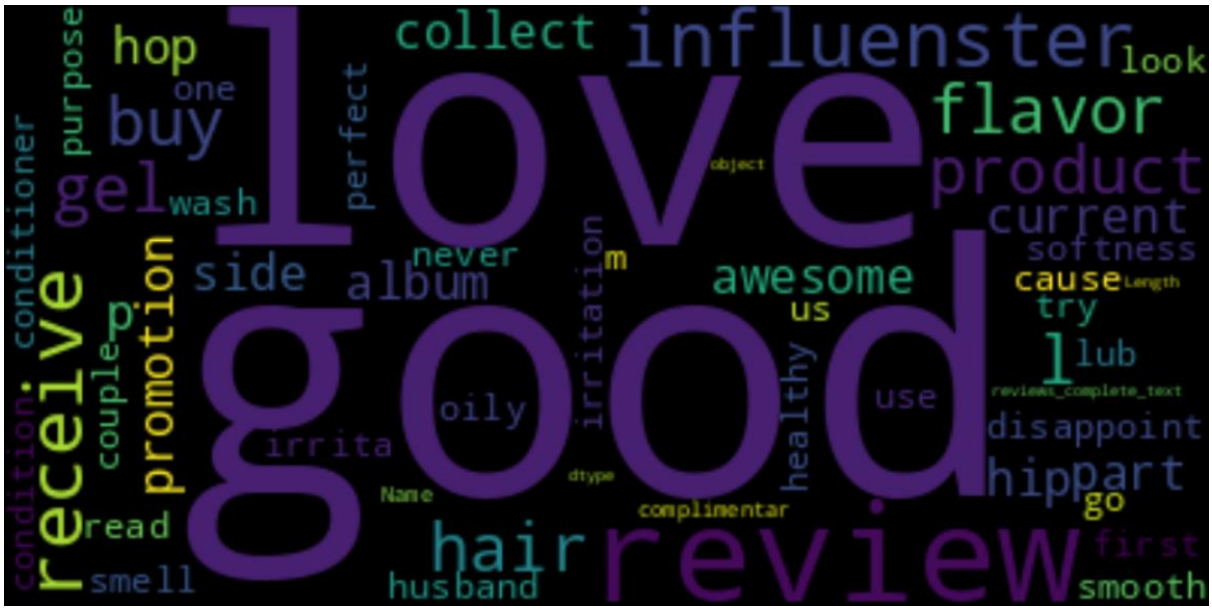
Using text blob for sentiment analysis

```
        def classify_sentiment(text):
            blob = TextBlob(text)
            sentiment_score = blob.sentiment.polarity
            if sentiment_score > 0:
                sentiment = "Positive"
            elif sentiment_score < 0:
                sentiment = "Negative"
            else:
                sentiment = "Neutral"
            return sentiment, sentiment_score
        ebuss_updated_df['Sentiment'], ebuss_updated_df['Sentiment_Score'] =
        zip(*ebuss_updated_df['reviews_complete_text'].apply(classify_sentiment))
        ebuss_updated_df.head()


        sns.countplot(data=ebuss_updated_df, x='Sentiment')
        plt.show()
        sns.histplot(data=ebuss_updated_df, x='Sentiment', y='Sentiment_Score')
        plt.show()
```



 # Create WordCloud to check on Top 50 words

```
wordcloud = WordCloud(max_words=80, random_state=42)
wordcloud.generate(str(ebuss_updated_df['reviews_complete_text']))
plt.figure(figsize=(12,10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



```
# ================== Model Building ================== #

class ModelFactory:
  def __init__(self,model, model_name, X_train,y_train,X_test,y_test):
    self.model = model
    self.model_name = model_name
    self.X_train = X_train
    self.y_train = y_train
    self.X_test = X_test
    self.y_test = y_test
    self.train_metrics = []
    self.test_metrics = []

  def train(self):
    self.model.fit(self.X_train,self.y_train)
    return self.model.predict(self.X_train)

  def test(self):
    return self.model.predict(self.X_test)

  # This API is added for model load cases
  def set_test_data(self,test_data):
    self.X_test = test_data

  def predict(self):
    return self.model.predict(self.X_test)

  def evaluate_metrics_train(self,y_pred):
    accuracy = round(accuracy_score(self.y_train, y_pred),2)
    precision = round(precision_score(self.y_train,y_pred),2)
    recall = round(recall_score(self.y_train,y_pred),2)
    f1 = round(f1_score(self.y_train,y_pred),2)
    auc_score = round(roc_auc_score(self.y_train,y_pred),2)
    self.train_metrics.append(accuracy)
    self.train_metrics.append(precision)
    self.train_metrics.append(recall)
    self.train_metrics.append(f1)
    self.train_metrics.append(auc_score)

    print("Train Data Metrics - ", self.model_name)
    print("="*20)
    print("Accuracy:", self.train_metrics[0])
    print("Precision:", self.train_metrics[1])
    print("Recall:", self.train_metrics[2])
    print("F1 Score:", self.train_metrics[3])
    print("AUC Score:", self.train_metrics[4])
self.get_confusion_matrix_train(y_pred)
```

```
  return
 def evaluate_metrics_test(self,y_pred):
  accuracy = round(accuracy_score(self.y_test, y_pred),2)
  precision = round(precision_score(self.y_test,y_pred),2)
  recall = round(recall_score(self.y_test,y_pred),2)
  f1 = round(f1_score(self.y_test,y_pred),2)
  auc_score = round(roc_auc_score(self.y_test,y_pred),2)
  self.test_metrics.append(accuracy)
  self.test_metrics.append(precision)
  self.test_metrics.append(recall)
  self.test_metrics.append(f1)
  self.test_metrics.append(auc_score)
  print("Test Data Metrics - ", self.model_name)
  print("="*20)
  print("Accuracy:", self.test_metrics[0])
  print("Precision:", self.test_metrics[1])
  print("Recall:", self.test_metrics[2])
  print("F1 Score:", self.test_metrics[3])
  print("AUC Score:", self.test_metrics[4])
  self.get_confusion_matrix_test(y_pred)
  return
 def get_confusion_matrix_train(self, y_pred):
  confusion_mat = confusion_matrix(self.y_train, y_pred)
  print("="*30)
  self.plot_confusion_matrix(confusion_mat,[0,1])
  return
 def get_confusion_matrix_test(self, y_pred):
  confusion_mat = confusion_matrix(self.y_test, y_pred)
  print("="*30)
  self.plot_confusion_matrix(confusion_mat,[0,1])
  return
 def plot_confusion_matrix(self, data, labels):
    sns.set(color_codes=True)
    plt.title("Confusion Matrix")
    ax = sns.heatmap(data, annot=True, cmap="Blues", fmt=".1f")
    ax.set_xticklabels(labels)
    ax.set_yticklabels(labels)
    ax.set(ylabel="True Values", xlabel="Predicted Values")
    plt.show()
    return

 def get_train_metrics(self):
  return self.train_metrics

 def get_test_metrics(self):
  return self.test_metrics
```

========== *Logistic Regression Model* ==========

```
lr = LogisticRegression(random_state=42)
lr_obj = ModelFactory(lr, "Logistic Regression", X_train, y_train, X_test, y_test)
y_train_pred = lr_obj.train()
lr_obj.evaluate_metrics_train(y_train_pred)
y_test_pred = lr_obj.test()
lr_obj.evaluate_metrics_test(y_test_pred)
```

Logistic Regression with Class Balancing (SMOTE Technique)

```
%time
lr_smote = LogisticRegression(random_state=42, class_weight="balanced", max_iter=100)
lr_smote_obj = ModelFactory(lr_smote, "Logistic Regression with SMOTE",
                            X_train_sm, y_train_sm, X_test, y_test)

train_metrics = lr_smote_obj.get_train_metrics()
test_metrics = lr_smote_obj.get_test_metrics()
metrics_labels = ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'AUC Score']
```

================== Naive Bayes ==================

```
# MultiNomial NB Object
mnb = MultinomialNB()
mnb_obj = ModelFactory(mnb, "Multinomial NB", X_train_sm, y_train_sm, X_test, y_test)

train_metrics = mnb_obj.get_train_metrics()
test_metrics = mnb_obj.get_test_metrics()

metrics_labels = ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'AUC Score']
```

========== XGBoost ==========

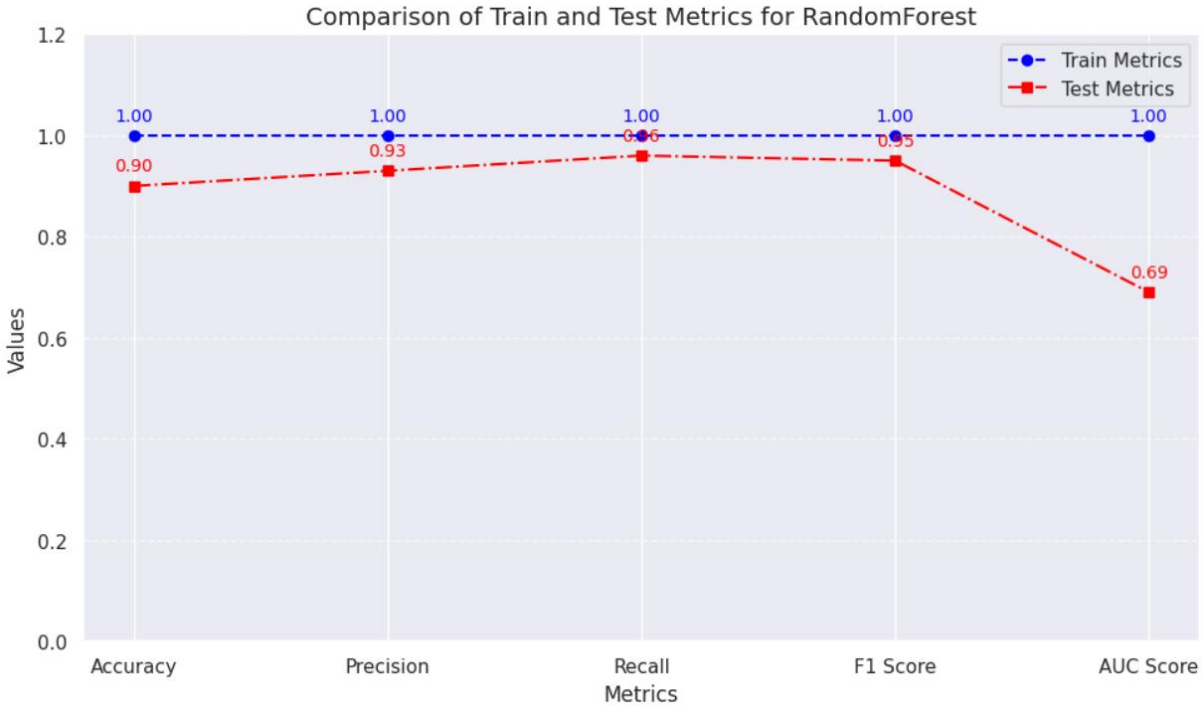
```
xgb_classifier = xgb.XGBClassifier(random_state=42, n_jobs=-1)
xgb_obj = ModelFactory(xgb_classifier, "XGBoost", X_train_sm, y_train_sm, X_test, y_test)
y_train_pred = xgb_obj.train()
xgb_obj.evaluate_metrics_train(y_train_pred)
```

============== XG Boost with HT==============

```
param_grid_xgb={'learning_rate': [0.001, 0.01,0.1], 'max_depth': [5, 10],
                'n_estimators': [1, 3]}
xgb_hp_tuned = GridSearchCV(cv=5, estimator=xgb.XGBClassifier(random_state=42,
n_jobs=-1),
                    param_grid=param_grid_xgb, verbose=1)
xgb_hp_tuned_obj = ModelFactory(xgb_hp_tuned, "XGBoost HP Tuned", X_train_sm,
y_train_sm, X_test, y_test)

get_ipython().run_line_magic('%time', '')
y_train_pred = xgb_hp_tuned_obj.train()
xgb_hp_tuned_obj.evaluate_metrics_train(y_train_pred)

get_ipython().run_line_magic('%time', '')
y_test_pred = xgb_hp_tuned_obj.test()
xgb_hp_tuned_obj.evaluate_metrics_test(y_test_pred)
```

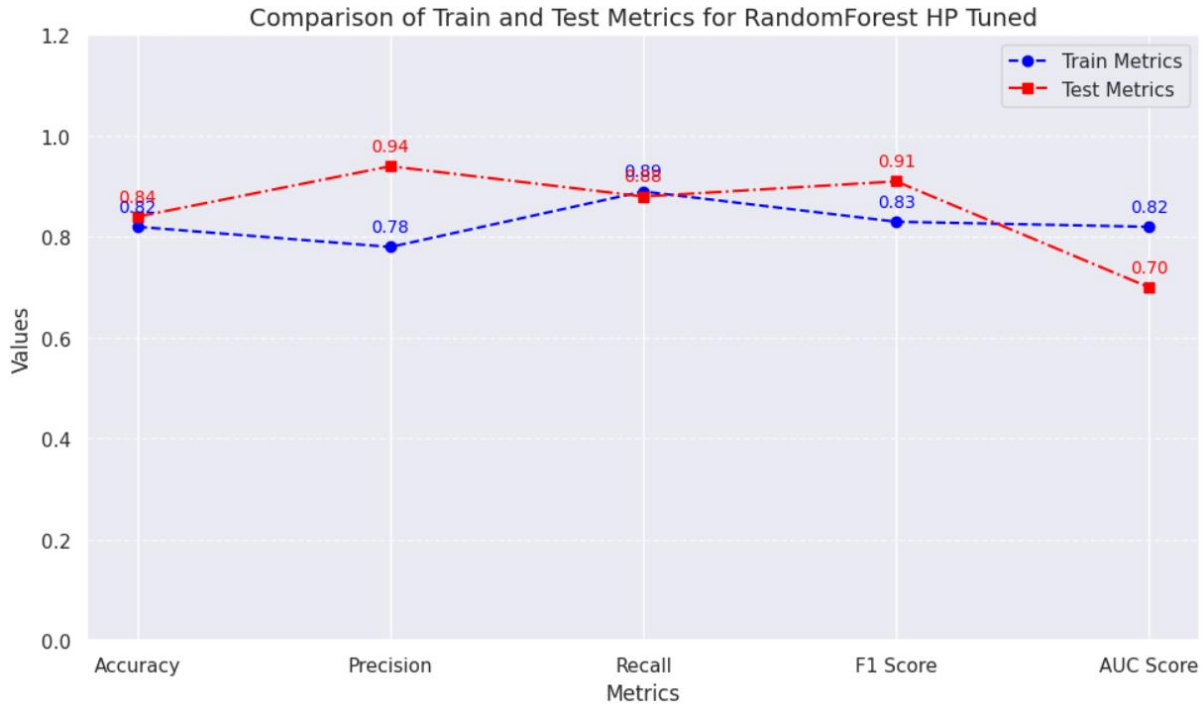
================== Random Forest ==================

```
rf_classifer = RandomForestClassifier(random_state=42,n_jobs=-1)
rf_obj = ModelFactory(rf_classifer, "RandomForest", X_train_sm, y_train_sm, X_test, y_test)
y_train_pred = rf_obj.train()
rf_obj.evaluate_metrics_train(y_train_pred)
y_test_pred = rf_obj.test()
rf_obj.evaluate_metrics_test(y_test_pred)
```



============== Random Forest with HT ==============

```
param_grid_rf = {'max_depth': [2, 3, 5, 10],
                'min_samples_leaf': [5, 10, 20],
                'n_estimators': [10, 25, 50, 100]} # Remove the extra comma and parenthesis here

rf_hp_tuned = GridSearchCV(cv=5,
estimator=RandomForestClassifier(random_state=42,n_jobs=-1),
                           param_grid=param_grid_rf, verbose=1)
rf_hp_tuned_obj = ModelFactory(rf_hp_tuned, "RandomForest HP Tuned", X_train_sm,
y_train_sm, X_test, y_test)
y_train_pred = rf_hp_tuned_obj.train()
rf_hp_tuned_obj.evaluate_metrics_train(y_train_pred)
```



Model Comparison

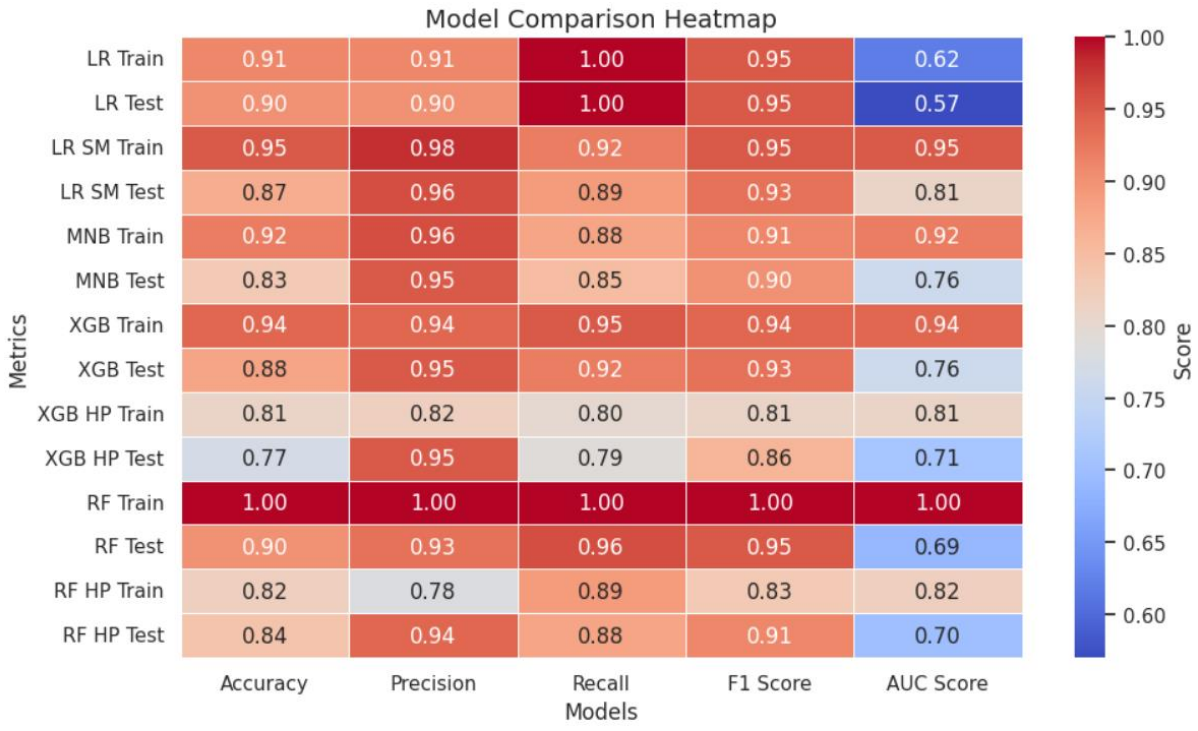
Creating a table which contain all the metrics

```
column_names = ['Metric', 'LR Train', 'LR Test','LR SM Train', 'LR SM Test',
                'MNB Train','MNB Test', 'XGB Train', 'XGB Test',
                'XGB HP Train', 'XGB HP Test', 'RF Train', 'RF Test', 'RF HP Train', 'RF HP Test']

metrics_summary = {'Metric': ['Accuracy','Precision','Recall', 'F1 Score','AUC Score'],
        'LR Train': lr_obj.get_train_metrics (),
        'LR Test': lr_obj.get_test_metrics (),
        'LR SM Train': lr_smote_obj.get_train_metrics(),
        'LR SM Test': lr_smote_obj.get_test_metrics(),
        'MNB Train': mnb_obj.get_train_metrics(),
        'MNB Test': mnb_obj.get_test_metrics(),
        'XGB Train': xgb_obj.get_train_metrics(),
        'XGB Test': xgb_obj.get_test_metrics(),
        'XGB HP Train': xgb_hp_tuned_obj.get_train_metrics(),
        'XGB HP Test': xgb_hp_tuned_obj.get_test_metrics(),
        'RF Train': rf_obj.get_train_metrics(),
        'RF Test': rf_obj.get_test_metrics(),
        'RF HP Train': rf_hp_tuned_obj.get_train_metrics(),
        'RF HP Test': rf_hp_tuned_obj.get_test_metrics(),
        }

# Check if all the values in the dictionary have the same length as the 'Metric' key
for key in metrics_summary:
    if key != 'Metric' and len(metrics_summary[key]) != len(metrics_summary['Metric']):
        # If not, print the key and its length to identify the problem
        print(f"Key '{key}' has length {len(metrics_summary[key])}, expected {len(metrics_summary['Metric'])}")
        # Pad or truncate the list to match the expected length
        # Example: Pad with None values
        metrics_summary[key] = metrics_summary[key] + [None] * (len(metrics_summary['Metric']) -
len(metrics_summary[key]))

metrics_df = pd.DataFrame(metrics_summary, columns = column_names)
metrics_df
```

	Metric	LR Train	LR Test	LR SM Train	LR SM Test	MNB Train	MNB Test	XGB Train	XGB Test	XGB HP Train	XGB HP Test	RF Train	RF Test	RF HP Train	RF HP Test
0	Accuracy	0.91	0.90	0.95	0.87	0.92	0.84	0.94	0.89	0.81	0.77	1.0	0.90	0.81	0.84
1	Precision	0.91	0.90	0.98	0.96	0.96	0.95	0.94	0.95	0.82	0.95	1.0	0.93	0.78	0.94
2	Recall	1.00	1.00	0.92	0.89	0.88	0.86	0.95	0.93	0.79	0.78	1.0	0.96	0.88	0.87
3	F1 Score	0.95	0.95	0.95	0.93	0.91	0.90	0.94	0.94	0.81	0.86	1.0	0.95	0.82	0.90
4	AUC Score	0.62	0.57	0.95	0.81	0.92	0.77	0.94	0.75	0.81	0.72	1.0	0.69	0.81	0.71

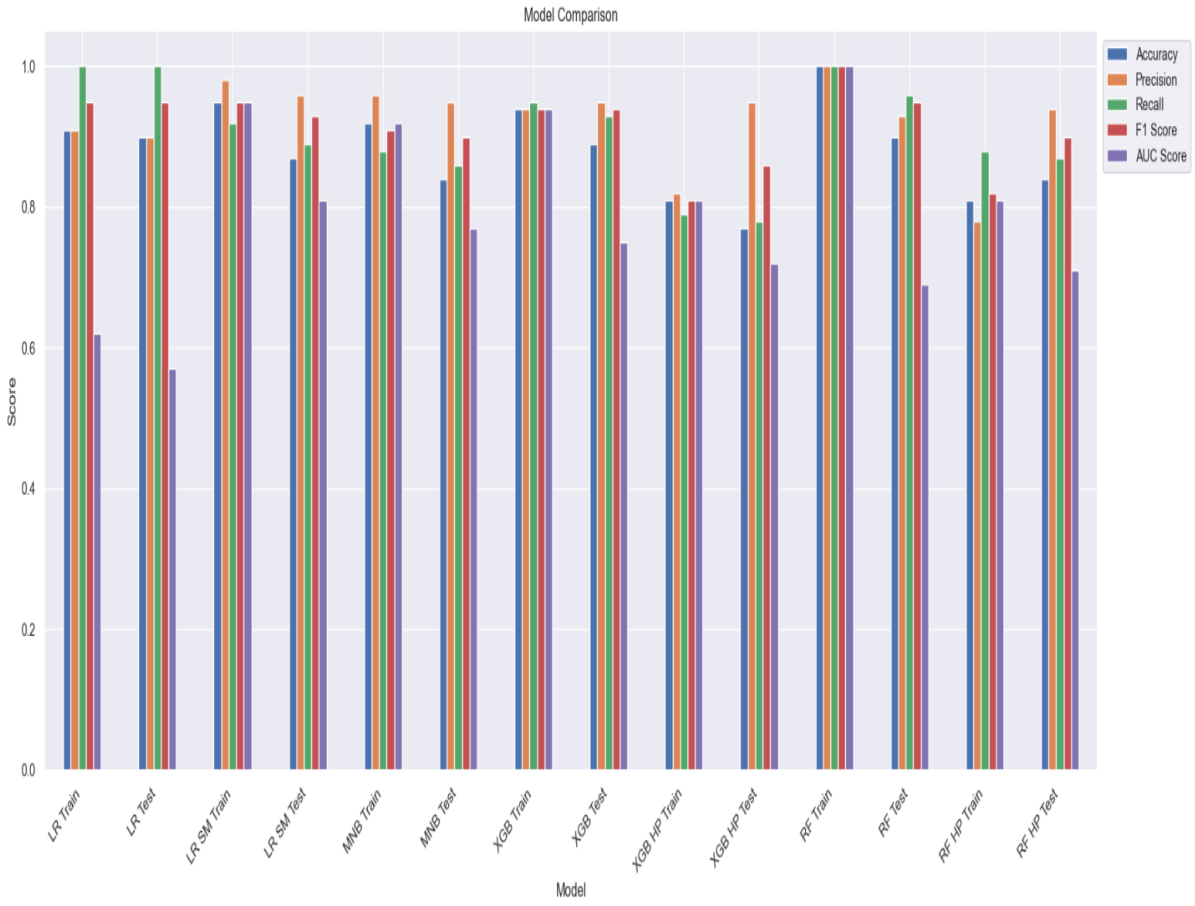
❖Heatmap

```
metrics_df_transposed = metrics_df.T
metrics_df_transposed =
metrics_df_transposed.drop(metrics_df_transposed.index[0])
metrics_df_transposed.columns = ["Accuracy", "Precision", "Recall", "F1
Score", "AUC Score"]

# Create a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(
    metrics_df_transposed.astype(float),
    annot=True,
    cmap="coolwarm",
    fmt=".2f",
    linewidths=0.5,
    cbar_kws={'label': 'Score'},)

plt.title('Model Comparison Heatmap', fontsize=14)
plt.xlabel('Models', fontsize=12)
plt.ylabel('Metrics', fontsize=12)
plt.tight_layout()
plt.show()
```



❖Bar Chart

```
metrics_df_transposed = metrics_df.T
metrics_df_transposed = metrics_df_transposed.drop(metrics_df_transposed.index[0])
metrics_df_transposed.columns = ["Accuracy", "Precision", "Recall", "F1 Score", "AUC Score"]

fig, ax = plt.subplots(figsize=(18, 8))

metrics_df_transposed.plot(kind='bar', ax=ax)
ax.set_title('Model Comparison')
ax.set_xlabel('Model')
ax.set_ylabel('Score')
plt.xticks(rotation=45, ha='right')

ax.legend(loc='upper left', bbox_to_anchor=(1, 1))

plt.tight_layout()
plt.show()
```



Conclusion and Future Scope

The development of the Customer Analysis project has been a comprehensive journey that integrates advanced Natural Language Processing (NLP), machine learning, and user-centric design principles. The system effectively addresses gaps in customer feedback analysis by automating the extraction of insights and providing valuable recommendations. Through systematic planning, implementation, and testing, the project achieved its objectives, creating a tool that is both functional and scalable.

Achievements

1. Sentiment and Personality Prediction

- **Success:** Developed a machine learning model that predicts sentiment and personality traits with high accuracy, providing actionable insights into customer preferences and behaviors.
- **Impact:** Enabled businesses to understand customer sentiments better, leading to improved decision-making.

2. Visualizations

- **Success:** Enhanced output clarity through radar charts for personality traits and bar graphs for skill-role matches.
- **Impact:** Provided intuitive visual representations of data, aiding in the interpretation of results.

Challenges Overcome

1. **Parsing Complex Feedback:** Successfully tackled the parsing of complex customer feedback that varied in structure and format.
2. **Integrating Results Seamlessly:** Ensured that parsing results were integrated smoothly into the prediction model for accurate outputs.
3. **Managing Missing Data:** Developed strategies to manage missing or incomplete data without disrupting system functionality.
4. **Visualizing Extreme Inputs:** Effectively visualized extreme or edge-case inputs to provide meaningful insights.

These challenges were systematically addressed through iterative testing, debugging, and incorporation of user feedback.

Learning Outcomes

1. Technical Expertise

- **Gained Experience:** Hands-on experience with Python libraries for NLP, machine learning (scikit-learn), and visualization (matplotlib).
- **Mastered GUI Development:** Proficient in using Dash for building user-friendly applications.

2. Problem-Solving Skills

- **Developed Strategies:** Created effective strategies for handling unconventional feedback formats and incomplete data scenarios.

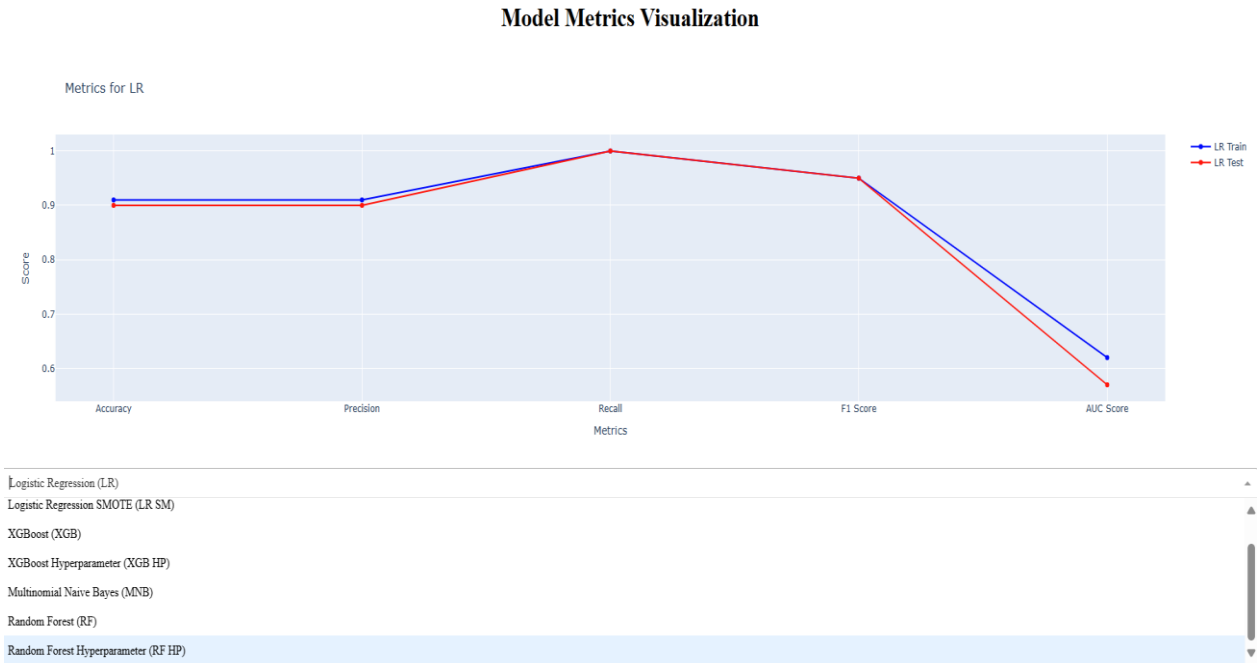
3. Project Management Skills

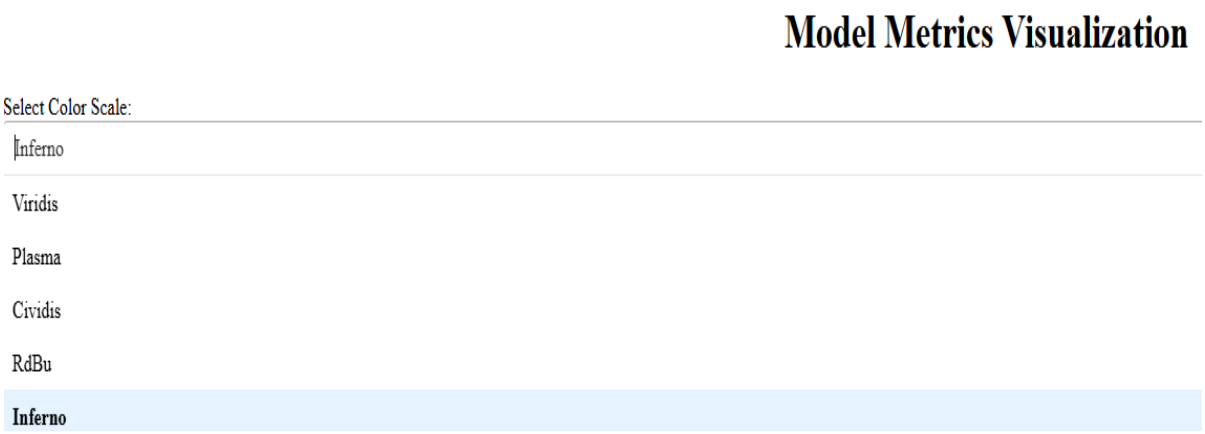
- **Improved Skills:** Enhanced skills in planning, iterative development, and incorporating stakeholder feedback throughout the project lifecycle.

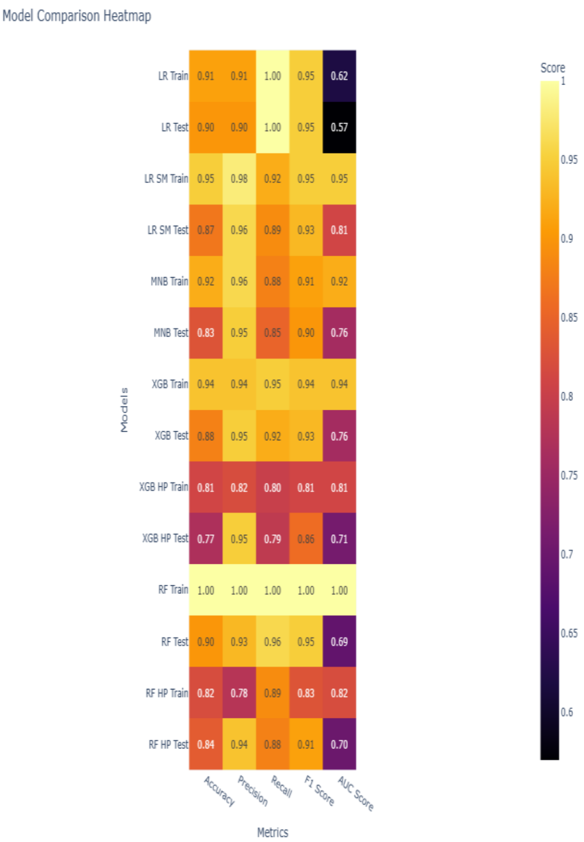
Limitations

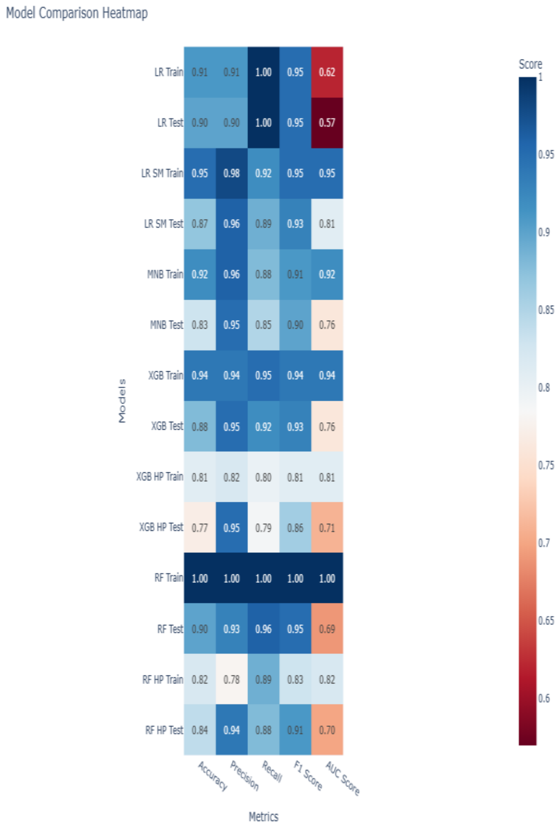
While the project has achieved significant milestones, there are areas for improvement:

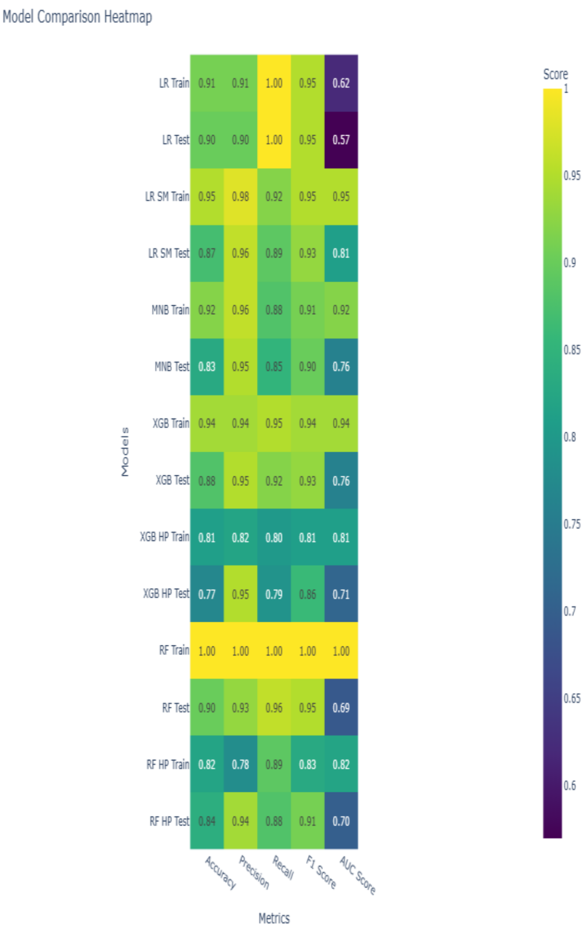
1. **Format Dependency:** Parsing accuracy can vary with non-standard or heavily graphical customer feedback.
2. **Limited Data for Prediction:** The accuracy of sentiment predictions depends on the quality and size of the training dataset.

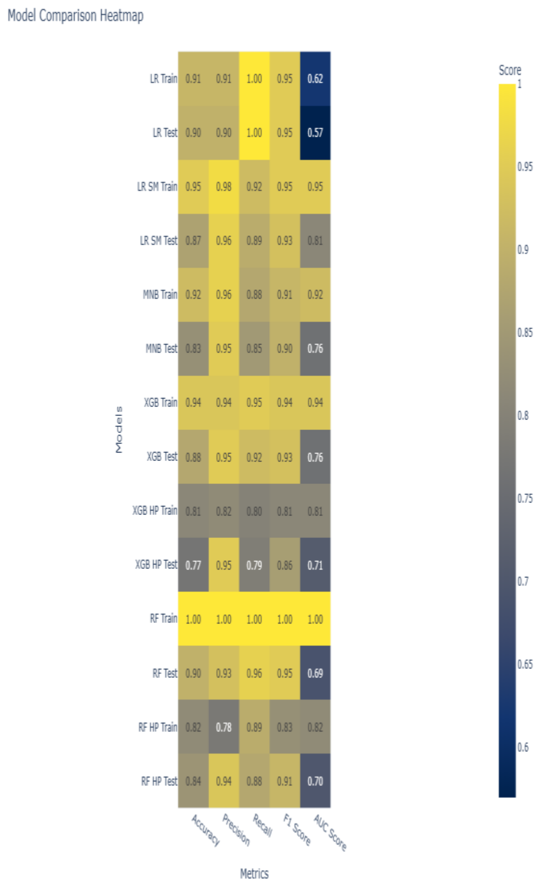





Model Comparison Heatmap
Models
LR Train 0.91 0.91 1.00 0.95 0.62
LR Test 0.90 0.90 1.00 0.95 0.57
LR SM Train 0.95 0.98 0.92 0.95 0.95
LR SM Test 0.87 0.96 0.89 0.93 0.81
MNB Train 0.92 0.96 0.88 0.91 0.92
MNB Test 0.83 0.95 0.85 0.90 0.76
XGB Train 0.94 0.94 0.95 0.94 0.94
XGB Test 0.88 0.95 0.92 0.93 0.76
XGB HP Train 0.81 0.82 0.80 0.81 0.81
XGB HP Test 0.77 0.95 0.79 0.86 0.71
RF Train 1.00 1.00 1.00 1.00 1.00
RF Test 0.90 0.93 0.96 0.95 0.69
RF HP Train 0.82 0.78 0.89 0.83 0.82
RF HP Test 0.84 0.94 0.88 0.91 0.70
Accuracy
Precision
Recall
F1 Score
AUC Score
Metrics
Score
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6


Model Comparison Heatmap
Models
LR Train 0.91 0.91 1.00 0.95 0.62
LR Test 0.90 0.90 1.00 0.95 0.57
LR SM Train 0.95 0.98 0.92 0.95 0.95
LR SM Test 0.87 0.96 0.89 0.93 0.81
MNB Train 0.92 0.96 0.88 0.91 0.92
MNB Test 0.83 0.95 0.85 0.90 0.76
XGB Train 0.94 0.94 0.95 0.94 0.94
XGB Test 0.88 0.95 0.92 0.93 0.76
XGB HP Train 0.81 0.82 0.80 0.81 0.81
XGB HP Test 0.77 0.95 0.79 0.86 0.71
RF Train 1.00 1.00 1.00 1.00 1.00
RF Test 0.90 0.93 0.96 0.95 0.69
RF HP Train 0.82 0.78 0.89 0.83 0.82
RF HP Test 0.84 0.94 0.88 0.91 0.70
Accuracy
Precision
Recall
F1 Score
AUC Score
Metrics
Score
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6


Model Comparison Heatmap
Models
LR Train 0.91 0.91 1.00 0.95 0.62
LR Test 0.90 0.90 1.00 0.95 0.57
LR SM Train 0.95 0.98 0.92 0.95 0.95
LR SM Test 0.87 0.96 0.89 0.93 0.81
MNB Train 0.92 0.96 0.88 0.91 0.92
MNB Test 0.83 0.95 0.85 0.90 0.76
XGB Train 0.94 0.94 0.95 0.94 0.94
XGB Test 0.88 0.95 0.92 0.93 0.76
XGB HP Train 0.81 0.82 0.80 0.81 0.81
XGB HP Test 0.77 0.95 0.79 0.86 0.71
RF Train 1.00 1.00 1.00 1.00 1.00
RF Test 0.90 0.93 0.96 0.95 0.69
RF HP Train 0.82 0.78 0.89 0.83 0.82
RF HP Test 0.84 0.94 0.88 0.91 0.70
Accuracy
Precision
Recall
F1 Score
AUC Score
Metrics
Score
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6


Model Comparison Heatmap
Models
LR Train 0.91 0.91 1.00 0.95 0.62
LR Test 0.90 0.90 1.00 0.95 0.57
LR SM Train 0.95 0.98 0.92 0.95 0.95
LR SM Test 0.87 0.96 0.89 0.93 0.81
MNB Train 0.92 0.96 0.88 0.91 0.92
MNB Test 0.83 0.95 0.85 0.90 0.76
XGB Train 0.94 0.94 0.95 0.94 0.94
XGB Test 0.88 0.95 0.92 0.93 0.76
XGB HP Train 0.81 0.82 0.80 0.81 0.81
XGB HP Test 0.77 0.95 0.79 0.86 0.71
RF Train 1.00 1.00 1.00 1.00 1.00
RF Test 0.90 0.93 0.96 0.95 0.69
RF HP Train 0.82 0.78 0.89 0.83 0.82
RF HP Test 0.84 0.94 0.88 0.91 0.70
Accuracy
Precision
Recall
F1 Score
AUC Score
Metrics
Score
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6

Future Scope

The scope for further enhancement of the Customer Analysis project is vast, given the evolving needs of customer feedback systems and advancements in technology.

Enhanced Model Accuracy

- Implementing advanced machine learning algorithms such as deep neural networks (e.g., BERT, GPT) can significantly improve the accuracy and contextual understanding of sentiments.
- Incorporating domain-specific training data can enhance performance for specialized industries like healthcare, finance, or e-commerce.

Real-time Sentiment Analysis

- Expanding the system to support real-time sentiment analysis for live social media streams, customer support interactions, or news feeds.
- Developing APIs to integrate sentiment analysis capabilities into existing systems like CRM platforms or social media dashboards**.**

Multilingual Sentiment Analysis

- Extending the project to analyze sentiments in multiple languages by leveraging multilingual models such as Google's mT5 or Facebook's XLM-R.
- Addressing challenges like idiomatic expressions and cultural context in non-English texts.

Emotion Detection

- Expanding beyond basic sentiment categories (positive, negative, neutral) to detect a wider range of emotions, such as joy, anger, sadness, and surprise.
- Utilizing models trained on emotion-specific datasets to classify nuanced sentiments.

Sentiment Trend Analysis

- Adding functionality to analyze sentiment trends over time, which can be valuable for market research, political campaigns, and brand reputation monitoring.
- Generating visual dashboards to showcase sentiment changes and patterns.

Explainability and Interpretability

- Incorporating techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) to make the model's decisions more interpretable to end-users.
- Providing reasons or key phrases influencing sentiment classifications.

□ Integration with Voice and Video Analysis

- Extending the project to process audio or video inputs, analyzing spoken words, tone, and facial expressions for a holistic sentiment analysis system.
- Applications in customer service, where both text and voice data can be analyzed.

□ Cross-domain Applications

- Applying the model to new domains such as mental health analysis, where sentiment insights can help identify individuals at risk of depression or anxiety.
- Leveraging sentiment analysis in fraud detection by identifying emotional patterns in customer communications.

□ Customizable Models for Enterprises

- Offering tailored sentiment analysis models for businesses to align with their specific needs and terminologies.
- Developing user-friendly interfaces for non-technical users to adapt the tool for various use cases.

□ Ethical and Privacy Considerations

- Ensuring the system adheres to ethical AI practices by preventing biases in sentiment classification.
- Introducing privacy-preserving methods like differential privacy to protect user data.

Appendices

Appendix A: Sample Dataset

- Example entries from the dataset used for training and testing sentiment analysis models.

Text	Sentiment
"The product quality is excellent!"	Positive
"Terrible service, very disappointed."	Negative
"Delivery was okay, nothing special."	Neutral

Appendix B: Preprocessing Techniques

- **Text Cleaning**: Removed special characters, URLs, and numbers.
- **Tokenization Example**:
  Input: "The product is amazing!"
  Output: ['The', 'product', 'is', 'amazing']
- **Stop-word Removal**:
  Input: ['The', 'product', 'is', 'amazing']
  Output: ['product', 'amazing']
- **Stemming Example**:
  Input: ['amazing', 'features']
  Output: ['amaz', 'featur']

Appendix C: Algorithms and Hyperparameters

- **Algorithms**:
  - Naive Bayes: Laplace smoothing = 1.0
  - Support Vector Machine: Kernel = 'linear', C = 1.0
- **Hyperparameters Tuned**:
  - Learning rate, number of epochs, regularization parameters.

Appendix D: Model Performance Metrics

- **Confusion Matrix**:

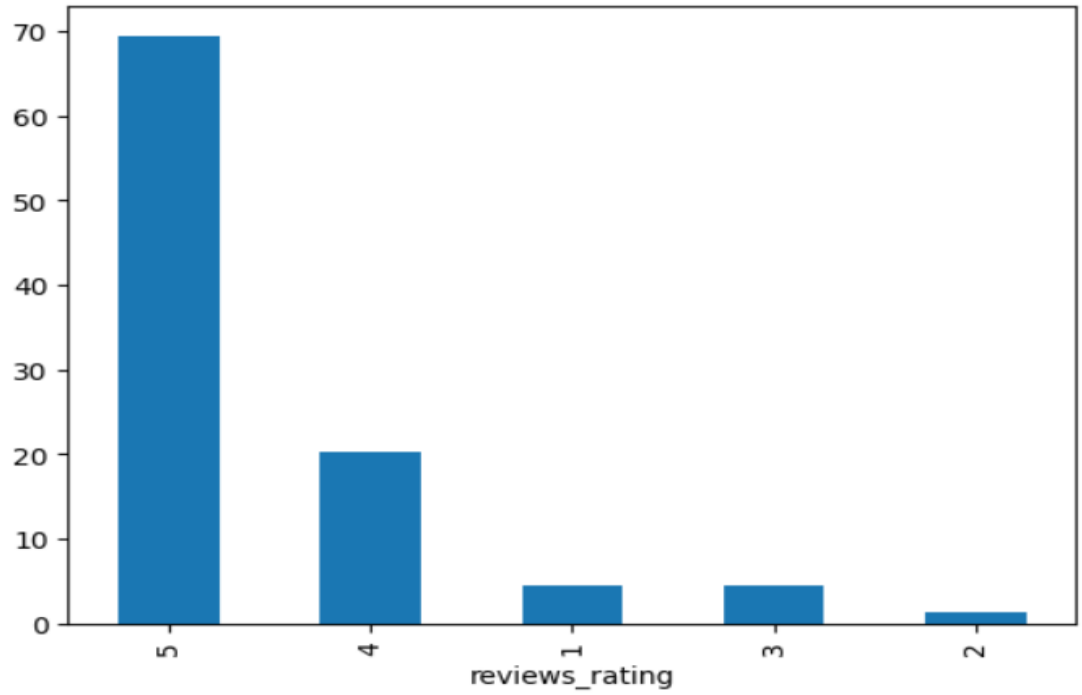
	Positive	Negative	Neutral
Positive	50	5	3
Negative	4	48	6
Neutral	3	7	45

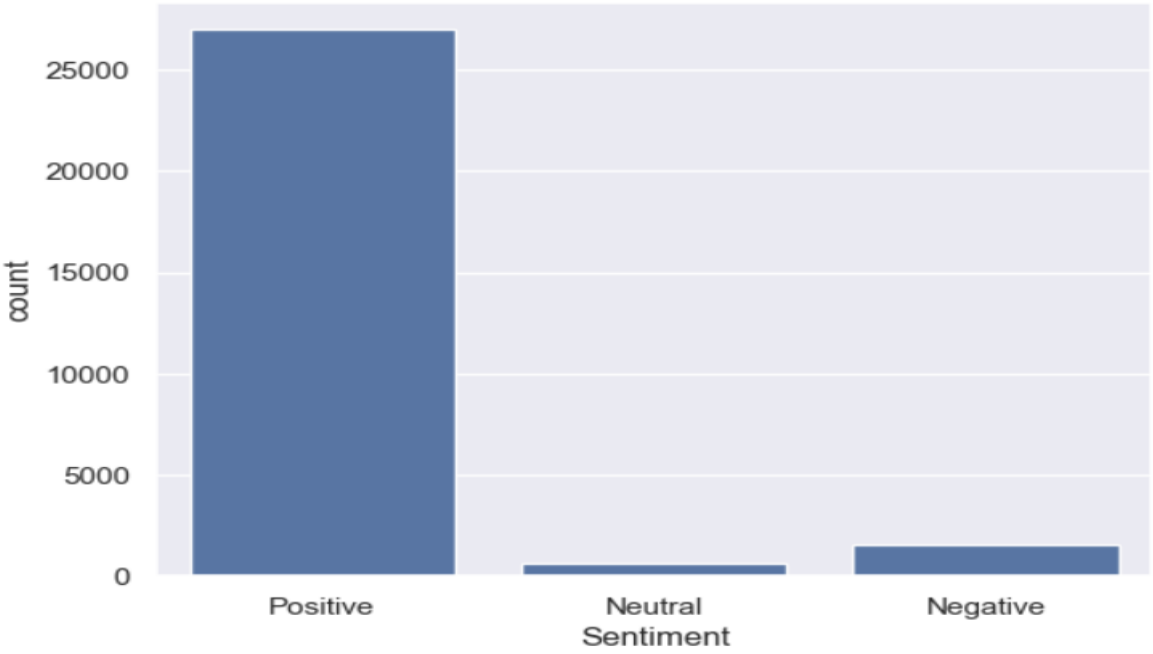
- **Classification Metrics**:
  - Accuracy: 92%
  - Precision, Recall, and F1-Score for each category.

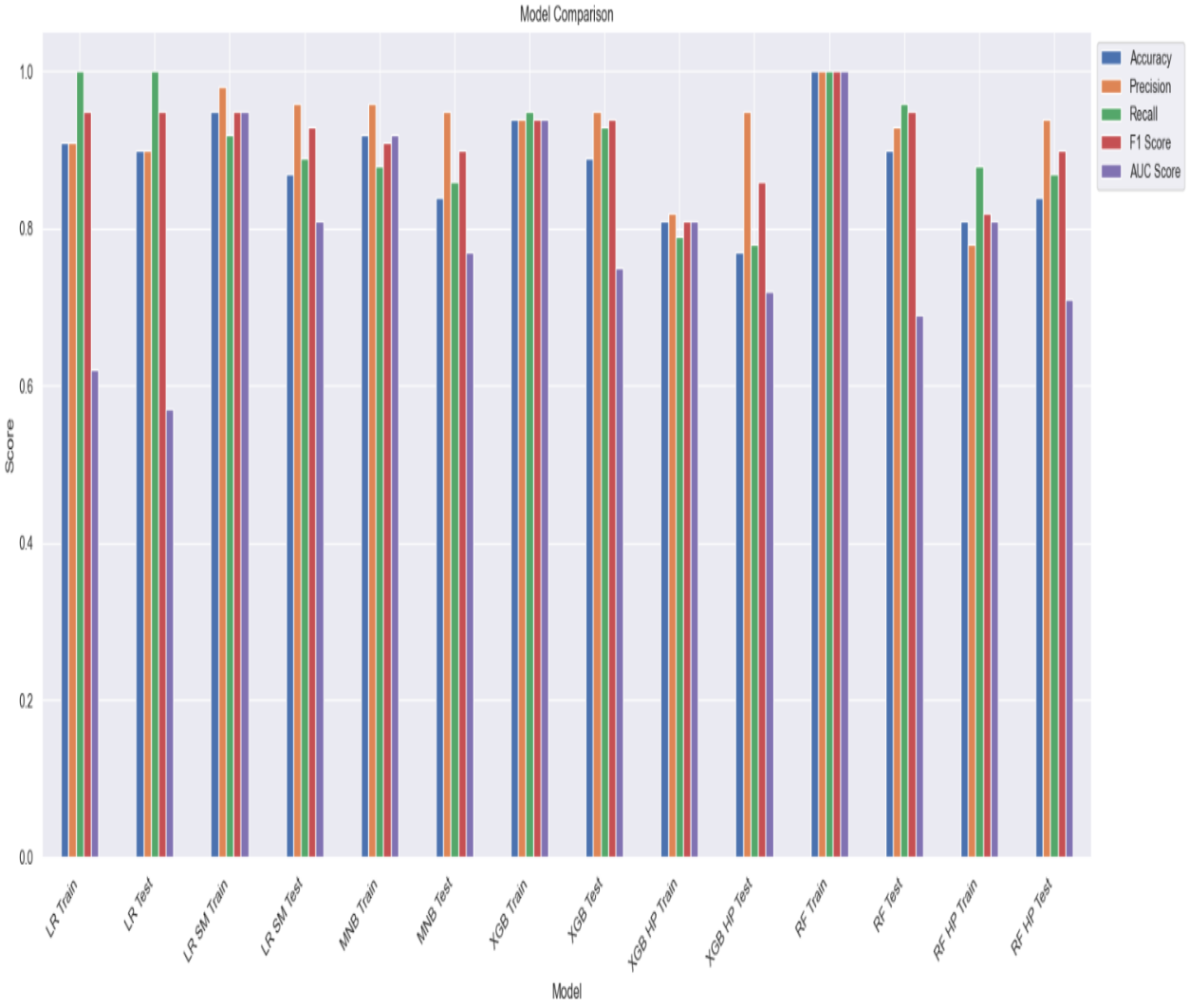
Appendix E: Tools and Libraries

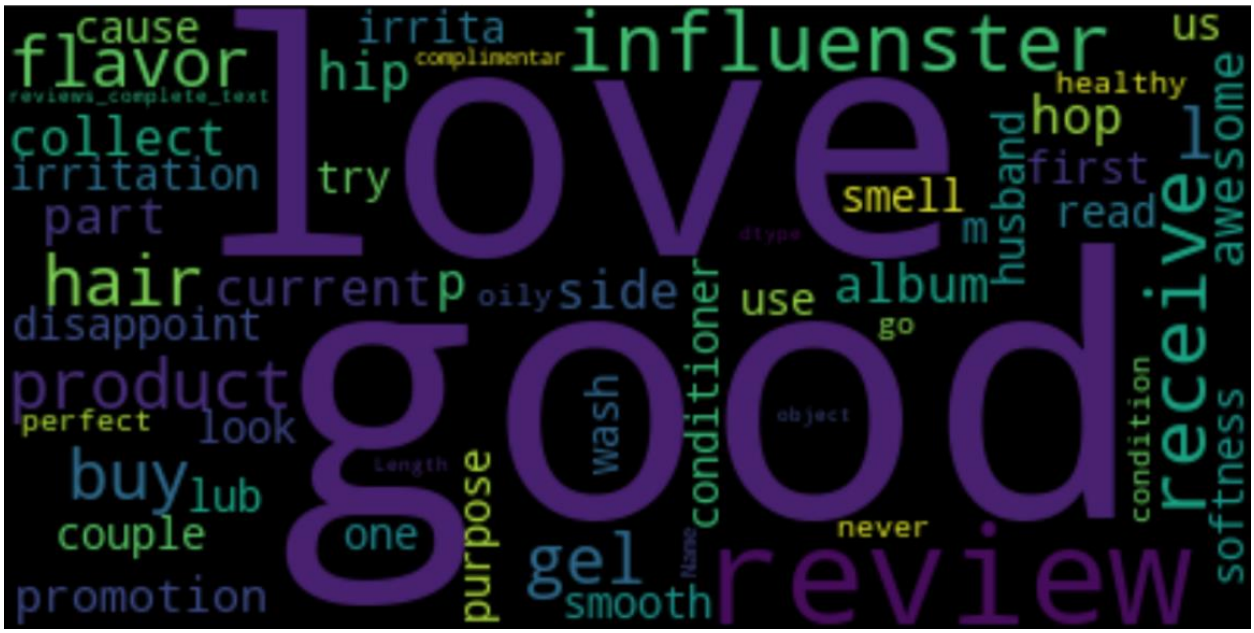
- **Programming Language**: Python.
- **Libraries**:
  - pandas for data manipulation.
  - nltk and spaCy for NLP preprocessing.
  - scikit-learn for machine learning models.
  - matplotlib and seaborn for visualizations.

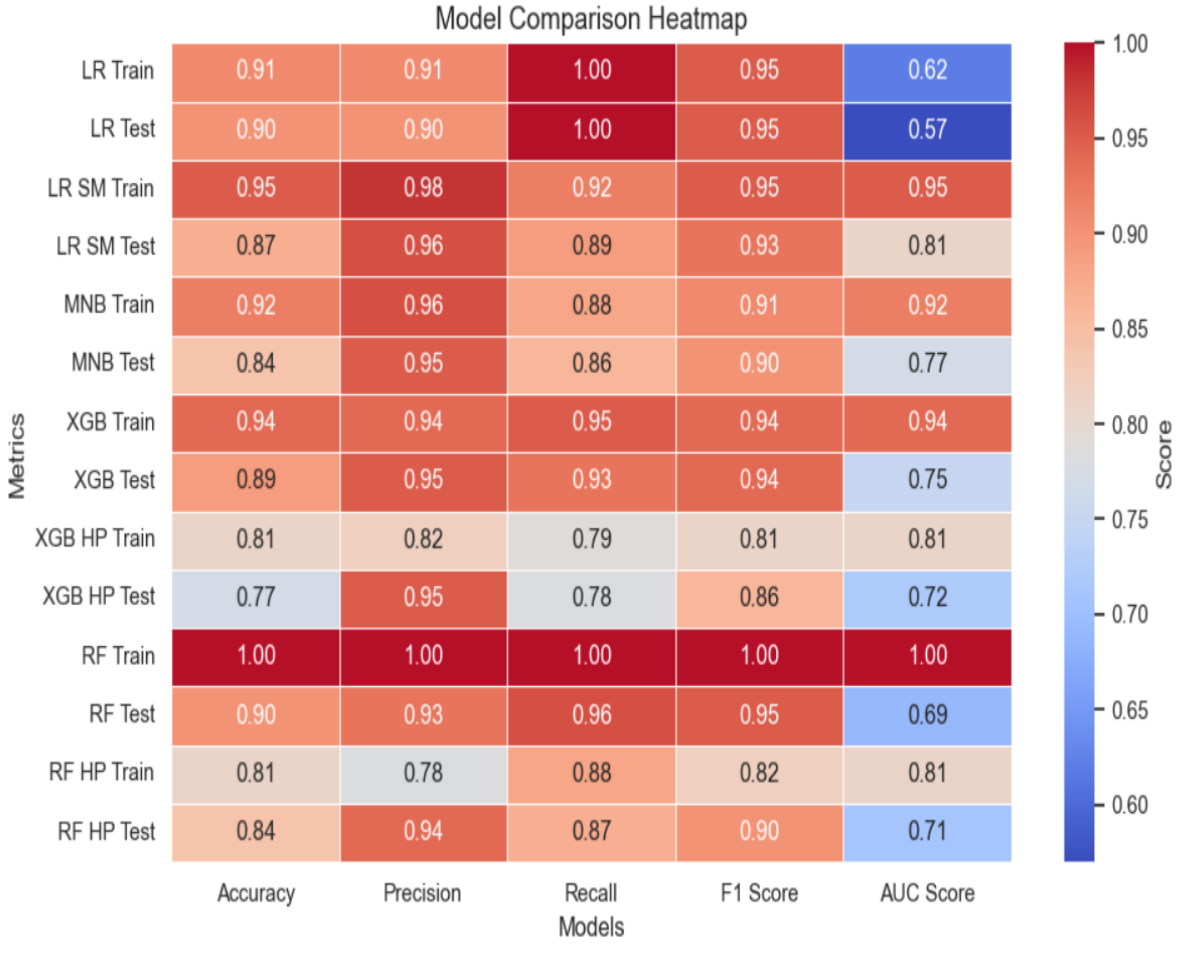
Appendix F: Visualizations




25000
20000
15000
10000
5000
0
count
Positive
Neutral
Negative
Sentiment


Model Comparison
Accuracy
Precision
Recall
F1 Score
AUC Score
1.0
0.8
0.6
0.4
0.2
0.0
Score
LR Train
LR Test
LR SM Train
LR SM Test
MNB Train
MNB Test
XGB Train
XGB Test
XGB HP Train
XGB HP Test
RF Train
RF Test
RF HP Train
RF HP Test
Model


love
good
review
influenster
flavor
cause
irrita
hip
complimentar
us
healthy
hop
awesome
collect
irritation
try
smell
husband
first
read
part
hair
currentp
oily
side
conditioner
use
album
go
disappoint
product
perfect
look
wash
object
condition
receive
buy
lub
couple
one
purpose
gel
smooth
never
softness
promotion
reviews_complete_text
dtype
Length
Name


Model Comparison Heatmap
LR Train 0.91 0.91 1.00 0.95 0.62
LR Test 0.90 0.90 1.00 0.95 0.57
LR SM Train 0.95 0.98 0.92 0.95 0.95
LR SM Test 0.87 0.96 0.89 0.93 0.81
MNB Train 0.92 0.96 0.88 0.91 0.92
MNB Test 0.84 0.95 0.86 0.90 0.77
XGB Train 0.94 0.94 0.95 0.94 0.94
XGB Test 0.89 0.95 0.93 0.94 0.75
XGB HP Train 0.81 0.82 0.79 0.81 0.81
XGB HP Test 0.77 0.95 0.78 0.86 0.72
RF Train 1.00 1.00 1.00 1.00 1.00
RF Test 0.90 0.93 0.96 0.95 0.69
RF HP Train 0.81 0.78 0.88 0.82 0.81
RF HP Test 0.84 0.94 0.87 0.90 0.71
Metrics
Accuracy
Precision
Recall
F1 Score
AUC Score
Models
Score
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60

References

1. **Research Papers and Articles**:
    - Pennington, J., Socher, R., & Manning, C. D. (2014). *GloVe: Global Vectors for Word Representation*. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).
    - Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv preprint.
2. **Datasets**:
    - Stanford Sentiment Treebank Dataset. Available at: https://nlp.stanford.edu/sentiment/
    - Kaggle Sentiment Analysis Datasets. Available at: https://www.kaggle.com/
3. **Books**:
    - Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing*. Prentice Hall.
4. **Online Resources**:
    - Scikit-learn Documentation. Available at: https://scikit-learn.org/
    - NLTK Documentation. Available at: https://www.nltk.org/
    - TensorFlow Tutorials. Available at: https://www.tensorflow.org/tutorials
5. **Tools and Software**:
    - Python 3.x: https://www.python.org/
    - Lucidchart (for designing DFDs): https://www.lucidchart.com/
6. **Blogs and Tutorials**:
    - Analytics Vidhya: *A Comprehensive Guide to Sentiment Analysis*. Available at: https://www.analyticsvidhya.com/
    - Towards Data Science: *Text Preprocessing for Sentiment Analysis*. Available at: https://towardsdatascience.com/
7. **Other Resources**:
    - Word embeddings and TF-IDF guides used in feature extraction.
    - Articles on handling imbalanced datasets and evaluation metrics.