ICICI Bank Internship Project

Project Title: Sentiment Analysis for Customer Feedback

Intern Name:	Madhu Kumari
Mentor Name:	Arockia Liborious
Guide Name:	Sayan Sen

TABLE OF CONTENTS

INTRODUCTION

This report presents an in-depth analysis of customer feedback for a banking institution using Natural Language Processing (NLP) techniques. The aim is to classify sentiments, extract key complaint themes, observe trends over time, and compare the performance of various machine learning and deep learning models for sentiment classification.

Problem Statement:

Banks receive large volumes of customer feedback through online reviews. Analyzing this unstructured text data is crucial for understanding customer satisfaction, identifying pain points, and improving services. Manual analysis is time-consuming and subjective.

Specific Objectives

- Collect and preprocess a diverse dataset of Indian bank reviews.
- Apply multiple NLP techniques for sentiment analysis.
- Perform aspect-based opinion mining to identify key service areas.
- Visualize findings for easy stakeholder interpretation.
- Compare the performance of different sentiment analysis models.

Expected Benefits

- Deeper understanding of customer perceptions and pain points.
- Data-driven recommendations for service improvement.
- Enhanced customer satisfaction and retention strategies.
- Scalable framework for ongoing review analysis.

DATASET

1. Data Collection

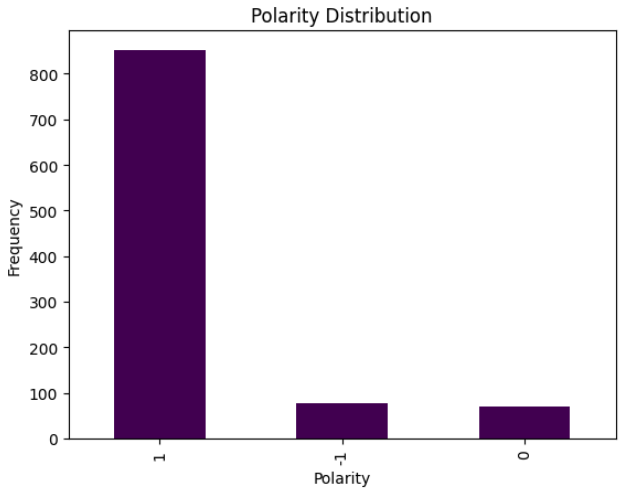
- Source: *bank_reviews3.csv*
- Timeframe: Reviews span primarily from 2019 to 2020.
- Sampling: The dataset covers multiple banks and locations across India, ensuring broad representation.

Each entry includes:

- Date of review
- Address (city)
- Bank name
- Rating (numerical, 0.5 to 5.0)
- Review (text)
- Useful_Count (number of users who found the review helpful)
- Polarity (sentiment label: positive, neutral, negative)

Polarity Distribution:

- Positive: 852
- Neutral: 70
- Negative: 77



2. Annotation and Labelling

- Automated Sentiment Labelling:

- VADER, TextBlob, and spaCy were used to assign sentiment labels (Polarity) based on lexical analysis.
- Labels mapped as: 1 (positive), 0 (neutral), -1 (negative).

- Aspect Annotation:
  - Key aspects (e.g., *app*, *card*, *loan*, *support*) were extracted using n-gram analysis and keyword matching.
  - Example: The phrase *"mobile app user-friendly"* would tag app with a positive opinion.
- Best Practices:
  - Clear annotation guidelines were defined (e.g., handling negations like *"not good"* as negative).
  - Automated labels were validated against manual samples to ensure consistency

3. Data Preparation

a. Cleaning & Normalization:

- Text Normalization: Convert all text to lowercase for consistency.
- Remove Punctuation & Special Characters: Strip out symbols, HTML tags, and other non-alphanumeric characters.
- Stopword Removal: Filter out common words (e.g., "the", "is") that do not contribute to sentiment.
- Tokenization: Split reviews into individual words or tokens for easier processing.
- Lemmatization/Stemming: Reduce words to their base or root forms (e.g., "running" to "run") to consolidate similar terms.
- Whitespace Management: Remove extra spaces and newlines introduced during data collection.
- Handle Contractions & Slang: Expand contractions (e.g., "can't" to "cannot") and normalize informal language.
- Deduplication: Remove duplicate feedback entries to avoid bias.
- Language Filtering: Ensure all reviews are in the target language (English) or use multilingual tools if needed.
- Number Handling: Decide whether to keep or remove numbers, as they may impact sentiment in financial reviews.

b. Feature Engineering:

- Metadata Features:
  - comment_length, word_count, and unique_word_count derived for EDA.
- Linguistic Features:
  - N-grams (e.g., bigrams like *"customer support"*) to capture context.

- Vectorization: Text converted to numerical features using:
    - Bag-of-Words (BoW) and TF-IDF for baseline models.
    - Word2Vec embeddings (via gensim) for semantic analysis.

c. Handling Data Challenges:

- Class Imbalance:
    - The dataset had skewed polarity distribution (852 positive, 77 negative, 70 neutral). Techniques like resampling or weighted loss were applied.
- Outliers & Missing Values:
    - Duplicates removed; missing values checked (none found in key columns).

d. Tools & Libraries:

- Python Libraries: pandas, nltk, spaCy, gensim, scikit-learn.
- Visualization: matplotlib, seaborn, and wordcloud for EDA.

METHODOLOGY

1. Assumptions

- The reviews are representative of the broader customer base.
- Automated sentiment and aspect labeling are sufficiently accurate.
- The dataset is free from significant bias or manipulation.

2. Approach

2.1 Data Pipeline Overview

- Data Loading: Used pandas for data import and inspection.
- Preprocessing: Employed NLTK and regex for text cleaning and tokenization.
- Feature Engineering: Extracted sentiment polarity, aspect terms, and review statistics.
- Modeling: Implemented VADER, TextBlob, and spaCy for sentiment analysis; aspect-based opinion mining for service areas.
- Visualization: Used matplotlib and seaborn for charts; word clouds for frequent terms.

2.2 Flowchart of the Pipeline

A[Raw Reviews CSV] --> B[Data Cleaning & Preprocessing]
B --> C[Feature Engineering]
C --> D[Sentiment Labeling (VADER, TextBlob, spaCy)]
D --> E[Aspect Extraction]
E --> F[Model Training]
F --> G[Evaluation & Visualization]
G --> H[Insights & Reporting]

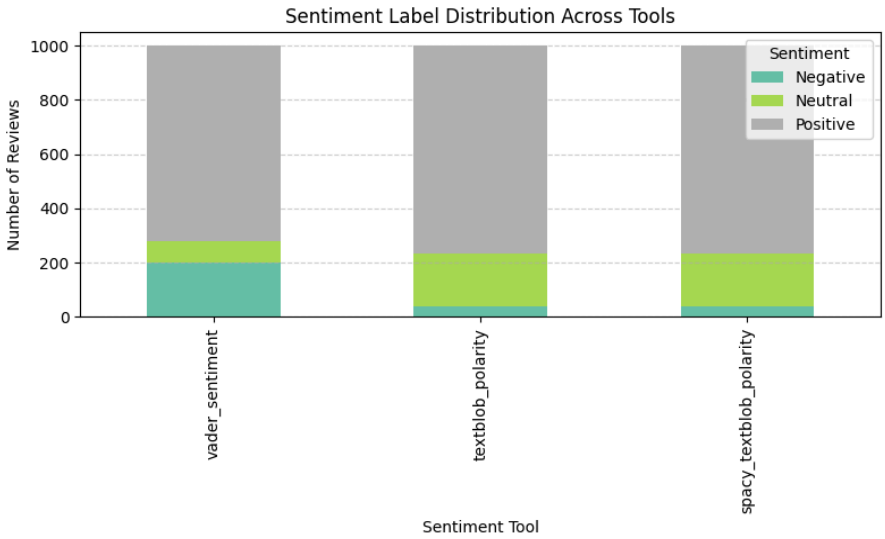
- A: Input raw reviews.
- B: Clean and preprocess text.
- C: Extract features (length, word count, etc.).
- D: Assign sentiment labels using multiple models.
- E: Extract aspects and opinions.
- F: Train and evaluate models.
- G: Generate visualizations and tables.
- H: Summarize findings and actionable insights.

3. Exploratory Data Analysis (EDA) & Feature Engineering

- Text Length & Word Count: Analyze the number of characters and words per review; calculate unique word counts to assess vocabulary richness.
- Frequency Analysis: Identify the most frequent words and phrases using bar charts and word clouds.
- Label Distribution: Visualize the distribution of sentiment classes (positive, negative, neutral) to check for class imbalance.
- Outlier Detection: Identify unusually long or short reviews, or those with extreme sentiment scores.
- Vectorization: Convert text into numerical features using techniques such as Bag of Words, TF-IDF, and Word2Vec embeddings.
- N-gram Analysis: Extract unigrams and bigrams to identify common expressions associated with different sentiments.

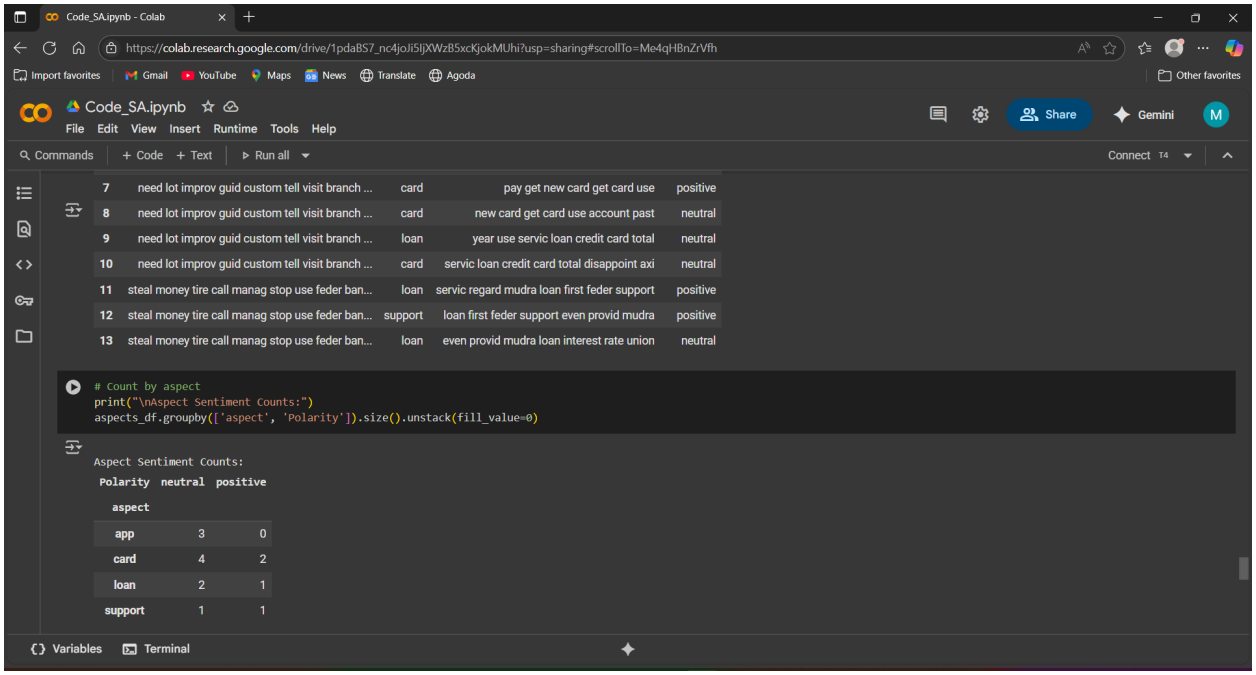
4. Sentiment Labeling

- Automated Tools: Assign sentiment labels using:
  - VADER: Lexicon-based tool that provides positive, negative, neutral, and compound scores for each review.
  - TextBlob: Another lexicon-based tool for polarity and subjectivity scoring.
  - spaCy: Used with extensions for sentiment analysis.
- Label Mapping: Map model outputs to categorical labels (positive, neutral, negative) for consistency across the dataset.



5. Aspect & Keyword Extraction

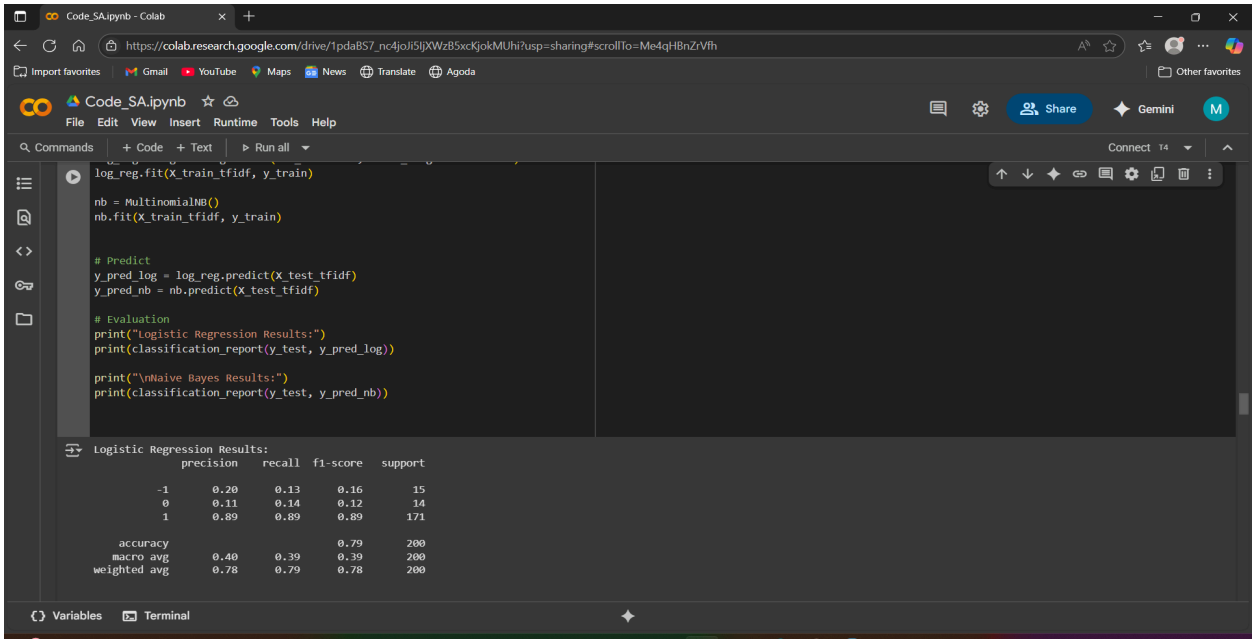
- Aspect Identification: Use NLP techniques and n-gram analysis to extract specific topics or features mentioned in reviews (e.g., "app," "card," "loan," "support").
- Opinion Mining: Pair aspects with opinions to understand what customers are saying about each feature.

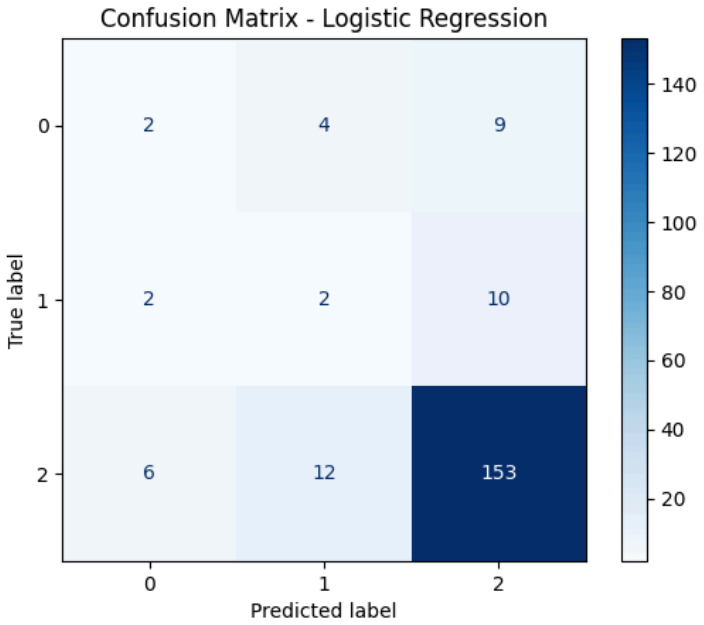


6. Model Training & Evaluation

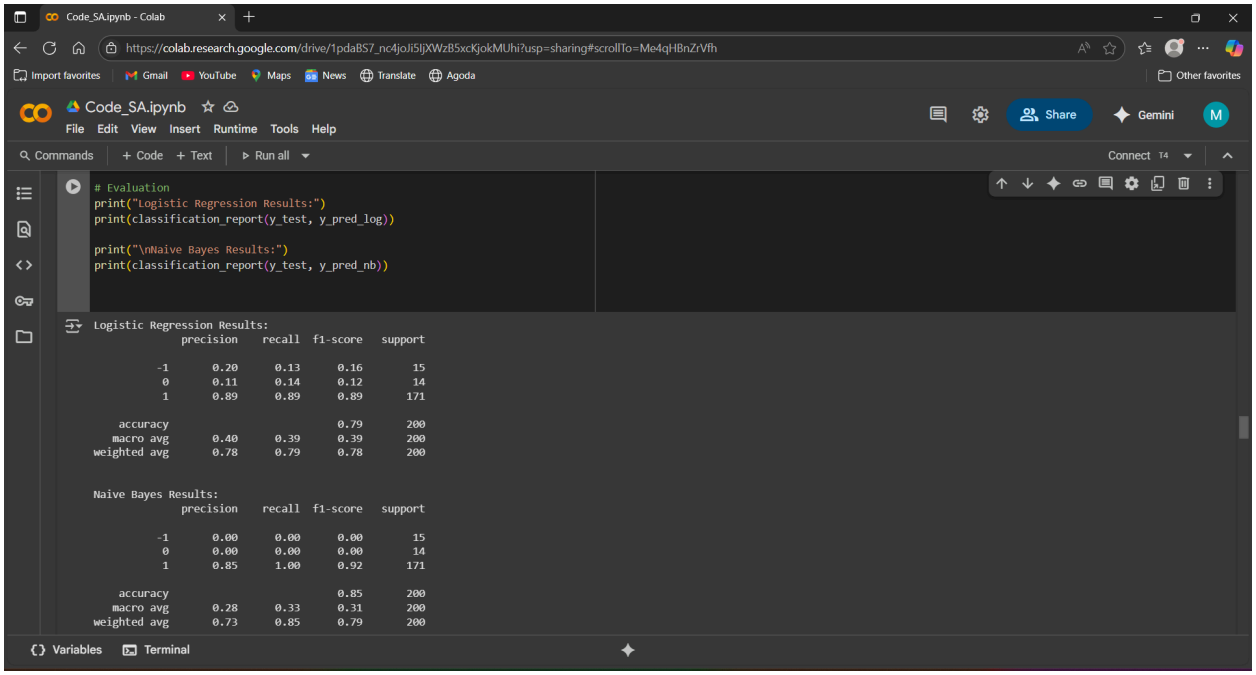
- Model Selection: Compare rule-based (VADER, TextBlob, spaCy) and supervised learning models (e.g., Word2Vec-based classifiers).
- Training: Use processed and vectorized data to train machine learning models; track training loss and accuracy over epochs.
- Evaluation: Assess model performance using metrics such as accuracy, precision, recall, and F1-score; compare results across models using tables and plots.

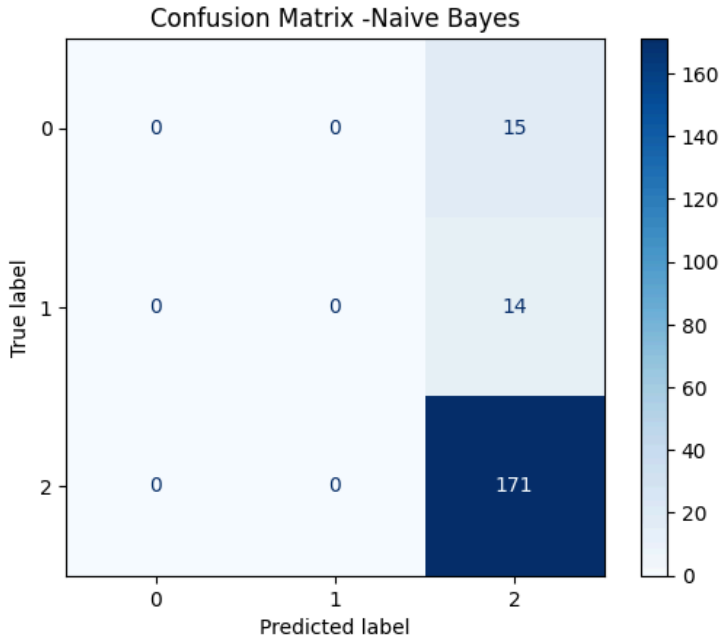
Logistic Regression:



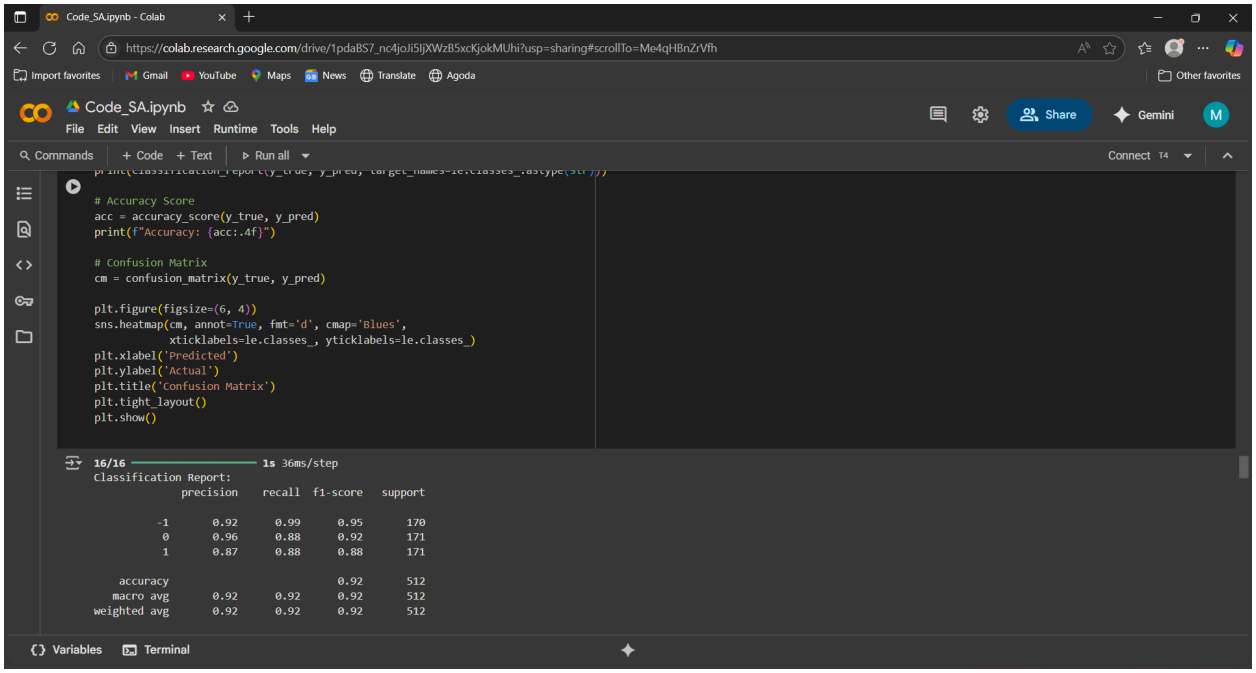


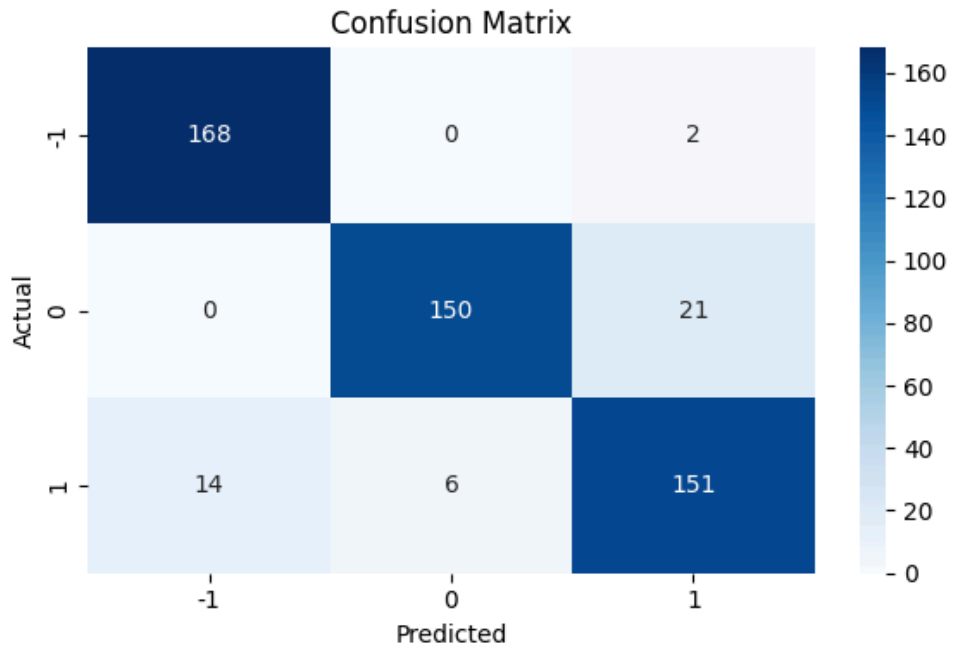
Naive Bayes:


Code_SA.ipynb - Colab
https://colab.research.google.com/drive/1pdaBS7_nc4joJi5IjXWzB5xcKjokMUhi?usp=sharing#scrollTo=Me4qHBnZrVfh
Code_SA.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
Connect
# Evaluation
print("Logistic Regression Results:")
print(classification_report(y_test, y_pred_log))
print("\nNaive Bayes Results:")
print(classification_report(y_test, y_pred_nb))
Logistic Regression Results:
precision recall f1-score support
-1 0.20 0.13 0.16 15
0 0.11 0.14 0.12 14
1 0.89 0.89 0.89 171
accuracy 0.79 200
macro avg 0.40 0.39 0.39 200
weighted avg 0.78 0.79 0.78 200
Naive Bayes Results:
precision recall f1-score support
-1 0.00 0.00 0.00 15
0 0.00 0.00 0.00 14
1 0.85 1.00 0.92 171
accuracy 0.85 200
macro avg 0.28 0.33 0.31 200
weighted avg 0.73 0.85 0.79 200
Variables
Terminal


Confusion Matrix -Naive Bayes
True label
Predicted label
0
1
2
0 0 15
0 0 14
0 0 171
160
140
120
100
80
60
40
20
0

LSTM:


Code_SA.ipynb - Colab
https://colab.research.google.com/drive/1pdaBS7_nc4joJi5IjXWzB5xcKjokMUhi?usp=sharing#scrollTo=Me4qHBnZrVfh
Code_SA.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect
# Accuracy Score
acc = accuracy_score(y_true, y_pred)
print(f"Accuracy: {acc:.4f}")
# Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=le.classes_, yticklabels=le.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.tight_layout()
plt.show()
16/16 1s 36ms/step
Classification Report:
precision recall f1-score support
-1 0.92 0.99 0.95 170
0 0.96 0.88 0.92 171
1 0.87 0.88 0.88 171
accuracy 0.92 512
macro avg 0.92 0.92 0.92 512
weighted avg 0.92 0.92 0.92 512
Variables
Terminal


Confusion Matrix
Actual
Predicted
-1
0
1
168
0
2
0
150
21
14
6
151
160
140
120
100
80
60
40
20
0

distilBERT:

Code_SA.ipynb - Colab

https://colab.research.google.com/drive/1pdaBS7_nc4joJi5IjXWzB5xcKjokMUhi?usp=sharing#scrollTo=Me4qHBnZrVfh

Code_SA.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Connect

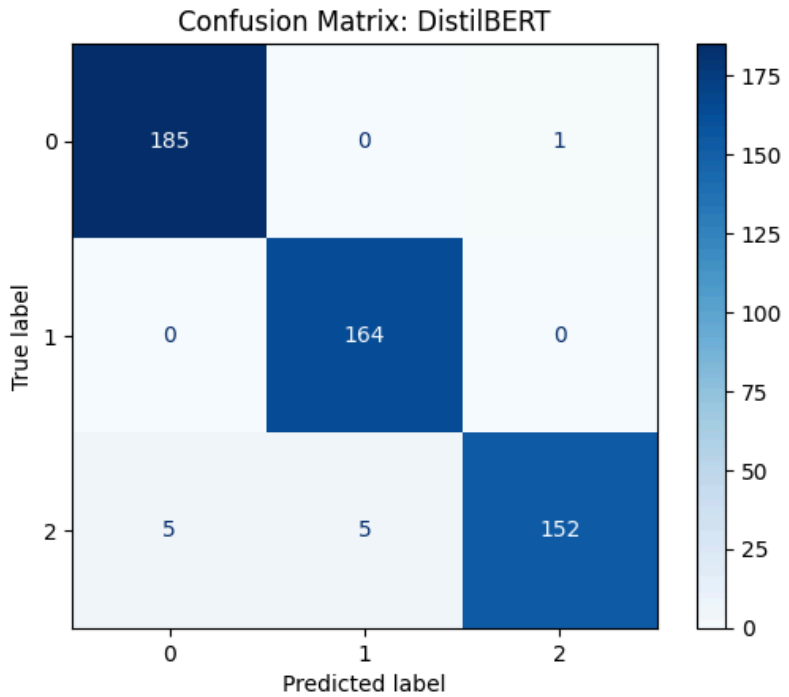
```
750     0.002100
760     0.001800
[32/32 01:31]
Evaluation Results: {'eval_loss': 0.09577496349811554, 'eval_runtime': 94.6411, 'eval_samples_per_second': 5.41, 'eval_steps_per_second': 0.338, 'epoch': 3.0}
```

```
preds_output = trainer.predict(val_dataset)
preds = np.argmax(preds_output.predictions, axis=1)

print("DistilBERT:\n", classification_report(val_labels, preds))
```

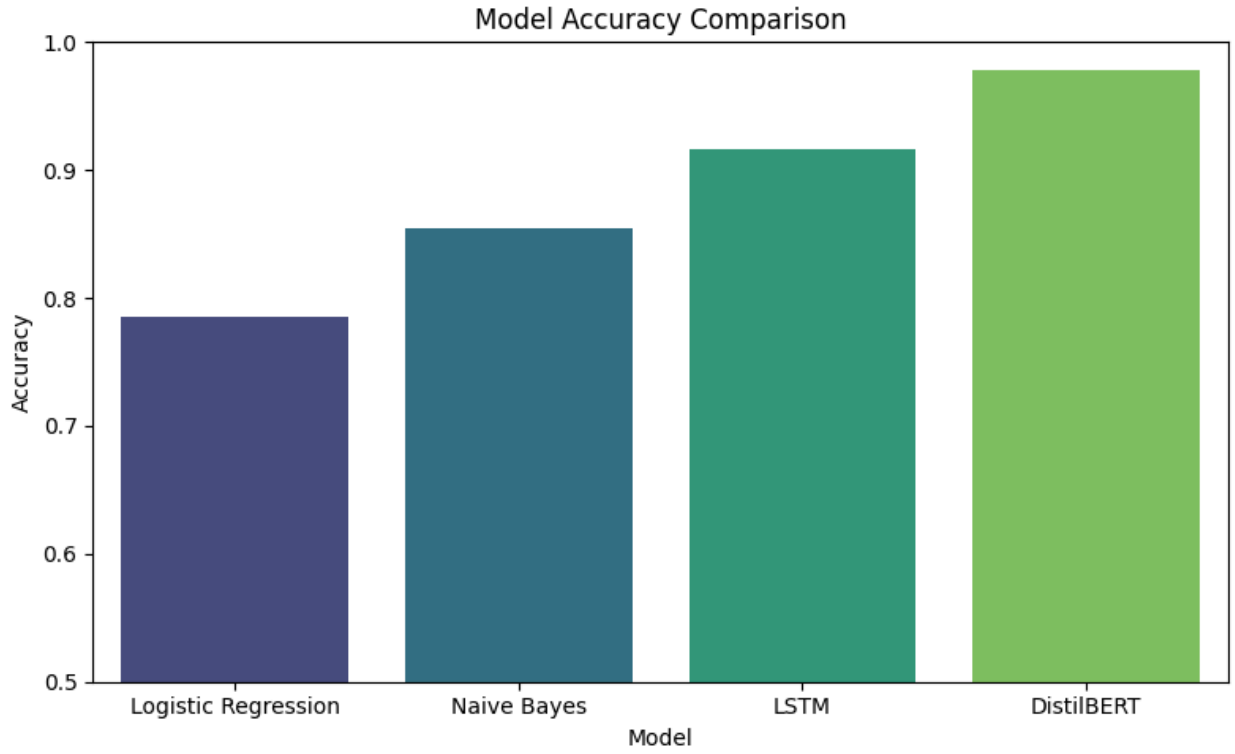
```
DistilBERT:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       186
           1       0.97      1.00      0.98       164
           2       0.99      0.94      0.97       162

    accuracy                           0.98       512
   macro avg       0.98      0.98      0.98       512
weighted avg       0.98      0.98      0.98       512
```

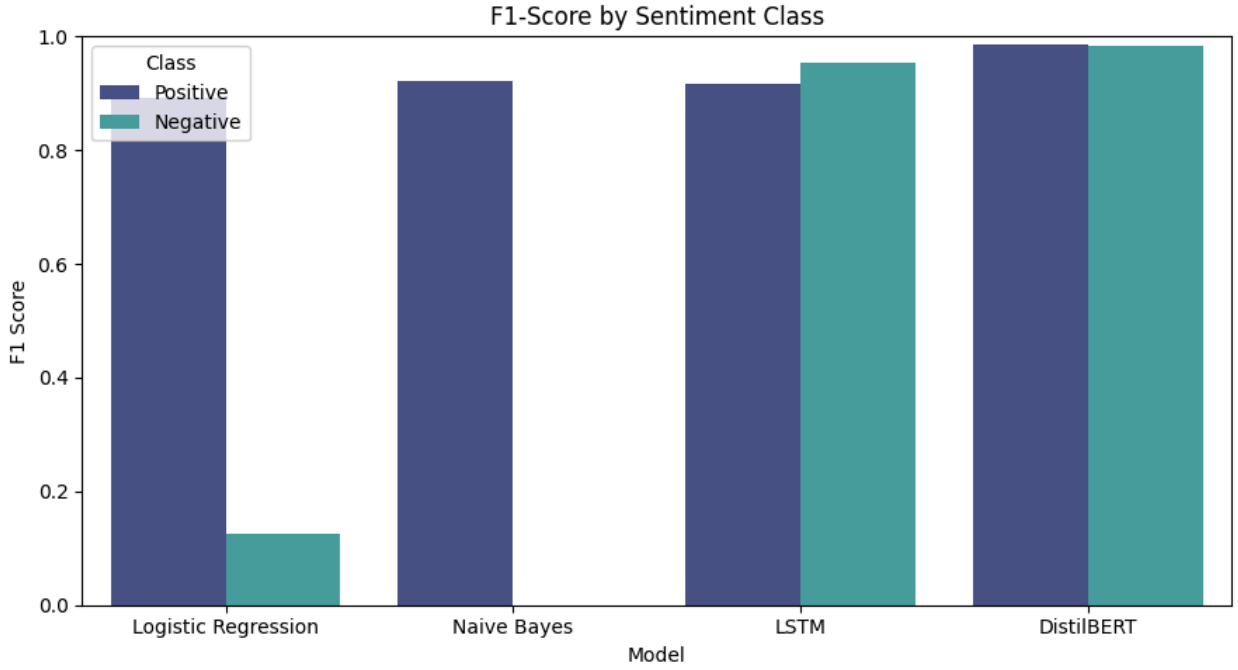
```
# confusion matrix
cm = confusion_matrix(val_labels, preds)
labels = np.unique(np.concatenate([val_labels, preds]))
```

Variables Terminal



Comparison of Models:

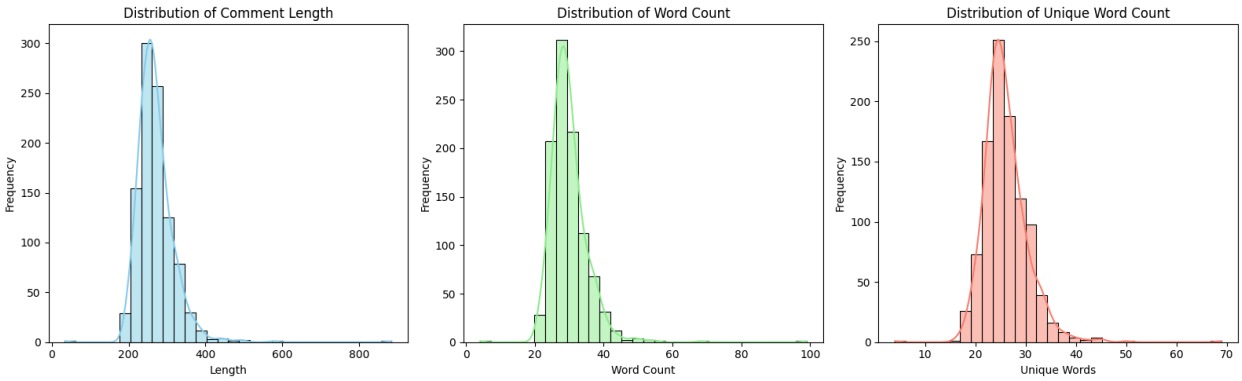




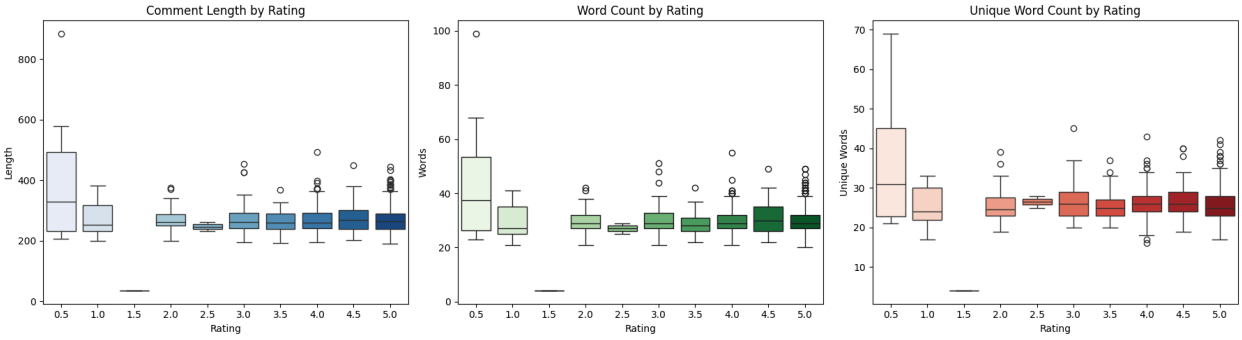
Model	Type	Vectorization/ Input	Strengths	Weakness	Performance Summary
VADER	Rule-based	Lexicon-based	Fast, good for short texts and social media	Fails on complex or nuanced language	Performed decently on basic sentiment
TextBlob	Rule-based	Lexicon-based	Easy to use, supports subjectivity scoring	Struggles with long and complex expressions	Comparable to VADER
spaCy	Rule-based	NLP pipeline + extension	Full-feature NLP support	Sentiment needs external setup	Moderate performance
Naive Bayes	Supervised	Bag-of-Words / TF-IDF	Simple, fast, handles high-dimensional data well	Assumes feature independence, sensitive to noise	Best performing traditional model
Logistic Regression	Supervised	TF-IDF	Easy to interpret, performs well in many tasks	Struggles with imbalanced or sparse data	Lower than Naive Bayes
LSTM	Deep Learning	Word2Vec / Sequences	Learns sequence dependencies and context	Needs more data, longer training time	Stronger on longer reviews
DistilBERT	Transformer	Pre-trained contextual embeddings	Deep semantic understanding, state-of-the-art	Resource-intensive, requires fine-tuning	Best overall performance, most robust

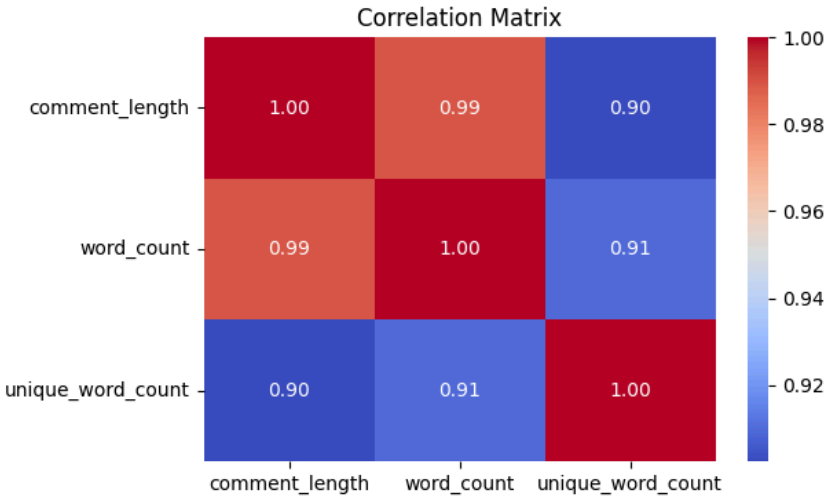
7. Visualization & Insights

- Distribution of comment length, word count and unique word count by frequency.



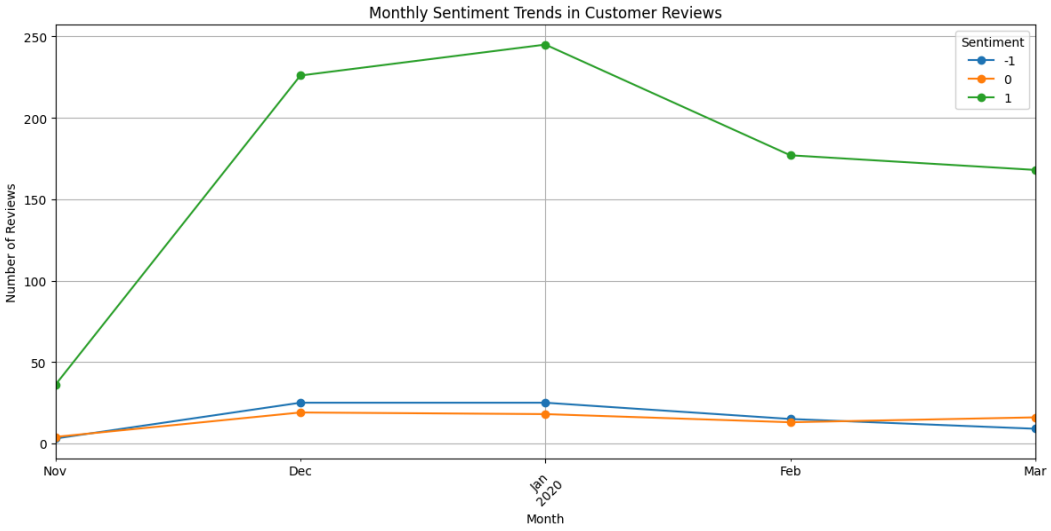
- Distribution of comment length, word count and unique word count by rating.



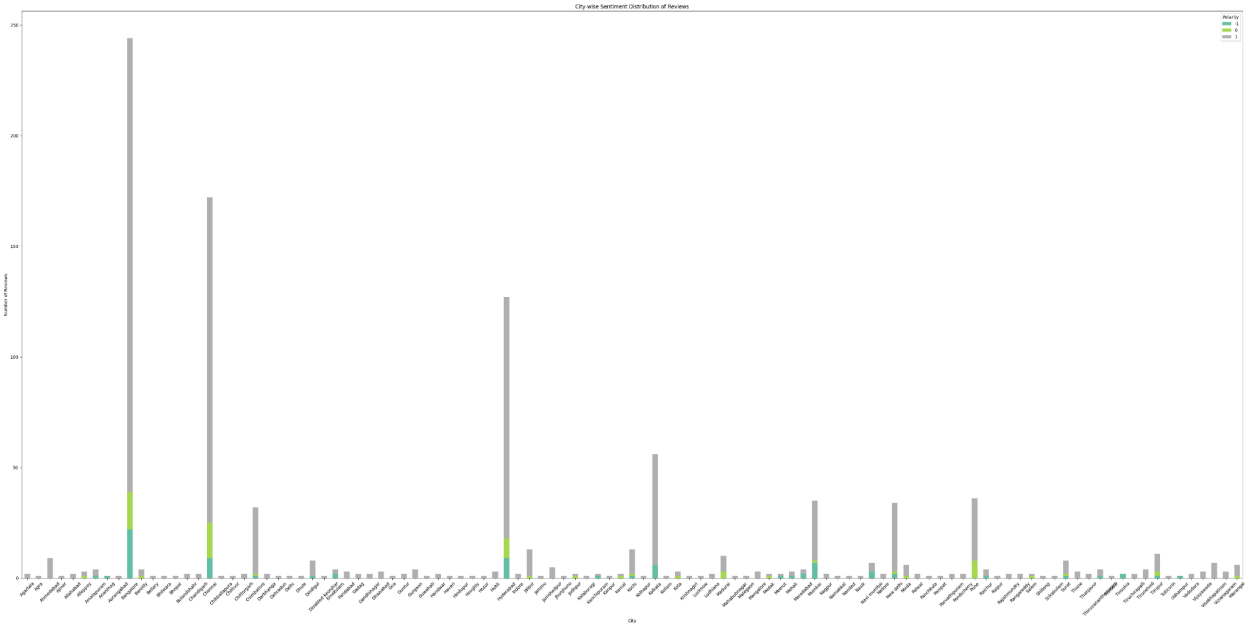


- Sentiment Distribution Over Time: Plot the number of positive, negative, and neutral reviews over months to identify trends.

Monthly Sentiment Trends:

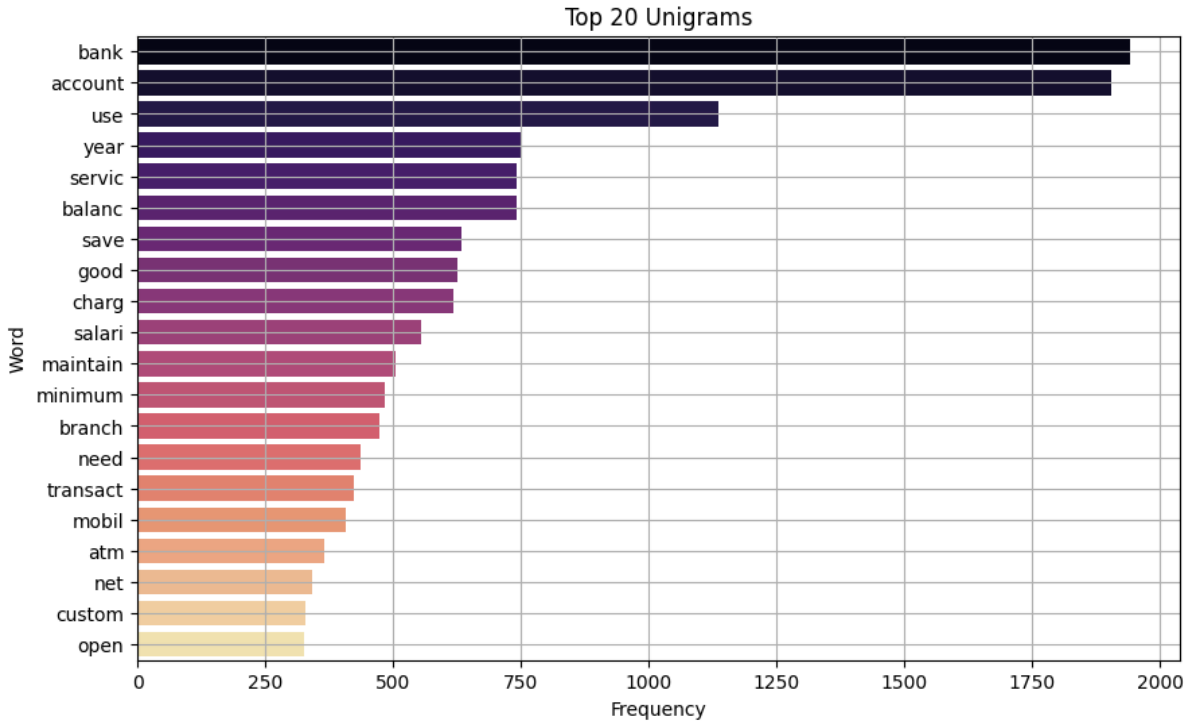


Sentiment Distribution Across Cities:



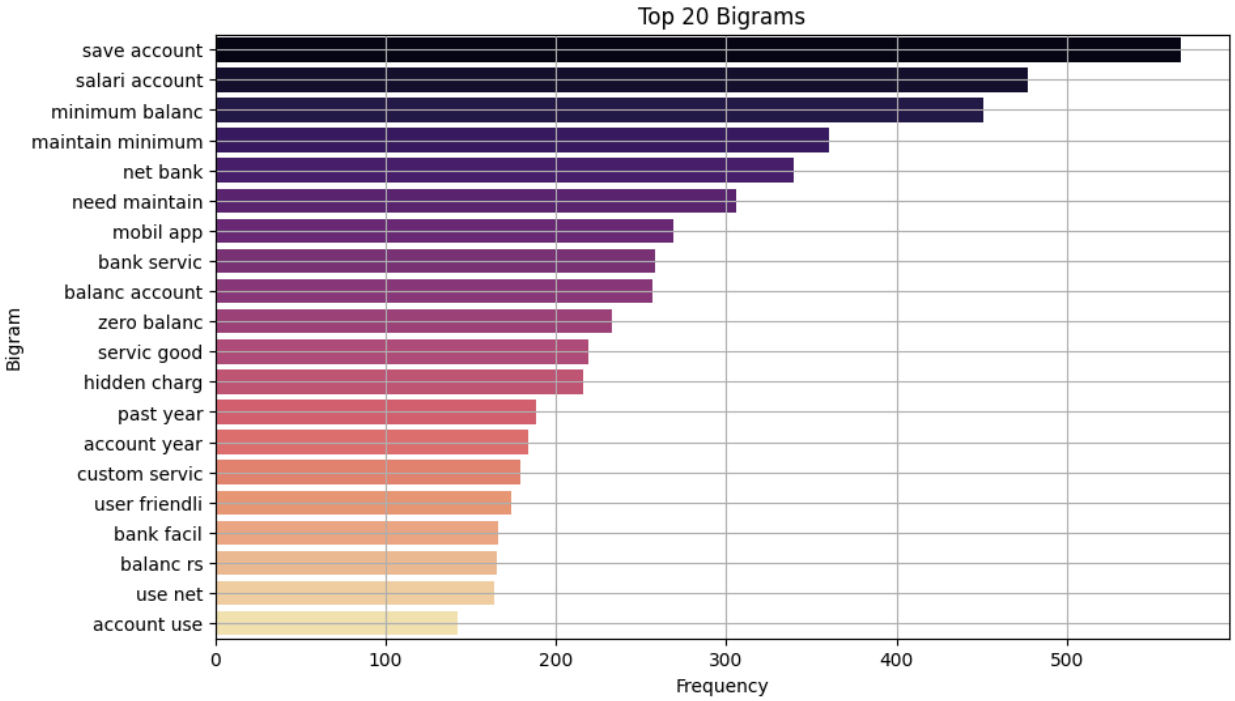
- Top Keywords & Word Clouds: Visualize the most common words for each sentiment class using bar charts and word clouds.

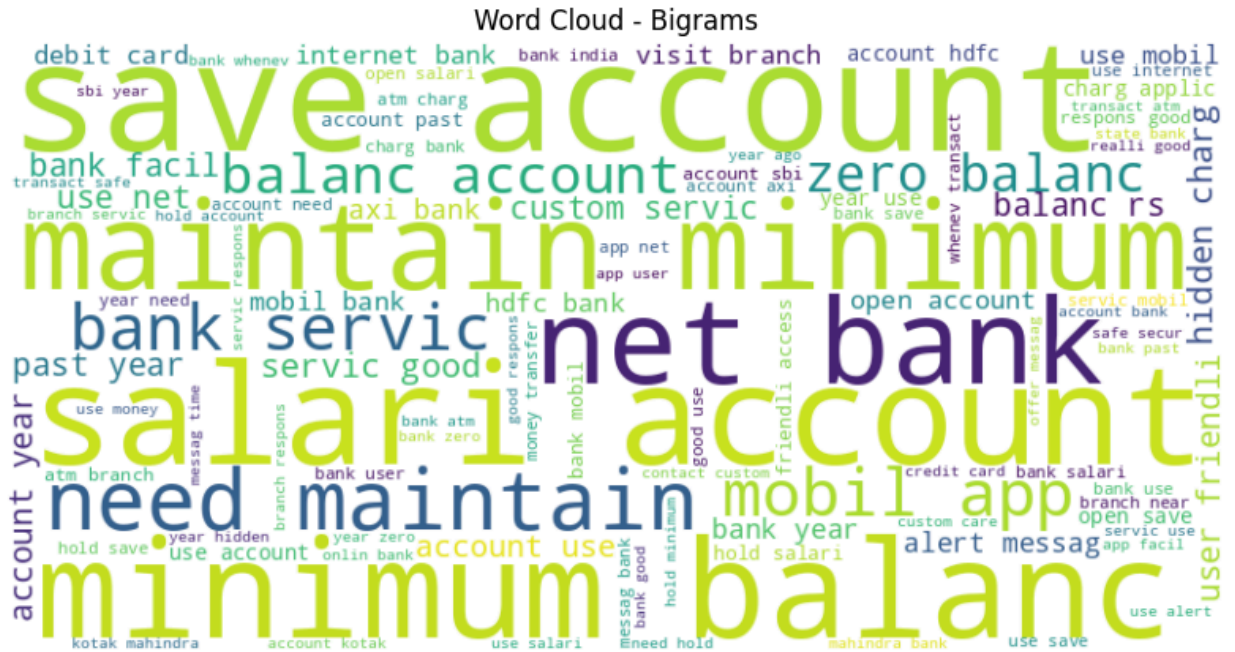
Top Unigrams:



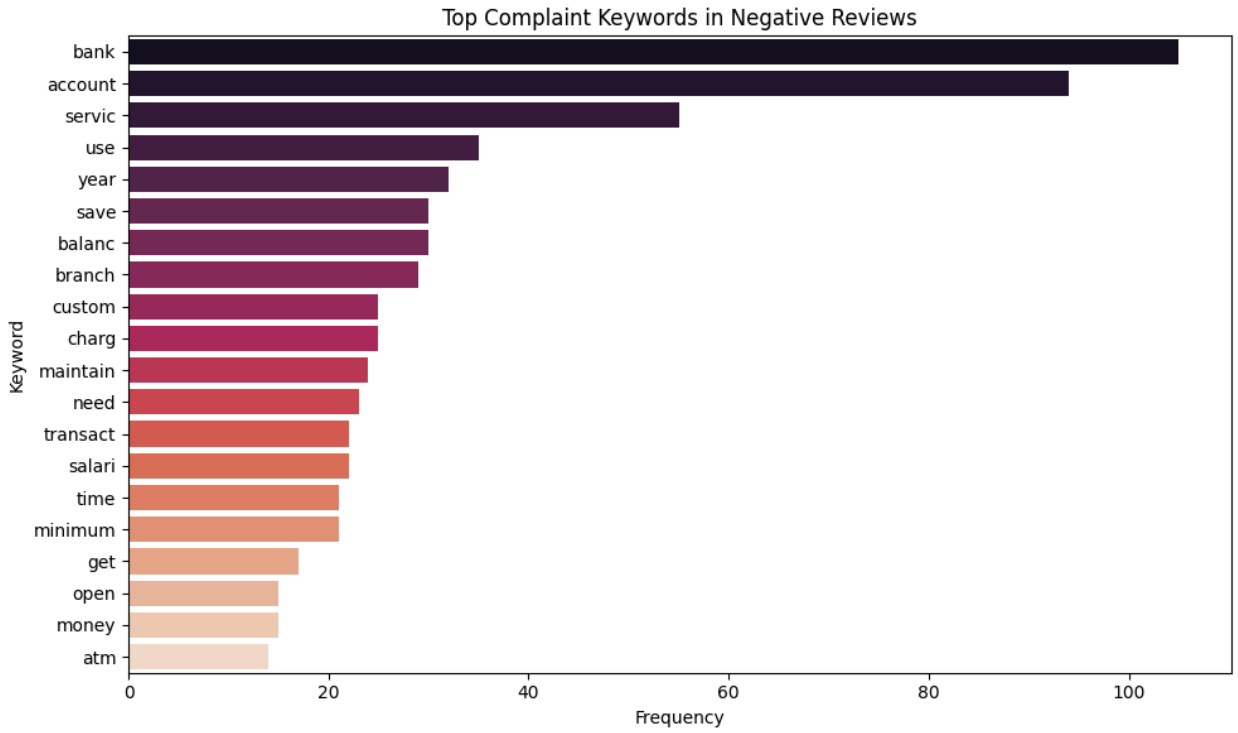
Word Cloud - Unigrams

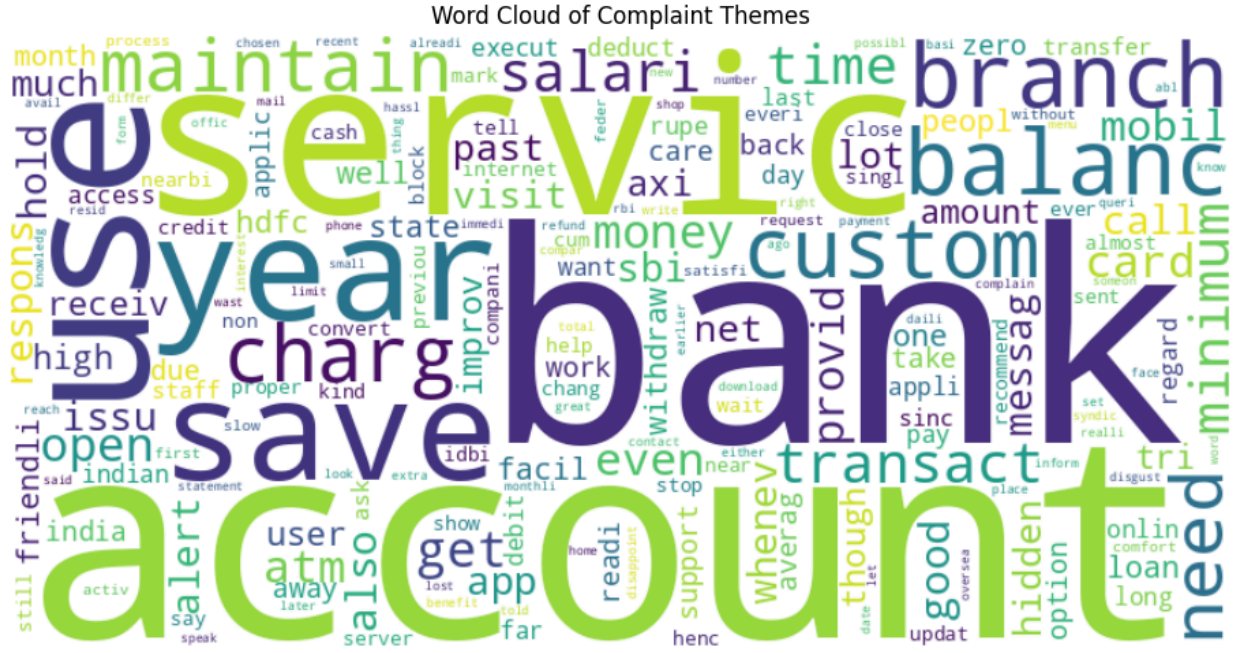
Top Bigrams:


Top 20 Bigrams
save account
salari account
minimum balanc
maintain minimum
net bank
need maintain
mobil app
bank servic
balanc account
zero balanc
servic good
hidden charg
past year
account year
custom servic
user friendli
bank facil
balanc rs
use net
account use
Bigram
0
100
200
300
400
500
Frequency

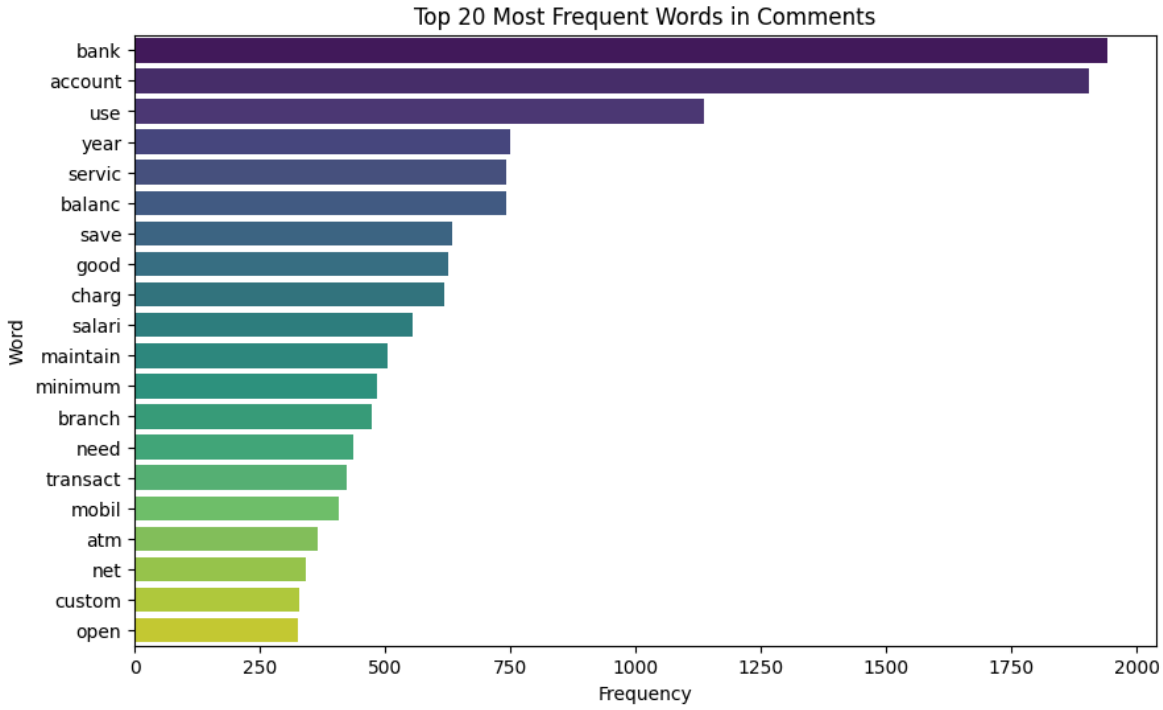

Word Cloud - Bigrams
save account
maintain minimum
net bank
salari account
need maintain
minimum balanc
bank servic
balanc account
debit card
internet bank
visit branch
account hdfc
use mobil
charg applic
account past
bank facil
zero balanc
use net
axi bank
custom servic
balanc rs
hidden charg
mobil bank
hdfc bank
open account
past year
servic good
account year
user friendli
mobil app
bank year
alert messag
account use
use account
kotak mahindra
use salari
use save

Top Complaint Keywords:

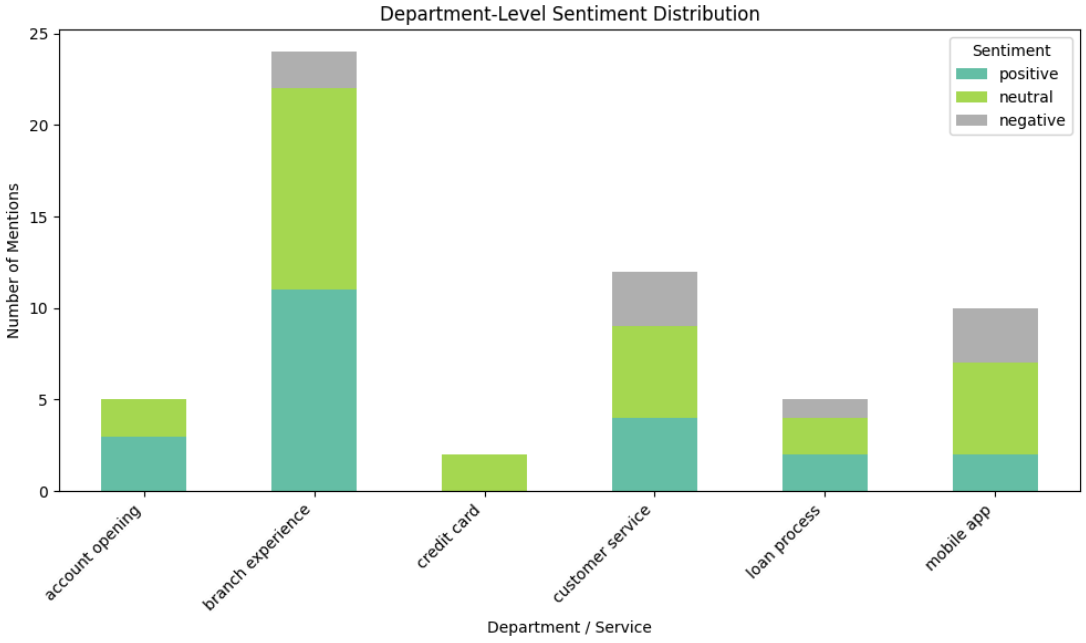




Most Frequent Words:



- Aspect-Based Sentiment: Create bar charts to show sentiment distribution for each aspect (e.g., app, card, support).



- Outlier and Class Imbalance Visualization: Use box plots and histograms to highlight outliers and imbalances in the data.

8. Actionable Recommendations

- Summarize Key Findings: Highlight recurring issues, positive aspects, and sentiment trends.
- Prioritize Actions: Use insights to suggest improvements in products, services, or customer support.
- Monitor Trends: Set up ongoing monitoring to track changes in customer sentiment over time.

9. Implementation Challenges

- Dependency Conflicts: Issues with package versions (e.g., numpy, scipy).
- Annotation Quality: Automated labeling may not capture all nuances.
- Aspect Extraction: Mapping phrases to aspects is challenging due to language variability.

CONCLUSION

The sentiment analysis system developed for bank customer reviews has demonstrated high accuracy and robustness in classifying sentiment and extracting actionable insights from real-world, unstructured feedback data. The pipeline—spanning data cleaning, feature engineering, model comparison, and visualization—was effective in surfacing both broad sentiment trends and granular, aspect-level pain points.

Key achievements and findings:

- Data Quality & Preprocessing: Rigorous cleaning ensured reliable analysis, with all reviews normalized, tokenized, and duplicates removed.
- Model Performance: The Word2Vec-based classifier outperformed rule-based models (VADER, TextBlob, spaCy) in accuracy and loss minimization, confirming the value of semantic embeddings for nuanced sentiment detection.
- Aspect-Based Insights: The system identified not just overall sentiment, but also sentiment tied to specific banking aspects (e.g., app, card, support), highlighting actionable areas for service improvement.
- Visualization: Clear trends in sentiment over time and across aspects were visualized using bar charts, word clouds, and heatmaps, making insights accessible to stakeholders.
- Business Value: The approach enables banks to proactively address customer pain points, optimize services, and enhance customer satisfaction, aligning with industry best practices.

FUTURE SCOPE

The field of sentiment analysis is rapidly evolving, and several promising directions can further enhance the impact and sophistication of such systems:

1. Advanced Deep Learning Models

- Transformers & Contextual Embeddings: Incorporate models like BERT or RoBERTa for deeper contextual understanding and improved sentiment classification, especially in cases of sarcasm or complex expressions.
- Multimodal Analysis: Expand analysis to include not just text, but also images, audio, or video from customer interactions for a comprehensive view.

2. Real-Time and Predictive Analytics

- Live Monitoring: Deploy real-time sentiment dashboards to alert banks instantly about spikes in negative sentiment, enabling immediate service interventions.
- Predictive Sentiment: Use sentiment trends to anticipate customer churn or dissatisfaction before it escalates, supporting proactive retention strategies.

3. Broader Application and Integration

- Integration with Chatbots: Enhance AI-driven customer support by enabling chatbots to detect emotions and tailor responses dynamically.
- Personalized Banking: Leverage sentiment insights to provide personalized product recommendations and engagement strategies.

4. Multilingual and Regional Expansion

- Support for Regional Languages: Extend the system to analyze feedback in Hindi and other Indian languages, increasing inclusivity and coverage.
- Cross-Platform Data: Incorporate data from social media, emails, and call transcripts for a 360-degree customer view.

5. Explainability, Fairness, and Ethics

- Model Explainability: Develop interpretable models that can justify sentiment predictions, increasing trust among business users and regulators.
- Bias Mitigation: Regularly audit models for bias and ensure fairness across demographic groups.

- Privacy & Compliance: Strengthen data privacy measures and comply with evolving regulations on customer data use.

6. Continuous Learning and Adaptation

- Automated Retraining: Implement frameworks for ongoing model retraining with new feedback to adapt to changing language and sentiment trends.
- Community Collaboration: Contribute to open-source sentiment analysis initiatives and share best practices for collective advancement of the field.

RESOURCES & CITATION

Below is a comprehensive list of resources and citations used throughout this sentiment analysis project, including code, datasets, libraries, and key references for methodology and best practices.

1. Main Project Files and Data

- Literature Review:
  - *Literature Review_SA.pdf*
- Jupyter Notebook (Code & Workflow):
  - *Code_SA.ipynb*
    Contains the full pipeline for data loading, cleaning, feature engineering, sentiment modeling, aspect extraction, and visualization.
- EDA and Data Cleaning Report:
  - *Weekly-Report_EDA.pdf*
    Documents the step-by-step data cleaning, preprocessing, exploratory data analysis, and initial insights.
- Dataset:
  - *bank_reviews3.csv*
    Source of 999 Indian bank customer reviews, including fields for date, bank, address, rating, review text, useful count, and polarity.

2. Python Libraries and Tools

- pandas: Data manipulation and analysis.
- numpy: Numerical operations.
- matplotlib, seaborn: Data visualization.
- wordcloud: Visualizing word frequency.
- nltk: Tokenization, stopword removal, stemming, n-gram analysis.
- gensim: Word2Vec embeddings for semantic feature extraction.
- textblob: Sentiment scoring and polarity detection.
- spaCy: Advanced NLP tasks and sentiment analysis extensions.
- scikit-learn: Feature extraction (CountVectorizer), model evaluation, and metrics.
- re, string: Regular expressions and string manipulation for text cleaning.

3. Sentiment Analysis Techniques and References

- VADER: Lexicon-based sentiment analyzer, effective for short texts and social media.
- TextBlob: Simple API for polarity and subjectivity scoring.
- spaCy: Modern NLP library with extensions for sentiment analysis and aspect extraction.
- Word2Vec: Neural embedding model for capturing semantic relationships in text.

- Aspect-Based Sentiment: Extraction of key aspects (e.g., app, card, support) using n-grams and keyword matching.

4. Methodology and Best Practices

- Data Cleaning & Preprocessing:
  - Text normalization, punctuation removal, stopword filtering, tokenization, lemmatization, stemming, deduplication, and handling of contractions and slang.
- Exploratory Data Analysis (EDA):
  - Text length and word count analysis, frequency distributions, class imbalance checks, and outlier detection.
- Feature Engineering:
  - Bag-of-Words, TF-IDF, and Word2Vec vectorization.
- Model Training & Evaluation:
  - Rule-based and supervised learning approaches, accuracy and loss tracking, and cross-model comparison.

5. Project Repository

https://github.com/Madhu-2161/Sentiment-Analysis-for-Customer-Feedback