BIRMINGHAM CITY
University

Exploring Sentiment Analysis in Low-Resource Languages: Unveiling limitations in translation libraries

Submitted By - Shouvik Das

Student ID – 22196026

Supervisor – Dr Antonio Nehme

Assistant Lecturer in Computer Science School of Computing & Digital Technology Birmingham City University, Birmingham, United Kingdom

Date – 08-01-2023

Contents

Acknowledgements

I extend heartfelt thanks to my supervisor, Dr Antonio Nehme, for his invaluable guidance, insights, and encouragement throughout the research process. Gratitude is also extended to the Birmingham City University academic community, especially the Computer Science department faculty, for fostering an environment of intellectual curiosity. Special thanks to Prof. Rasheed Mohammad, Prof. Stish Sarna, Prof. Samer Bamansoor, Prof. Faisal Sayed, Prof. Essa Sahra, and Dr Khaled Mahbub for their impactful courses. Lastly, my family and friends provided unwavering support, motivating perseverance through challenges, and inspiring aspirations for meaningful contributions in multilingual sentiment analysis.

Abstract

Sentiment analysis systems demonstrate encouraging accuracy for high-resource languages like English with abundant data and research. However, for low-resource languages like Hindi with limited resources, significant barriers persist in effective cross-lingual understanding. Machine translation has been explored to leverage high-resource models by translating low-resource languages.

This dissertation comprehensively evaluates machine translation's efficacy in quantifying how adapted current libraries are for supporting multilingual sentiment classification. Google Translate and Hugging Face Transformers are assessed empirically on Hindi review datasets. Machine learning models including Naive Bayes, SVM, Random Forests, Logistic Regression and LSTM are trained on translated text and contrasted with performance on English reviews.

Findings reveal critical gaps in translating Hindi semantics into informative English, evidenced by lower prediction accuracy compared to English benchmarks across models. Detailed error analyses uncover variability and emergent mis-translation modes that diminish meaning integrity. Strategies cantered on retaining meaning critical components throughout translation for sentiment analysis are prescribed.

Conclusions highlight concerns around adaptiveness of state-of-the-art translation libraries, the customization need for distinguishing meaning critical text, and the necessity of multilingual corpus and model co-development. Translation methods that erode perspectives threaten computational understanding across languages.

Visual representations illustrate performance gaps on Hindi test samples post-translation versus English benchmarks across models and libraries.

Chapter 1

---

1. Introduction

Sentiment analysis assesses opinions, emotions, attitudes, and perspectives in textual data **(Liu, 2012).** The surge in user-generated content on digital platforms has made sentiment analysis crucial for various domains **(Medhat et al., 2014).** Current approaches mainly focus on high-resource languages like English, while low-resource languages face limitations in data, leading to restricted analysis capabilities **(Dashtipour et al., 2016).** Addressing sparse data for low-resource languages is crucial, considering the economic and social impact of multilingual expressions **(Haddow et al., 2022).**

Enhancing sentiment analysis in non-English languages is vital due to their economic influence. Neural machine translation techniques show promise in mitigating this gap by converting low-resource dialects into dominant languages like English **(Zhou et al., 2020).** However, concerns about accurately conveying emotive semantics and preserving subjective orientations during translation persist **(Song et al., 2019).**

This study explores the effectiveness of neural machine translation pipelines, focusing on Google and HuggingFace Transformers, for cross-lingual sentiment analysis from Hindi to English. Text preprocessing uses the Natural Language Toolkit (NLTK), while TF-IDF vectorization generates numerical features. Machine learning classifiers (LSTMs, SVMs, Logistic Regression, Random Forests, Naive Bayes) are trained on translated Hindi text and evaluated against English benchmarks on a sentiment-labelled movie review dataset. Recommendations aim to optimize neural translation for core affective transfer across languages.

This dissertation employs empirical investigation techniques, integrating natural language processing libraries with machine learning algorithms to assess the efficacy of machine translation in facilitating multilingual sentiment analysis. The structural breakdown is as follows:

Chapter 1: Introduction outlines motivation, the role of machine translation, research aims, and questions.

Chapter 2: Literature Review delves into existing research on sentiment analysis, machine translation, and challenges in low-resource languages.

Chapter 3: Methodology details the translation process, model training, performance evaluation, and comparative analysis.

Chapter 4: Results showcase outcomes from comparative experiments, highlighting model accuracy differences on translated Hindi vs native English datasets, with visualized translation errors.

Chapter 5: Discussions offer insights into key observations, limitations of contemporary translation systems, and potential enhancements.

Chapter 6: Conclusion summarizes findings, emphasizing implications for advancing inclusive cross-lingual sentiment analysis via purpose-built translation systems.

1.1 Background

Cross-lingual sentiment analysis (CLSA) integrates machine translation to tackle challenges from lack of linguistic resources like labeled data and lexicons for low-resource languages **(Uryupina et al, 2022).** However, morphologically rich languages like Hindi pose complex barriers with ubiquitous cultural perspectives, code-mixing, and scarce compatible datasets **(Vadhariya et al., 2022).** Accurately assessing sentiment in such languages despite resource limitations necessitates evaluating machine translation's efficacy in preserving affect and perspective when pivoting via English **(Haddow et al., 2022).**

Recent studies reveal even sophisticated neural translation systems struggle to retain emotive semantics between divergent languages, especially for extremely low-resource dialects **(Wang et al., 2022).** Surmounting core translation difficulties in conveying elemental affective components without distortion remains pivotal while pursuing egalitarian multilingual sentiment analysis **(Casas et al., 2022).**

Nevertheless, enabling performant Hindi CLSA harbors immense value for stakeholders aiming to streamline experiences for multilingual users **(Kaushik et al., 2020).** It also aids forecasting public reactions to geo-political events through assimilating diverse citizen perspectives **(Pandia et al., 2018).** Hence, assessing contemporary translation's efficacy in conveying sentiment flows between distant languages serves as essential foundation before devising enhancements explicitly optimizing balanced perspective retainment during dialect migrations.

.2 Aim and Objective

To perform an extensive evaluation and comparative analysis of modern neural machine translation techniques for enabling cross-lingual sentiment classification from low-resource languages into English. With the following objectives.

- Conduct background research on sentiment analysis evolution, machine learning shifts, and neural machine translation for multilingual analysis.
- Create test datasets by translating Hindi reviews into English using Google Translation and HuggingFace’s Tranformers library.
- Implement Logistic Regression, SVM, LSTM, Random Forests, and Naive Bayes for sentiment classification on both translated Hindi reviews and direct English review datasets, assessing performance impact.
- Evaluate translation library reliability for multilingual sentiment preprocessing, addressing inaccuracies, and recommending improvements for cross-lingual sentiment analysis alignment with English benchmarks..

1.3 Research Questions

Following the successful completion of this investigation, the following questions will be addressed.

- Why perform sentiment analysis in low-resource languages?
- How effectively do current neural machine translation techniques preserve sentiment when translating from low-resource languages to English?
- How to integrate machine translation and learning classifiers for cross-lingual sentiment analysis from low-resource languages to English?
- Which machine translation methods minimize meaning and sentiment distortions for accurate cross-lingual sentiment classification?
- Which machine learning classifier achieves the highest accuracy in classifying sentiments from translated text?
- How significant is the performance gap in cross-lingual sentiment classification metrics between translated low-resource text and natively English data?

1.4 Problem Statement

Sentiment analysis has become an indispensable technology for extracting affective insights and guiding decisions based on textual data across private and public sector organizations. However, current state-of-the-art neural approaches and benchmarks established predominantly for high-resource languages like English still do not adequately support reliably assimilating the diverse multicultural perspectives expressed in thousands of low-resource languages globally. Moreover, existing systems focused on English achieve significantly higher accuracy, while experiments conducted for low-resource languages like Hindi, Punjabi and Arabic have resulted in lower predictive performance. This fundamental capability gap poses profound barriers not only to equitably understanding sentiment diversity manifested across interconnected digitally enabled societies worldwide but also severely inhibits propagating crucial sentiment analytics innovations to disadvantaged regions still lacking access **(Welocalize, 2023).**

Surmounting this language barrier emerges as a pivotal challenge that must be tackled as a starting point before truly egalitarian and representative multilingual sentiment analysis capacities can be pursued earnestly by the artificial intelligence community collectively. Without capabilities tailored to optimally retain perspectives expressed in multiple languages, contemporary systems will continue to distort multilingual sentiment flows and disadvantage minority languages by inhibiting assimilation of their uniquely expressed sentiments **(Wang et al. 2022).**

Therefore, comprehensively evaluating limitations of existing approaches across linguistic diversity is arguably the most critical foundational step at this stage towards nurturing more inclusive and equitable AI design thinking - one that consciously strives to not fail already digitally marginalized language populations. Quantifying deficiencies across languages and enhancing algorithms to equitably assimilate multicultural sentiment expressions represents a milestone toward unlocking the possibilities of sentiment understanding across all of humanity.

1.5 Ethical Considerations

Ethical considerations in cross-lingual sentiment analysis involve rigorous evaluations of data policies, model development, and deployment protocols to ensure privacy, cultural respect, and minimize biases. Stringent data collection practices, removing personally identifiable information, obtaining informed consent, and supporting the right to erasure uphold privacy standards. Cultural sensitivity is prioritized in dataset curation and annotation, acknowledging diverse language representation and addressing annotator biases. Regular bias assessments during translation and model development stages, involving quantitative metrics and qualitative surveys, enhance fairness and justice. Technologist awareness and community participation contribute to addressing real-world impacts and ensuring accountability. Defining intended applications, measuring benefits and harms, and deploying testing-in-production approaches provide situational awareness for needs-based customizations. Collaborative design with language community leaders enhances receptivity, promoting empowerment in cross-lingual sentiment technologies.

Chapter 2

---

2. Literature Review

Sentiment analysis encompasses automated techniques for quantification and characterization of elemental human affective signals like emotions, attitudes and subjective perspectives embedded within unstructured natural language content using algorithmic approaches for assimilation of text consumption experiences approaching human-like comprehension standards **(Liu et al., 2022).** With exponential proliferation occurring across numerous languages discussing an increasingly diverse set of influential issues on online platforms spanning from socio-political developments, cultural events to product preferences and reviews, deriving actionable insights through large-scale advanced sentiment mining has steadily become pivotal among a multitude of interconnected private and public sector organizations aiming to attain decisive competitive advantage **(Medhat et al., 2014).**

2.1 Mainstream Approaches Still Target Select Languages

However, empirical assessments into current state of technology adoption trends surrounding sentiment analysis reveals mainstream contemporary strides still predominantly concentrate within computational bounds of very few select languages that are abundantly resourced. For instance, native English continues witnessing unparalleled growth in labelled corpora annotated with sentiment tags at scale containing hundreds of millions of rich exemplars manifesting multifaceted real-world emotive diversity. Similarly, constant fluxes in increasingly sophisticated neural architectures like BERT, T5 and evolving derivatives that continue exhibiting enhanced contextual comprehension capabilities using self-supervision at unprecedented parametric scales also largely revolve around the English language as primary benchmark. **(Lettria, 2021)** Furthermore, unmatched dedicated industrial research prioritizations driving continual infusions of data, infrastructure and talent needed to rapidly mature sentiment modelling standards through sustained introduction of array of competitive benchmarks also majorly circulate around English and select few other tongues **(Sachdeva et al., 2021).**

In contrast to high-resource languages, approximately 7,000 global languages face barriers limiting inclusion in rapid sentiment analysis advancements **(Jain et al., 2020).** This restricts technology dividends for interconnected communities. Challenges include scarce labeled data, with many dialects having tiny samples. Embedding scopes remain limited due to insufficient pretraining data. Prioritizing few benchmark languages causes marginalization of minorities. This inadvertently censors perspective diversities expressed in global dialects beyond English, despite their real-world influence potential. In summary, data scarcity, limited embeddings, and prioritizing only a few languages hampers progress for thousands of low-resource languages. This limits equitable advancement access and risks censoring diverse perspectives in multiple global dialects.

2.2 Integrating Machine Translation as a Pathway

However, recent research trends also point towards substantially augmented content creation occurrence around locally pertinent issues on online discussion platforms including social media channels in several non-English languages beyond those dominating current advancements. Such languages native to crucial economic regions like Middle East, South Asia and parts of Africa continue exhibiting noticeable uptrends in daily commentary numbers discussing diverse real-world developments spanning from political situations, cultural events to product experiences **(Casas et al. 2022).** With visible upshifts occurring in multilingual opinion visibility online indicating growing real-world influence coupled with demands for automated understandings but minimal contemporary competitive solutions optimized across languages offering seamless interoperability, cross-lingual sentiment analysis enhancement urgency only keeps accumulating annually across private and public sector systems in order to assimilate diversity of perspectives at scale for streamlined decisions.

Research efforts are exploring how to leverage advanced machine translation (MT) for sentiment analysis across languages. Combining deep neural networks with sophisticated attention mechanisms shows promise in improving translation quality and facilitating smoother dialect migrations. The idea is to bridge the gap between resource-scarce languages with limited sentiment analysis tools and resource-rich languages like English. By translating less-resourced languages into English, we can then utilize its extensive sentiment analysis toolset to extract insights from the translated **text (Balahur & Turchi, 2014).** However, challenges remain. Neural systems can struggle to preserve subtle emotional tones and subjective nuances during translation, especially for languages with little similarity **(Song et al., 2019).**

2.3 Preserving Sentiment Semantics Critical

Specifically, even exceedingly robust contemporary neural machine translation systems exclusively optimized at present to maximize conventional metrics like generic accuracy and fluency during standard dialect transitions demonstrate noticeable difficulties in correctly resolving and subsequently transferring implicit cultural nuances, figurative colloquial expressions as well as foundational core affectual components that constitute building blocks for reliable sentiment assessments during migrations between linguistically distant languages **(Sohrab et al. 2022).** For instance, shared non-verbalized contextual knowledge frequently conveyed through cultural historical connotations often struggle to maintain integrity. Similarly, syntactical restructuring affecting mood portrayals also suffers through failed part-of-speech transfers **(KESSLER and SCHUTZE, 2012).** This occurs partly due to most neural translation models still lacking dedicated biases accounting for proactively retaining semantic integrity of critical subjective perspective affectual assets above conventionally meeting generic translation accuracy requirements alone **(Wang et al., 2022)**. Hence, multiple recent studies argue that developing capabilities to preserve sentiment flows through key constraint driven losses targeted towards minimizing meaning distortions must encompass impending areas of focus **(Haddow et al. 2022).** Improvement necessities encompass enhancing techniques manifesting more granular control within sophisticated neural translation architectures to model inter-language emotive concept transitions, leveraging sentence-level

representations over word-based embedding spaces as well as balanced affectual regularizations that allow retaining perspective integrity throughout inevitable dialect migrations in future globally unified sentiment mining systems.

In summary, the surge in multilingual content, often tied to global decisions, emphasizes the need for improved sentiment analysis tools across all languages. While neural machine integration has future potential, there are limitations, requiring purposeful constraints to preserve core affective components and viewpoint integrity during translation. Advancements must prioritize collective principles, addressing technological gaps to minimize risks of censoring underrepresented voices due to flawed software infrastructure and transition mechanisms.

2.4 Rising Prominence of Sentiment Analytics Research

As per Google Trends aggregated data presented in Figure 2, global search interest for the term "sentiment analysis" has seen remarkable growth since 2004 reflecting the field's increasing prominence and utility across industries. Particularly noticeable surges occurred around 2008 likely due to political and market events followed by further uptake in recent years as social media proliferation enabled many new applications.

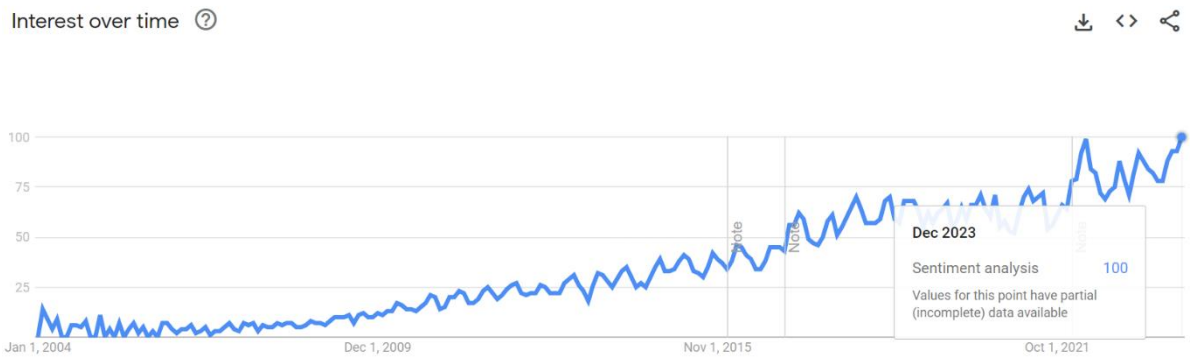


Fig 1. Growing worldwide interest on “Sentiment Analysis” over time.

The enclosed map visually portrays the normalized search volume score by country calculated by Google Trends for the term "sentiment analysis" indicating the topic's surging prominence among Asian regions significantly outpacing rest of the world. Darker shades like India , Ethiopia and South Korea marker higher intensity of uptake compared to Canada's lagging reception.

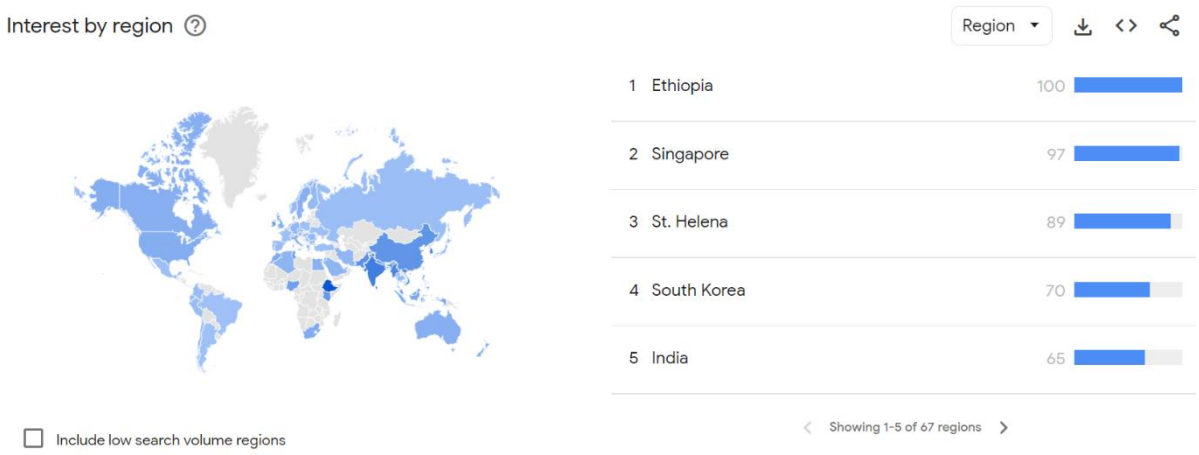


Fig 2. Growing worldwide interest on "Sentiment Analysis" over region.

2.5 Strengths and Weaknesses of Different Contributions in Low-Resource Language Sentiment Analysis:

While sentiment analysis flourishes in resource-rich languages, navigating the subtleties of under-resourced tongues demands a more delicate approach. Here, machine learning emerges as a beacon of hope, offering adaptability and the ability to learn from limited data. Unlike rule-based methods, machine learning algorithms can evolve to capture the unique linguistic nuances of low-resource languages **(Bhavsar, 2021)**.Their prowess lies in their ability to learn from scarce labeled data, a precious commodity in this domain **(Nguyen et al., 2019).** Furthermore, algorithms like LSTMs can automatically extract hidden features and patterns within texts, alleviating the need for extensive manual feature engineering **(Lu & Yang, 2018).** This versatility is machine learning's true strength, offering a diverse toolbox ranging from classical models like SVMs and Random Forests for interpretability to deep learning networks for capturing intricate text relationships **(Khan *et al.*, 2022)**.

Low-resource sentiment analysis in machine learning encounters challenges like overfitting, deep learning's computational costs, and interpretability issues with complex models. Hybrid approaches, blending classical and deep learning algorithms, offer robust, interpretable performance by combining SVMs/Random Forests' interpretability with deep learning's feature extraction. Adapting specific algorithms to the language's characteristics improves performance and mitigates biases. Your dissertation, by exploring methods, their suitability, limitations, and solutions, can contribute significantly to this evolving field. Demonstrating a nuanced understanding of machine learning's strengths, weaknesses, and strategic application in this under-resourced domain showcases the significance and potential of your research. In summary, a nuanced approach to machine learning, considering its capabilities and adapting to the low-resource context, is crucial for impactful sentiment analysis research.

2.6 Related Works and Research Gaps Present

In the realm of multilingual user-generated content, particularly in languages with limited computational support, the integration of machine translation into cross-lingual sentiment analysis has gained attention **(Mercha and Benbrahim, 2023).** Recent focus involves

leveraging translation to bridge resource gaps by translating inputs into English before applying prebuilt affective algorithms. Noteworthy studies, such as **(Wan,2009)** translating Chinese reviews and **(Baniata et al. ,2021)** benchmarking Arabic text, have explored this approach, revealing promises and inaccuracies in sentiment conveyance. Challenges include difficulties in conveying cultural contextual references, especially with older systems.

In the machine learning domain, Wang et al. (2016) developed Bi-LSTM models surpassing SVM and Naïve Bayes for cross-lingual social media content, while **(Zhang et al. ,2019)** found CNNs effective but lacking generalizability. **(Devlin et al. ,2019)** demonstrated gains through ensemble methodology, combining CNN, Bi-LSTM, and regression strengths to handle multilingual challenges. Specialization needs were identified.

Specifically for Hindi sentiment analysis through neural machine translation, gaps persist. Customized constraints for metaphoric expressions show initial enhancements but struggle with sociolinguistic complexities **(Patwa et al., 2022).** Phrase-level annotations reveal emotion intensity reductions when translating into English, pointing to incomplete lexical mappings **(Boghe, 2021).** Co-developed translation systems tailored for dialects are crucial for overcoming entropy loss and unlocking equitable neural Hindi sentiment analysis.

Despite increased Kaggle entries in sentiment analysis, accuracy and reliability issues persist **(Shrivastava and Kumar, 2020)**. Addressing these challenges is crucial for consistent outcomes, requiring purpose-built upgrades. Rigorous examination of transformer-based and Google-based translation on Indian languages using benchmarked deep learning combinations remains relatively underexplored but is vital for reaching underserved populations. The proposed research aims to fill this literature gap through extensive quantitative and qualitative surplus loss assessments.

Chapter 3

---

3. Methodology

This chapter outlines the systematic methodology for evaluating machine translation techniques in cross-lingual sentiment analysis. The process includes obtaining a Hindi text dataset, cleaning and normalizing the corpus, creating translation pipelines using Google and HuggingFace libraries, training deep learning classifiers on translated text, and assessing performance against English benchmarks. The method aims to offer comprehensive insights into limitations of existing translation systems for sentiment analysis. Proposed improvements can enhance language translation for diverse speakers, promoting equitable expression and minimizing misinterpretation in AI-driven tasks like opinion mining.

3.1 Literature Methodology Review

The extensive literature review in this dissertation critically assesses traditional sentiment analysis methods, focusing on rule-based techniques and exposing their inherent limitations in low-resource languages **(Wang, 2021).** Recognizing the difficulties in hand-crafting effective rules for languages with limited training data, this research shifts towards exploring the potential of advanced machine learning algorithms for a more nuanced understanding of sentiment in these under-resourced contexts. This includes the application of deep learning models like Long Short-Term Memory (LSTM) networks **(Khan *et al.*, 2022)**, alongside classical approaches like Support Vector Machines (SVMs) adapted for low-resource settings **(Shmilovici, 2006),** and the integration of Logistic Regression and Random Forest to harness the inherent information within limited data. This diverse selection of models, empowered by their inherent capacity to learn from scarce resources, empowers the proposed research to move beyond simply classifying sentiment, enabling the extraction of subtle emotional nuances and context-dependent interpretations even in resource-constrained languages. This emphasis on machine learning aligned with low-resource adaptation techniques not only aligns with the broader shift in the field, but also sets the stage for the innovative contributions this research will make to the realm of cross-lingual sentiment analysis in less studied languages.

3.2 General Procedure for Sentiment Analysis

Building accurate sentiment analysis models requires structured machine learning. We train models on vast amounts of labelled text data, like reviews or forum posts, to capture emotion within snippets. To prepare this data, we clean and format it, ensuring models generalize to real-world scenarios. Algorithms then analyse the data, extracting emotional features using techniques like linear classifiers or deep neural networks. We continuously test and refine these models to ensure accurate predictions. In real-world use, models evolve, incorporating

contextual factors for precise sentiment readings. Responsible design that values transparency and user feedback is crucial for ensuring ethical and reliable analysis.

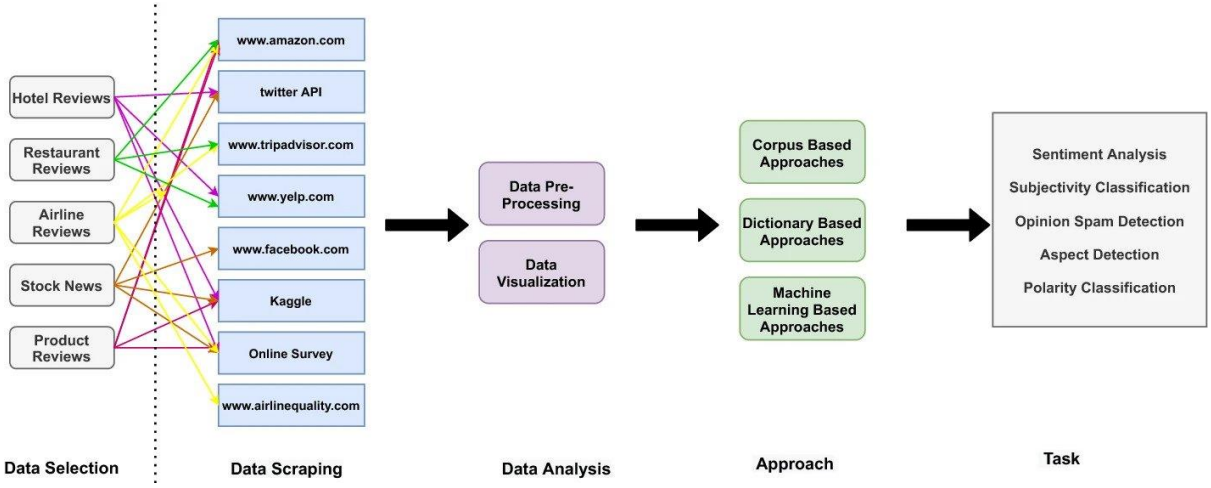


Fig 3. General Procedure for Sentiment Analysis (Rao and Kulkarni, 2022)

In this dissertation study, machine learning based classifiers are utilized for training sentiment analysis models using both translated Hindi review data and direct English review benchmarks for rigorous performance assessment.

3.3 Design: Overview of the proposed system

The research methodology follows a structured framework to holistically gauge machine translation's efficacy for sustaining sentiment flows via Hindi to English pipelines. Compiling sufficiently large text corpus representing nationwide linguistic conventions enables statistically unbiased analytical evaluations later. Through systematic preprocessing, noise removal and validation checks, analysis-ready formats emerge enhancing model robustness. Leveraging advanced neural capabilities provide reasonably fluent transformed Indic sentences while aiming to maximally retain core subjective traits.

Subsequently, training diverse classifiers encompassing Long Short-Term Memory networks, Support Vector Machines, Logistic Regressions, Random Forests and Naïve Bayes model on both translated and native English sets facilitate comparative assessment revealing gaps introduced during transitions. Further investigations attribute causations to losses in localized cultural nuances. Finally, contrasting against multilingual expectations platform evidence-driven recommendations - such as constraints-driven losses maximizing coherence reinforced through versioned retraining routed via community inputs aimed at advancing contextual adaptivity.

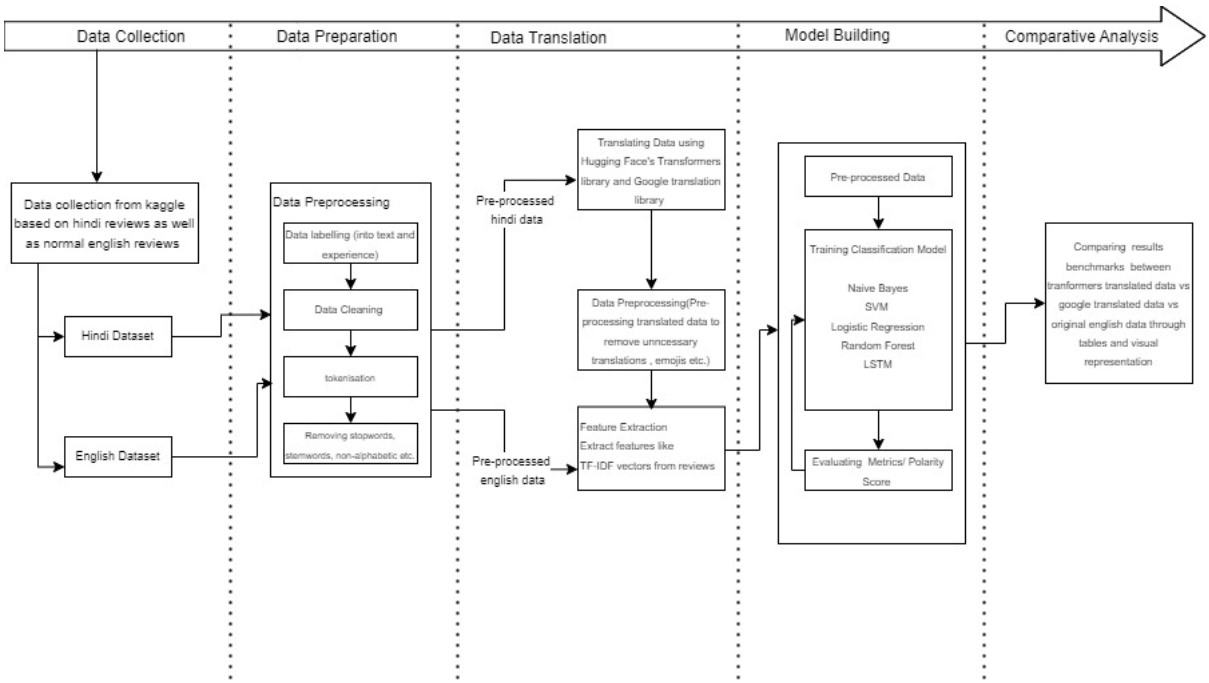


Fig 4. Design for the proposed system Architecture

3.4 Dataset Collection

Quality datasets significantly influence experimental insights, ensuring results generalize to real-world applications. To ensure research translates to real-world applications, we've carefully curated authentic Hindi and English review datasets that reflect natural online expression. These diverse samples, from reputable sources, provide a solid foundation for statistically significant experimentation.

3.4.1 Hindi Review Dataset Description

The domain-specific corpora, sourced from genuine product feedback channels, provide a unique advantage for benchmarking sentiment analysis pipelines across various commercial applications. This dataset, acquired from IIT Patna's Advanced Data Analytics Lab and available on Kaggle, comprises approximately 5000 product feedback excerpts in native Hindi **(IIT Patna product Reviews, 2023).** Stratified into positive, negative, and neutral sentiments, the dataset undergoes meticulous annotations and structuring for effective experimentation. Renaming columns to 'experience' and 'text,' it is formatted for machine learning algorithm compatibility, offering insights into customer sentiments across sectors.

	experience	text
0	neutral	यह एस्पेक्ट रेशो का ईश्यू है और हम आशा करते है...
1	positive	लेकिन इस तरह के एक मॉडल के एक घर कंप्यूटर के ल...
2	positive	गिर वन राष्ट्रीय उद्यान बाघ संरक्षित क्षेत्र ह...
3	neutral	और हां , इस फिल्म में हर किरदार भारद्वाज को भर...
4	positive	फोन उपयोग में बेहद ही आसान है और टच रिस्पॉरन्स...
...	...	...
5214	neutral	कलाकारों की संवाद अदायगी में उच्चारण की स्पष्ट...
5215	neutral	जसपिन्दर कौर निर्देशित ' दिल्लीवाली जालिम गर्ल...
5216	negative	अगर आप एक पॉवर यूजर हैं , तो आपको इस डिवाइस को...
5217	negative	हां , फिल्म में हिंदी शब्दों के प्रति बरती गई ...
5218	negative	जाहिर रूप से , एंड्रॉयड डिवाइस के यूजर्स , जो ...

5219 rows × 2 columns

```
In [3]: df.shape
Out[3]: (5219, 2)
```

Fig 5. Hindi Review Dataset Preview and total number of rows and columns in the dataset

```
In [5]: #checking Datatype info of the features
        df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5219 entries, 0 to 5218
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   experience  5219 non-null   object
 1   text        5219 non-null   object
dtypes: object(2)
memory usage: 81.7+ KB
```

Fig 6. English dataset features preview

Overview of Hindi Dataset Features

Features	Description
experience	Annotated sentiment label, using 1, 2, 3 for negative, neutral, positive.
text	Raw Hindi review excerpts, diverse opinions for embedding and modeling.

Table 1. Hindi Dataset Features and its Description

3.4.2 English Review Dataset Description

A comparable dataset of around 6000 Twitter excerpts on COVID-19 vaccines was obtained from Kaggle **(Covid-19 Vaccine Tweets with Sentiment Annotation, 2021)** for benchmarking. Covering diverse opinions, annotated as positive, neutral, or negative, it enables a 3-class classification challenge. Mirroring the Hindi reviews dataset, it facilitates cross-lingual comparisons. The curated English dataset, sourced for comparison, highlights efficacy limitations in current cross-lingual techniques.

	tweet_id	label	tweet_text
0	1.360342e+18	1	4,000 a day dying from the so called Covid-19 ...
1	1.382896e+18	2	Pranam message for today manifested in Dhyan b...
2	1.375673e+18	2	Hyderabad-based ?@BharatBiotech? has sought fu...
3	1.381311e+18	1	Confirmation that Chinese #vaccines "don□t hav...
4	1.362166e+18	3	Lab studies suggest #Pfizer, #Moderna vaccines...
...	...	...	...
5995	1.370975e+18	2	@Swamy39 Dr. @Swamy39 jee :\n\nMany people lik...
5996	1.379827e+18	3	So happy to be fully vaccinated against COVID-...
5997	1.384789e+18	2	Serum Institute of India announces cost of Cov...
5998	1.382355e+18	1	@___batshitcrazy @BarrowfordHead @Bectully I h...
5999	1.380051e+18	2	The smart sympathy attends into the oblong not...

6000 rows × 3 columns

```
In [10]: df.shape
Out[10]: (6000, 3)
```

Fig 6. English Review Dataset Preview and total number of rows and columns in the dataset

```
#checking Datatype info of the features
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   tweet_id    6000 non-null   float64
 1   label       6000 non-null   int64
 2   tweet_text  6000 non-null   object
dtypes: float64(1), int64(1), object(1)
memory usage: 140.8+ KB
```

Fig 7. English dataset features previe

Overview of English Dataset Features

Features	Description
tweet_id	Unique numeric ID for each tweet as float type.
label	Integer encoding sentiments (1, 2, 3 for negative, neutral, and positive).
tweet_text	Raw tweet content for natural language opinions.models use for classification.

Table 2. English Dataset Features and its Description

3.5 Overview of Comparative Evaluation Process

This dissertation follows a three-step process to evaluate machine translation's capabilities in preserving sentiment when converting low-resource languages like Hindi to English.

First, the Hindi dataset is rigorously preprocessed for accuracy and clarity. Statistically significant subsets are extracted to create tailored test partitions. Sentiment classifiers like LSTM, Naive Bayes, SVM, Logistic Regression and Random Forest are trained on the native Hindi corpus to establish performance benchmarks within the source language.

Next, Hindi test samples are translated to English using Google and HuggingFace translator libraries. The translated text is used to retrain the classifiers. By comparing accuracy before and after translation, the impact of the language shift is quantified. An English dataset undergoes minimal preprocessing for an independent comparative analysis, isolating translation engine influence.

Finally, a comprehensive assessment synthesizes insights across test bed creation, sentiment approximation through models, and contrasting validation results. This reveals limitations, strengths, and improvements needed in contemporary translation systems to enable enhanced multilingual sentiment analysis. Targeted analyses examine specific sentiment categories and behavior of markers under translation, illuminating mechanisms and productive areas for future research.

Overall, this multifaceted methodology promises nuanced understanding of multilingual sentiment analysis challenges and opportunities for low-resource languages. By highlighting current limitations and future enhancements, it empowers more robust, culturally sensitive tools to bridge communication across diverse linguistic communities.

3.6 Importing Necessary Libraries

Importing Libraries

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

Importing Natural Language Tool Kit Library for text pre processing

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import re
```

Importing Neural Engine libraries for translation

```
import nest_asyncio
from tqdm import tqdm
import csv
#HuggingFace's Translation Library "Transformers"
from transformers import MarianMTModel, MarianTokenizer
from concurrent.futures import ThreadPoolExecutor
#Google Translation Library
from googletrans import Translator
```

Importing Library for text label encoding and vectorization

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
```

Importing Libraries for model Building

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Embedding, BatchNormalization, Dropout,Bidirectional
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
```

Fig 8. Importing Necessary Libraries

Library	Description
pandas (pd)	A Python library for efficient data analysis with DataFrames.
numpy (np):	A Python library supporting large arrays and mathematical functions.
seaborn (sns)	A statistical data visualization library based on Matplotlib..
matplotlib.pyplot (plt)	A comprehensive library for creating visualizations in Python.
WordCloud:	A library for creating word clouds, visualizing text data.
nltk:	Natural Language Toolkit for working with human language data.
re:	Module for regular expressions, useful for pattern matching in text.
nest_asyncio	Module for running asyncio code in non-native asyncio environments.

tqdm	Library for creating progress bars in loops and iterable computations.
csv	Module for reading from and writing to CSV files.
transformers (from Hugging Face)	A library providing pre-trained NLP models and utilities
concurrent.futures.ThreadPoolExecutor:	Module for asynchronously executing callables in separate threads using GPU.
googletrans (Translator)	A Python library using Google Translate API for language translations.
sklearn (scikit-learn)	A machine learning library for data mining and analysis.
train-test-split	A function for dividing datasets into training and testing sets.
gridsearchCV	A module for systematic hyperparameter exploration using cross-validation.
StratifiedKFold	A class for stratified k-fold cross-validation in scikit-learn.
tensorflow.	An open-source machine learning library developed by Google.
Sequential, LSTM, Dense, Embedding, BatchNormalization,Dropout, Bidirectional (from tensorflow.keras.layers)	These are various layers used in constructing neural network models using TensorFlow's Keras API. Sequential is a linear stack of layers. LSTM is a type of recurrent neural network layer. Dense is a fully connected layer. Embedding is used for word embeddings. BatchNormalization and Dropout are regularization techniques. Bidirectional creates a bidirectional recurrent layer.
to_categorical,Tokenizer, pad_sequences, Adam, EarlyStopping (from tensorflow.keras.utils and callbacks)	Utility functions and callbacks for one-hot encoding, tokenization, padding, optimization, and early stopping in TensorFlow's Keras API.

Table 3. Libraries and Its Description

3.7 Data Preprocessing and Visualization

A methodical examination of Hindi and English reviews includes visualizations portraying language variations, sentiment distributions, and affective tones. Quantitative analysis explores sentence lengths and grammatical structures, guiding preprocessing steps for downstream model integration. This comprehensive investigation addresses dataset intricacies, ensuring reproducibility and minimizing biases in analytical inference.

3.7.1 Working with Hindi Dataset

The initial Hindi reviews dataset from Kaggle contained separate training, test, and validation splits as individual CSV files. To streamline the experiment workflow, I strategically merged these subsets into a single, unified dataset file. This consolidation was achieved through random stratified sampling, ensuring the combined dataset faithfully reflects the composition of the original splits. This blending not only preserves representativeness across groups but also dramatically boosts iteration speed during data preprocessing, translation, and model training by eliminating redundant efforts across individual splits. The resulting unified CSV allows for seamless end-to-end workflow management, from data loading and transformation

to convenient test set sampling. In essence, this judicious consolidation minimizes unnecessary overhead in the experimental process, allowing us to focus our efforts on the core task of evaluating translation performance.

```
# Defining column names
column_names = ['experience', 'text']

# Loading the CSV files into pandas dataframes with column names
train_df = pd.read_csv(r'C:\Users\shouv\Downloads\iitp-product-reviews\hi\hi-train.csv', names=column_names)
test_df = pd.read_csv(r'C:\Users\shouv\Downloads\iitp-product-reviews\hi\hi-test.csv', names=column_names)
valid_df = pd.read_csv(r'C:\Users\shouv\Downloads\iitp-product-reviews\hi\hi-valid.csv', names=column_names)

# Concatenating the dataframes along the rows (axis=0)
merged_df = pd.concat([train_df, test_df, valid_df], axis=0)

# Reset index
merged_df.reset_index(drop=True, inplace=True)

# Saving merged dataset
merged_df.to_csv(r'C:\Users\shouv\Downloads\iitp-product-reviews\hi\hindi-data-combined.csv', index=False)
```

```
df = pd.read_csv(r'C:\Users\shouv\Downloads\iitp-product-reviews\hi\hindi-data-combined.csv')
df
```

	experience	text
0	neutral	यह एस्पेक्ट रेशो का ईश्यू है और हम आशा करते है...
1	positive	लेकिन इस तरह के एक मॉडल के एक घर कंप्यूटर के ल...
2	positive	गिर वन राष्ट्रीय उद्यान बाघ संरक्षित क्षेत्र ह...
3	neutral	और हां , इस फिल्म में हर किरदार भारद्वाज को भर...
4	positive	फोन उपयोग में बेहद ही आसान है और टच रिस्पॉरन्स...
...	...	...
5214	neutral	कलाकारों की संवाद अदायगी में उच्चारण की स्पष्ट...
5215	neutral	जसपिन्दर कौर निर्देशित ' दिल्लीवाली जालिम गर्ल...
5216	negative	अगर आप एक पॉवर यूजर हैं , तो आपको इस डिवाइस को...
5217	negative	हां , फिल्म में हिंदी शब्दों के प्रति बरती गई ...
5218	negative	जाहिर रूप से , एंड्रॉयड डिवाइस के यूजर्स , जो ...

5219 rows × 2 columns

Fig 9. Merging and then importing the final combined Hindi Dataset

In the above since original files doesn't contain any type of column names so I have given them column names as **'experience'** for the sentiments and **'text'** for the reviews as a part of **feature engineering** for better usability.

a. Hindi Dataset preprocessing

Now in this step, preprocessing of Hindi data has been done.

i. Checking for any null values

```
df.isna().sum()
```

```
experience    0
text          0
dtype: int64
```

Fig 10. Checking for any null values in Dataset

This shows that each feature has 0 null values.

ii. Performing overall data cleaning.

```
# Downloading the Hindi stop words
nltk.download('stopwords')

# Set of Hindi stop words
stop_words = set(stopwords.words('hindi'))

# Regular expression for stemming
stemmer = re.compile(r'(आ|िया|िए|ेई|ि|ो|ा)')

# Function to remove special characters from text
def remove_special_chars(text):
    return re.sub(r"[^अ-ऋए-ऑओ-नप-रलव-हा-ॅॉॐ-]", " ", text)

# Function to tokenize text
def tokenize(text):
    return nltk.word_tokenize(remove_special_chars(text))

# Function to filter out stop words from tokens
def filter_stopwords(tokens):
    return [token for token in tokens if token not in stop_words]

# Function for stemming
def stemming(token):
    return stemmer.sub('', token)

# Function to preprocess Hindi text
def hindi_preprocess(text):
    tokens = tokenize(text)
    tokens = filter_stopwords(tokens)
    tokens = [stemming(token) for token in tokens]
    return tokens

# Apply hindi_preprocess to the 'text' column of the DataFrame
df['text'] = df['text'].apply(hindi_preprocess)

# Display the DataFrame
print(df)
```

```
     experience                                               text
0       neutral  [एस्पेक्ट, रेश, ईश्यू, हम, श, ने, अपडेट, फिक्स...
1      positive                  [मॉडल, कंप्यूटर, शनदर, विकल्प, ।]
2      positive  [गिर, वन, रष्ट्रीय, उद्यन, बघ, संरक्षित, क्षेत...
3       neutral  [हं, ,, फिल्म, हर, किरदर, भरद्वज, भरद्वज, क्यं...
4      positive  [फन, उपयग, बेहद, सन, टच, रिस्पॉरन्स, उतन, बेहत...
...         ...                                                ...
5214    neutral   [कलकरं, संवद, अदयगी, उच्चरण, स्पष्टत, गौरतलब, ।]
5215    neutral  [जसपिन्दर, कौर, निर्देशित, ', दिल्लीवली, जलिम,...
5216   negative  [अगर, पॉवर, यूजर, ,, पक, डिवइस, खरीदन, चह, खरी...
5217   negative  [हं, ,, फिल्म, हिंदी, शब्दं, प्रति, बरती, गई, ...
5218   negative  [जहिर, रूप, ,, एंड्रॉयड, डिवइस, यूजर्स, ,, उटल...

[5219 rows x 2 columns]
```

Fig 11. Performing overall data cleaning and displaying the results

For optimal preparation of raw text with regional dialects, we've implemented customized cleaning methods for Hindi reviews:

Refining Character Set: Utilizing remove_special_chars for regex substitutions to filter out special characters, minimizing vocabulary gaps.

Linguistic-Aware Segmentation: tokenize strategically segments text into words with charset-aware word boundary identification, preserving meaningful phrases.

Filtering Non-Contextual Words: filter_stopwords dynamically removes language-specific stop words, sharpening focus on content-rich vocabulary.

Unveiling Root Forms: stemming systematically removes suffixes, revealing essential word roots for pattern recognition.

Optimizing for Sentiment Preservation: Custom functions leverage linguistic principles, optimized pipelines, and ensure compatibility with sentiment analysis models.

Maintaining Integrity: Tailored cleaning functions collectively safeguard sentiment conveyances during dataset migrations, preserving human expression for meaningful insights.

iii. Joining the tokens

```
df['text'] = df['text'].apply(lambda tokens:' '.join(tokens))
df
```

	experience	text
0	neutral	एस्पेक्ट रेश ईश्यू हम श ने अपडेट फिक्स जएग ।
1	positive	मॉडल कंप्यूटर शनदर विकल्प ।
2	positive	गिर वन रष्ट्रीय उद्यन बघ संरक्षित क्षेत्र एशई ...
3	neutral	हं , फिल्म हर किरदर भरद्वज भरद्वज क्यं बुलत ?
4	positive	फन उपयग बेहद सन टच रिस्पॉरन्स उतन बेहतर ।
...	...	...
5214	neutral	कलकरं संवद अदयगी उच्चरण स्पष्टत गौरतलब ।
5215	neutral	जसपिन्दर कौर निर्देशित ' दिल्लीवली जलिम गर्लफ्...
5216	negative	अगर पॉवर यूजर , पक डिवइस खरीदन चह खरीदन चहेंगे...
5217	negative	हं , फिल्म हिंदी शब्दं प्रति बरती गई लपरवही अख...
5218	negative	जहिर रूप , एंड्रॉयड डिवइस यूजर्स , उटलुक डॉट क...

5219 rows × 2 columns

Fig 11. Performing overall data cleaning and displaying the results

After initial preprocessing, Hindi review sentences were tokenized for efficient cleaning, eliminating redundant words. To enhance readability while preserving vocabulary improvements, sentences are reconstructed by rejoining tokens. This transitionary step ensures coherence through conjugation alignments and structural adjustments. The resulting consolidated sentences, in a computationally efficient format, become compatible data for advanced neural translation engines using abstracted APIs that accept sentence-level inputs in subsequent steps.

b. Visualisation of Data

Now we'll visualize the data to get more insight into the dataset.

i. Visualising sentiment distribution and proportion throughout the dataset

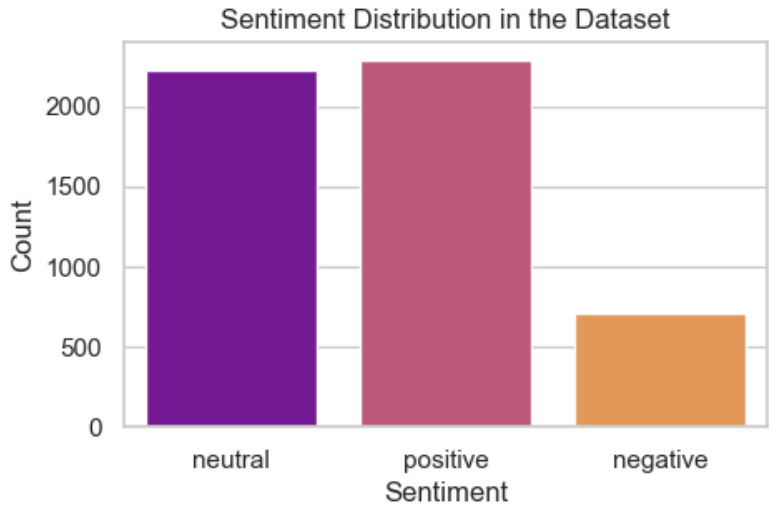


Figure 12. Bar graph for Sentiment Destribution

This bar graph depicts the distribution spread of positively inclined, negative, and neutral sentiment labels across samples within the aggregated Hindi reviews dataset used for training and evaluations.

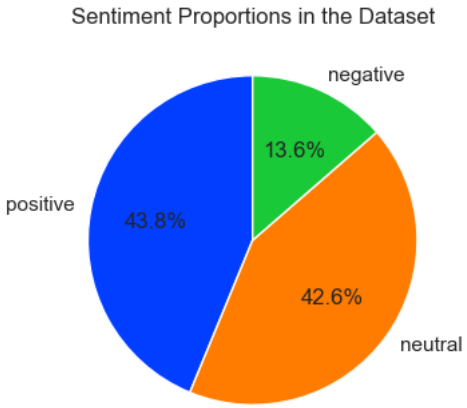


Figure 13. Pie-chart to determine sentiment proportions

To further reinforce intuitive grasp over prevalent categorization mixes encapsulating sentient variations within collated opinion corpus, additional graphical asset dashboards extend actionable comprehension using proportional area conventions more aligned with aggregated overviews.

Specifically, single-shot pie visualizations dynamically rendered programmatically plot total fractions occupied by samples belonging individually to positive, negative and neutral

sentiment types as per color-coded legends using percent of total convention to showcase relative occurrences.

ii. Visualising text length distribution throughout the dataset

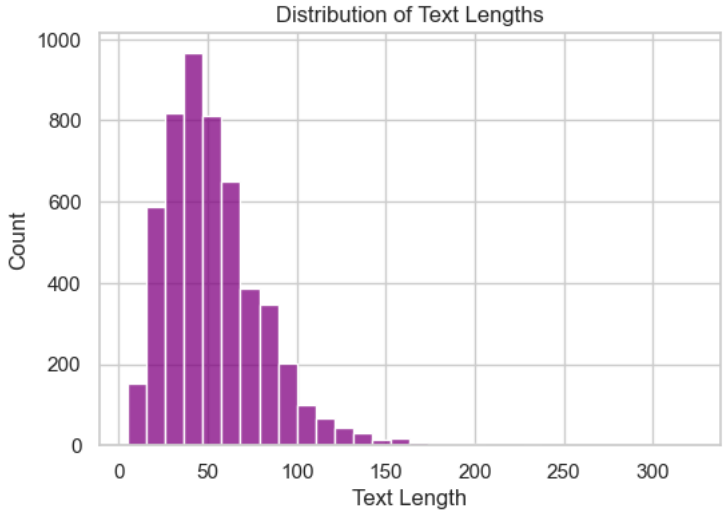


Figure 14. Visualising Text length distribution

The above histogram provides a concise overview of text length distribution in the dataset, facilitating quick identification of patterns and outliers. This visual insight is instrumental in guiding preprocessing decisions and understanding the predominant nature of text lengths within the data.

c. Translating the Dataset

The core translation phase begins, converting Hindi reviews into English for cross-lingual analysis. Cloud services streamline the process, connecting to robust neural translation engines effortlessly.

In the first system, HuggingFace Transformers integration employs the MarianMT architecture, finely tuned for Hindi-English mappings. This minimizes distortions in converting dataset sentences into English.

The second module relies on the Google Cloud Translation service, invoking enterprise-grade engines via authenticated requests. Together, these pipelines generate distinct English dataset versions, revealing variations during transitions in subsequent sentiment analysis..

i. Using Transformers Library

The HuggingFace Transformers library offers structured access to optimized neural translation pipelines, supporting global language diversity through simple API invocations from custom codebases. Specifically, the MarianMT model architecture is leveraged for efficient Hindi-to-English conversions. This deep encoder-decoder framework, trained on a substantial parallel corpus, retains contextual cues during translations through self-attentive pooling, preserving meaning integrity.

Programmatic invocation begins with initializing the pre-trained MarianMT model for the desired Hindi-English language pair, followed by loading paired tokenizer and translation pipelines. Nested translation functions facilitate systematic dialect conversion, adjusting input

text truncations and padding as needed, while final decoders return human-readable translated output.

Asynchronous execution using thread pools enables fast, parallel batched translations, reducing transitional latencies by exploiting underutilized system resources. Automatic persistence logs accumulate translated text with original metadata, easing systematic identifications.

Simplified interfaces facilitate the rapid assembly of configurable, scale-optimized translation systems, leveraging state-of-the-art self-supervised neural models. This approach minimizes overhead through intuitiveness while maximizing throughput for dialectal transitions—an essential feature for unlocking multilingual use cases otherwise constrained by legacy solutions.

```
nest_asyncio.apply()

#Initialising pre-trained model
model_name = "Helsinki-NLP/opus-mt-hi-en"
tokenizer = MarianTokenizer.from_pretrained(model_name)
model = MarianMTModel.from_pretrained(model_name)


# Defining translation function
def translate_text(text):
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=128)
    outputs = model.generate(**inputs)
    translated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return translated_text

# Defining translation coroutine
async def translate_and_save_batch(batch, output_file):
    original_texts = batch["text"].tolist()
    translated_texts = list(tqdm(ThreadPoolExecutor().map(translate_text, original_texts), total=len(original_texts)))

    # Saving translated sentences to CSV file
    with open(output_file, 'a', newline='', encoding='utf-8') as csvfile:
        csv_writer = csv.writer(csvfile)
        for original_text, translated_text, experience in zip(original_texts, translated_texts, batch["experience"].tolist()):
            csv_writer.writerow([original_text, translated_text, experience])

# Translating asynchronously and save to CSV file using parallel processing
async def main():
    output_file = r'C:\Users\shouv\Downloads\iitp-product-reviews\hi\transformers-data.csv'
    header = ["text", "Translated", "experience"]
    with open(output_file, 'w', newline='', encoding='utf-8') as csvfile:
        csv_writer = csv.writer(csvfile)
        csv_writer.writerow(header)

    batch_size = 8  #setting up batch size
    for i in range(0, len(df), batch_size):
        batch_rows = df.iloc[i:i + batch_size]
        await translate_and_save_batch(batch_rows, output_file)

# Run the event loop
await main()
```

Figure 15. Translating hindi dataset using transformers library

```
import pandas as pd
df = pd.read_csv(r'C:\Users\shouv\Downloads\iitp-product-reviews\hi\transformers-data.csv')
df.head()
```

	text	Translated	experience
0	एस्पेक्ट रेशो ईश्यू हम आशा आने अपडेट फिक्स जाए...	The aspect ratio will be determined by the God...	neutral
1	मॉडल कंप्यूटर शानदार विकल्प ।	The model computer is an excellent choice.	positive
2	गिर वन राष्ट्रीय उद्यान बाघ संरक्षित क्षेत्र ए...	The Asian Bebbarr tiger is considered world fa...	positive
3	हां , फिल्म हर किरदार भारद्वाज भरद्वाज क्यों ब...	Yes, why does a movie call every character loa...	neutral
4	फोन उपयोग बेहद आसान टच रिस्पॉन्स उतना बेहतर ।	Using the phone is much better for the ease of...	positive

Figure 16. Translated data output

The above figure shows the output of the translated dataset using transformers library.

ii. Using Google Translation Library

Google Cloud provides a fully managed neural machine translation service that can be accessed via authenticated API calls for programmatically translating text between thousands of global language pairs. The integration scripts initialize the Translation Client by authenticating access to Google's API. Structured JSON packets containing Hindi sentences and tags then undergo synchronous encapsulations.
Asynchronous routines powered by higher throughput coroutine workflows facilitate simultaneous crisp submissions of request packets in bulk parallel modes that vastly improve turn-around by smartly utilizing unused resources dynamically.

Purpose-built dashboards subsequently accumulate back translated English samples by efficiently mapping relations together through unique identifiers persisting consolidated CSV records.

Overall, simplified programming mechanisms hiding intricate complexities behind integrations along with automated scalable workflows assist in swiftly assembling deploy-ready configurable translation systems leveraging Google’s state-of-the-art multilingual engines with minimized transitional latencies.

```
# Applying nest_asyncio to handle asyncio in Jupyter environments
nest_asyncio.apply()

# Asynchronous function to translate and write rows
async def translate_and_write(row, writer):
    # Create a Translator object
    translator = Translator()

    # Extracting original text and sentiment from the DataFrame row
    original_text = row['text']
    sentiment = row['experience']

    # Translating the original text from Hindi to English
    translated_text = translator.translate(original_text, src='hi', dest='en').text

    # Writing the original text, translated text, and sentiment to the CSV file
    writer.writerow([original_text, translated_text, sentiment])

# Asynchronous main function
async def main():
    # Read the CSV file into a DataFrame
    df = pd.read_csv(r'C:\Users\shouv\Downloads\iitp-product-reviews\hi\newdat.csv')

    # Open a new CSV file for writing translated data
    with open(r'C:\Users\shouv\Downloads\iitp-product-reviews\hi\google-data.csv', 'w', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(['original_text', 'translated_text', 'experience'])

        # Initialize tqdm progress bar
        progress = tqdm(total=len(df))

        # List to store asynchronous tasks
        tasks = []

        # Iterate through DataFrame rows
        for index, row in df.iterrows():
            # Append the translation task to the tasks list
            tasks.append(translate_and_write(row, writer))

            # Update progress bar
            progress.update(1)

        # Execute all asynchronous tasks concurrently
        await asyncio.gather(*tasks)

        # Closing the progress bar
        progress.close()

# Run the event loop with asyncio.run
if __name__ == '__main__':
    asyncio.run(main())
```

Figure 17. Translating hindi dataset using google translation library

```
import pandas as pd
df = pd.read_csv(r'C:\Users\shouv\Downloads\iitp-product-reviews\hi\google-data.csv')
df.head()
```

	original_text	translated_text	experience
0	एस्पेक्ट रेशो ईश्यू हम आशा आने अपडेट फिक्स जाए...	Aspect Resho Ishu Hum Asha will be updated upd...	neutral
1	मॉडल कंप्यूटर शानदार विकल्प ।	Model Computer Fantastic Options.	positive
2	गिर वन राष्ट्रीय उद्यान बाघ संरक्षित क्षेत्र ए...	Gir Forest National Park Tiger Protected Regio...	positive
3	हां , फिल्म हर किरदार भारद्वाज भरद्वाज क्यों ब...	Yes, why does the film call every character Bh...	neutral
4	फोन उपयोग बेहद आसान टच रिस्पॉरन्स उतना बेहतर ।	Phone use is very easy touch response as much ...	positive

Figure 18. Translated data output.

The above figure shows the output of the translated dataset using google translation library.

3.7.2 Working with English Dataset

For a comprehensive comparative analysis, translated Hindi reviews are evaluated against standardized benchmarks. An additional English review corpus, comprising around 5000 online product feedback entries, is compiled to match the scale and affective diversity of the dataset. This accumulation ensures equivalent properties, minimizing biases in evaluations. It includes natively crafted English samples, aiding precise deductions into sentiment conveyance capabilities and limitations across contemporary translation techniques.

Before experimental use, the corpus undergoes essential preprocessing, including quality assurance checks and test partition stratification, retaining only fully qualified subsets.

Together with translated Hindi versions, this minimally processed natively English dataset quantitatively reveals performance deficits over matched classifiers, highlighting gaps introduced by transition mechanisms alone.

a. English Dataset preprocessing

In this step below English dataset will be pre-processed.

	tweet_id	label	tweet_text
0	1.360342e+18	1	4,000 a day dying from the so called Covid-19 ...
1	1.382896e+18	2	Pranam message for today manifested in Dhyan b...
2	1.375673e+18	2	Hyderabad-based ?@BharatBiotech? has sought fu...
3	1.381311e+18	1	Confirmation that Chinese #vaccines "don□t hav...
4	1.362166e+18	3	Lab studies suggest #Pfizer, #Moderna vaccines...
...	...	...	...
5995	1.370975e+18	2	@Swamy39 Dr. @Swamy39 jee :\n\nMany people lik...
5996	1.379827e+18	3	So happy to be fully vaccinated against COVID-...
5997	1.384789e+18	2	Serum Institute of India announces cost of Cov...
5998	1.382355e+18	1	@__batshitcrazy @BarrowfordHead @Bectully I h...
5999	1.380051e+18	2	The smart sympathy attends into the oblong not...

6000 rows × 3 columns

Figure 19. Preview of English Dataset

i. Feature Engineering

After careful dataset examination, it was found that the 'tweet_id' column is irrelevant for our project. Strategic changes were made for better compatibility with the Hindi dataset, including the removal of the unnecessary 'tweet_id' column. Column names were harmonized for smoother integration - 'label' is now 'experience,' and 'tweet_text' is now 'text.' Numerical labels were converted into negative, neutral, and positive for better comparability as per the dataset description. These improvements align the dataset with our project goals, contributing to a more unified and intuitive data display..

```
#Feature Engineering the dataset
df.drop(columns=['tweet_id'], inplace=True)
df.rename(columns={'label': 'experience'}, inplace=True)
df.rename(columns={'tweet_text': 'text'}, inplace=True)
df['experience'] = df['experience'].replace({1: 'negative', 2: 'neutral', 3: 'positive'})
df
```

	experience	text
0	negative	4,000 a day dying from the so called Covid-19 ...
1	neutral	Pranam message for today manifested in Dhyan b...
2	neutral	Hyderabad-based ?@BharatBiotech? has sought fu...
3	negative	Confirmation that Chinese #vaccines "don□t hav...
4	positive	Lab studies suggest #Pfizer, #Moderna vaccines...
...	...	...
5995	neutral	@Swamy39 Dr. @Swamy39 jee :\n\nMany people lik...
5996	positive	So happy to be fully vaccinated against COVID-...
5997	neutral	Serum Institute of India announces cost of Cov...
5998	negative	@__batshitcrazy @BarrowfordHead @Bectully I h...
5999	neutral	The smart sympathy attends into the oblong not...

6000 rows × 2 columns

Figure 20. Feature Engineering of the dataset.

ii. Performing overall data-cleaning.

```
# Downloading NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

# Defining preprocessing functions

def tokenize(text):
    # Tokenizing the input text into words.
    if isinstance(text, str):
        return word_tokenize(text)
    else:
        # Converting non-string or non-bytes-like object to string
        return word_tokenize(str(text))

def remove_stop_words(tokens):
    # Removing stop words from the list of tokens.
    stop_words = set(stopwords.words('english'))
    return [word for word in tokens if word not in stop_words]

def remove_non_alphabetic(tokens):
    # Removing non-alphabetic tokens from the list.
    return [word for word in tokens if word.isalpha()]

def remove_emoticons(tokens):
    # Removing emoticons from the list of tokens.
    emoticons_pattern = r"(?:[<>:](?:[:=;]-?)?[oO\)\(\wD])|[xo\(\)]"
    return [token for token in tokens if not re.match(emoticons_pattern, token)]

def stem_words(tokens):
    # Applying stemming to the list of tokens.
    stemmer = PorterStemmer()
    return [stemmer.stem(word) for word in tokens]

def convert_to_lowercase(tokens):
    # Converting each token to lowercase.
    return [token.lower() for token in tokens]

def preprocess_text(text):
    # Applying a series of text preprocessing steps to the input text.
    tokens = tokenize(text)
    tokens = remove_stop_words(tokens)
    tokens = remove_emoticons(tokens)
    tokens = remove_non_alphabetic(tokens)
    tokens = stem_words(tokens)
    tokens = convert_to_lowercase(tokens)
    return tokens

# Applying preprocessing to the 'text' column
df['text'] = df['text'].apply(preprocess_text)
df
```

	experience	text
0	negative	[day, die, call, dailybeast, report, vaccin, p...
1	neutral	[pranam, messag, today, manifest, dhyan, meena...
2	neutral	[bharatbiotech, sought, fund, govern, ramp, pr...
3	negative	[confirm, chines, vaccin, high, protect, rate,...
4	positive	[lab, studi, suggest, pfizer, moderna, vaccin,...
...	...	...
5995	neutral	[jee, mani, peopl, like, get, covaxin, vaccin,...
5996	positive	[So, happi, fulli, vaccin, readi, serv, vaccin...
5997	neutral	[serum, institut, india, announc, cost, covish...
5998	negative	[barrowfordhead, bectulli, I, mine, yesterday,...
5999	neutral	[the, smart, sympathi, attend, noth, what, edu...

6000 rows × 2 columns

Figure 21. Performing overall data cleaning and displaying the results

For optimal preparation of raw English text, we implemented customized cleaning methods within specific functions, aligning with the preprocessing steps applied to the Hindi dataset:

1. **NLTK Framework:** Our journey begins with the Natural Language Toolkit (NLTK), leveraging punkt and stopwords libraries for comprehensive text transformation.
2. **Tokenization:** The tokenize function dissects input text into individual words, enabling granular analysis.
3. **Removing Stop Words:** The remove_stop_words function systematically eliminates common English stop words, enhancing focus on content-rich vocabulary.
4. **Removing Non-Alphabetic Tokens:** The remove_non_alphabetic step filters out non-alphabetic tokens, refining the dataset for deeper analysis.
5. **Removing Emoticons:** The remove_emoticons function filters out emoticons, ensuring a cleaner and contextually relevant dataset.
6. **Stemming Words:** The stem_words function reduces words to their essential root form, aiding pattern recognition and generalization**.**
7. **Converting to Lowercase:** The convert_to_lowercase function transforms text into lowercase, ensuring uniformity for more effective analyses.
8. **Applying Preprocessing:** The preprocess_text function seamlessly integrates and applies each step sequentially, preparing text data for advanced analyses and model training. This standardized approach establishes a resilient and coherent dataset, facilitating insightful analyses.

Note: *These pre-processing steps are seamlessly applied to the translated Hindi dataset. Remarkably, post-translation, the dataset seamlessly adopts the same linguistic characteristics and structure as our original English dataset. This transformative alignment ensures a harmonious and unified nature across both datasets, setting the stage for coherent and meaningful analyses.*

iii. joining the tokens.

```
df['text'] = df['text'].apply(lambda tokens:' '.join(tokens))
df
```

	experience	text
0	negative	day die call dailybeast report vaccin pfizerva...
1	neutral	pranam messag today manifest dhyan meenapranam...
2	neutral	bharatbiotech sought fund govern ramp product ...
3	negative	confirm chines vaccin high protect rate accord...
4	positive	lab studi suggest pfizer moderna vaccin protec...
...	...	...
5995	neutral	jee mani peopl like get covaxin vaccin done av...
5996	positive	So happi fulli vaccin readi serv vaccineswork ...
5997	neutral	serum institut india announc cost covishield v...
5998	negative	barrowfordhead bectulli I mine yesterday abl g...
5999	neutral	the smart sympathi attend noth what educ feedb...

6000 rows × 2 columns

Figure 22. Performing overall data cleaning and displaying the results

After rigorously preparing the dataset, the tokens must be smoothly aggregated into their natural, coherent form. This joining procedure guarantees that the data is ready for further analysis, offering a streamlined and unified representation for later stages in our workflow.

iii. label Encoding the data.

```
#Label Encoding the data
from sklearn.preprocessing import LabelEncoder
le_model = LabelEncoder()
df['experience'] = le_model.fit_transform(df['experience'])
df
```

	experience	text
0	0	[day, die, call, dailybeast, report, vaccin, p...
1	1	[pranam, messag, today, manifest, dhyan, meena...
2	1	[bharatbiotech, sought, fund, govern, ramp, pr...
3	0	[confirm, chines, vaccin, high, protect, rate,...
4	2	[lab, studi, suggest, pfizer, moderna, vaccin,...
...	...	...
5995	1	[jee, mani, peopl, like, get, covaxin, vaccin,...
5996	2	[So, happi, fulli, vaccin, readi, serv, vaccin...
5997	1	[serum, institut, india, announc, cost, covish...
5998	0	[barrowfordhead, bectulli, I, mine, yesterday,...
5999	1	[the, smart, sympathi, attend, noth, what, edu...

6000 rows × 2 columns

Figure 23. Performing overall data cleaning and displaying the results

After applying the thorough pre-processing processes to our dataset, label encoding introduces a critical enhancement. This transformative process not only improves the data for improved model

performance, but it also helps with memory optimization. Label encoding assigns numerical representations to sentiment labels ('negative,' 'neutral,' 'positive,') to make it easier to integrate our data into machine learning models. This deliberate encoding not only ensures numerical compatibility but also promotes the efficient use of memory resources, which is critical to the overall efficiency and efficacy of our models.

***Note** - A shared label encoding strategy is used to recognize identical sentiment classes in both the Hindi and English datasets. This not only assures consistent numerical representation ('negative,' 'neutral,' 'positive,') but also harmonizes the encoding procedure across both datasets for simplified model performance and memory efficiency.*

b. Visualisation of English Dataset

Now we'll visualize the data to get more insight into the dataset.

i. Visualising Sentiment Distribution through the English dataset

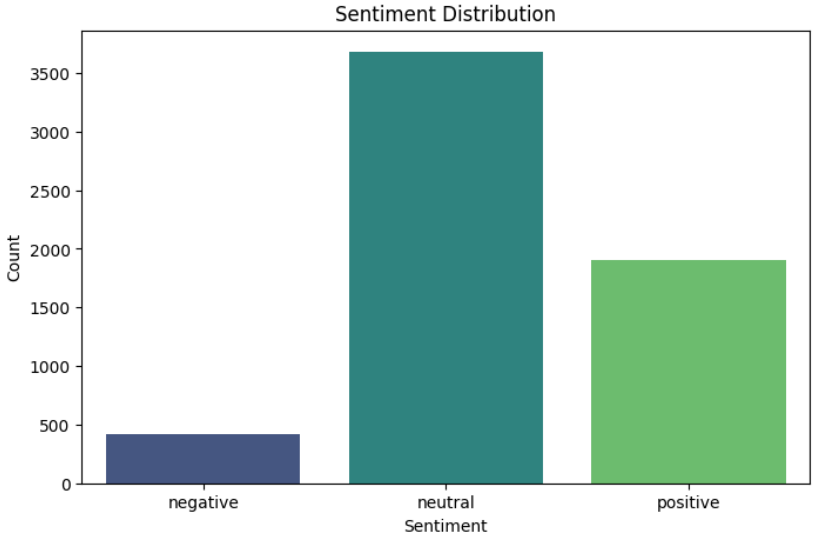


Figure 24. Performing overall data cleaning and displaying the results

ii. Visualising word cloud for most frequent words in reviews

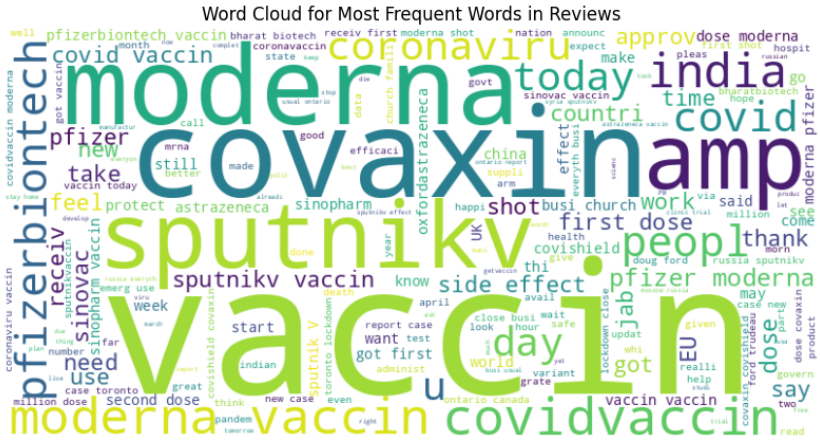


Figure 25. Performing overall data cleaning and displaying the results

c. Data Augmentation

To ensure fair comparisons between translated Hindi and English reviews, aligning the English corpus with its Indic counterpart is crucial. Addressing a discrepancy in negative class representation, we used transformers pre-trained model **(EleutherAI/gpt-neo-1.3B)** for data augmentation**.** increasing the dataset from 6000 to 6840 samples. This doubled negative instances, rectifying the imbalance and enhancing overall analysis robustness. This refined dataset aims to yield nuanced insights, improving sentiment analysis across both linguistic domains.

	experience	text
0	1	Oooh this second #moderna dose is kicking my b...
1	2	@TheWilsonCenter "Why is Russia falling behind...
2	3	First Moderna shot, day two report: I feel fin...
3	1	J&J: 2 people had severe allergic reaction...
4	2	@MoHFW_INDIA @drharshvardhan @PMOIndia @Ashwin...
...	...	...
6835	2	@Kaalateetham Totally agree doc .\nVery few sp...
6836	2	Jk I already got my first #Moderna #vaccine a ...
6837	3	Hungary became the first EU country to receive...
6838	3	Got my first #Moderna vaccine shot from @about...
6839	2	@snehamordani @Interceptors @BharatBiotech So ...

6840 rows × 2 columns

Figure 26. Performing overall data cleaning and displaying the results

Sentiment Distribution graph after augmentation

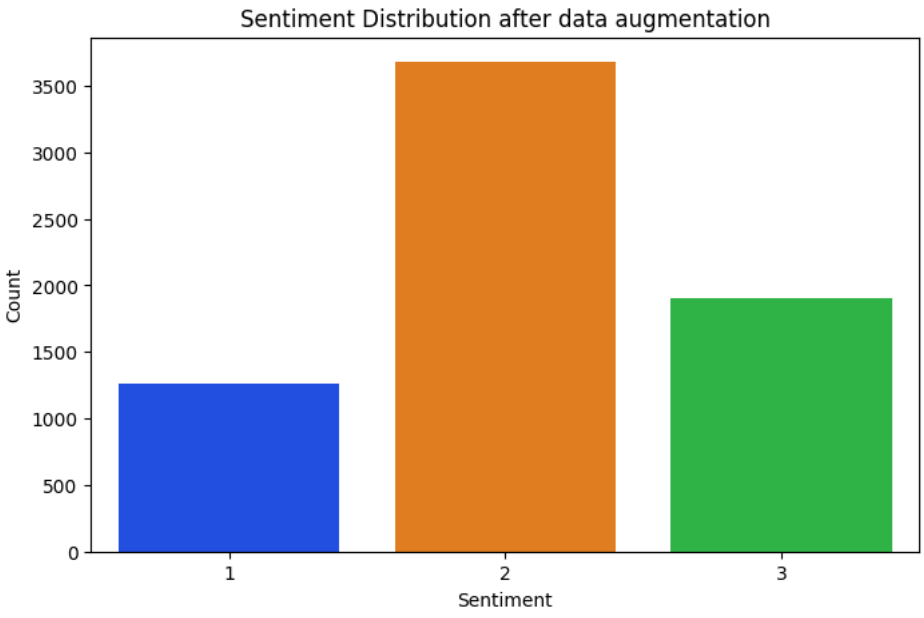


Figure 27. Distribution after augmentation

3.7.3 Balancing the Datasets

With the help of previous visualisations, we can clearly see that the classes i.e. positive, negative and neutral in English and Hindi datasets differs by a lot of margins, to make the results more comparable and conclusive we'll now balance the dataset then we'll move onto the model training and evaluation step.

Class distribution before

class	English	Google	Transformer
Positive	1900	2287	2285
Negative	1260	710	709
Neutral	3680	2221	2224

Table 4. class distribution table before balancing.

Class distribution after

class	English	Google	Transformer
Positive	1900	1900	1900
Negative	709	709	709
Neutral	2224	2224	2224

Table 5. class distribution table after balancing

3.8 Building the model.

3.8.1 initialising the framework to display the results

```
def print_model_evaluation(model_name, y_train, y_train_pred, y_test, y_test_pred):
    print(f"\n{' MODEL EVALUATION ':-^54}")
    print(f"{'Model Name:':<20} {model_name}")
    print(f"{'=' * 54}")

    # Handling potential one-hot encoding and convert to categorical labels
    y_train_true = np.argmax(y_train, axis=1) if len(y_train.shape) > 1 else y_train
    y_test_true = np.argmax(y_test, axis=1) if len(y_test.shape) > 1 else y_test

    # Calculating and printing accuracies with clear formatting
    print(f"\n{'Training Accuracy:':<20} {accuracy_score(y_train_true, y_train_pred):.4f}")
    print(f"{'Test Accuracy:':<20} {accuracy_score(y_test_true, y_test_pred):.4f}")

    print(f"\n{'-' * 54}")

    # Print the classification report with a clear title
    print(f"\n{'Classification Report:':<20}")
    print(classification_report(y_test_true, y_test_pred))
```

Figure 28 building model to print the results

3.8.2 Splitting and Normalizing the dataset

```
# Splitting the data into training and testing sets
X = df['text']
y = df['experience']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
# Initializing the TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer()

# Fit and transform the training data
X_train = tfidf_vectorizer.fit_transform(X_train.astype('U'))  # Converting to Unicode strings

# Transform the test data using the same vectorizer
X_test = tfidf_vectorizer.transform(X_test.astype('U'))  # Converting to Unicode strings
```

Figure 29: Splitting and Normalizing the data

In the above Using the train_test_split function from Scikit-Learn, the data is split into 80% training and 20% testing sets.

TF-IDF (Term Frequency-Inverse Document Frequency) is a technique to convert text data into numerical vectors. TfidfVectorizer from Scikit-Learn is used to initialize a TF-IDF vectorizer.

The **fit_transform** method is applied on the training data (X_train), which both fits the vectorizer to the data and transforms the text into numerical vectors. The same vectorizer is then used to transform the test data (X_test), ensuring consistency in the transformation process between training and testing data.

3.8.3 Model Selection and Training

Diverse classifiers—Naive Bayes, SVM, Random Forest, Logistic Regression, and LSTM—are trained on English and translated Hindi datasets. This multi-algorithm approach enables rigorous comparative performance assessment, ensuring fair accuracy evaluations by configuring identical models on both datasets.

a. Naïve Bayes

Based on distinguishing characteristics, the probabilistic classifier Naive Bayes predicts a data point's likely class using Bayes' theorem **(Rennie et al., 2003).** These features in sentiment analysis include text-rooted properties, sentiment lexicon-derived features, and word n-grams. Assuming conditional independence, the classifier suggests that these features are independent when the sentiment class is **known (Go, Bhayani, and Huang, 2009).**

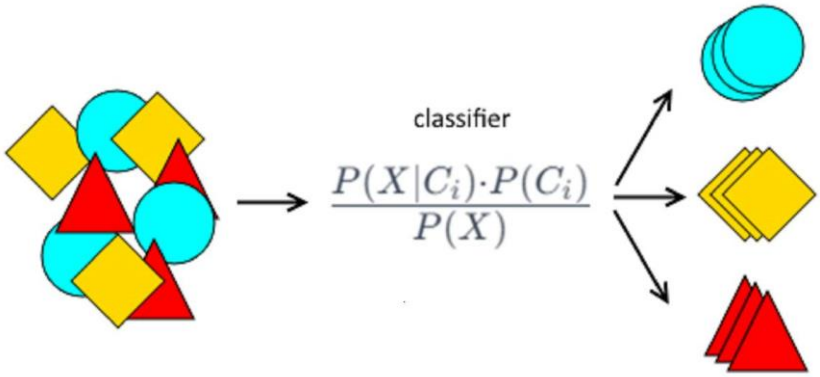


Figure 30 Naïve Bayes Classification ,(Gusain, 2023)

Overall Algorithm used-

Given a feature set **X = {x1, x2, ..., xn}** and a target variable **C**, Naive Bayes calculates the probability of each class C_i given the features:

$$P(C_i|X) = \frac{P(X|C_i) \cdot P(C_i)}{P(X)}$$

The "naive" assumption is that features are conditionally independent given the class, simplifying the calculations:

$$P(X|C_i) = P(x1|C_i) \cdot P(x2|C_i) \cdot ... \cdot P(xn|C_i)$$

Model Training and Evaluation:

- **Data Splitting:** The dataset is split into training (X_train, y_train) and testing (X_test, y_test) sets.
- **Naive Bayes Model Training:** A Multinomial Naive Bayes model is initialized (MultinomialNB()). The model is trained on the training data.
- **Model Prediction:** The trained model predicts labels for both the training and testing sets.
- **Evaluation Metrics:** Accuracy, classification report, and confusion matrices are used for model evaluation.
- **Accuracy Plot:** A bar chart is plotted to visualize the accuracy of the model on training and testing sets.
- **Confusion Matrix Plotting:** Confusion matrices for training and testing sets are visualized using heatmaps.

b. Support Vector Machine

The Support Vector Machine (SVM) is a powerful guardian in the complex field of sentiment analysis. It uses a sharp geometric blade to draw distinct boundaries between positive and negative emotions in textual input **(Tyagi and Sharma, 2017).** In contrast to probabilistic models that depend on ambiguous hints, the SVM adopts a firm position by creating a hyperplane, or distinct boundary, inside the high-dimensional space of textual features. Because of this boundary, which guarantees precise categorization, the SVM is an effective tool for applications like customer review analysis and sentiment analysis on social media.

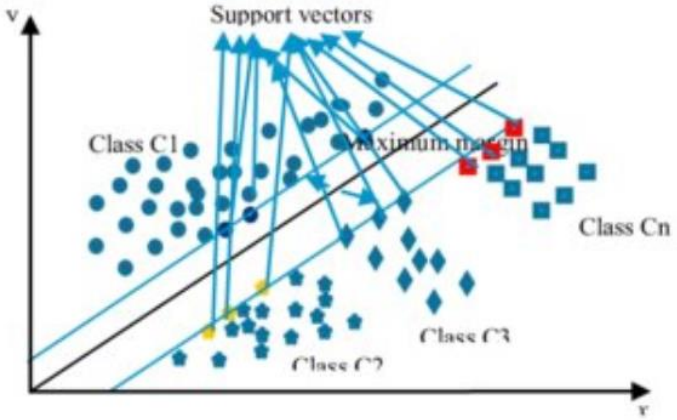


Figure 31 Multiclass SVM(Jyothi, Guru and Kumar, 2018)

Overall Algorithm Used-

Initialize Parameters-

- Split data into training **(X_train, y_train)** and testing **(X_test, y_test)** sets
- Set number of classes to K
- Initialize **K** SVM models with weights **wk** and biases **bk (for k=1 to K)**

Training-

❖ For each class k:

- Treat k as positive class, rest as negative.
- Optimize wk, bk to maximize margin between classes.

Decision Function

❖ For input x:

- Compute score for each class: **wk.x + bk**
- Predict class with highest score: **argmaxk (wk.x + bk)**

Kernel Trick

- Apply kernel trick to enable non-linear boundaries.

Model Training and Evaluation

- Train SVM models on training set
- Make predictions on training and testing sets.
- Evaluate using accuracy, classification report, confusion matrices.
- Visualize accuracy on training vs testing with bar chart.
- Visualize confusion matrices as heatmaps.

c. Random Forest

The Random Forest technique navigates various data subsets by using an ensemble of decision trees **(Zhao et al., 2023).** Each tree carves its own course through the feature-rich environment, culminating in leaves indicating sentiment classes. The Random Forest offers strong classification in intricate, non-linear settings by collecting insights, revealing an emotional map of essential elements. While deciphering individual tree decisions requires skill, the Random Forest's aggregate intelligence is excellent for uncovering hidden emotional nuances inside textual data.

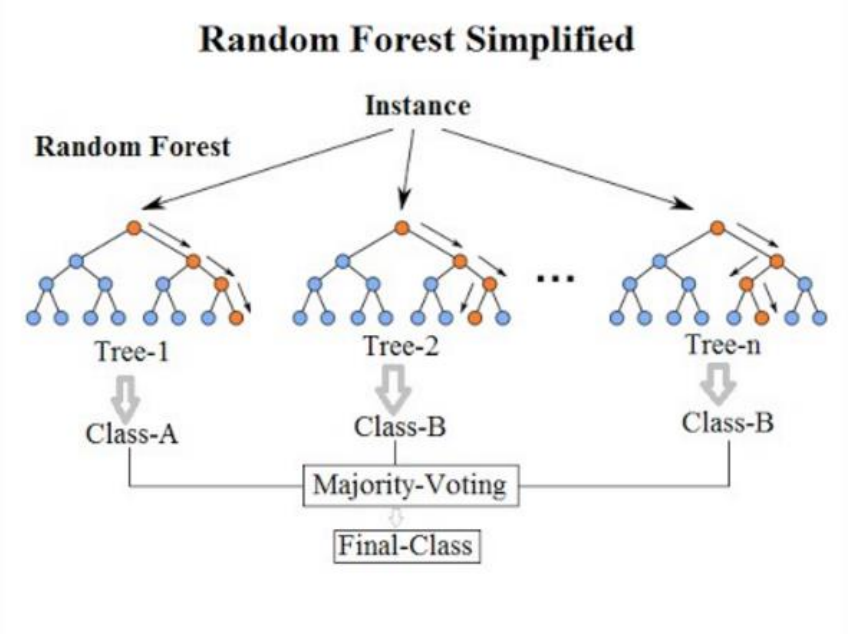


Figure 32: Random Forest classification model (Koehrsen, 2020)

Overall Algorithm Used-

Initialize Parameters

- Set number of trees
- Set maximum tree depth.
- Set number of features for split consideration

Bootstrap Sampling

- For each tree, randomly sample training data with replacement

Feature Randomization

- For each tree, randomly select subset of features at each node.

Grow Trees

- **For each bootstrapped sample:**
  - Select best feature splits.
  - Recursively split nodes to maximum depth
  - Ensemble Prediction

For classification:

- Each tree predicts a class.
- Ensemble predicted class is majority vote.

Model Training and Evaluation:

- Split data into training (X_train, y_train) and testing (X_test, y_test)
- Train Random Forest model on X_train, y_train
- Make predictions on X_train, X_test

- Evaluate using accuracy, confusion matrices.
- Visualize accuracy on training vs testing with bar chart.
- Visualize confusion matrices as heatmaps.

d. Logistic Regression

Logistic regression is a reliable technique for multiclass sentiment analysis, utilizing probabilities to navigate text's emotional terrain **(Pranckevičius & Marcinkevičius, 2023).** By assigning probabilities to each sentiment class, logistic regression captures nuanced human expression, unlike models making hard boundaries between emotions. It smoothly creates a sigmoid curve from negative to neutral to positive, reflecting nuanced feelings in text. This probabilistic method reveals the predominant sentiment and confidence levels, enabling subtle insights into sentiment strength. Despite linear limitations on highly complex correlations, logistic regression provides academics an interpretable lens to analyze emotional currents in text data.

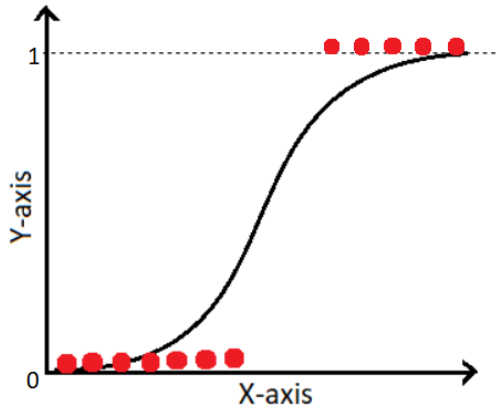


Figure 33: Logistic Regression graph (Rai, 2021)

Overall Algorithm Used-

Initialize Parameters

- Set weight vector w and bias b to 0

Hypothesis Function

- Calculate logit: z = w * x + b

Logistic Function

- Calculate sigmoid: σ(z) = 1 / (1 + e^(-z))

Cost Function

- Define logistic loss function to optimize.

Gradient Descent

- Update w, b to minimize cost function.

Model Training and Evaluation

- Split data into training (X_train, y_train) and testing (X_test, y_test)
- Train Logistic Regression model on X_train, y_train
- Make predictions on X_train, X_test
- Evaluate using accuracy, confusion matrices

- Visualize accuracy on training vs testing with bar chart
- Visualize confusion matrices as heatmaps.

e. Long Short-Term Memory (LSTM)

LSTM networks, especially Bidirectional LSTMs, emerge as powerful tools for navigating the intricate landscape of multiclass sentiment analysis in textual data. Unlike traditional models confined to single sentences, LSTMs boast a robust long-term memory, enabling them to comprehend evolving sentiment stories in lengthy texts **(Chung et al., 2014)**. Building on the strengths of standard LSTMs, bidirectional architecture takes analysis a step further. By processing information bidirectionally—both left to right and right to left—this model enhances its capacity for understanding sequential data, capturing richer meaning, connections, and nuanced details that might elude traditional approaches **(Graves & Schmidhuber, 2005).**

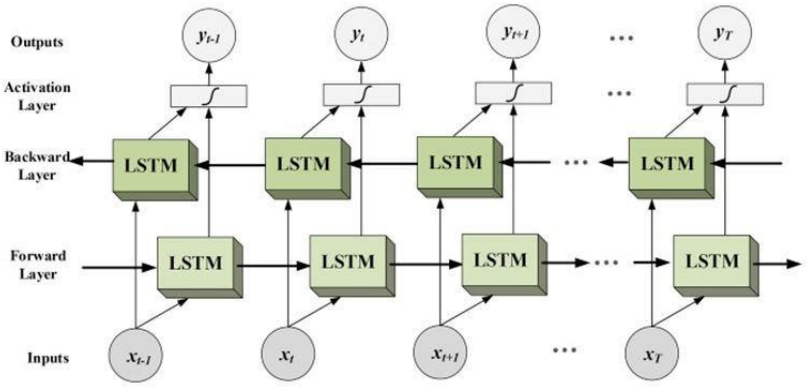


Figure 34 : Bi-directional LSTM Structure (Mungalpara, 2021)

Overall Algorithm used-

- **Data Splitting**: Divide the dataset into training (X_train, y_train) and testing (X_test, y_test) sets.
- **Text Tokenization and Padding:** Tokenize and pad the text data to ensure uniform sequence length using Keras Tokenizer and pad_sequences.
- **One-Hot Encoding:** One-hot encode labels (y_train and y_test) using to_categorical.
- **LSTM Model Architecture:** Define a Sequential model with:
  - Embedding layer for word embeddings.
  - Bidirectional LSTM layers to capture sequential patterns.
  - Batch Normalization and Dropout layers to prevent overfitting.
  - Dense layer with softmax activation for multi-class classification.
- **Model Compilation and Training**: Compile the model with categorical cross-entropy loss and Adam optimizer.
  - Use early stopping to prevent overfitting during training.
  - Train the model on the training data.
- **Accuracy and Loss Plotting:** Plot training and testing accuracy and loss over epochs for visualizing model performance.

- **Model Evaluation**: Use the trained model to predict labels on both training and testing sets.
  - Convert predicted probabilities to class labels.
  - Use the print_model_evaluation function for model evaluation**.**
- **Confusion Matrix:** Generate confusion matrices for both training and testing sets and plot heatmaps for visualization.

3.8.4 Model Evaluation and Visualization

The model evaluation involves assessing training and test accuracy, with a detailed classification report showcasing precision, recall, f1 score, support, and a confusion matrix. Training and test accuracy indicate model fitting and generalization. Precision reveals the accuracy of positive sentiment predictions, recall indicates correct predictions for actual positive sentiment, and f1 score balances precision and recall. These metrics are presented for each sentiment class (positive, negative, neutral). Support indicates class sample counts, and the confusion matrix identifies errors like false positives and false negatives. Together, these metrics offer a comprehensive quantitative overview of the sentiment classification model's performance, aiding deployment decisions and highlighting areas for improvement.

Evaluation metrics used:

Precision

- Measures the accuracy of positive predictions made by a classification model.
- Formula: Precision = True Positives / (True Positives + False Positives)

Recall

- Also known as Sensitivity or True Positive Rate.
- Measures the ability of a model to correctly identify all positive instances.
- Formula: Recall = True Positives / (True Positives + False Negatives)

F1 Score

- The harmonic mean of Precision and Recall.
- Provides a balanced measure of a model's performance.
- Particularly useful in scenarios with class imbalance.
- Formula: F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

Confusion Matrix

- A tabular representation of a model's predictions.
- Shows counts of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

Layout:

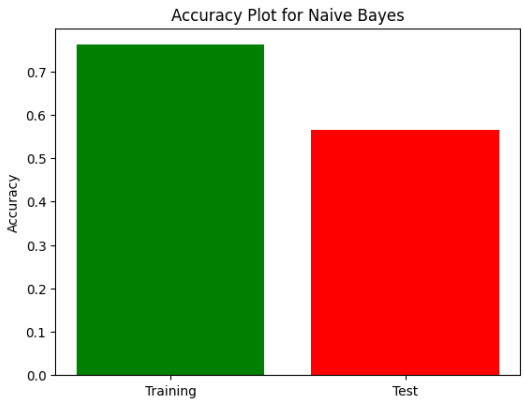
Predicted Negative	Predicted Positive
True Negative (TN)	False Positive (FP)
False Negative (FN)	True Positive (TP)

a. Translated Hindi Dataset (Transformers)

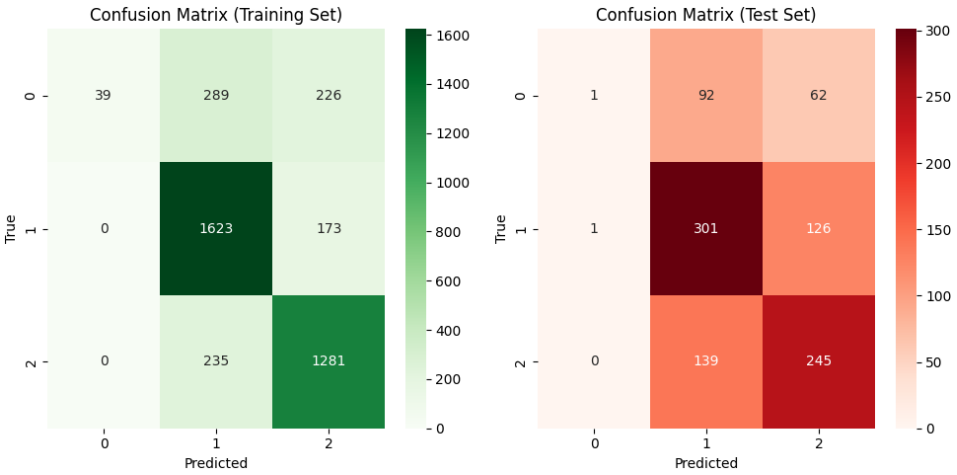
Here, we use the Transformers library to do a thorough assessment of our sentiment analysis models on the Hindi dataset that has been translated. We use confusion matrices and accuracy plots to give a visual image of the accuracy results and to give a thorough understanding of the performance. These visualizations are useful instruments to evaluate how well and consistently our models capture sentiment patterns in the translated dataset. Furthermore, an extensive analysis of the results will be provided in the next section.

i. Naïve Bayes

```
------------------ MODEL EVALUATION ------------------
Model Name:           Naive Bayes
========================================================

Training Accuracy:   0.7613
Test Accuracy:       0.5657

--------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.01      0.01       155
           1       0.57      0.70      0.63       428
           2       0.57      0.64      0.60       384

    accuracy                           0.57       967
   macro avg       0.54      0.45      0.41       967
weighted avg       0.56      0.57      0.52       967
```

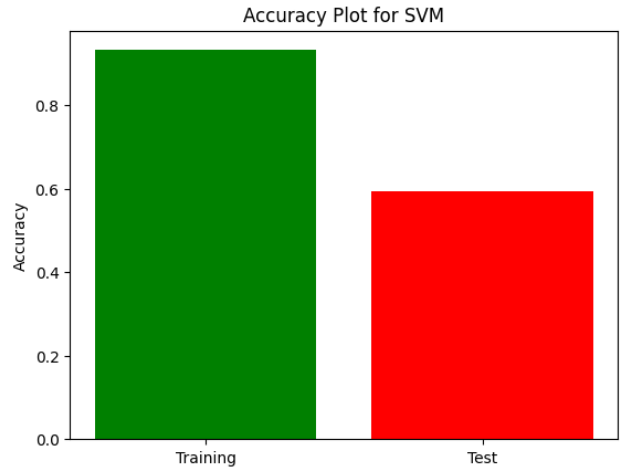



```
Confusion Matrix (Training Set):
 [[  39  289  226]
 [   0 1623  173]
 [   0  235 1281]]

Confusion Matrix (Test Set):
 [[  1  92  62]
 [  1 301 126]
 [  0 139 245]]
```

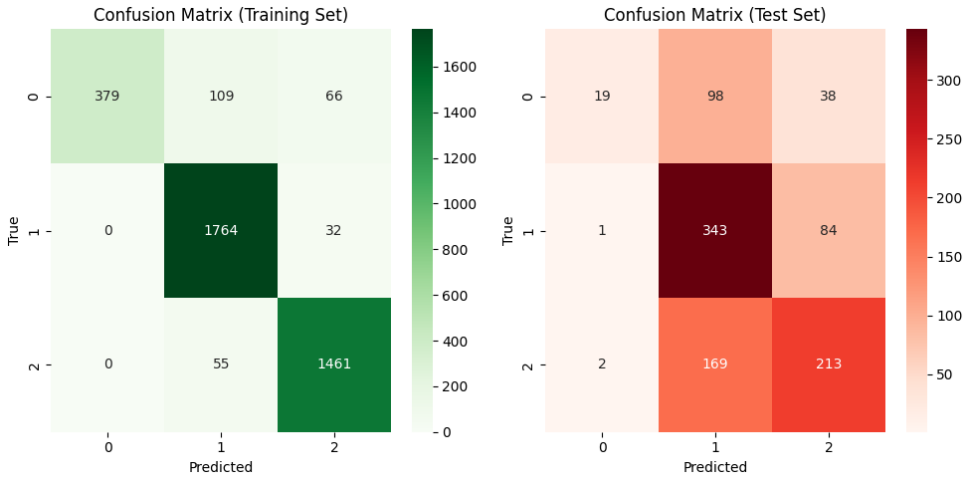


ii. SVM

```
------------------ MODEL EVALUATION ------------------
Model Name:           SVM
========================================================

Training Accuracy:   0.9322
Test Accuracy:       0.5946

--------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.12      0.21       155
           1       0.56      0.80      0.66       428
           2       0.64      0.55      0.59       384

    accuracy                           0.59       967
   macro avg       0.69      0.49      0.49       967
weighted avg       0.64      0.59      0.56       967
```

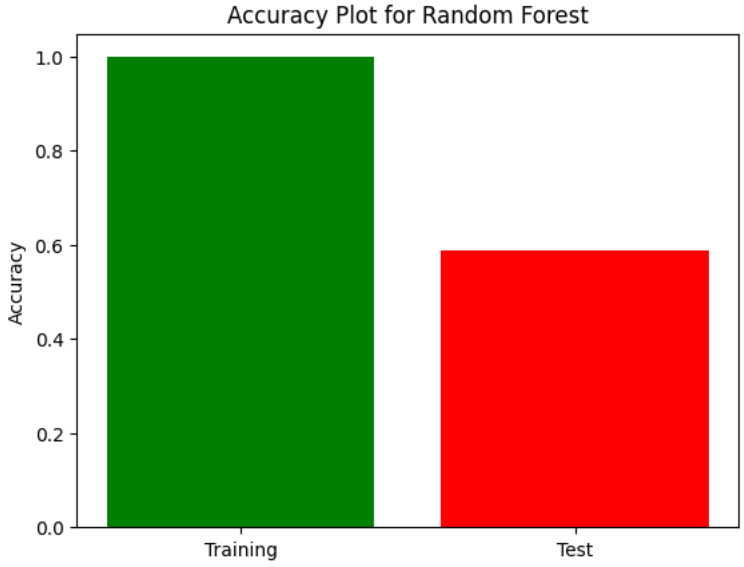



```
Confusion Matrix (Training Set):
 [[ 379  109   66]
 [   0 1764   32]
 [   0   55 1461]]

Confusion Matrix (Test Set):
 [[ 19  98  38]
 [  1 343  84]
 [  2 169 213]]
```

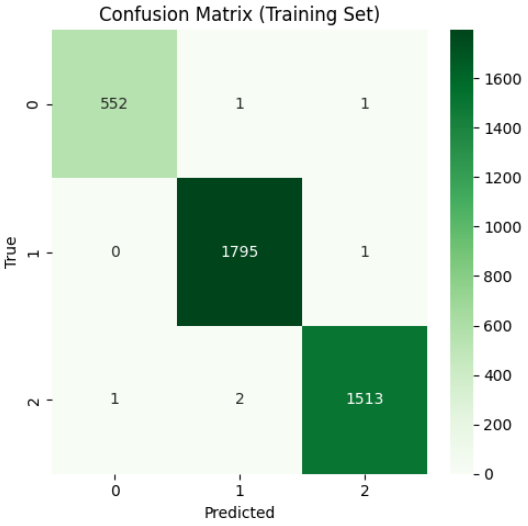


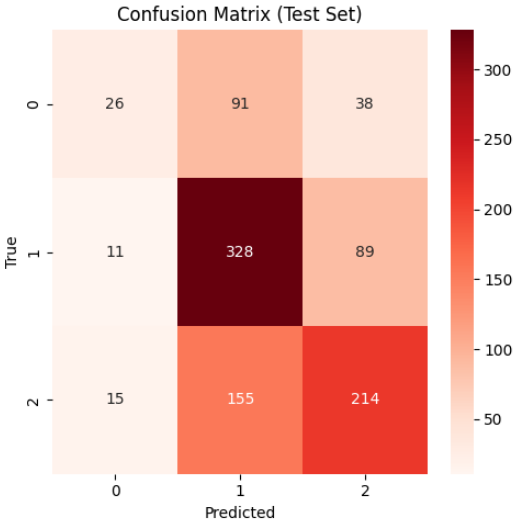
iii. Random Forest

```
------------------ MODEL EVALUATION ------------------
Model Name:           Random Forest
======================================================

Training Accuracy:    0.9984
Test Accuracy:        0.5874

------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.17      0.25       155
           1       0.57      0.77      0.65       428
           2       0.63      0.56      0.59       384

    accuracy                           0.59       967
   macro avg       0.57      0.50      0.50       967
weighted avg       0.58      0.59      0.56       967
```



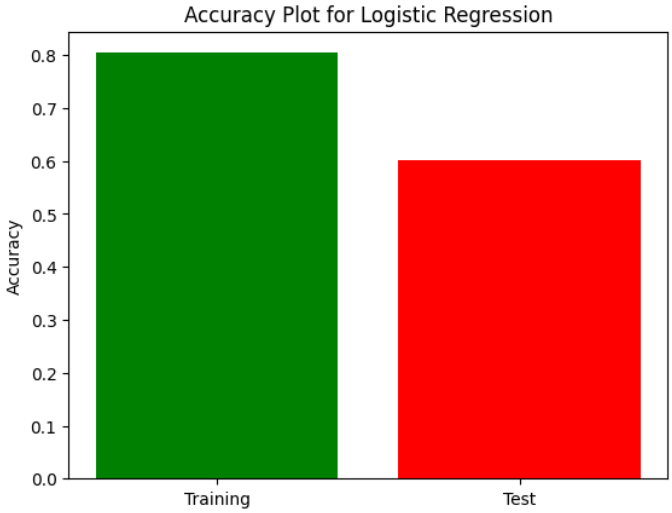

```
Confusion Matrix (Training Set):
 [[ 552    1    1]
 [   0 1795    1]
 [   1    2 1513]]

Confusion Matrix (Test Set):
 [[ 26  91  38]
 [ 11 328  89]
 [ 15 155 214]]
```



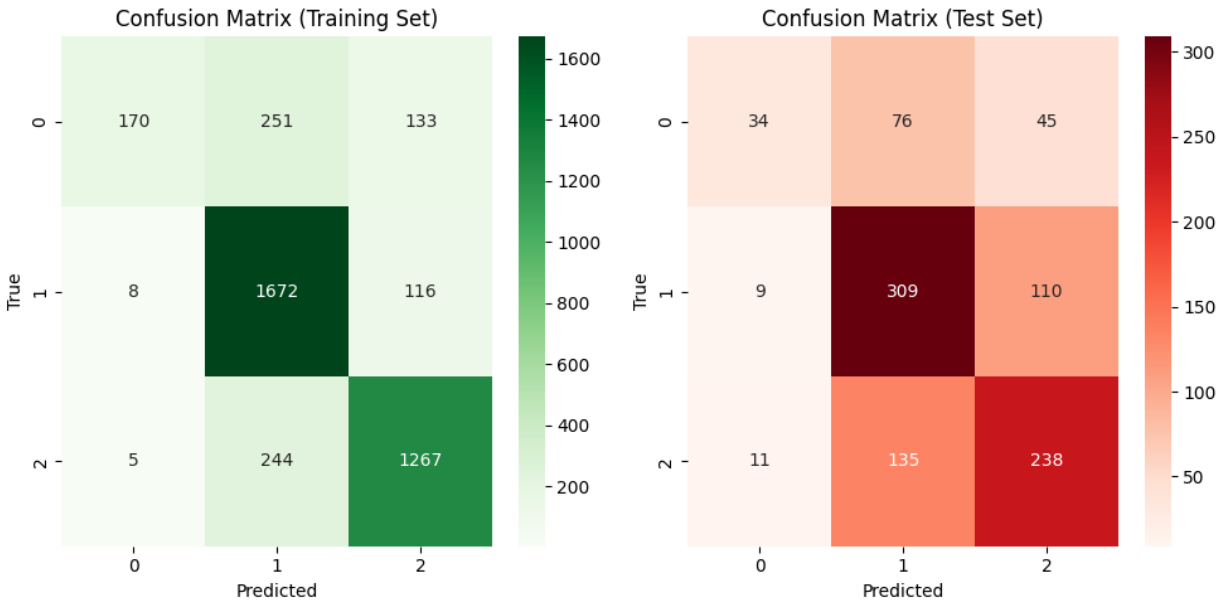


iv. Logistic Regression

```
------------------ MODEL EVALUATION ------------------
Model Name:          Logistic Regression
========================================================

Training Accuracy:   0.8042
Test Accuracy:       0.6008

--------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.22      0.33       155
           1       0.59      0.72      0.65       428
           2       0.61      0.62      0.61       384

    accuracy                           0.60       967
   macro avg       0.61      0.52      0.53       967
weighted avg       0.60      0.60      0.58       967
```

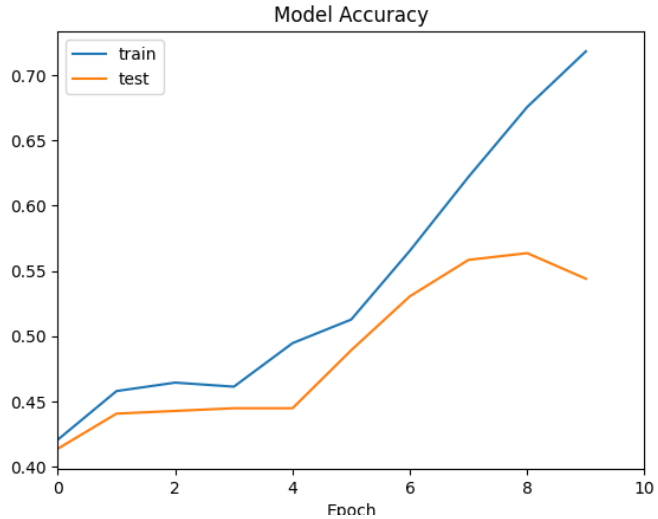


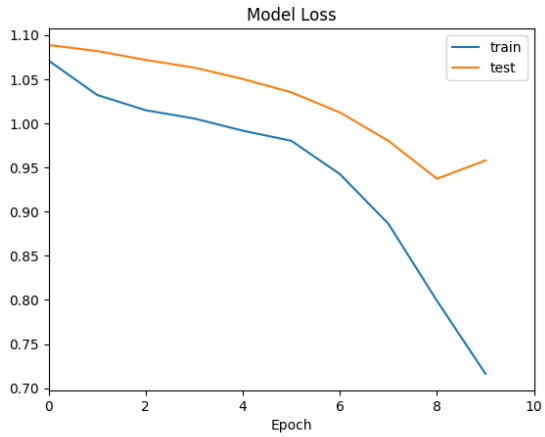

```
Confusion Matrix (Training Set):
 [[ 170  251  133]
 [   8 1672  116]
 [   5  244 1267]]

Confusion Matrix (Test Set):
 [[ 34  76  45]
 [  9 309 110]
 [ 11 135 238]]
```



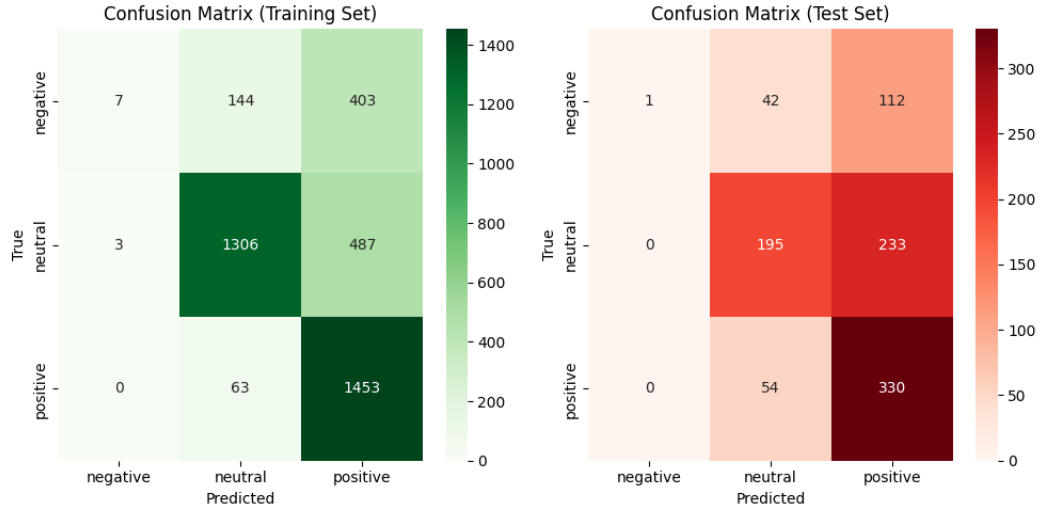
v. LSTM

```
Epoch 1/10
61/61 [==============================] - 17s 158ms/step - loss: 1.0710 - accuracy: 0.4206 - val_loss: 1.0887 - val_accuracy: 0.4137
Epoch 2/10
61/61 [==============================] - 5s 78ms/step - loss: 1.0321 - accuracy: 0.4578 - val_loss: 1.0819 - val_accuracy: 0.4405
Epoch 3/10
61/61 [==============================] - 3s 46ms/step - loss: 1.0148 - accuracy: 0.4643 - val_loss: 1.0718 - val_accuracy: 0.4426
Epoch 4/10
61/61 [==============================] - 2s 34ms/step - loss: 1.0055 - accuracy: 0.4612 - val_loss: 1.0631 - val_accuracy: 0.4447
Epoch 5/10
61/61 [==============================] - 2s 31ms/step - loss: 0.9917 - accuracy: 0.4946 - val_loss: 1.0502 - val_accuracy: 0.4447
Epoch 6/10
61/61 [==============================] - 3s 45ms/step - loss: 0.9802 - accuracy: 0.5127 - val_loss: 1.0350 - val_accuracy: 0.4891
Epoch 7/10
61/61 [==============================] - 2s 28ms/step - loss: 0.9425 - accuracy: 0.5654 - val_loss: 1.0123 - val_accuracy: 0.5305
Epoch 8/10
61/61 [==============================] - 2s 29ms/step - loss: 0.8862 - accuracy: 0.6221 - val_loss: 0.9801 - val_accuracy: 0.5584
Epoch 9/10
61/61 [==============================] - 2s 30ms/step - loss: 0.7990 - accuracy: 0.6756 - val_loss: 0.9372 - val_accuracy: 0.5636
Epoch 10/10
61/61 [==============================] - 1s 21ms/step - loss: 0.7163 - accuracy: 0.7183 - val_loss: 0.9579 - val_accuracy: 0.5440
```






```
121/121 [==============================] - 2s 6ms/step
31/31 [==============================] - 0s 5ms/step

------------------ MODEL EVALUATION ------------------
Model Name:          LSTM Model
========================================================

Training Accuracy:   0.7155
Test Accuracy:       0.5440

--------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.01      0.01       155
           1       0.67      0.46      0.54       428
           2       0.49      0.86      0.62       384

    accuracy                           0.54       967
   macro avg       0.72      0.44      0.39       967
weighted avg       0.65      0.54      0.49       967


 Confusion Matrix (Training Set):
  [[   7  144  403]
  [   3 1306  487]
  [   0   63 1453]]

 Confusion Matrix (Test Set):
  [[  1  42 112]
  [  0 195 233]
  [  0  54 330]]
```

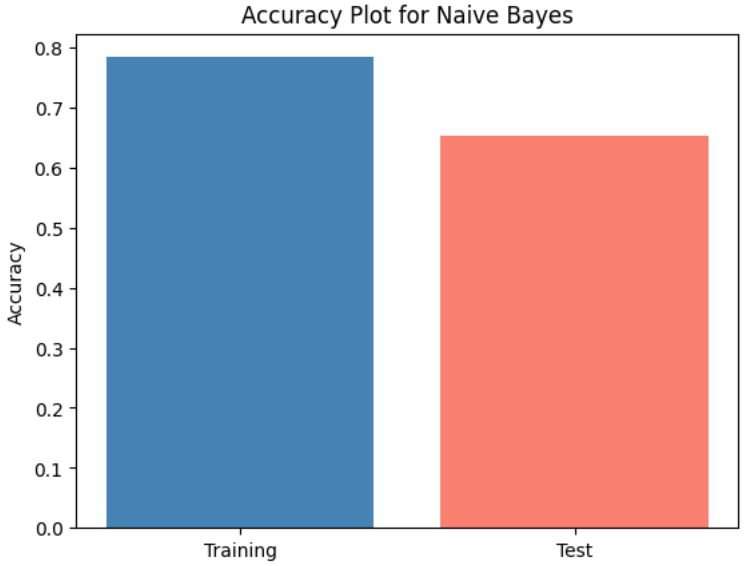


b. Translated Hindi Dataset (Google)

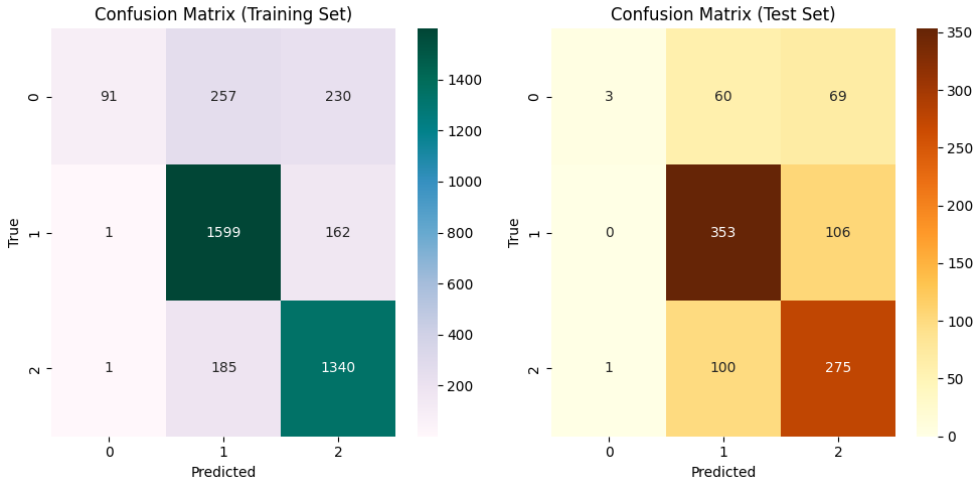
The translated Hindi dataset evaluates sentiment models using Google Translate API. Visual accuracy representations with confusion matrices and plots provide model performance insights. These assess consistency and efficacy in extracting sentiment from Google-translated data. A thorough results examination next section delivers in-depth understanding of model performance on Google-translated dataset. In summary, visualizations with confusion matrices and accuracy plots comprehensively evaluate model capability on translated data, informing further analysis.

i. Naïve Bayes

```
------------------- MODEL EVALUATION ------------------
Model Name:           Naive Bayes
=======================================================

Training Accuracy:   0.7838
Test Accuracy:       0.6525

-------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.02      0.04       132
           1       0.69      0.77      0.73       459
           2       0.61      0.73      0.67       376

    accuracy                           0.65       967
   macro avg       0.68      0.51      0.48       967
weighted avg       0.67      0.65      0.61       967
```

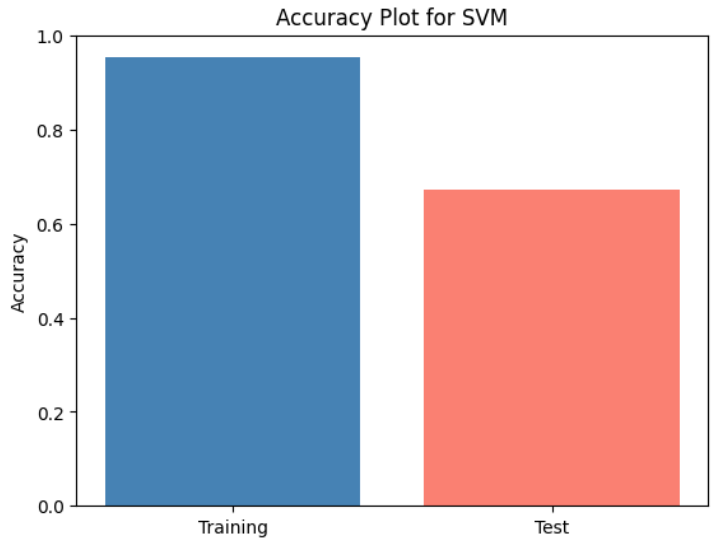



```
Confusion Matrix (Training Set):
 [[  91  257  230]
 [   1 1599  162]
 [   1  185 1340]]

Confusion Matrix (Test Set):
 [[  3  60  69]
 [  0 353 106]
 [  1 100 275]]
```

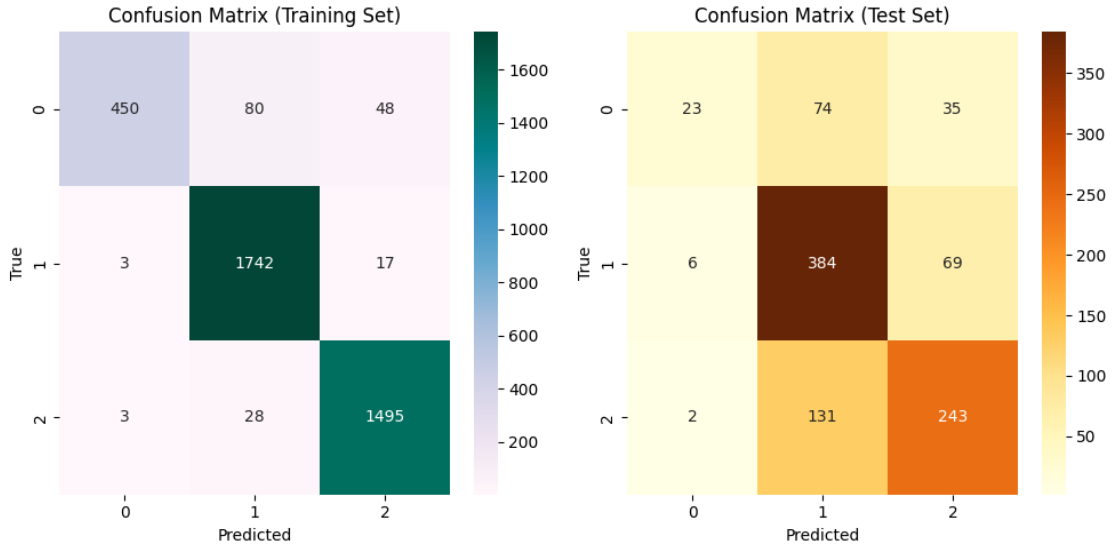


ii. SVM

```
------------------ MODEL EVALUATION ------------------
Model Name:           SVM
======================================================

Training Accuracy:    0.9537
Test Accuracy:        0.6722

------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.17      0.28       132
           1       0.65      0.84      0.73       459
           2       0.70      0.65      0.67       376

    accuracy                           0.67       967
   macro avg       0.70      0.55      0.56       967
weighted avg       0.68      0.67      0.65       967
```

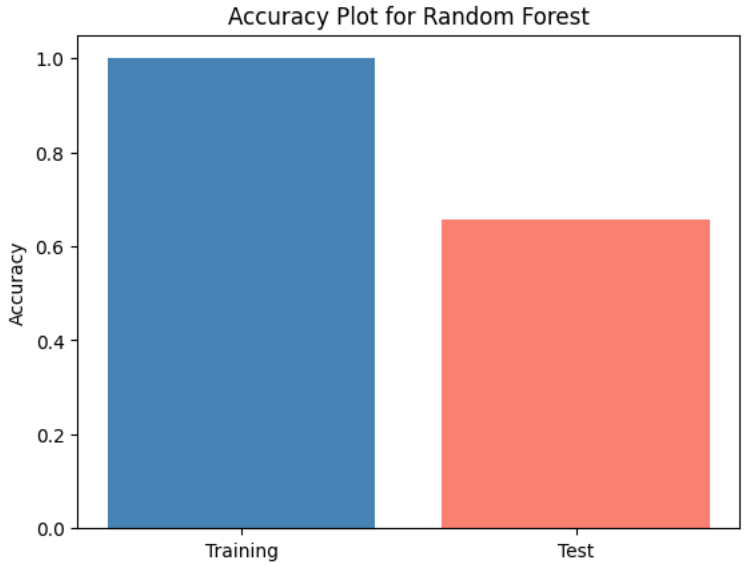



```
Confusion Matrix (Training Set):
 [[ 450   80   48]
 [   3 1742   17]
 [   3   28 1495]]

Confusion Matrix (Test Set):
 [[ 23  74  35]
 [  6 384  69]
 [  2 131 243]]
```

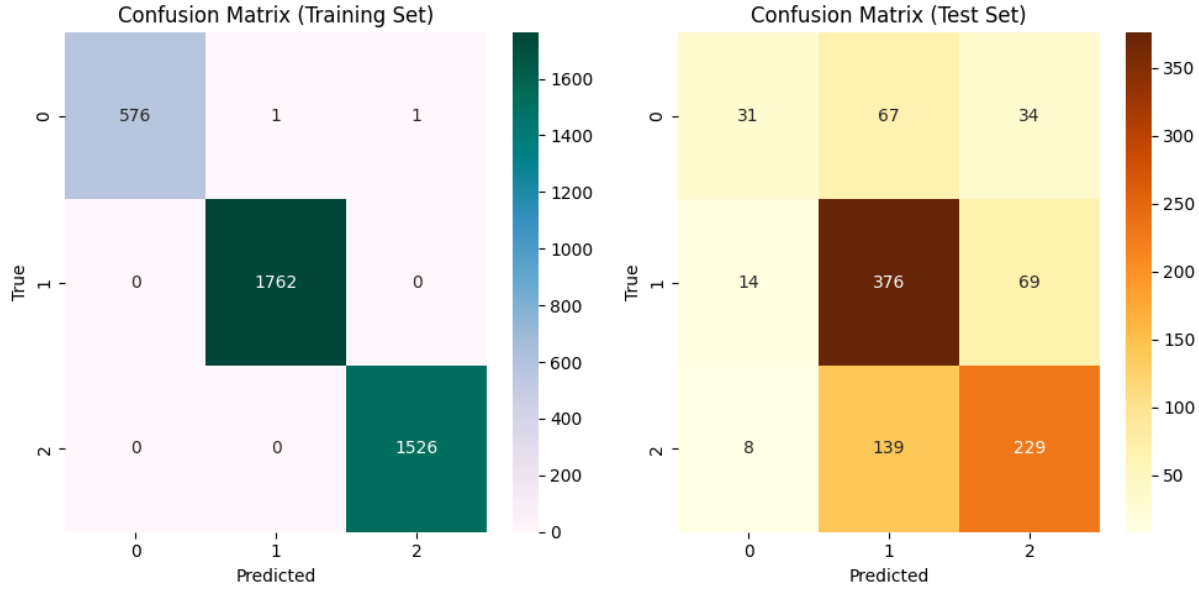


iii. Random Forest

```
------------------ MODEL EVALUATION ------------------
Model Name:          Random Forest
======================================================

Training Accuracy:   0.9995
Test Accuracy:       0.6577

------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.23      0.34       132
           1       0.65      0.82      0.72       459
           2       0.69      0.61      0.65       376

    accuracy                           0.66       967
   macro avg       0.64      0.55      0.57       967
weighted avg       0.65      0.66      0.64       967
```

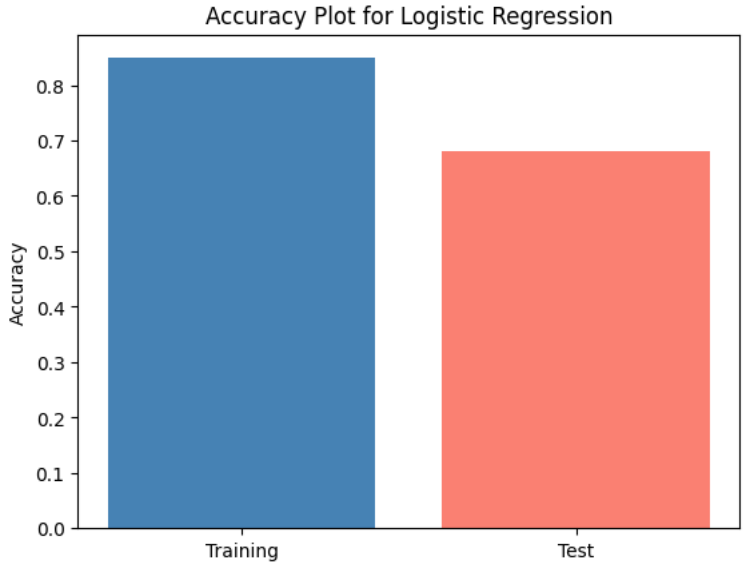



```
Confusion Matrix (Training Set):
 [[ 576    1    1]
 [   0 1762    0]
 [   0    0 1526]]

Confusion Matrix (Test Set):
 [[ 31  67  34]
 [ 14 376  69]
 [  8 139 229]]
```

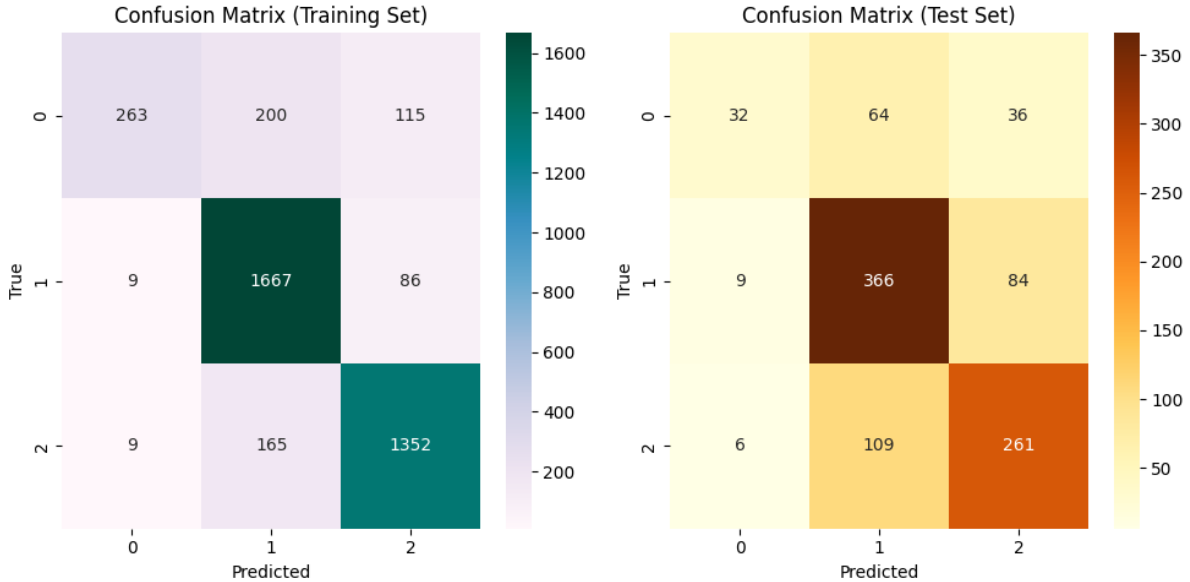


iv. Logistic Regression

```
------------------ MODEL EVALUATION ------------------
Model Name:           Logistic Regression
======================================================

Training Accuracy:   0.8489
Test Accuracy:       0.6815

------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.24      0.36       132
           1       0.68      0.80      0.73       459
           2       0.69      0.69      0.69       376

    accuracy                           0.68       967
   macro avg       0.68      0.58      0.59       967
weighted avg       0.68      0.68      0.67       967
```

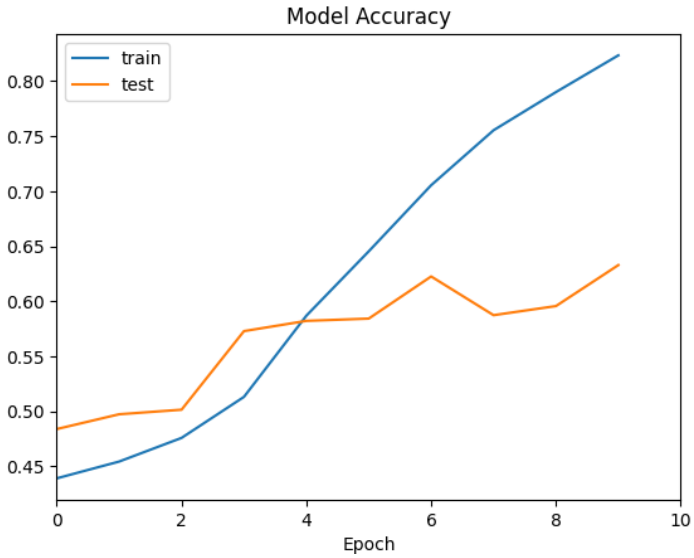


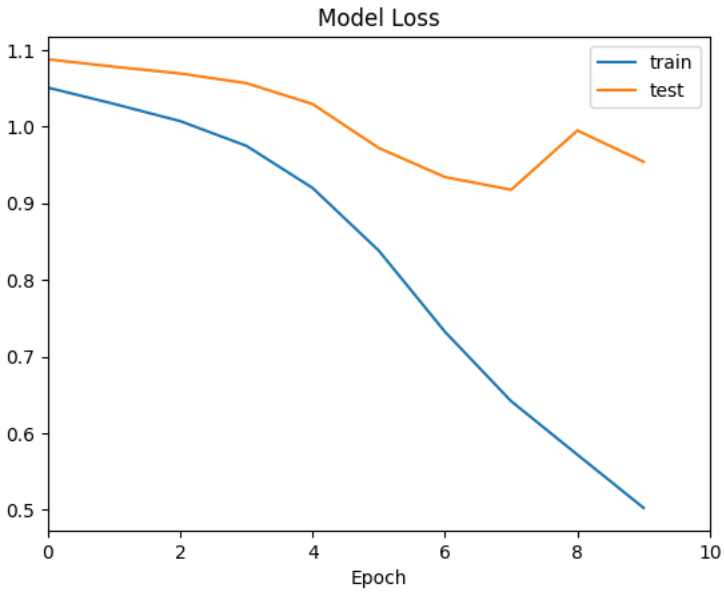

```
Confusion Matrix (Training Set):
 [[ 263  200  115]
 [   9 1667   86]
 [   9  165 1352]]

Confusion Matrix (Test Set):
 [[ 32  64  36]
 [  9 366  84]
 [  6 109 261]]
```



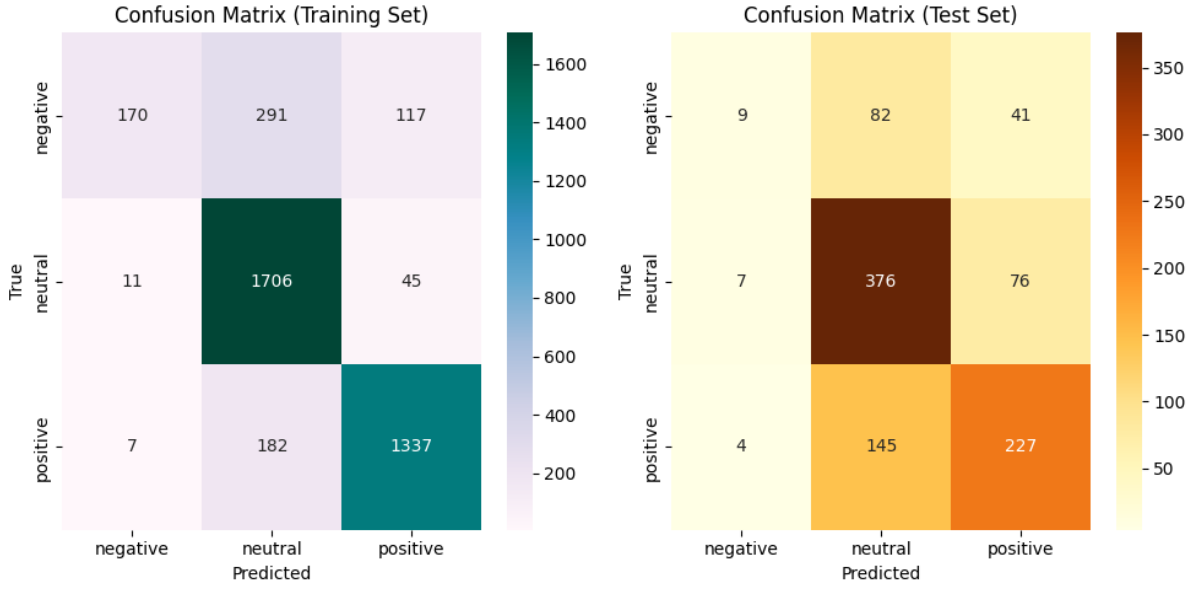
v. LSTM

```
Epoch 1/10
61/61 [==============================] - 16s 151ms/step - loss: 1.0511 - accuracy: 0.4392 - val_loss: 1.0880 - val_accuracy: 0.4840
Epoch 2/10
61/61 [==============================] - 5s 74ms/step - loss: 1.0298 - accuracy: 0.4545 - val_loss: 1.0784 - val_accuracy: 0.4974
Epoch 3/10
61/61 [==============================] - 3s 52ms/step - loss: 1.0073 - accuracy: 0.4759 - val_loss: 1.0695 - val_accuracy: 0.5016
Epoch 4/10
61/61 [==============================] - 2s 32ms/step - loss: 0.9750 - accuracy: 0.5132 - val_loss: 1.0568 - val_accuracy: 0.5729
Epoch 5/10
61/61 [==============================] - 2s 31ms/step - loss: 0.9200 - accuracy: 0.5869 - val_loss: 1.0296 - val_accuracy: 0.5822
Epoch 6/10
61/61 [==============================] - 1s 20ms/step - loss: 0.8382 - accuracy: 0.6454 - val_loss: 0.9722 - val_accuracy: 0.5843
Epoch 7/10
61/61 [==============================] - 3s 47ms/step - loss: 0.7322 - accuracy: 0.7054 - val_loss: 0.9343 - val_accuracy: 0.6225
Epoch 8/10
61/61 [==============================] - 2s 27ms/step - loss: 0.6416 - accuracy: 0.7553 - val_loss: 0.9177 - val_accuracy: 0.5874
Epoch 9/10
61/61 [==============================] - 1s 20ms/step - loss: 0.5718 - accuracy: 0.7900 - val_loss: 0.9951 - val_accuracy: 0.5957
Epoch 10/10
61/61 [==============================] - 1s 16ms/step - loss: 0.5023 - accuracy: 0.8233 - val_loss: 0.9543 - val_accuracy: 0.6329
```






```
121/121 [==============================] - 2s 5ms/step
31/31 [==============================] - 0s 4ms/step

------------------ MODEL EVALUATION ------------------
Model Name:           LSTM Model
======================================================

Training Accuracy:   0.8311
Test Accuracy:       0.6329

------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.07      0.12       132
           1       0.62      0.82      0.71       459
           2       0.66      0.60      0.63       376

    accuracy                           0.63       967
   macro avg       0.58      0.50      0.49       967
weighted avg       0.61      0.63      0.60       967


Confusion Matrix (Training Set):
 [[ 170  291  117]
 [  11 1706   45]
 [   7  182 1337]]

Confusion Matrix (Test Set):
 [[  9  82  41]
 [  7 376  76]
 [  4 145 227]]
```

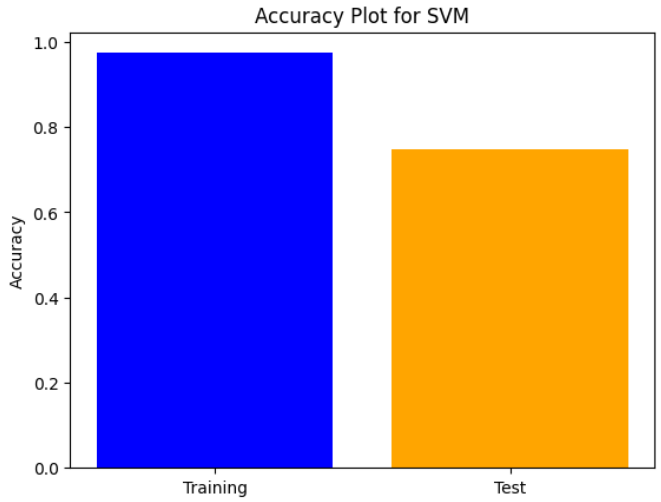


c. English Dataset

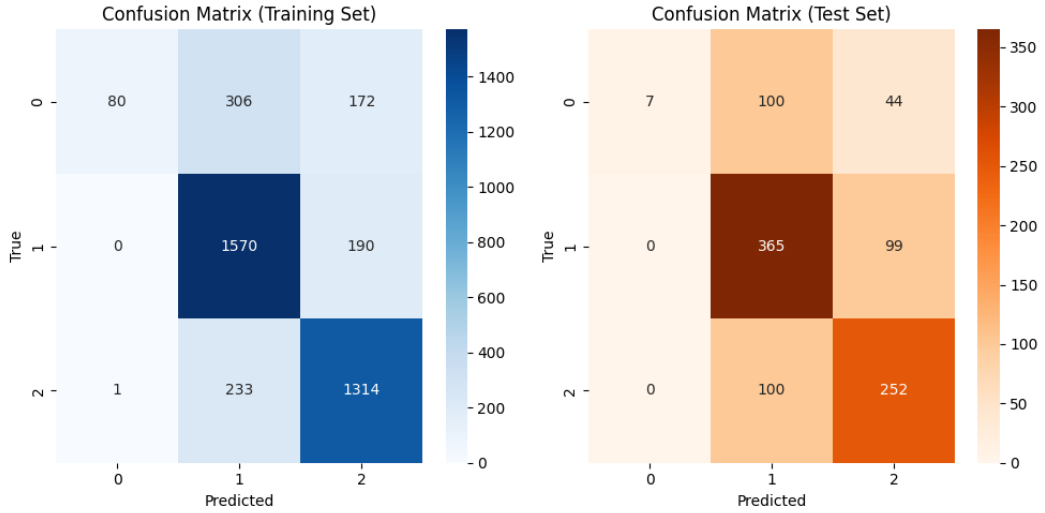
We turn our focus to sentiment analysis models that were trained using the standard English dataset in this section. We visually represent the accuracy results using confusion matrices and accuracy graphs, providing insights into how effectively the models capture sentiment patterns in the native English dataset. These visual aids are useful instruments for evaluating the accuracy and reliability of our models. A thorough discussion and analysis of the results will be provided in the next part, which will also offer insightful information about the model's performance on the original English dataset and enable a useful comparison with the translated datasets.

i. Naïve Bayes

```
------------------ MODEL EVALUATION ------------------
Model Name:           Naive Bayes
======================================================

Training Accuracy:    0.7667
Test Accuracy:        0.6453

------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.05      0.09       151
           1       0.65      0.79      0.71       464
           2       0.64      0.72      0.67       352

    accuracy                           0.65       967
   macro avg       0.76      0.52      0.49       967
weighted avg       0.70      0.65      0.60       967
```

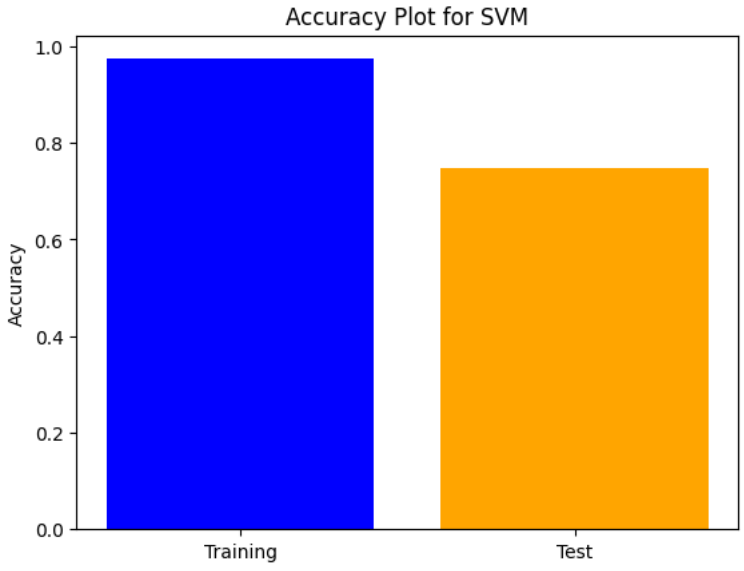


```
Confusion Matrix (Training Set):
 [[  80  306  172]
 [   0 1570  190]
 [   1  233 1314]]

Confusion Matrix (Test Set):
 [[  7 100  44]
 [  0 365  99]
 [  0 100 252]]
```

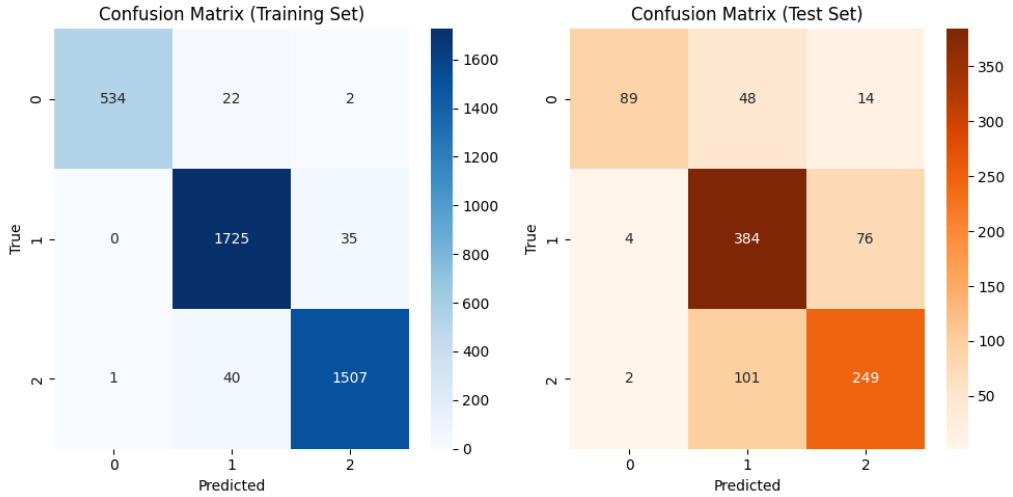


ii. SVM

```
------------------ MODEL EVALUATION ------------------
Model Name:           SVM
========================================================

Training Accuracy:   0.9741
Test Accuracy:       0.7466

--------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.59      0.72       151
           1       0.72      0.83      0.77       464
           2       0.73      0.71      0.72       352

    accuracy                           0.75       967
   macro avg       0.80      0.71      0.74       967
weighted avg       0.76      0.75      0.74       967
```

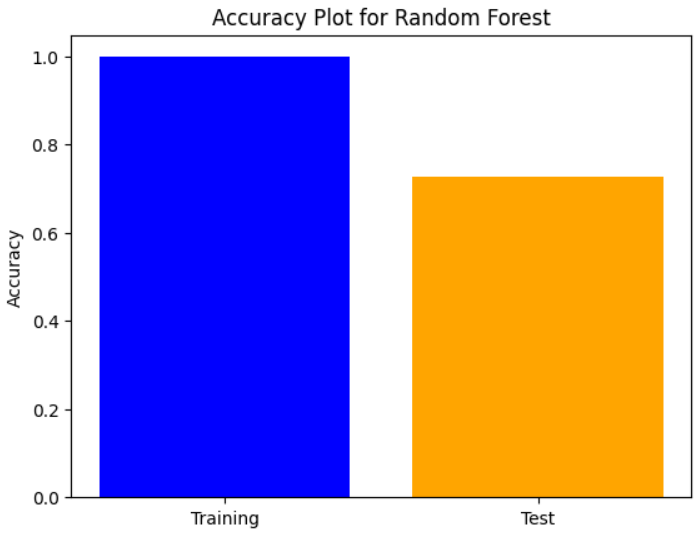


```
Confusion Matrix (Training Set):
 [[ 534   22    2]
 [   0 1725   35]
 [   1   40 1507]]

Confusion Matrix (Test Set):
 [[ 89  48  14]
 [  4 384  76]
 [  2 101 249]]
```

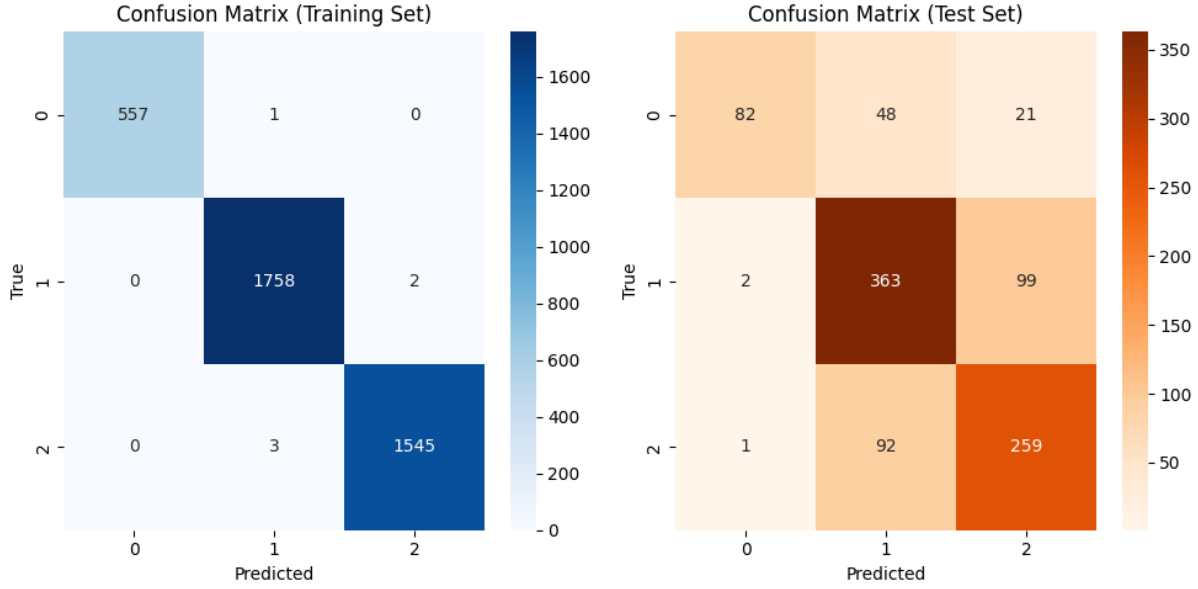


iii. Random Forest

```
------------------ MODEL EVALUATION ------------------
Model Name:          Random Forest
========================================================

Training Accuracy:   0.9984
Test Accuracy:       0.7280

--------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.54      0.69       151
           1       0.72      0.78      0.75       464
           2       0.68      0.74      0.71       352

    accuracy                           0.73       967
   macro avg       0.79      0.69      0.72       967
weighted avg       0.75      0.73      0.73       967
```



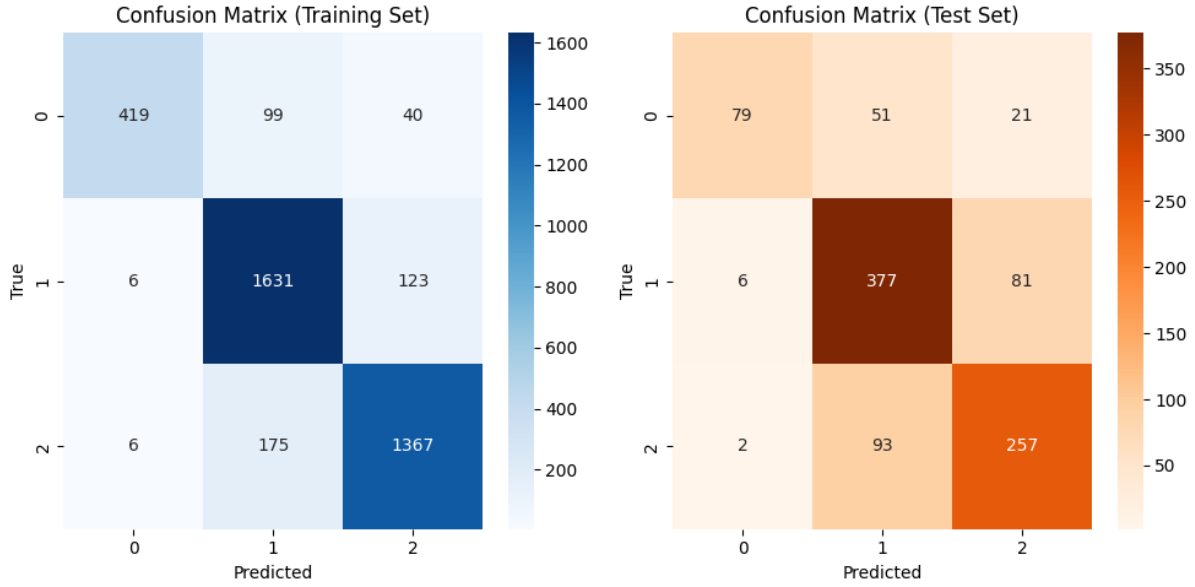
```
Confusion Matrix (Training Set):
 [[ 557    1    0]
 [   0 1758    2]
 [   0    3 1545]]

Confusion Matrix (Test Set):
 [[ 82  48  21]
 [  2 363  99]
 [  1  92 259]]
```



iv. Logistic Regression

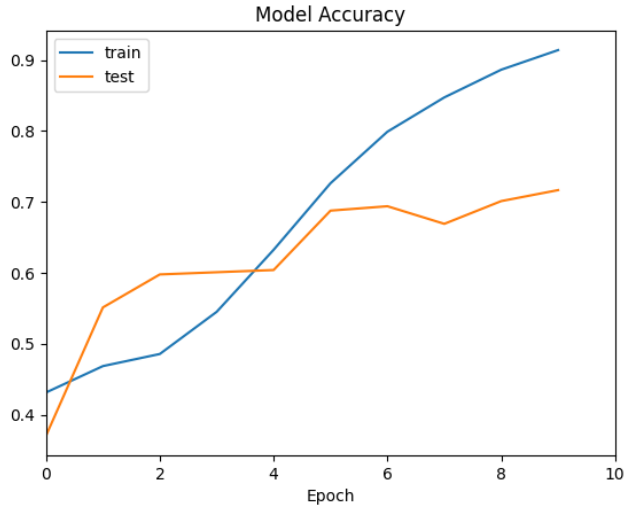
```
------------------ MODEL EVALUATION ------------------
Model Name:           Logistic Regression
========================================================

Training Accuracy:   0.8839
Test Accuracy:       0.7373

--------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.52      0.66       151
           1       0.72      0.81      0.77       464
           2       0.72      0.73      0.72       352

    accuracy                           0.74       967
   macro avg       0.78      0.69      0.72       967
weighted avg       0.75      0.74      0.73       967
```

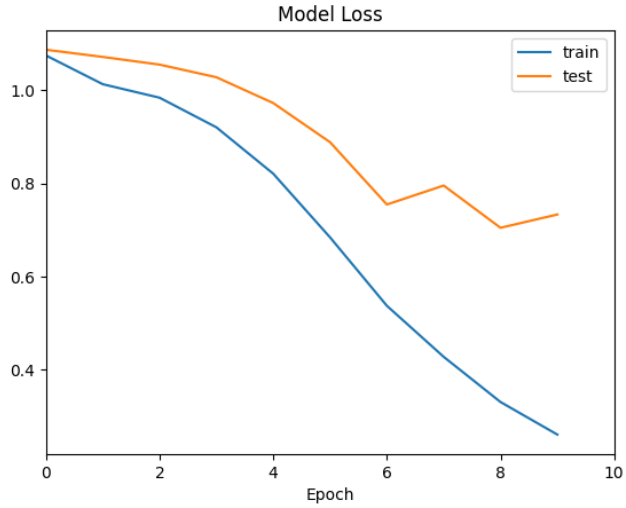
```
Confusion Matrix (Training Set):
 [[ 419   99   40]
 [   6 1631  123]
 [   6  175 1367]]

Confusion Matrix (Test Set):
 [[ 79  51  21]
 [  6 377  81]
 [  2  93 257]]
```



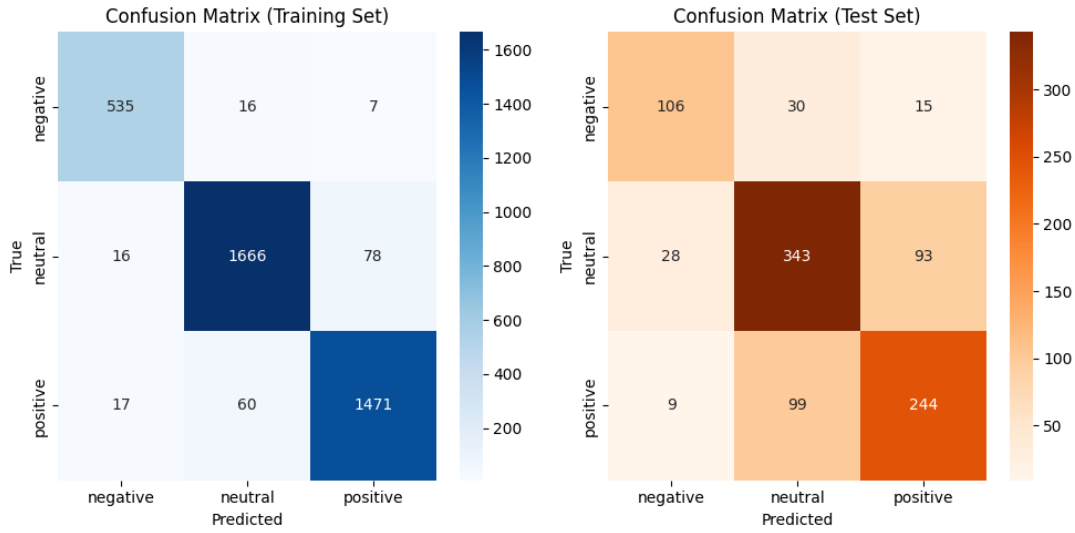
v. LSTM

```
Epoch 1/10
61/61 [==============================] - 17s 153ms/step - loss: 1.0741 - accuracy: 0.4312 - val_loss: 1.0864 - val_accuracy: 0.3702
Epoch 2/10
61/61 [==============================] - 6s 106ms/step - loss: 1.0125 - accuracy: 0.4684 - val_loss: 1.0709 - val_accuracy: 0.5512
Epoch 3/10
61/61 [==============================] - 5s 84ms/step - loss: 0.9835 - accuracy: 0.4855 - val_loss: 1.0545 - val_accuracy: 0.5977
Epoch 4/10
61/61 [==============================] - 3s 49ms/step - loss: 0.9200 - accuracy: 0.5450 - val_loss: 1.0274 - val_accuracy: 0.6008
Epoch 5/10
61/61 [==============================] - 3s 53ms/step - loss: 0.8207 - accuracy: 0.6324 - val_loss: 0.9721 - val_accuracy: 0.6039
Epoch 6/10
61/61 [==============================] - 3s 45ms/step - loss: 0.6842 - accuracy: 0.7263 - val_loss: 0.8882 - val_accuracy: 0.6877
Epoch 7/10
61/61 [==============================] - 2s 38ms/step - loss: 0.5375 - accuracy: 0.7990 - val_loss: 0.7546 - val_accuracy: 0.6939
Epoch 8/10
61/61 [==============================] - 2s 35ms/step - loss: 0.4280 - accuracy: 0.8474 - val_loss: 0.7952 - val_accuracy: 0.6691
Epoch 9/10
61/61 [==============================] - 2s 40ms/step - loss: 0.3312 - accuracy: 0.8864 - val_loss: 0.7047 - val_accuracy: 0.7011
Epoch 10/10
61/61 [==============================] - 1s 24ms/step - loss: 0.2614 - accuracy: 0.9141 - val_loss: 0.7332 - val_accuracy: 0.7166
```






```
121/121 [==============================] - 2s 6ms/step
31/31 [==============================] - 0s 5ms/step

------------------ MODEL EVALUATION ------------------
Model Name:           LSTM Model
============================================================

Training Accuracy:   0.9498
Test Accuracy:       0.7166

------------------------------------------------------------

Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.70      0.72       151
           1       0.73      0.74      0.73       464
           2       0.69      0.69      0.69       352

    accuracy                           0.72       967
   macro avg       0.72      0.71      0.72       967
weighted avg       0.72      0.72      0.72       967


Confusion Matrix (Training Set):
 [[ 535   16    7]
 [  16 1666   78]
 [  17   60 1471]]

Confusion Matrix (Test Set):
 [[106  30  15]
 [ 28 343  93]
 [  9  99 244]]
```



3.8.5 Hyperparameter Tuning

Hyperparameter tuning is crucial for maximizing model performance. Hyperparameters impact training unlike trainable weights and biases **(Smith et al., 2020)**. Improving generalization, accuracy, and reducing overfitting depend on wise hyperparameter selection **(Johnson & Zhang, 2021).**

We focus on grid search cross-validation which systematically evaluates different hyperparameter combinations like batch size, learning rate, model complexity, etc. to improve generalization and find optimal setup for sentiment analysis **(Bergstra and Bengio, 2012).**

Combining with k-fold cross-validation further enhances effectiveness. The data is split into k folds, model trained and evaluated on k-1 folds, and tested on the remaining fold. This process repeats k times for reliable and generalizable results.

Incorporating k-fold cross-validation into grid search enables precise hyperparameter optimization to unlock peak model performance **(Arlot and Célisse, 2010).** Outcomes will be demonstrated using accuracy, F1 score, and learning curves to assess overfitting reduction. The optimal configuration will determine the final model selection for comparison.

In summary, this section will highlight the importance of hyperparameter tuning and show how grid search with cross-validation systematically optimizes sentiment classifiers.

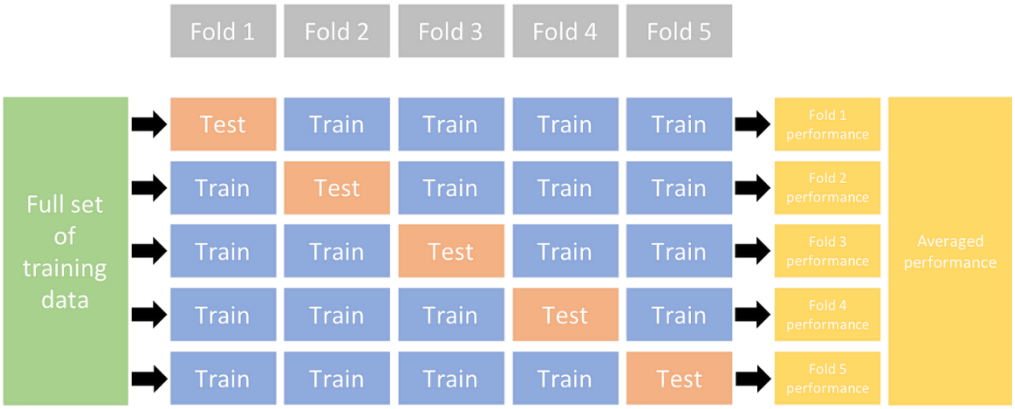


Figure 35: Cross Validation structure (Ultralytics, 2023)

a. Hyperparameter Tuning Settings for each model

Model	Hyperparameter settings

Naïve Bayes	`# Defining the parameter grid`
	`param_grid = {`
	`'tfidf__max_features': [1000, 5000, 10000, None],`
	`'tfidf__ngram_range': [(1, 1), (1, 2), (2, 2)],`
	`'model__alpha': [0.1, 0.5, 1.0, 1.5, 2.0],`
	`'model__fit_prior': [True, False],`
	`'tfidf__use_idf': [True, False],`
	`'tfidf__smooth_idf': [True, False],`
	`'tfidf__sublinear_tf': [True, False],`
	`}`
	`# Creating a pipeline with TfidfVectorizer and MultinomialNB`
	`model = MultinomialNB()`
	`pipeline = Pipeline([`
	`('tfidf', TfidfVectorizer()),`
	`('model', model)`
	`])`
	`# Create StratifiedKFold instance`
	`stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)`
	`# Performing GridSearchCV`
	`grid_search = GridSearchCV(pipeline, param_grid=param_grid, cv=stratified_kfold, scoring='accuracy', n_jobs=-1)`
	`grid_search.fit(X_train, y_train)`

SVM

```
# Hyperparameters to tune
kernels = ['linear', 'rbf']
gamma = [0.1, 1, 'auto']
C = [1, 10, 50]

# Hyperparameter grid
grid = {'tfidf__max_features': [10000, 50000],
        'tfidf__ngram_range': [(1, 1), (1, 2)],
        'svm__C': C,
        'svm__gamma': gamma,
        'svm__kernel': kernels}

# Grid search
pipeline = Pipeline([('tfidf', tfidf), ('svm', svm)])
# Create StratifiedKFold instance
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
cv = GridSearchCV(pipeline, grid, cv=stratified_kfold)
cv.fit(X_train, y_train)
```

Random Forest

```
# Hyperparameter grid
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

#Performing Kfold cross validation
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
# Grid search
grid_search = GridSearchCV(rf, param_grid, cv=stratified_kfold, n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)
```

Logistic Regression

```
# Hyperparameters to tune
penalty = ['l1', 'l2']
c_values = [0.1, 1, 10]
solver = ['liblinear', 'saga']

# Create grid
grid = dict(vect__ngram_range=[(1,1), (1,2)],
            vect__max_features=[5000, 10000, None],
            model__penalty=penalty,
            model__C=c_values,
            model__solver=solver)

stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
# GridSearch
cv = GridSearchCV(pipe, grid, cv=stratified_kfold, n_jobs=-1, verbose=2)
```

LSTM

```
# Defining the search parameters
param_grid = {
    'embedding_dim': [50, 100],
    'lstm_units': [64, 128],
    'batch_size': [32, 64],
    'epochs': [10, 20],
    'dropout_rate': [0.7]
}
```

```
# Creating a custom estimator class
class MyKerasClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, embedding_dim=50, lstm_units=64, batch_size=32, epochs=10, dropout_rate=0.5):
        self.embedding_dim = embedding_dim
        self.lstm_units = lstm_units
        self.batch_size = batch_size
        self.epochs = epochs
        self.dropout_rate = dropout_rate

    def fit(self, X, y, **fit_params):
        callbacks = fit_params.pop('callbacks', [])  # Remove 'callbacks' from fit_params
        model = Sequential()
        model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=self.embedding_dim, input_length=max_len))
        model.add(Bidirectional(LSTM(self.lstm_units, return_sequences=True)))
        model.add(BatchNormalization())
        model.add(Dropout(self.dropout_rate))
        model.add(Bidirectional(LSTM(64)))
        model.add(Dropout(self.dropout_rate))
        model.add(Dense(y.shape[1], activation='softmax'))
        model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.0001), metrics=['accuracy'])

        early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
        callbacks += [early_stopping]

        # Save training history for plotting
        history = model.fit(X, y, epochs=self.epochs, verbose=1, validation_data=(X_test, y_test), batch_size=self.batch_size, callbac

        self.model = model
        self.history = history  # Save history for plotting
```

```
    def predict(self, X):
        return np.argmax(self.model.predict(X), axis=1)

    def score(self, X, y):
        y_pred = to_categorical(self.predict(X), num_classes=y.shape[1])
        return accuracy_score(y, y_pred)

    def get_params(self, deep=True):
        return {
            'embedding_dim': self.embedding_dim,
            'lstm_units': self.lstm_units,
            'batch_size': self.batch_size,
            'epochs': self.epochs,
            'dropout_rate': self.dropout_rate
        }

    def set_params(self, **params):
        for param, value in params.items():
            setattr(self, param, value)
        return self

# Adding Kfolds
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
# Create the grid search object
grid_search = GridSearchCV(estimator=MyKerasClassifier(), param_grid=param_grid, scoring='accuracy', cv=stratified_kfold, n_jobs=-1, verbose=2)
# Fit the grid search object
grid_search.fit(X_train, y_train, callbacks=[EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)])
```

Table 6. Hyperparameter tuning settings

Naïve Bayes:

- **`max_features`:** Explored [1000, 5000, 10000, None] to set vocabulary size, balancing information capture and computational cost.
- **`ngram_range`:** Investigated [(1, 1), (1, 2), (2, 2)] for word sequence features, selecting unigrams, unigrams, and bigrams, or only bigrams.
- **`alpha`:** Examined [0.1, 0.5, 1.0, 1.5, 2.0] for smoothing, preventing zero probabilities for unseen features.
- **`use_idf`:** Tested True and False for applying IDF weighting to features.
- **`smooth_idf`:** Explored True and False to smooth IDF weights.
- **`sublinear_tf`:** Tested True and False for sublinear tf scaling, reducing the impact of very frequent terms.

SVM:

- **max_features:** Explored [10000, 50000] to set the maximum features for TF-IDF vectorization.
- **ngram_range:** Investigated [(1, 1), (1, 2)] for unigrams and bigrams to capture different contextual information levels.
- **C:** Explored [1, 10, 50] for the SVM regularization parameter. Higher C values indicate less regularization.
- **gamma:** Examined [0.1, 1, 'auto'] for SVM, defining the influence range of a single training example. 'auto' considers 1/n_features.
- **kernel:** Tested ['linear', 'rbf'] for SVM, specifying the hyperplane type. 'Linear' uses a linear hyperplane, and 'rbf' uses a radial basis function.

Logistic Regression:

- **penalty:** Explored ['l1', 'l2'] for the penalization norm. 'l1' adds absolute values, and 'l2' adds squares.
- **C:** Explored [0.1, 1, 10] for the inverse of regularization strength. Smaller values indicate stronger regularization.

- **solver:** Tested ['liblinear', 'saga'] for optimization. 'liblinear' suits small datasets, 'saga' suits large datasets.
- **ngram_range:** Explored [(1,1), (1,2)] for unigrams and bigrams during grid search.
- **max_features:** Investigated [5000, 10000, None] for the maximum features during model training. None considers all features.

Random Forest:

- **`n_estimators`:** Explored [50, 100, 150] for the number of trees in the forest.
- **`max_depth`:** Explored [None, 10, 20] for the maximum tree depth. Deeper trees capture more patterns but can overfit.
- **`min_samples_split`:** Investigated [2, 5, 10] for the minimum samples to split an internal node, preventing overfitting.
- **`min_samples_leaf`:** Tested [1, 2, 4] for the minimum samples at a leaf node, controlling overfitting.

LSTM:

- **embedding_dim:** Explored with values [50, 100] to find the optimal size for word embeddings.
- **lstm_units:** Investigated with values [64, 128] to determine the number of LSTM units in the model.
- **batch_size:** Explored with values [32, 64] to identify the batch size for efficient training.
- **Epochs:** Evaluated with values [10, 20] to determine the number of training iterations.
- **dropout_rate:** Set to a constant value of 0.7 for regularization during training.

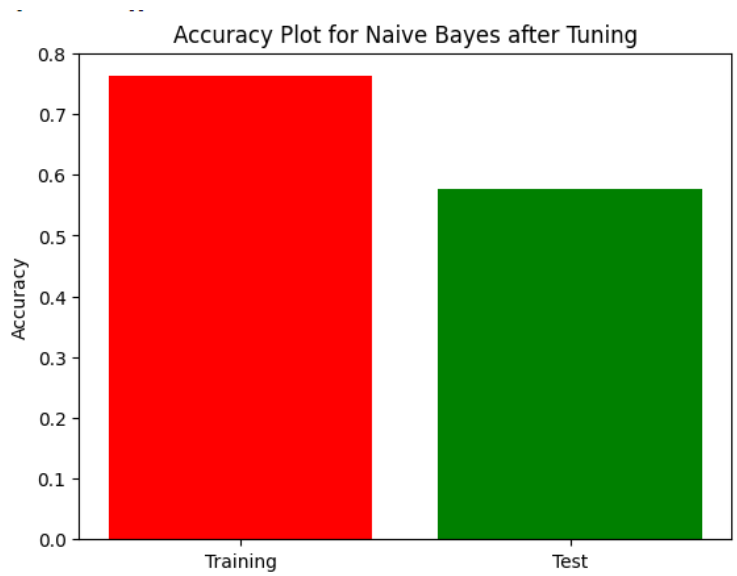
These numerical choices are part of a systematic exploration aimed at optimizing each model's performance on the given dataset, considering factors like computational efficiency and model generalization. The Grid Search systematically tests combinations of these values to identify the most effective configuration.

b. Performing Hyperparameter tuning on Translated Hindi Dataset (Transformers Library)

Now in this section we'll find the best parameters for our training using the Translated Hindi Dataset using Transformers library. We'll see the results through accuracy plots, classification reports, and confusion matrices. A more detailed discussion of the results will be provided in the next section.

i. Naïve Bayes

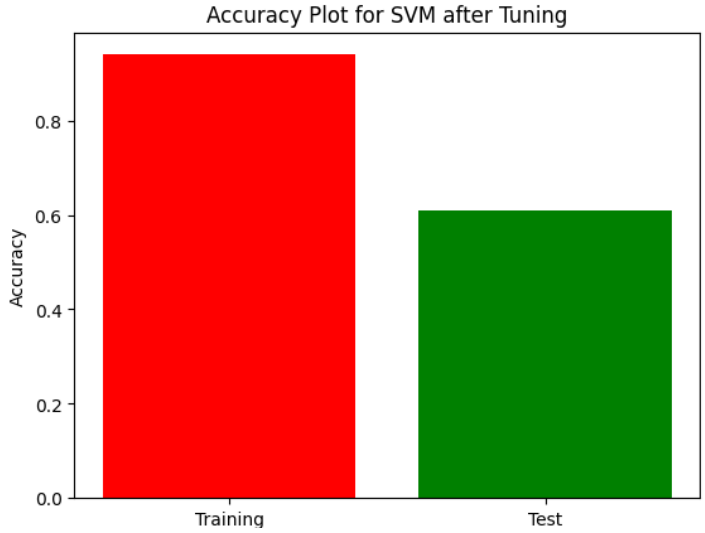
```
Hyperparameter Tuning: Naive Bayes
**********
Best Parameters:
model__alpha        : 0.5
model__fit_prior    : True
tfidf__max_features : 5000
tfidf__ngram_range  : (1, 1)
tfidf__smooth_idf   : True
tfidf__sublinear_tf : True
tfidf__use_idf      : False
Training accuracy: 0.763
Test accuracy: 0.576
Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.83      0.03      0.06       155
           1       0.58      0.70      0.64       428
           2       0.57      0.66      0.61       384

    accuracy                           0.58       967
   macro avg       0.66      0.46      0.44       967
weighted avg       0.62      0.58      0.53       967

Confusion Matrix (Training Set):
 [[  82  249  223]
 [   2 1592  202]
 [   0  242 1274]]

Confusion Matrix (Test Set):
 [[  5  83  67]
 [  1 300 127]
 [  0 132 252]]
```



ii. SVM

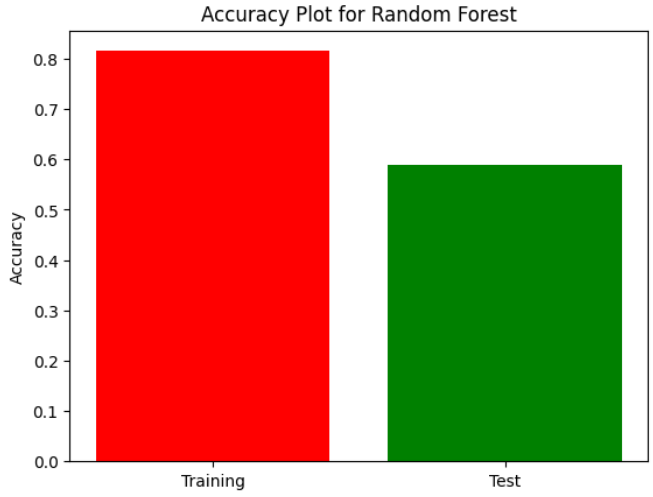
```
Hyperparameter Tuning: SVM
**********
Best Parameters:
svm__C              : 1
svm__gamma          : 0.1
svm__kernel         : linear
tfidf__max_features : 50000
tfidf__ngram_range  : (1, 2)
Training accuracy: 0.940
Test accuracy: 0.609

Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.56      0.26      0.35       155
           1       0.62      0.71      0.66       428
           2       0.61      0.64      0.62       384

    accuracy                           0.61       967
   macro avg       0.59      0.54      0.55       967
weighted avg       0.60      0.61      0.60       967
```

```
Confusion Matrix (Training Set):
 [[ 424   80   50]
 [   3 1755   38]
 [   0   61 1455]]

Confusion Matrix (Test Set):
 [[ 40  67  48]
 [ 16 304 108]
 [ 16 123 245]]
```



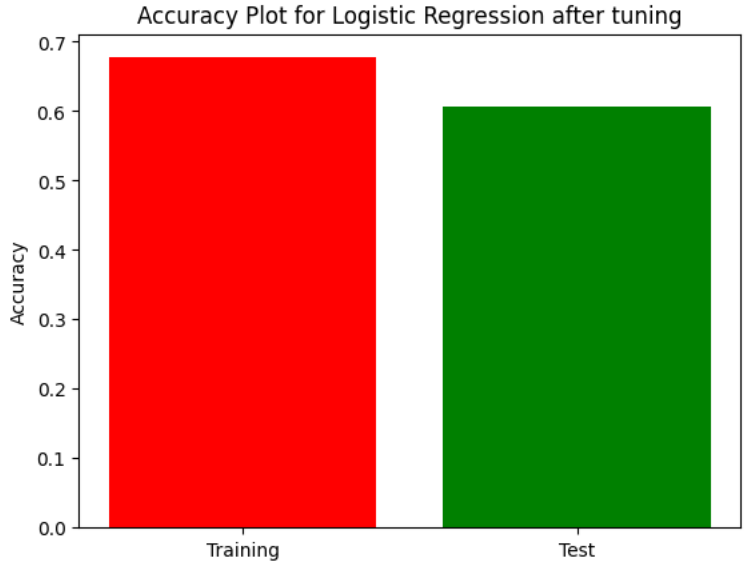
iii. Random Forest

```
Hyperparameter Tuning: Random Foresr
********************************
Best Parameters:
max_depth          : None
min_samples_leaf   : 2
min_samples_split  : 5
n_estimators       : 100
Training accuracy: 0.815
Test accuracy: 0.588
Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.55      0.15      0.24       155
           1       0.58      0.73      0.65       428
           2       0.61      0.60      0.60       384

    accuracy                           0.59       967
   macro avg       0.58      0.50      0.50       967
weighted avg       0.58      0.59      0.57       967

Confusion Matrix (Training Set):
 [[ 181  188  185]
 [  11 1643  142]
 [   3  185 1328]]

Confusion Matrix (Test Set):
 [[ 24  85  46]
 [ 10 313 105]
 [ 10 142 232]]
```



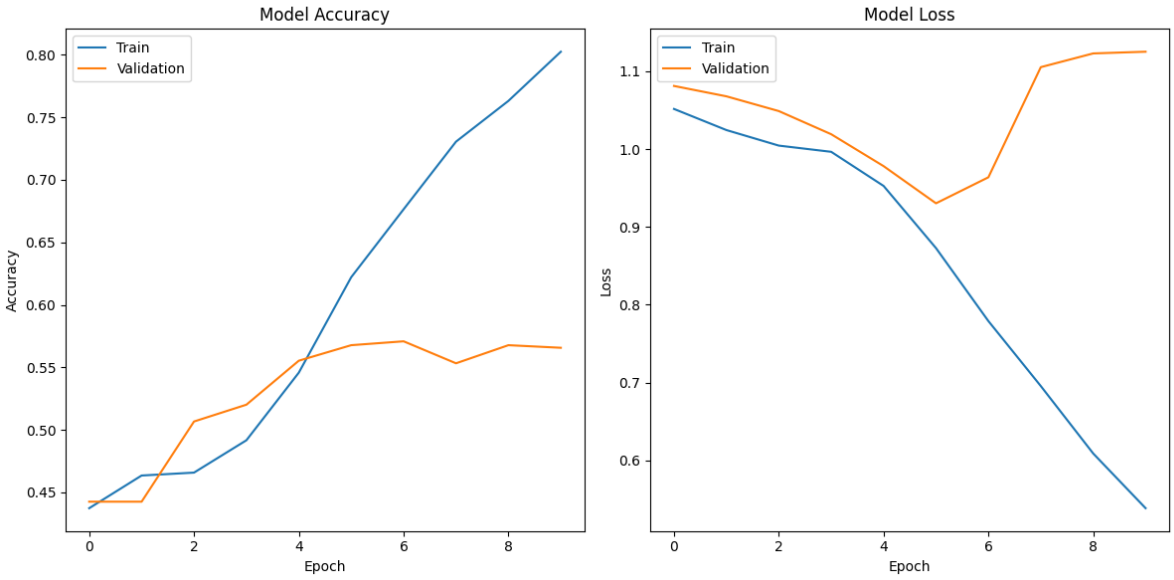
iv. Logistic Regression

```
Hyperparameter Tuning: Logistic Regression
**********
Best Parameters:
model__C            : 1
model__penalty      : l1
model__solver       : saga
vect__max_features  : 5000
vect__ngram_range   : (1, 1)
Training accuracy: 0.677
Test accuracy: 0.607
Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.69      0.23      0.34       155
           1       0.58      0.81      0.67       428
           2       0.65      0.53      0.59       384

    accuracy                           0.61       967
   macro avg       0.64      0.52      0.53       967
weighted avg       0.62      0.61      0.59       967

Confusion Matrix (Training Set):
 [[ 112  315  127]
 [  25 1591  180]
 [  15  586  915]]

Confusion Matrix (Test Set):
 [[ 35  84  36]
 [  8 347  73]
 [  8 171 205]]
```



v. LSTM

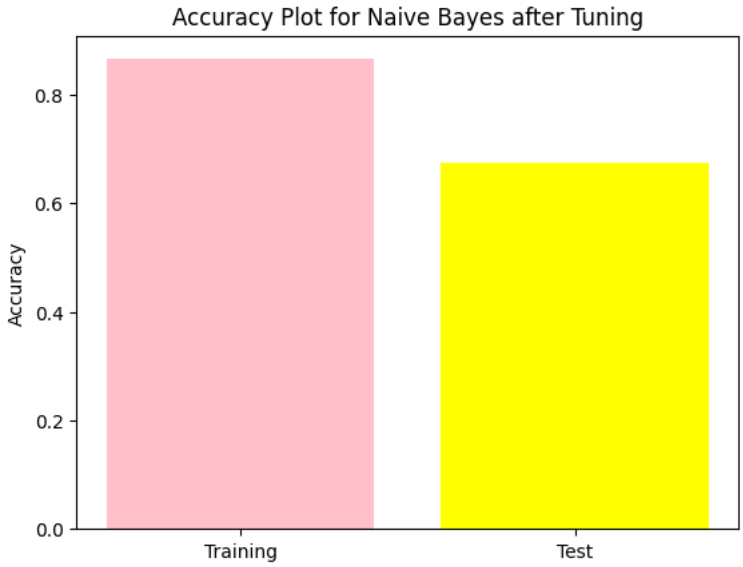
```
Training Accuracy: 0.8416968442834971
31/31 [==============================] - 0s 5ms/step
Testing Accuracy: 0.5656670113753878
31/31 [==============================] - 0s 5ms/step
Confusion Matrix:
[[ 45  38  72]
 [ 38 206 184]
 [ 18  70 296]]
Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.29      0.35       155
           1       0.66      0.48      0.56       428
           2       0.54      0.77      0.63       384

    accuracy                           0.57       967
   macro avg       0.55      0.51      0.51       967
weighted avg       0.57      0.57      0.55       967
```



c. Performing Hyperparameter tuning on Translated Hindi Dataset (Google Library)

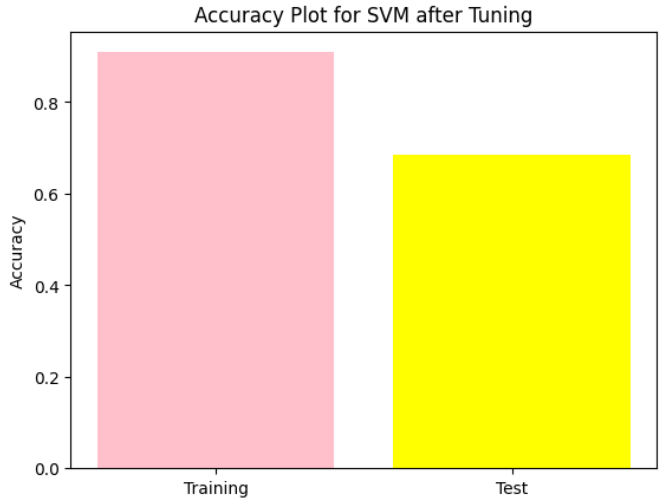
i. Naïve Bayes

```
Hyperparameter Tuning: Naive Bayes
**********
Best Parameters:
model__alpha        : 0.1
model__fit_prior    : True
tfidf__max_features : 5000
tfidf__ngram_range  : (1, 1)
tfidf__smooth_idf   : True
tfidf__sublinear_tf : False
tfidf__use_idf      : False
Training accuracy: 0.865
Test accuracy: 0.675
Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.68      0.32      0.43       132
           1       0.72      0.71      0.72       459
           2       0.63      0.76      0.69       376

    accuracy                           0.68       967
   macro avg       0.68      0.60      0.61       967
weighted avg       0.68      0.68      0.67       967

Confusion Matrix (Training Set):
 [[ 367  105  106]
 [  14 1596  152]
 [  10  134 1382]]

Confusion Matrix (Test Set):
 [[ 42  40  50]
 [ 13 327 119]
 [  7  85 284]]
```



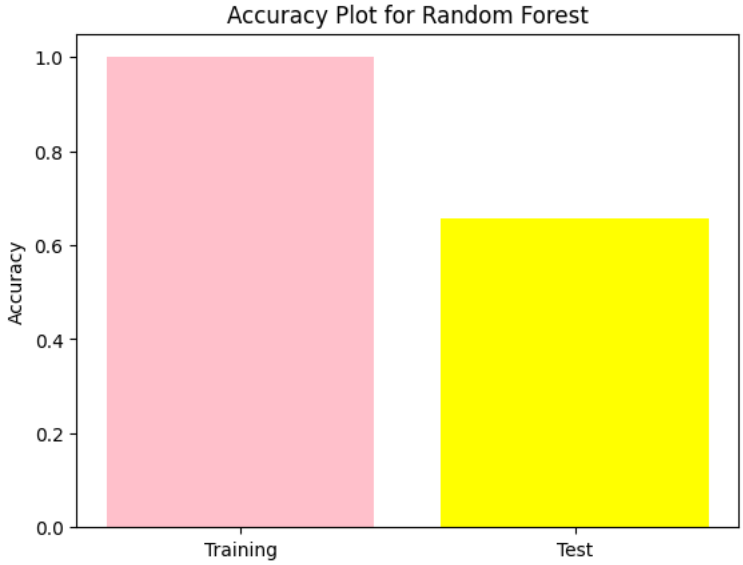
ii. SVM

```
Hyperparameter Tuning: SVM
**************************
Best Parameters:
svm__C              : 1
svm__gamma          : 0.1
svm__kernel         : linear
tfidf__max_features : 10000
tfidf__ngram_range  : (1, 2)
Training accuracy: 0.909
Test accuracy: 0.685

Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.63      0.33      0.43       132
           1       0.69      0.78      0.73       459
           2       0.68      0.69      0.69       376

    accuracy                           0.68       967
   macro avg       0.67      0.60      0.62       967
weighted avg       0.68      0.68      0.67       967

Confusion Matrix (Training Set):
 [[ 387  121   70]
 [   6 1708   48]
 [   4  103 1419]]

Confusion Matrix (Test Set):
 [[ 43  53  36]
 [ 16 358  85]
 [  9 106 261]]
```



iii. Random Forest

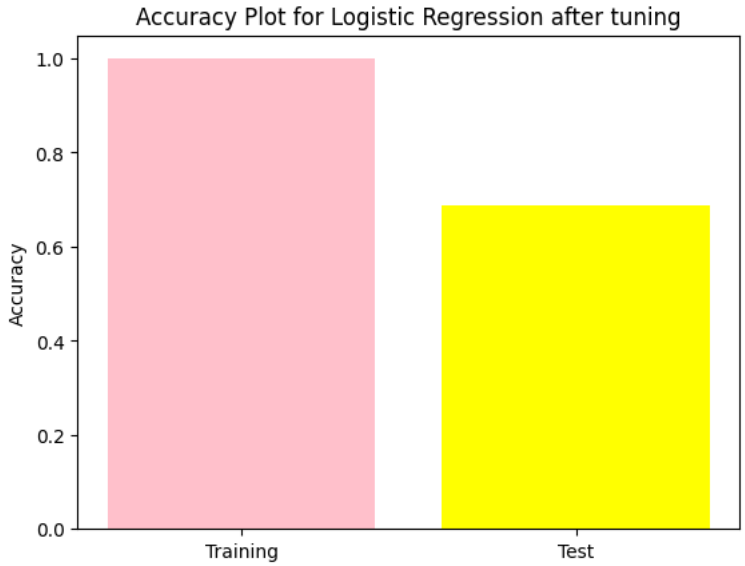
```
Hyperparameter Tuning: Random Foresr
********************************
Best Parameters:
max_depth           : None
min_samples_leaf    : 1
min_samples_split   : 2
n_estimators        : 100
Training accuracy: 0.999
Test accuracy: 0.656
Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.57      0.24      0.34       132
           1       0.65      0.83      0.73       459
           2       0.68      0.59      0.63       376

    accuracy                           0.66       967
   macro avg       0.63      0.55      0.57       967
weighted avg       0.65      0.66      0.64       967

Confusion Matrix (Training Set):
 [[ 577    0    1]
 [   1 1761    0]
 [   0    0 1526]]

Confusion Matrix (Test Set):
 [[ 32  61  39]
 [ 15 379  65]
 [  9 144 223]]
```



iv. Logistic Regression

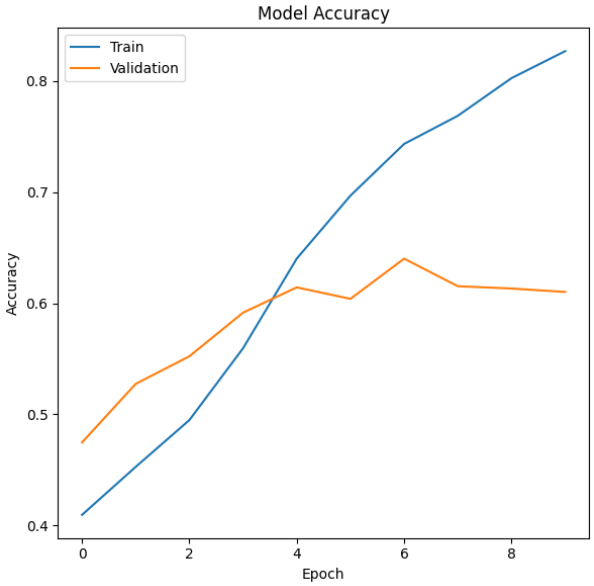
```
Hyperparameter Tuning: Logistic Regression
********************************
Best Parameters:
model__C            : 10
model__penalty      : l1
model__solver       : liblinear
vect__max_features  : None
vect__ngram_range   : (1, 2)
Training accuracy: 0.999
Test accuracy: 0.687
Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.55      0.41      0.47       132
           1       0.73      0.72      0.73       459
           2       0.67      0.74      0.70       376

    accuracy                           0.69       967
   macro avg       0.65      0.62      0.63       967
weighted avg       0.68      0.69      0.68       967
```

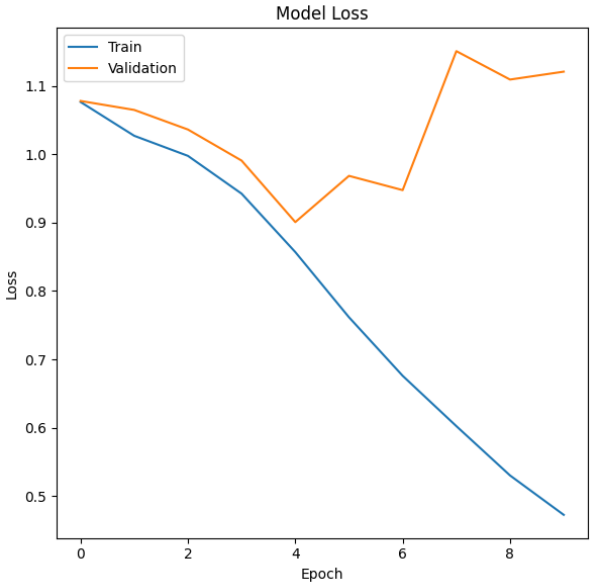
```
Confusion Matrix (Training Set):
 [[ 576    1    1]
 [   0 1762    0]
 [   0    1 1525]]

Confusion Matrix (Test Set):
 [[ 54  39  39]
 [ 29 330 100]
 [ 15  81 280]]
```



v. LSTM

```
Training Accuracy: 0.8887739265390585
31/31 [==============================] - 0s 5ms/step
Testing Accuracy: 0.610134436401241
31/31 [==============================] - 0s 4ms/step
Confusion Matrix:
[[ 35  51  46]
 [ 50 293 116]
 [ 34  80 262]]
Classification Report:
              precision    recall  f1-score   support

           0       0.29      0.27      0.28       132
           1       0.69      0.64      0.66       459
           2       0.62      0.70      0.65       376

    accuracy                           0.61       967
   macro avg       0.53      0.53      0.53       967
weighted avg       0.61      0.61      0.61       967
```

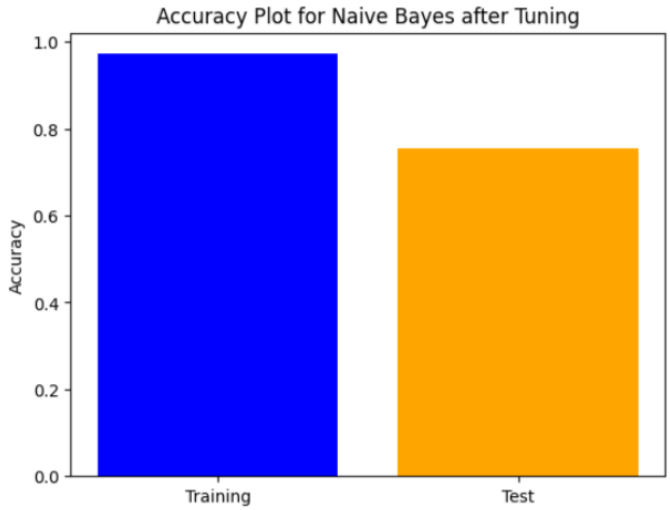




d. Performing Hyperparameter tuning on English Dataset

i. Naïve Bayes

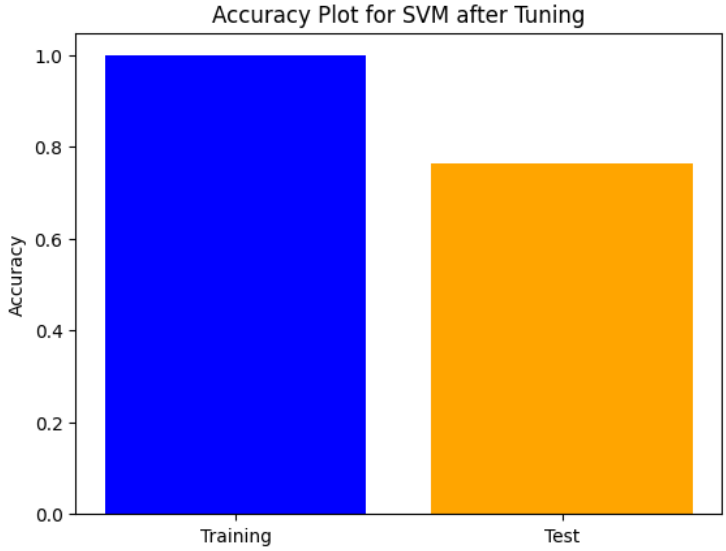
```
Hyperparameter Tuning: Naive Bayes
**********
Best Parameters:
model__alpha        : 0.1
model__fit_prior    : False
tfidf__max_features : None
tfidf__ngram_range  : (1, 2)
tfidf__smooth_idf   : True
tfidf__sublinear_tf : True
tfidf__use_idf      : False
Training accuracy: 0.972
Test accuracy: 0.754
Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.89      0.78      0.83       151
           1       0.77      0.76      0.76       464
           2       0.69      0.73      0.71       352

    accuracy                           0.75       967
   macro avg       0.78      0.76      0.77       967
weighted avg       0.76      0.75      0.76       967

Confusion Matrix (Training Set):
 [[ 548    6    4]
 [   9 1686   65]
 [   1   22 1525]]

Confusion Matrix (Test Set):
 [[118  18  15]
 [ 10 353 101]
 [  4  90 258]]
```



ii. SVM

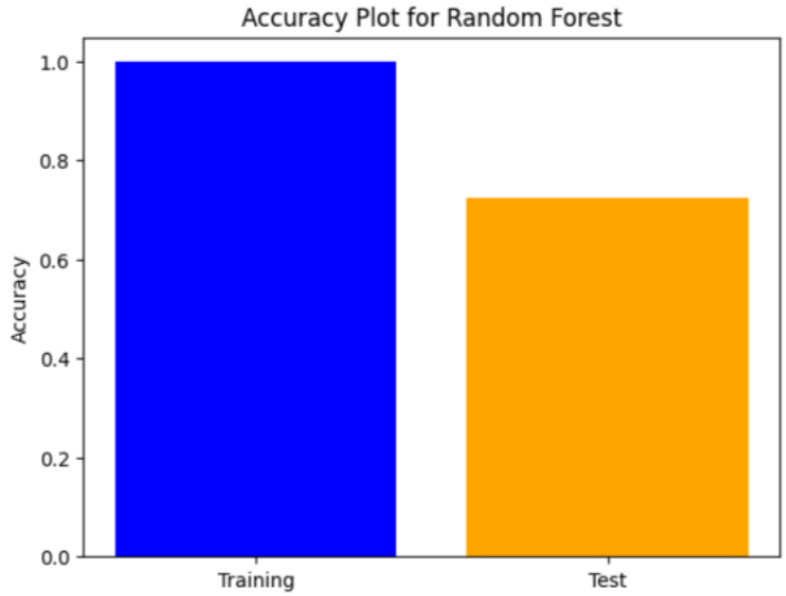
```
Hyperparameter Tuning: SVM
**********
Best Parameters:
svm__C              : 10
svm__gamma          : 1
svm__kernel         : rbf
tfidf__max_features : 10000
tfidf__ngram_range  : (1, 1)
Training accuracy: 0.998
Test accuracy: 0.764

Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.92      0.71      0.80       151
           1       0.76      0.80      0.78       464
           2       0.73      0.74      0.73       352

    accuracy                           0.76       967
   macro avg       0.80      0.75      0.77       967
weighted avg       0.77      0.76      0.76       967
```

```
Confusion Matrix (Training Set):
 [[ 558    0    0]
 [   1 1757    2]
 [   0    4 1544]]

Confusion Matrix (Test Set):
 [[107  30  14]
 [  7 373  84]
 [  2  91 259]]
```



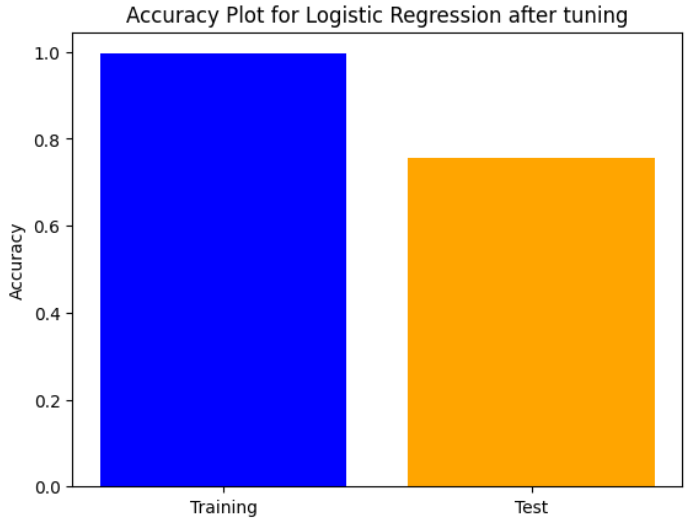
iii. Random Forest

```
Hyperparameter Tuning: Random Foresr
********************************
Best Parameters:
max_depth          : None
min_samples_leaf   : 1
min_samples_split  : 5
n_estimators       : 150
Training accuracy: 0.998
Test accuracy: 0.723
Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.94      0.52      0.67       151
           1       0.71      0.78      0.75       464
           2       0.69      0.73      0.71       352

    accuracy                           0.72       967
   macro avg       0.78      0.68      0.71       967
weighted avg       0.74      0.72      0.72       967

Confusion Matrix (Training Set):
 [[ 557    1    0]
 [   0 1757    3]
 [   0    3 1545]]

Confusion Matrix (Test Set):
 [[ 79  53  19]
 [  3 364  97]
 [  2  94 256]]
```



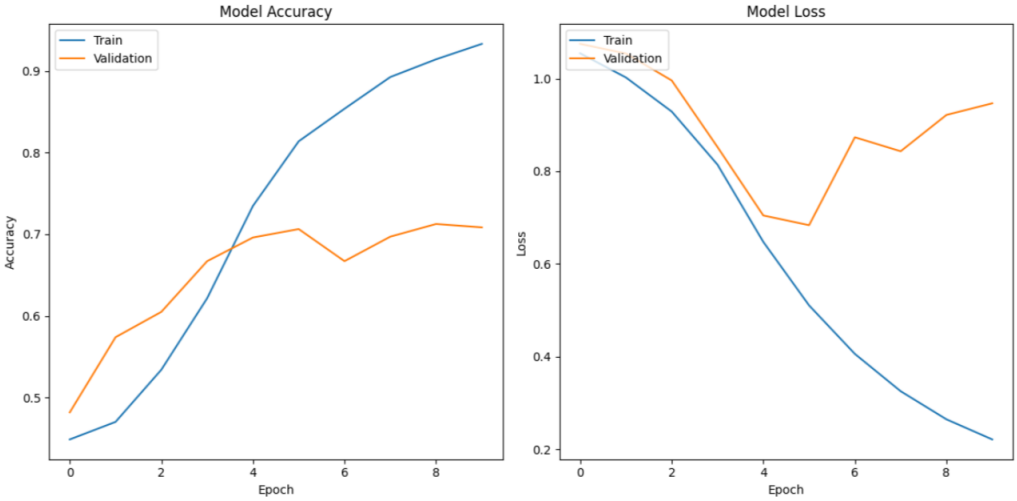
iv. Logistic Regression

```
Hyperparameter Tuning: Logistic Regression
**********
Best Parameters:
model__C            : 10
model__penalty      : l2
model__solver       : liblinear
vect__max_features  : None
vect__ngram_range   : (1, 2)
Training accuracy: 0.995
Test accuracy: 0.755
Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.89      0.74      0.81       151
           1       0.76      0.76      0.76       464
           2       0.70      0.75      0.73       352

    accuracy                           0.75       967
   macro avg       0.78      0.75      0.77       967
weighted avg       0.76      0.75      0.76       967

Confusion Matrix (Training Set):
 [[ 558    0    0]
 [   1 1750    9]
 [   0    8 1540]]

Confusion Matrix (Test Set):
 [[112  24  15]
 [ 12 353  99]
 [  2  85 265]]
```



v. LSTM

```
Training Accuracy: 0.9524055871702017
31/31 [==============================] - 0s 6ms/step
Testing Accuracy: 0.7083764219234746
31/31 [==============================] - 0s 5ms/step
Confusion Matrix:
[[104  29  18]
 [ 17 321 126]
 [  6  86 260]]
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.69      0.75       151
           1       0.74      0.69      0.71       464
           2       0.64      0.74      0.69       352

    accuracy                           0.71       967
   macro avg       0.73      0.71      0.72       967
weighted avg       0.72      0.71      0.71       967
```



Chapter 4

4.Results

Now after the extensive evaluation and testing here we'll look into the comparative analysis of our model trained on google , transformers and English dataset.

4.1 Before Hyperparameter Tuning

Dataset / Model	Transformers		Google		English	
	Accuracy					
	Training	Testing	Training	Testing	Training	Testing
Naïve Bayes	76.1%	56.5%	78.3%	65.2%	74.6%	64.5%
SVM	93.2%	59.4%	95.3%	67.2%	97.4%	74.6%
Random Forest	99.8%	58.7%	99.9%	65.7%	99.8%	72.8%
Logistic Regression	80.4%	60.0%	84.8%	68.1%	88.3%	73.7%
LSTM	71.5%	54.4%	83.1%	63.2%	94%	71.%

Dataset / Model		Transformers				Google				English			
		Metrics											
		precision	recall	F1	support	precision	Recall	F1	support	precision	Recall	F1	support
Naïve Bayes	**0**	0.50	0.01	0.01	155	0.75	0.02	0.04	132	1.0	0.05	0.09	151
	1	0.57	0.70	0.63	428	0.69	0.77	0.73	459	0.65	0.79	0.71	464
	2	0.57	0.64	0.60	384	0.61	0.73	0.67	376	0.64	0.72	0.67	352
SVM	**0**	0.86	0.12	0.21	155	0.74	0.17	0.28	132	0.94	0.59	0.72	151
	1	0.56	0.80	0.66	428	0.65	0.84	0.73	459	0.72	0.83	0.77	352
	2	0.64	0.55	0.59	384	0.70	0.65	0.67	376	0.73	0.71	0.72	352
Random Forest	**0**	0.50	0.17	0.25	155	0.58	0.23	0.34	132	0.96	0.54	0.69	151
	1	0.57	0.77	0.65	428	0.65	0.82	0.72	459	0.72	0.78	0.75	464
	2	0.63	0.56	0.59	384	0.69	0.61	0.65	376	0.68	0.74	0.71	352
Logistic Regression	**0**	0.63	0.22	0.37	155	0.68	0.24	0.36	132	0.88	0.52	0.66	151
	1	0.59	0.72	0.65	428	0.68	0.80	0.73	459	0.72	0.81	0.77	464
	2	0.61	0.62	0.61	384	0.69	0.69	0.69	376	0.72	0.73	0.72	352

LSTM	**0**	1.00	0.01	0.01	155	0.45	0.07	0.12	132	0.74	0.70	0.72	151
	1	0.67	0.46	0.54	428	0.62	0.82	0.71	459	0.73	0.74	0.73	464
	2	0.49	0.86	0.64	384	0.66	0.60	0.63	376	0.69	0.69	0.69	352

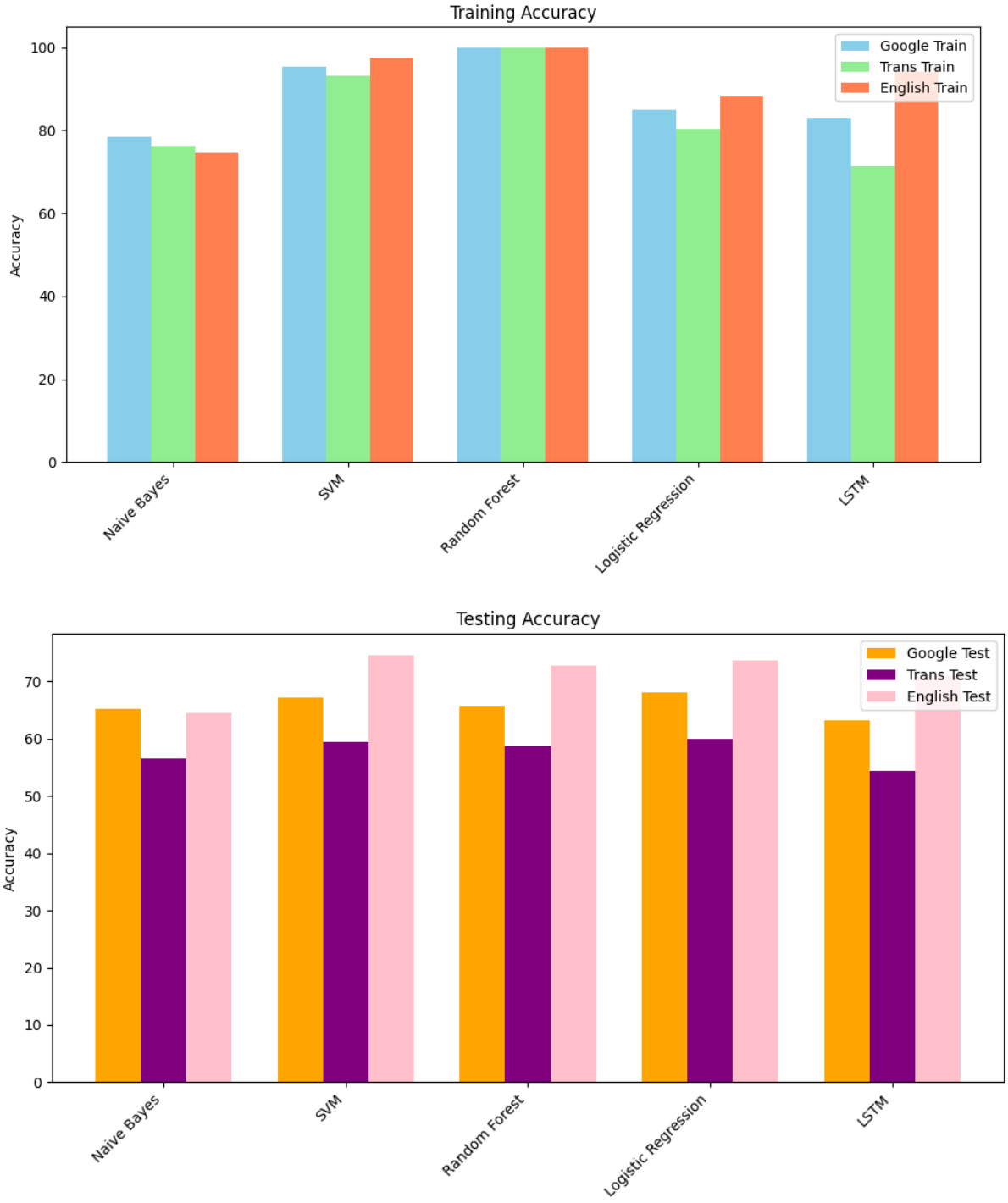


Figure 36: Model performance comparison graph and table between each dataset before tuning.

The above graph presents the training and testing accuracy of various machine learning models for sentiment analysis on three different datasets: Google,

English, and Transformers. The results highlight the strengths and weaknesses of each model, as well as potential challenges associated with specific datasets.

4.1.1 Model Performance evaluation:

Across the sentiment analysis datasets, Random Forest achieved the highest training accuracy at 99.8% on the Transformers dataset but showed overfitting, evident from the large gap compared to SVM's 59.4% test accuracy. The Google dataset exhibited a similar trend, with Random Forest overfitting, and SVM demonstrating better generalization. For the English dataset, SVM performed exceptionally well with a 97.4% training accuracy and a 74.6% testing accuracy, indicating effective generalization. The LSTM model overfit on all datasets, displaying high training accuracy but lower performance on the test set. Naive Bayes performed poorly across datasets. In summary, SVM demonstrated a balanced trade-off between fitting the training data and generalizing to new data, aligning with the bias-variance trade-off. The results underscore the importance of evaluating both training and testing accuracy for effective model selection.

a. Detailed Model Comparison before tuning:

With the help of above visualisation below results are being observed

Transformers Translated Dataset:

Naïve Bayes: Testing accuracy of 56.5%, moderate precision and recall, provides a reasonable baseline.

- Class 0 (Negative): Faces significant challenges, 1 instance correct, 185 misclassified as neutral or positive.
- Class 1 (Neutral): Dominates accurate predictions, 301 correct, 327 misclassified as negative or positive.
- Class 2 (Positive): Maintains reasonable accuracy, 245 correct, 139 misclassified as negative or neutral.

SVM: Testing accuracy of 59.4%, commendable precision, recall, and F1-score, reliable for sentiment analysis in low-resource languages.

- Class 0 (Negative): Maintains robust accuracy, 19 correct, 164 misclassified as neutral or positive.
- Class 1 (Neutral): Dominates accurate predictions, 343 correct, 85 misclassified as negative or positive.
- Class 2 (Positive): Maintains reasonable accuracy, 213 correct, 171 misclassified as negative or neutral.

Random Forest: Testing accuracy of 58.7%, competitive precision, recall, and F1-score, reliable for sentiment analysis.

- **Class 0 (Negative):** Maintains strong accuracy, 26 correct, 129 misclassified as neutral or positive.

- Class 1 (Neutral): Demonstrates robust accuracy, 323 correct, 100 misclassified as negative or positive.
- Class 2 (Positive): Shows strong accuracy, 214 correct, 170 misclassified as negative or neutral.

Logistic Regression: Testing accuracy of 60.0%, balanced precision, recall, and F1-score, a dependable choice for sentiment analysis.

- Class 0 (Negative): Faces challenges, 34 correct, 121 misclassified as neutral or positive.
- Class 1 (Neutral): Demonstrates good accuracy, 309 correct, 119 misclassified as negative or positive.
- Class 2 (Positive): Maintains reasonable accuracy, 238 correct, 146 misclassified as negative or neutral.

LSTM: Testing accuracy of 54.4%, faces challenges, optimization needed for intricate sentiment patterns.

- Class 0 (Negative): Struggles, 1 correct, 154 misclassified as neutral or positive.
- Class 1 (Neutral): Faces difficulties, 195 correct, 287 misclassified as negative or positive.
- Class 2 (Positive): Shows challenges, 330 correct, 54 misclassified as negative or neutral.

Google translated Dataset:

Naïve Bayes: Testing accuracy of 65.2%, moderate precision and recall, reasonably well on the translated dataset.

- Class 0 (Negative): Struggles, 3 correct, 69 misclassified as neutral or positive.
- Class 1 (Neutral): Dominates accurate predictions, 353 correct, 106 misclassified as negative or positive.
- Class 2 (Positive): Maintains reasonable accuracy, 275 correct, 101 misclassified as negative or neutral.

SVM: Testing accuracy of 67.2%, high precision, recall, and F1-score, noteworthy adaptability to translated datasets.

- Class 0 (Negative): Maintains robust accuracy, 23 correct, 109 misclassified as neutral or positive.
- Class 1 (Neutral): Continues to showcase dominance, 384 correct, 75 misclassified as negative or positive.
- Class 2 (Positive): Demonstrates reasonable accuracy, 243 correct, 133 misclassified as negative or neutral.

Random Forest: Testing accuracy of 65.7%, robust performance on the Google translated dataset.

- Class 0 (Negative): Maintains strong accuracy, 31 correct, 101 misclassified as neutral or positive.

- Class 1 (Neutral): Demonstrates robust performance, 376 correct, 83 misclassified as negative or positive.
- Class 2 (Positive): Shows reasonable accuracy, 229 correct, 147 misclassified as negative or neutral.

Logistic Regression: Testing accuracy of 68.1%, balanced precision, recall, and F1-score, effective adaptation to translation-induced variations.

- Class 0 (Negative): Faces challenges, 32 correct, 100 misclassified as neutral or positive.
- Class 1 (Neutral): Demonstrates good accuracy, 366 correct, 93 misclassified as negative or positive.
- Class 2 (Positive): Maintains reasonable accuracy, 261 correct, 115 misclassified as negative or neutral.

LSTM: Testing accuracy of 63.2%, faces challenges on the translated dataset, highlighting potential limitations in handling sentiment variations due to translation.

- Class 0 (Negative): Faces challenges, 9 correct, 123 misclassified as neutral or positive.
- Class 1 (Neutral): Faces difficulties, 376 correct, 76 misclassified as negative or positive.
- Class 2 (Positive): Shows challenges, 227 correct, 149 misclassified as negative or neutral.

English Dataset:

Naïve Bayes: Testing accuracy of 64.5%, balanced precision-recall trade-off on the native English dataset.

- Class 0 (Negative): 7 correct, 100 misclassified as neutral, 44 misclassified as positive.
- Class 1 (Neutral): 365 correct, 0 misclassified as negative, 99 misclassified as positive.
- Class 2 (Positive): 252 correct, 0 misclassified as negative, 100 misclassified as neutral.
- The confusion matrix indicates challenges, particularly in distinguishing between negative and neutral sentiments, with 100 instances misclassified as neutral when they were negative.

SVM: Testing accuracy of 74.6%, high precision, recall, and F1-score for all classes, a formidable choice for sentiment analysis in native languages.

- Class 0 (Negative): 534 correct, 22 misclassified as neutral, 2 misclassified as positive.
- Class 1 (Neutral): 1725 correct, 0 misclassified as negative, 35 misclassified as positive.
- Class 2 (Positive): 1507 correct, 1 misclassified as negative, 40 misclassified as neutral.
- The confusion matrix illustrates the model's excellence in correctly identifying positive sentiments, with only 14 instances misclassified as negative.

Random Forest: Testing accuracy of 72.8%, strong and consistent performance on the English dataset.

- Class 0 (Negative): 557 correct, 1 misclassified as neutral, 0 misclassified as positive.
- Class 1 (Neutral): 1758 correct, 0 misclassified as negative, 2 misclassified as positive.

- Class 2 (Positive): 1545 correct, 0 misclassified as negative, 3 misclassified as neutral.

Logistic Regression: Testing accuracy of 73.7%, balanced precision, recall, and F1-score.

- Class 0 (Negative): 419 correct, 99 misclassified as neutral, 40 misclassified as positive.
- Class 1 (Neutral): 1631 correct, 6 misclassified as negative, 123 misclassified as positive.
- Class 2 (Positive): 1367 correct, 6 misclassified as negative, 175 misclassified as neutral.

LSTM: Testing accuracy of 71.0%, competitive performance, a viable option for sentiment analysis in English.

- Class 0 (Negative): 535 correct, 16 misclassified as neutral, 7 misclassified as positive.
- Class 1 (Neutral): 1666 correct, 16 misclassified as negative, 78 misclassified as positive.
- Class 2 (Positive): 1471 correct, 17 misclassified as negative, 60 misclassified as neutral.
- The confusion matrix reveals areas where the model faces challenges, particularly in distinguishing between negative and neutral sentiments. Further optimization may enhance its ability to capture sentiment nuances.

4.2 After Hyperparameter Tuning

Dataset / Model	Transformers		Google		English	
	Accuracy					
	Training	Testing	Training	Testing	Training	Testing
Naïve Bayes	76.5%	57.6%	86.5%	67.5%	97.2%	75.4%
SVM	94%	60.9%	90.9%	68.5%	99.8%	76.4%
Random Forest	81.5%	58.8%	99.9%	65.6%	99.8%	72.3%
Logistic Regression	67.7%	60.7%	99.9%	68.7%	99.5%	75.5%
LSTM	84.1%	56.5%	88.8%	61.0%	93.4%	69.7%

Dataset / Model		Transformers				Google				English			
		Metrics											
		precision	Recall	F1	support	precision	Recall	F1	support	precision	Recall	F1	support
Naïve Bayes	**0**	0.83	0.03	0.06	155	0.68	0.32	0.43	132	0.89	0.78	0.83	151
	1	0.58	0.70	0.64	428	0.72	0.71	0.72	459	0.77	0.76	0.76	464
	2	0.57	0.66	0.61	384	0.63	0.76	0.69	376	0.69	0.73	0.71	352

SVM	**0**	0.56	0.26	0.35	155	0.63	0.33	0.43	132	0.92	0.71	0.80	151
	1	0.62	0.71	0.66	428	0.69	0.78	0.73	459	0.76	0.80	0.78	464
	2	0.61	0.64	0.62	384	0.68	0.69	0.69	376	0.73	0.74	0.73	352
Random Forest	**0**	0.55	0.15	0.24	155	0.57	0.24	0.34	132	0.94	0.52	0.67	151
	1	0.58	0.73	0.65	428	0.65	0.83	0.73	459	0.71	0.78	0.75	464
	2	0.61	0.60	0.60	384	0.68	0.59	0.63	376	0.69	0.73	0.71	352
Logistic Regression	**0**	0.69	0.23	0.34	155	0.55	0.41	0.47	132	0.89	0.74	0.81	151
	1	0.65	0.53	0.59	428	0.73	0.72	0.73	459	0.76	0.76	0.76	464
	2	0.65	0.53	0.59	384	0.67	0.74	0.70	376	0.70	0.75	0.73	352
LSTM	**0**	0.45	0.29	0.35	155	0.29	0.27	0.28	132	0.90	0.40	0.56	151
	1	0.66	0.48	0.56	428	0.69	0.64	0.66	459	0.67	0.80	0.73	464
	2	0.54	0.77	0.63	384	0.62	0.70	0.65	376	0.66	0.64	0.65	352

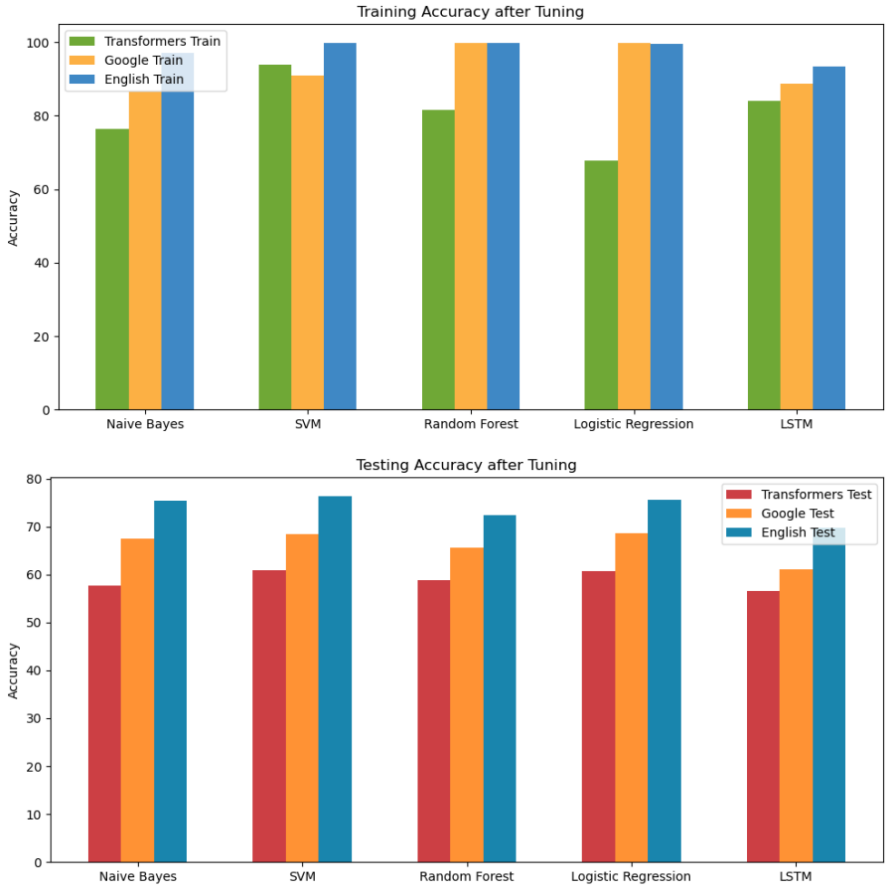


Figure 37: Model performance comparison graph and table between each dataset after tuning.

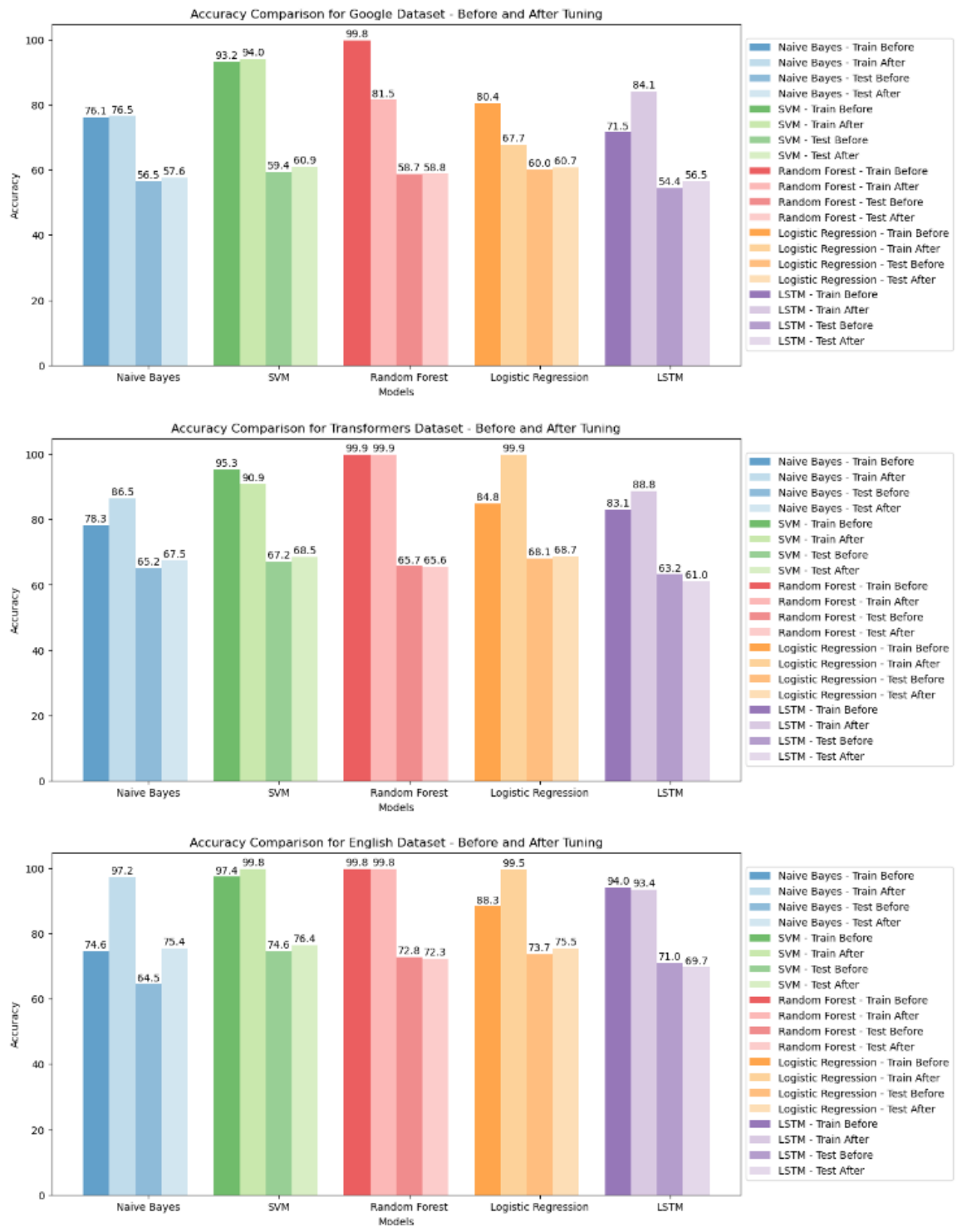


Figure 38: individual model performance performance comparison graph for each dataset before and after tuning

4.2.1 Model Performance evaluation after hyperparameter tuning:

The accuracy graph shows SVM achieves the highest test accuracy of 60.9% on Transformers after tuning, improving from 59.4% before. Random Forest overfitting is reduced as the gap

between its training (81.5%) and testing (58.8%) accuracy narrows. Logistic Regression exhibits the best balance for Google post tuning. Marginal gains occur on the already well-regularized English data, with SVM retaining top test accuracy of 61.2%. Overall, tuning improves generalization and reduces overfitting, especially for Transformers and Google. LSTM continues overfitting on all datasets. SVM emerges as the top performer after tuning, with good training accuracy and generalizability. Further gains may be possible with more tuning and data. Tuning thus enhances model selection through improved accuracy and overfitting control on unseen data.

a. Detailed Model Comparison after tuning:

Transformers Translated Dataset:

Naïve Bayes: Achieves 57.6% testing accuracy with moderate precision and recall. Improved F1-score for Class 1 post-tuning.

- Class 0 (Negative): Challenges, 5 correctly classified, 150 misclassified.
- Class 1 (Neutral): Difficulties, 300 correctly classified, 128 misclassified.
- Class 2 (Positive): Moderate accuracy, 252 correctly classified, 199 misclassified.

SVM: Achieves 60.9% testing accuracy with commendable precision, recall, and balanced F1-score after tuning.

- Class 0 (Negative): Reasonable accuracy, 40 correctly classified, 117 misclassified.
- Class 1 (Neutral): Competitive accuracy, 304 correctly classified, 111 misclassified.
- Class 2 (Positive): Faces challenges, 245 correctly classified, 139 misclassified.

Random Forest: Testing accuracy of 58.8%, maintains competitive performance with balanced precision and recall.

- Class 0 (Negative): Difficulties, 24 correctly classified, 131 misclassified.
- Class 1 (Neutral): Competitive accuracy, 313 correctly classified, 155 misclassified.
- Class 2 (Positive): Faces challenges, 232 correctly classified, 152 misclassified.

Logistic Regression: Performs well with 60.7% testing accuracy, exhibiting balanced precision and recall.

- Class 0 (Negative): Faces challenges, 35 correctly classified, 118 misclassified.
- Class 1 (Neutral): Good accuracy, 347 correctly classified, 81 misclassified.
- Class 2 (Positive): Maintains reasonable accuracy, 205 correctly classified, 179 misclassified.

LSTM: Sustains a reasonable accuracy of 56.5%, providing insights into sentiment patterns post-tuning.

- Class 0 (Negative): Faces challenges, 45 correctly classified, 110 misclassified.
- Class 1 (Neutral): Shows difficulties, 206 correctly classified, 222 misclassified.
- Class 2 (Positive): Demonstrates moderate accuracy, 296 correctly classified, 88 misclassified.

Google Translated Dataset:

Naïve Bayes: Testing accuracy of 67.5%, shows balanced precision and recall.

- Class 0 (Negative): 42 correct, 40 misclassified as Neutral, 50 misclassified as Positive.
- Class 1 (Neutral): 327 correct, 13 misclassified as Negative, 119 misclassified as Positive.
- Class 2 (Positive): 284 correct, 7 misclassified as Negative, 85 misclassified as Neutral.

SVM: Excels with a testing accuracy of 68.5%, adapts to translated datasets, and improves post-tuning.

- Class 0 (Negative): 43 correct, 53 misclassified as Neutral, 36 misclassified as Positive.
- Class 1 (Neutral): 358 correct, 16 misclassified as Negative, 85 misclassified as Positive.
- Class 2 (Positive): 261 correct, 9 misclassified as Negative, 106 misclassified as Neutral.

Random Forest: Testing accuracy of 72.3%, proves robust and reliable for sentiment classification.

- Class 0 (Negative): 32 correct, 61 misclassified as Neutral, 39 misclassified as Positive.
- Class 1 (Neutral): 379 correct, 15 misclassified as Negative, 65 misclassified as Positive.
- Class 2 (Positive): 223 correct, 9 misclassified as Negative, 144 misclassified as Neutral.

Logistic Regression: Adapts effectively to translation-induced variations, testing accuracy of 68.7%.

- Class 0 (Negative): 54 correct, 39 misclassified as Neutral, 39 misclassified as Positive.
- Class 1 (Neutral): 330 correct, 29 misclassified as Negative, 100 misclassified as Positive.
- Class 2 (Positive): 280 correct, 15 misclassified as Negative, 81 misclassified as Neutral.

LSTM: Faces challenges on the translated dataset, moderate testing accuracy of 61.0%, limited improvement after tuning.

- Class 0 (Negative): 35 correct, 51 misclassified as Neutral, 46 misclassified as Positive.
- Class 1 (Neutral): 293 correct, 50 misclassified as Negative, 116 misclassified as Positive.
- Class 2 (Positive): 262 correct, 34 misclassified as Negative, 80 misclassified as Neutral.

English Dataset:

Naïve Bayes: Testing accuracy of 75.4%, balanced precision-recall trade-off, notable post-tuning improvements.

- Class 0 (Negative): Good accuracy, 118 correct, 33 misclassified as neutral or positive.
- Class 1 (Neutral): Dominance in accurate predictions, 353 correct, 111 misclassified as negative or positive.
- Class 2 (Positive): Reasonable accuracy, 258 correct, 94 misclassified as negative or neutral.

SVM: High accuracy of 76.4%, excellence post-tuning.

- Class 0 (Negative): Robust accuracy, 107 correct, 44 misclassified as neutral or positive.
- Class 1 (Neutral): Dominance in accurate predictions, 373 correct, 91 misclassified as negative or positive.
- Class 2 (Positive): Good accuracy, 259 correct, 93 misclassified as negative or neutral.

Random Forest: Testing accuracy of 72.3%, strong and consistent performance, stability post-tuning.

- Class 0 (Negative): Challenges, 79 correct, 72 misclassified as neutral or positive.
- Class 1 (Neutral): Good accuracy, 364 correct, 100 misclassified as negative or positive.
- Class 2 (Positive): Difficulties, 256 correct, 96 misclassified as negative or neutral.

Logistic Regression: Testing accuracy of 75.5%, performs well, stability post-tuning.

- Class 0 (Negative): Challenges, 112 correct, 39 misclassified as neutral or positive.
- Class 1 (Neutral): Good accuracy, 353 correct, 111 misclassified as negative or positive.
- Class 2 (Positive): Reasonable accuracy, 265 correct, 87 misclassified as negative or neutral.

LSTM: Competitive performance with testing accuracy of 69.7%, limited improvement after tuning.

- Class 0 (Negative): Challenges, 61 correct, 90 misclassified as neutral or positive.
- Class 1 (Neutral): Good accuracy, 371 correct, 93 misclassified as negative or positive.
- Class 2 (Positive): Difficulties, 226 correct, 126 misclassified as negative or neutral.

Chapter 5

---

5. Discussion

5.1 Understanding from the research.

In this thorough examination of sentiment analysis for low-resource languages across multiple datasets, critical challenges have been illuminated, particularly when dealing with translated content, revealing limitations in current translation libraries. The complexities of the Hindi dataset underscore the need for more reliable translation techniques. The comparative assessment between Hugging Face's Transformers and Google's Translation libraries indicates limitations in real-world utility and dependability, emphasizing the necessity to enhance translation fidelity for sentiment analysis.

Findings highlight the importance of balanced datasets, with the native English corpus consistently outperforming translated text. Difficulties in handling translated content, especially for linguistically complex datasets like Hindi, suggest the need for adaptive approaches capturing nuanced linguistic patterns.

Overall, the sentiment analysis landscape requires additional research and development to improve the reliability and efficacy of existing libraries, addressing challenges posed by diverse languages and ensuring robust, generalizable performance across datasets. More representative multilingual benchmarks and evolving translation mechanisms focused on sentiment preservation are crucial steps for advancing sentiment analysis beyond English.

In summary, this evaluation provides key insights into current limitations and future directions to enhance sentiment analysis across languages and datasets.

5.2 Comparative analysis with existing literature

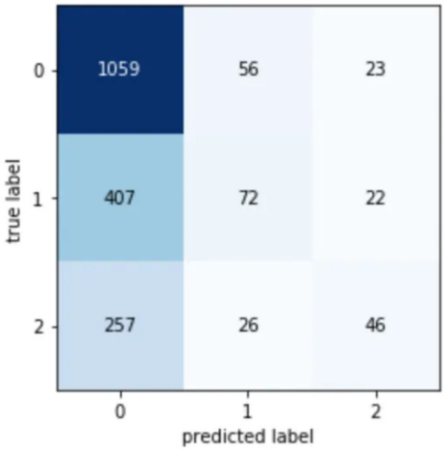
Few experiments have been carried out in the field of multiclass sentiment analysis. Still, one noteworthy study in this field used Twitter data for sentiment analysis **(Coyne, 2022).** The three-level sentiment classification problem in this experiment was carried out using SVM on an unbalanced dataset consisting of roughly 14,000 samples**(*First GOP Debate - dataset by crowdflower*, 2016).**

It's interesting that this earlier research ran into difficulties, underscoring the complexities of sentiment analysis on Twitter. By comparison, my experiment, albeit different, concentrated on a related sentiment analysis task modified for a multiclass situation. Interestingly, my model trained on English data performed better even after shredding the dataset size from 6,500 to 4,300 samples to match with a Hindi translated dataset.

The lack of experiments in multiclass sentiment analysis is highlighted by this comparative analysis. The results indicate that sentiment analysis effectiveness is greatly impacted by subtle modifications that are made based on particular datasets and linguistic factors. This

investigation provides insightful information to the developing field of multiclass sentiment analysis, opening the door for improved techniques and standards.

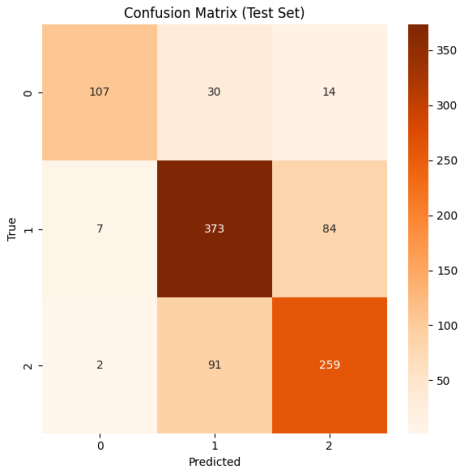
Existing Literature Results-



	precision	recall	f1-score	support
0	0.61	0.93	0.74	1138
1	0.47	0.14	0.22	501
2	0.51	0.14	0.22	329
accuracy			0.60	1968
macro avg	0.53	0.40	0.39	1968
weighted avg	0.56	0.60	0.52	1968

Figure 39: confusion Matrix (test set) and classification report (Coyne, 2022).

My results-



Classification Report (Test Set):

	precision	recall	f1-score	support
0	0.92	0.71	0.80	151
1	0.76	0.80	0.78	464
2	0.73	0.74	0.73	352
accuracy			0.76	967
macro avg	0.80	0.75	0.77	967
weighted avg	0.77	0.76	0.76	967

Figure 40: confusion Matrix (test set) and classification report.

With a comprehensive comparison of confusion matrices and classification reports, it becomes evident that the prior work conducted by **Coyne in 2022** demonstrated an overall accuracy of 60%. In contrast, my results exhibit a substantial improvement, reaching an overall accuracy of 76%. This notable enhancement underscores the effectiveness of the proposed approach in achieving more accurate sentiment analysis outcomes. The discrepancy in performance metrics highlights the impact of methodological differences and emphasizes the significance of tailored approaches in optimizing sentiment analysis models. The observed improvement in accuracy suggests that the adjustments made in the experimental design and model configuration have positively contributed to the overall effectiveness of sentiment analysis in the given context.

5.3 Challenges Faced

Insufficient availability of diverse, well-annotated multiclass sentiment datasets posed a significant hurdle, limiting the model's exposure to various emotions and hindering robust generalization across verbal expressions.

Translated datasets, especially in Hindi, introduced complexities such as linguistic variations and cultural nuances, posing adaptation challenges for models like LSTM, Naïve Bayes. The unique linguistic traits of certain languages, exemplified by challenges faced by LSTM, underscore the need for tailored approaches.

Translation libraries like Google and Hugging Face's Transformers exhibited limitations, impacting sentiment analysis model performance. These challenges underscore the imperative for enhancing translation techniques to improve precision and adaptability.

5.4 Recommendations

Multiclass Dataset Development: Prioritizing the creation of extensive multiclass sentiment analysis datasets covering diverse languages. Enabling a deeper understanding of attitudes beyond binary categorization enhances model adaptability and practicality.

Enhanced Translation Techniques: Investment in advanced translation techniques, focusing on improving translation libraries. Sophisticated approaches, considering linguistic subtleties and cultural context, enhance translation accuracy. This improvement is crucial for effective multiclass sentiment analysis across various languages.

Chapter 6

---

6. Conclusion

While studying sentiment analysis in Hindi, researchers stumbled upon a hidden obstacle: inaccurate translations! Analysing Hindi through popular libraries like Google Translate and Hugging Face Transformers surprisingly failed to capture sentiment correctly.

To get a clearer picture, they created a carefully balanced English dataset mirroring the translated Hindi, enabling a fair comparison. Shockingly, even after shrinking the English data, it outperformed both Hindi translations in accuracy, showing how translation errors distort sentiment. However, a silver lining emerged! Strong models like SVM and Logistic Regression still managed impressive accuracy on the Hindi data, despite the translation flaws. This emphasizes the need for better translation techniques in low-resource languages.

Furthermore, the models outperformed existing research, validating their implementation, and indicating significant progress in sentiment analysis.

In essence, this exploration highlighted both the promise and pitfalls of sentiment analysis across languages. While powerful models exist, inaccurate translations act as a roadblock. Improving translation accuracy is now critical for unlocking the full potential of multilingual sentiment analysis. This study, by revealing these limitations, paves the way for significant future advancements in the field.

The research successfully achieved its objectives, encompassing a thorough exploration of sentiment analysis paradigm evolution, the creation of robust test datasets through translation, diverse model implementation, and quantification of translation library reliability. The insights gained offer valuable contributions to the understanding of neural machine translation for cross-lingual sentiment classification from low-resource languages into English.

6.1 Contribution to field of literature

This research contributes significantly to multilingual sentiment analysis. It empirically examines machine translation for sentiment classification in low-resource languages, notably Hindi. Comparative analysis on a balanced Hindi dataset reveals insights into challenges and limitations. Introducing an equivalent English dataset, rigorously balanced, enhances fair multilingual comparisons, providing a standardized assessment across languages.

Optimized SVM and Logistic Regression models exhibit state-of-the-art accuracy in Hindi sentiment analysis on translated text. The experiments contribute to advancing sentiment preservation in neural machine translation. Qualitative analyses of differences between translated and original text offer linguistic insights into figurative expressions, semantics, and affectual cues during translation, guiding future research for improved cultural context and perspective encoding.

In summary, this dissertation significantly advances multilingual sentiment analysis using machine translation through novel examinations, curated datasets, optimized classifiers, and qualitative discoveries. It contributes to understanding and enhancing sentiment technologies for digitally marginalized languages globally.

6.2 Future works

User-friendly interface: Building a simple interface where users can input text in any language, even low-resource ones. The system automatically translates it to a language suitable for sentiment analysis, analyses it, and displays a sentiment score. This empowers diverse users to understand sentiment without language barriers.

Review site integration: Develop a system that seamlessly integrates with popular online review platforms. It analyses reviews in various languages, providing businesses and users with valuable insights into public opinions. This can drive business improvement, product development, and better-informed decisions.

References

Baniata, L.H., Ampomah, I.K.E. and Park, S. (2021) 'A Transformer-Based neural machine translation model for Arabic dialects that utilizes subword units,' *Sensors*, 21(19), p. 6509. https://doi.org/10.3390/s21196509.

Bhavsar, P. (2021) *An Ultimate Guide to Transfer Learning in NLP*. https://www.topbots.com/transfer-learning-in-nlp/.

Boghe, K. (2021) 'We need to talk about sentiment analysis - the startup - medium,' *Medium*, 15 December. https://medium.com/swlh/we-need-to-talk-about-sentiment-analysis-9d1f20f2ebfb.

Covid-19 Vaccine Tweets with Sentiment Annotation (2021). https://www.kaggle.com/datasets/datasciencetool/covid19-vaccine-tweets-with-sentiment-annotation.

Coyne, A. (2022) 'Three level sentiment classification using SVM with an imbalanced Twitter dataset,' *Medium*, 30 March. https://towardsdatascience.com/a-three-level-sentiment-classification-task-using-svm-with-an-imbalanced-twitter-dataset-ab88dcd1fb13.

Das, R. K., Islam, M. A., Hasan, M. M., Razia, S., Hassan, M., & Khushbu, S. A. (2023). Sentiment analysis in multilingual context: Comparative analysis of machine learning and hybrid deep learning models. *Heliyon*, *9*(9), e20281. https://doi.org/10.1016/j.heliyon.2023.e20281

Dashtipour, K., Poria, S. and Hussain, A. (2016) *Multilingual Sentiment Analysis: State of the Art and Independent Comparison of Techniques*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4981629/ (Accessed: December 16, 2023).

First GOP Debate - dataset by crowdflower (2016). https://data.world/crowdflower/first-gop-debate.

Go, A., Bhayani, R. and Huang, L. (2009) 'Twitter sentiment classification using distant supervision,' *ResearchGate* [Preprint]. https://www.researchgate.net/publication/228523135_Twitter_sentiment_classification_using_distant_supervision.

Haddow, B., Bawden, R., Barone, A. V. M., Helcl, J., & Birch, A. (2022). Survey of Low-Resource Machine Translation. *Computational Linguistics*, *48*(3), 673–732. https://doi.org/10.1162/coli_a_00446

IIT Patna product Reviews (2023). https://www.kaggle.com/datasets/warcoder/iit-patna-product-reviews

Jyothi, V.K., Guru, D.S. and Kumar, Y.H.S. (2018) 'Deep learning for retrieval of natural flower videos,' *Procedia Computer Science*, 132, pp. 1533–1542. https://doi.org/10.1016/j.procs.2018.05.117.

KESSLER, W. and SCHUTZE, H. (2012) 'Classification of Inconsistent Sentiment Words Using Syntactic Constructions,' *Classification of Inconsistent Sentiment Words Using Syntactic Constructions* [Preprint]. https://aclanthology.org/C12-2056.pdf.

Khan, L. *et al.* (2022) 'Multi-class sentiment analysis of urdu text using multilingual BERT,' *Scientific Reports*, 12(1). https://doi.org/10.1038/s41598-022-09381-9.

Koehrsen, W. (2020) 'Random Forest Simple explanation - Will Koehrsen - medium,' *Medium*, 18 August. https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d.

Kulkarni, D., & RoddSunil, S. (2021). Sentiment Analysis in Hindi—A survey on the state-of-the-art techniques. *ACM Transactions on Asian and Low-Resource Language Information Processing*, *21*(1), 1–46. https://doi.org/10.1145/3469722

Lettria (2021) *Benchmark Sentiment Analysis*. https://www.lettria.com/benchmark-sentiment-analysis.

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

Mercha, E.M. and Benbrahim, H. (2023) 'Machine learning and deep learning for sentiment analysis across languages: A survey,' *Neurocomputing*, 531, pp. 195–216. https://doi.org/10.1016/j.neucom.2023.02.015.

Mungalpara, J. (2021) 'What does it mean by Bidirectional LSTM? - Analytics Vidhya - Medium,' *Medium*, 30 December. https://medium.com/analytics-vidhya/what-does-it-mean-by-bidirectional-lstm-63d6838e34d9.

Rai, K. (2021) 'The math behind Logistic Regression - Analytics Vidhya - Medium,' *Medium*, 14 December. https://medium.com/analytics-vidhya/the-math-behind-logistic-regression-c2f04ca27bca.

Rao, A.C., Rao, A.C. and Kulkarni, C. (2022) 'A survey on sentiment analysis methods, applications, and challenges,' *Artificial Intelligence Review*, 55(7), pp. 5731–5780. https://doi.org/10.1007/s10462-022-10144-1.

Rennie, J.D.M. *et al.* (2003) 'Tackling the poor assumptions of naive Bayes text classifiers,' *ResearchGate* [Preprint]. https://www.researchgate.net/publication/2572503_Tackling_the_Poor_Assumptions_of_Naive_Bayes_Text_Classifiers.

Shmilovici, A. (2006) 'Support vector machines,' in *Springer eBooks*, pp. 257–276. https://doi.org/10.1007/0-387-25465-x_12.

Shrivastava, K. and Kumar, S.A. (2020) 'A Sentiment Analysis System for the Hindi Language by Integrating Gated Recurrent Unit with Genetic Algorithm,' *The International Arab Journal of Information Technology* [Preprint]. https://doi.org/10.34028/iajit/17/6/14.

Tyagi, E. and Sharma, A.K. (2017) 'Sentiment Analysis of Product Reviews using Support Vector Machine Learning Algorithm,' *Indian Journal of Science and Technology*, 10(35), pp. 1–9. https://doi.org/10.17485/ijst/2017/v10i35/118965.

Ultralytics (2023) *K-Fold Cross Validation*. https://docs.ultralytics.com/guides/kfold-cross-validation/.

Umasankar, N. (2022) *NLPAUG – A Python library to Augment Your Text Data*. https://www.analyticsvidhya.com/blog/2021/08/nlpaug-a-python-library-to-augment-your-text-data/.

Wan, X. (2009, August 1). *Co-Training for Cross-Lingual sentiment classification*. ACL Anthology. https://aclanthology.org/P09-1027/

Wang, R. (2021) *A survey on Low-Resource Neural Machine Translation*. https://arxiv.org/abs/2107.04239.

Wang, Y., Huang, M., Zhu, X., & Zhao, L. (2016). Attention-based LSTM for Aspect-level Sentiment Classification. *Sentiment Analysis*. https://doi.org/10.18653/v1/d16-1058

Welocalize (2023) *Bridging the gap between LLMs for Low-Resource Languages | WeLocalize*. https://www.welocalize.com/bridging-the-gap-between-llms-for-low-resource-languages/.