

# **Sentiment Analysis on Code-Mixing: A Study On Malaysian's Social Media Content**

**Fung Jie Deng**

**XIAMEN UNIVERSITY MALAYSIA**

**2023**



**XIAMEN UNIVERSITY MALAYSIA**  
**廈門大學 馬來西亞分校**

**FINAL YEAR PROJECT REPORT**


**SENTIMENT ANALYSIS ON CODE MIXING:  
A STUDY ON MALAYSIAN'S SOCIAL MEDIA  
CONTENT**

NAME OF STUDENT : FUNG JIE DENG  
STUDENT ID : SWE1909758  
SCHOOL/ FACULTY : SCHOOL OF COMPUTING AND DATA  
SCIENCE  
PROGRAMME : BACHELOR OF ENGINEERING IN  
SOFTWARE ENGINEERING  
(HONOURS)  
INTAKE : 2019/09  
SUPERVISOR : DR SHAIDAH BINTI JUSOH  
ASSISTANT PROFESSOR

**AUGUST 2023**

## DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at Xiamen University Malaysia or other institutions.

Signature : \_\_\_\_\_

Name : Fung Jie Deng\_\_\_\_\_

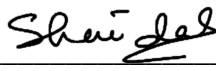
ID No. : SWE1909758\_\_\_\_\_

Date : 02/07/2023\_\_\_\_\_

## APPROVAL FOR SUBMISSION

I certify that this project report entitled “SENTIMENT ANALYSIS ON CODE MIXING: A STUDY ON MALAYSIAN’S SOCIAL MEDIA CONTENT” that was prepared by FUNG JIE DENG has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering in Software Engineering (Honours) at Xiamen University Malaysia.

Approved by,

Signature : 

Supervisor : Dr Shaidah Binti Jusoh

Date : 2 July 2023

The copyright of this report belongs to the author under the terms of Xiamen University Malaysia copyright policy. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this project report/ thesis.

©2023, Fung Jie Deng. All right reserved.

## **ACKNOWLEDGEMENTS**

I would like to thank all who have contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Prof. Dr. Shaidah Binti Jusoh for her invaluable advice, guidance and her enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who have helped and given me encouragement.

## ABSTRACT

In the recent years, sentiment analysis has gained quite a significant traction across various domains, including online reviews, social media, customer support, etc. However, sentiment analysis on code-mixed data, where multiple languages are used within a single sentence, presents a unique challenge for the task. The realized research gap had instigated the aim for this thesis research, which is to focus on conducting sentiment analysis on code-mixed data involving the English and Malay languages. Various machine learning models, including on Naive Bayes, Linear Support Vector Classification (LSVC), and also logistic regression are employed to scrutinize on the sentiment of the code-mixed English-Malay data. Each model is language-specific, and they're trained and implemented for both the English and Malay languages. Additionally, Bidirectional Encoder Representations from Transformers (BERT) models, notorious for their powerful contextual understanding, are also employed as two language-specific models for the sentiment analysis establishment. To optimize on the sentiment classification accuracy, an ensemble/integrated model is being proposed. The said ensemble model will combine the outputs of the language-specific machine learning models and BERT models through a voting system, and it gathers the collective decision of each model and leverages their individual strengths to enhance sentiment classification on code-mixed data. Experimental predictions are then conducted on a code-mixed dataset comprising of English and Malay languages. The performance of each individual model, as well as the ensemble model, is evaluated in terms of several evaluation metrics, including on accuracy, precision, recall, and F1-score. The output results are compared with their strengths and weaknesses, which the concluded comparison demonstrates on the effectiveness of the ensemble model in improving sentiment classification accuracy compared to individual models. To conclude, this conducted research explores on the sentiment analysis on code-mixed data and addresses the challenges associated with the English and Malay languages. By combining on machine learning models and BERT models through an ensemble approach, the proposed methodology achieves optimized sentiment classification accuracy for code-mixed datasets.

**Keywords:** Sentiment Analysis, Code-mixing, Machine Learning Models, Deep Learning Models, BERT Models

## TABLE OF CONTENTS

<b>DECLARATION</b>	<b>ii</b>
<b>APPROVAL FOR SUBMISSION</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF TABLES</b>	<b>viii</b>
<b>LIST OF ALGORITHMS</b>	<b>x</b>
<b>LIST OF FIGURES</b>	<b>xi</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xiii</b>

## CHAPTER

<b>1. INTRODUCTION</b>	<b>1</b>
1.1 Overview	<b>1</b>
1.2 Problem Statement	<b>4</b>
1.3 Research Questions	<b>5</b>
1.4 Aim and Objectives	<b>5</b>
1.5 Project Scope	<b>6</b>
1.6 Contributions	<b>7</b>
1.7 Thesis organisation	<b>7</b>
<b>2. LITERATURE REVIEW</b>	<b>9</b>
2.1 Sentiment Analysis on Code-Mixed Data	<b>13</b>
2.2 Data Pre-processing	<b>14</b>
2.3 Feature Extraction Technique	<b>15</b>
2.4 Sentiment Analysis Approaches/Methods	<b>18</b>
2.4.1 LR1: Decision Tree, Random Forest, Logistic Regression, & Naïve Bayes	<b>18</b>
2.4.2 LR2: BERT Models	<b>20</b>
2.4.3 LR3: Bidirectional Long Short-Term Memory (Bi-LSTM)	<b>21</b>
2.4.4 LR4: Support Vector Machine (SVM) & Naïve Bayes	<b>22</b>
2.4.5 LR5: Support Vector Machine (SVM) & Naïve Bayes	<b>23</b>



2.5 Conclusion	24
<b>3. METHODOLOGY</b>	<b>26</b>
3.1 Research Methodology Framework	26
3.2 Data Compositions	27
3.2.1 Data Collection	28
3.2.2 Data Analysis	29
3.2.3 Data Pre-Processing	33
3.3 Sentiment Analysis Models	35
3.3.1 Model Selection	36
3.3.2 Model Implementation	42
3.4 Evaluation Metrics	44
<b>4. EXPERIMENT AND IMPLEMENTATIONS</b>	<b>48</b>
4.1 Implement Machine Learning Models	48
4.2 Implement BERT Models	56
4.3 Proposed Code-mixed Sentiment Classification Model	63
4.4 Conclusion	70
<b>5. RESULTS AND DISCUSSION</b>	<b>72</b>
5.1 Results of Machine Learning Models	72
5.2 Results of BERT Models	83
5.3 Result of Proposed Ensemble/Integrated Model	86
5.4 Discussions	88
5.4.1 Comparison of Individual Models	89
5.5 Conclusion	91
<b>6. RESEARCH CONCLUSION AND FUTURE WORK</b>	<b>93</b>
6.1 Research Conclusion	94
6.2 Future Work	95
6.3 Current Working Prototype	95
6.4 Limitations	102
6.5 Research Achievements	103
<b>REFERENCES</b>	<b>106</b>

## LIST OF TABLES

<b>Table 1.1:</b> General classification to Sentiment Analysis	2
<b>Table 2.1:</b> Related Works on Code-Mixing Sentiment Analysis	11
<b>Table 3.1:</b> S140 English 10 Random Data Samples	30
<b>Table 3.2:</b> Malay Dataset 10 Random Data Samples	32
<b>Table 3.3:</b> Dataframe layout for data consistency	34
<b>Table 3.4:</b> Stopwords example for English and Malay	35
<b>Table 3.5:</b> Models intended to be implemented	42
<b>Table 3.6:</b> Structure for Confusion Matrix	44
<b>Table 5.1:</b> Summary of Eng machine learning model's split test validation result	74
<b>Table 5.2:</b> Summary of BM machine learning model's split test validation result	77
<b>Table 5.3:</b> Summary of Eng machine learning model's ground-truth validation result	80
<b>Table 5.4:</b> Summary of BM machine learning model's ground-truth validation result	82
<b>Table 5.5:</b> Summary of both BERT model's split test data validation result	84
<b>Table 5.6:</b> Summary of both BERT model's ground-truth data validation result	86
<b>Table 5.7:</b> Summary of Ensemble model's ground-truth data validation result	87
<b>Table 5.8:</b> Sentiment Accuracy of each implemented model	89

## LIST OF ALGORITHMS

<b>Algorithm 4.1:</b> data Preprocessing	53
<b>Algorithm 4.2:</b> TF-IDF vectorizer	55
<b>Algorithm 4.3:</b> Train models	55
<b>Algorithm 4.4:</b> Evaluation models	56
<b>Algorithm 4.5:</b> Fine tune bert model	62
<b>Algorithm 4.6:</b> Translation	66
<b>Algorithm 4.7:</b> Voting system	69

## LIST OF FIGURES

<b>Figure 1.1:</b> Approaches in establishing SA (Medhat et al., 2014)	3
<b>Figure 3.1:</b> Established Research Methodology Framework	26
<b>Figure 3.2:</b> S140 English dataset distribution	30
<b>Figure 3.3:</b> Malay dataset distribution	31
<b>Figure 3.4(a):</b> Malay Distribution of data before down-sampling	33
<b>Figure 3.4(b):</b> Malay Distribution of data after down-sampling	33
<b>Figure 3.5:</b> Concept of Linear Support Vector Machine	38
<b>Figure 3.6:</b> Concept of Logistic Regression	40
<b>Figure 3.7:</b> Overall pre-training and fine-tuning procedure for BERT	42
<b>Figure 4.1:</b> Implementation Framework for machine learning models	49
<b>Figure 4.2:</b> Loading dataset to panda's dataframe	50
<b>Figure 4.3:</b> Randomize dataset with sample function	51
<b>Figure 4.4(a):</b> Visualize english dataset class distribution	51
<b>Figure 4.4(b):</b> Visualize malay dataset class distribution	52
<b>Figure 4.5:</b> Splitting data to 80:20 train/test	54
<b>Figure 4.6:</b> Implementation Framework for BERT Models	57
<b>Figure 4.7:</b> BERT autoTokenizer and special tokens	59
<b>Figure 4.8:</b> BERT data split and dataloaders	61
<b>Figure 4.9:</b> Implementation Framework of the Ensemble/Integration Model	64
<b>Figure 5.1:</b> Split Test data validation result on English Gaussian Naive Bayes	73
<b>Figure 5.2:</b> Split test data validation result on English LSVC	73
<b>Figure 5.3:</b> Split test data validation result on English Logistic Regression	74
<b>Figure 5.4:</b> Split Test data validation result on Malay Gaussian Naive Bayes	75
<b>Figure 5.5:</b> Split Test data validation result on Malay LSVC	76
<b>Figure 5.6:</b> Split Test data validation result on Malay Logistic Regression	76
<b>Figure 5.7:</b> Ground-truth data validation result on English Gaussian Naive Bayes	78
<b>Figure 5.8:</b> Ground-truth data validation result on English LSVC	79
<b>Figure 5.9:</b> Ground-truth data validation result on English Logistic Regression	79
<b>Figure 5.10:</b> Ground-truth data validation result on Malay Gaussian Naive Bayes	81
<b>Figure 5.11:</b> Ground-truth data validation result on Malay LSVC	81
<b>Figure 5.12:</b> Ground-truth data validation result on Malay Logistic Regression	82
<b>Figure 5.13:</b> BERT English validation result on split test data	83

<b>Figure 5.14:</b> BERT Malay validation result on split test data	84
<b>Figure 5.15:</b> BERT English validation result on ground-truth data	85
<b>Figure 5.16:</b> BERT Malay validation result on ground-truth data	85
<b>Figure 5.17:</b> Ensemble/Integration model validation result on ground-truth data	87
<b>Figure 6.1:</b> Visual representation of the working prototype	96
<b>Figure 6.2(a):</b> Output result for demo input 1	97
<b>Figure 6.2(b):</b> Output result for demo input 2	97
<b>Figure 6.2(c):</b> Output result for demo input 3	97
<b>Figure 6.2(d):</b> Output result for demo input 4	98
<b>Figure 6.3(a):</b> Output result for positive input 1	99
<b>Figure 6.3(b):</b> Output result for positive input 2	99
<b>Figure 6.3(c):</b> Output result for positive input 3	99
<b>Figure 6.3(d):</b> Output result for negative input 4	100
<b>Figure 6.3(e):</b> Output result for negative input 5	100
<b>Figure 2.3(f):</b> Output result for negative input 6	100

## LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
BERT	Bidirectional Encoders Representations from Transformers
Bi-LSTM	Bidirectional Long Short-Term Memory
BM	Bahasa Melayu
CLS	Classification
ENG	English
FP	False Positive
FN	False Negative
KL	Kuala Lumpur
LR	Logistic Regression
LSVM	Linear Support Vector Machine
LSVC	Linear Support Vector Classification
ML	Machine Learning
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
SA	Sentiment Analysis
SEP	Separator
SVC	Support Vector Classification
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
TP	True Positive
TN	True Negative
URL	Uniform Resource Locator
UNK	Unknown

# Chapter 1

## Introduction

### 1.1 Overview

With the continuous escalation within the competitive business world, it became a vital necessity for brands, companies, or even practitioner to gain an overview of a wider opinion towards particular subjects. The notion of opinion mining, or as well Sentiment Analysis (SA) can be the veracity source to cognise on a large populace's opinion towards a subject matter, where the analysis of the collected opinion can be reformed into a valuable advantage over competitors. Howbeit of that, there are instances where complications exist within such notion, such as within a multilingual community where the practice of code-mixing or code-switching is the norm.

Code Mixing is a natural phenomenon of bilingualism (Myers-Scotton, 2002), transpiring the mix of two or more languages within oral-speeches or texts. A code may be referred as a language or a dialect (Thara & Poornachandran, 2018), where a bilingual speaker who partake in such phenomenon may switch, or mix different codes (languages) under the same pronouncement, texts, or maybe even thoughts. This phenomenon can be commonly observed upon a bilingual society, especially noticeable within social media, comparatively Facebook, Twitter, Reddit, etc.

To emphasize on the prior articulation on code-mixing, Malaysia is one of the prime example which consist of its community actively partaking code-mixing within their daily conversations. According to (Ahmad Bukhari et al., 2015), Malaysians had coined upon this phenomenon, naming it as “Bahasa Rojak”, where it code-mixes various of languages, mainly between English and Bahasa Melayu, but every so often, Chinese and Tamil are also added to the mix. This phenomenon not only took a huge impact upon Malaysian's day-to-day conversation, it also influenced the way Malaysians articulate upon their wordings and format on social media. To accentuate on the influences, below shows the example of code-mixed sentences (English and Malay), taken from Twitter:

- E.g. 1: *Buku ini memang* best, everyone should read it!!
- E.g. 2: Jammed *teruk* from KL to Ipoh, *dah* stuck 2 jam kat sini

On the other hand, Sentiment Analysis (SA) is a computational study field astride to people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards an entity (Liu, 2011). It is the in process in deriving a user's emotion or feeling towards an entity, extracting from the contextual polarity of a given texts. The intent of composing SA was to perceive opinions and identify upon the expressed sentiments, in which a polarity can be based on and classified (Medhat et al., 2014). Nonetheless, some researchers would be more incline to classify sentiments into rankings or levels of granularity, such that it determines whether a given text/sentences is being Positive, Neutral, or Negative manner (Pak & Paroubek, 2011).

Plenty of research works are being held towards the field of sentiment analysis, as the result gained from it will provides a useful indicator for many different purposes, an example for it is that Sentiment analysis provides a notable impact towards the marketing needs (Devika et al., 2016), where positive feedbacks and brand loyalties are significant for a business' success. In this case, social media is one of the main medium for users to post their opinions and reviews towards anything, which in turn is also the main medium to garner and collect data to be used for sentiment classification.

There are three main classifications to SA in general. Though as (Wilson et al., 2005) regards, sentiment expressions are not quite necessarily subjective in nature, however it is still significant to understand the different classifications of sentiment analysis, listed in Table 1.1:

**Table 1.1:** General classification to Sentiment Analysis

Level of Analysis	Description
Document-level SA	Considers a whole document as a basic information unit to classify sentiment on.
Sentence-level SA	Classify sentiment expressed within each sentence.
Aspect-level SA	Classify sentiment in respect to the specific aspects of entities.

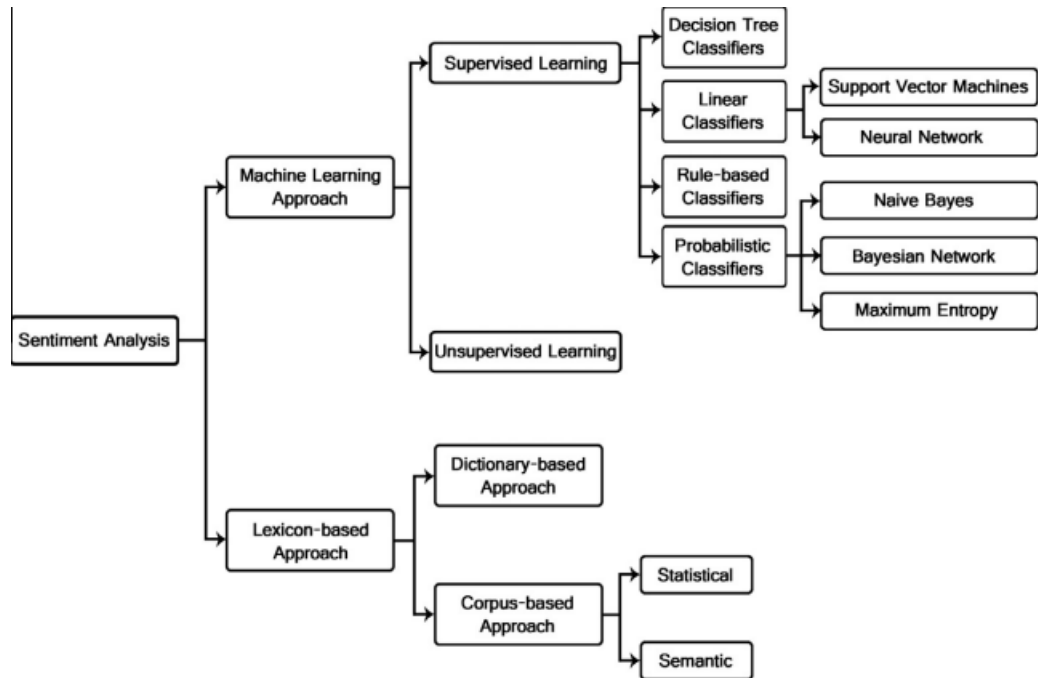
(Medhat et al., 2014)

Document-level Sentiment Analysis regards in classifying the sentiment/opinion of a whole document, where it is to assume that each document expresses sentiments as a single entity (Medhat et al., 2014). Sentence-level Sentiment Analysis on the other



hand, regards on classifying the sentiments on each sentences (Medhat et al., 2014), which relates closely to subjectivity classification, where it discerns between objective sentences that manifests factual sentiments, and subjective sentences that manifests opinionated sentiments (Devika et al., 2016). Furthermore, with the Aspect-level Sentiment Analysis, it regards in conducting a more specific analysis which discerns directly on the specific opinion itself instead of the language construct (Devika et al., 2016).

There are many different approach in imposing Sentiment Analysis, the most broadly used approach in establishing Sentiment Analysis are either through lexicon-based approach, or machine learning approach as portrayed in **Figure 1**. The machine learning approach generally requires two sets of data to be put in use: a training set, and a testing set. It utilizes on either supervised learning or unsupervised learning, and trains itself in differentiating the attributes of texts using the training dataset. Then, the performance of the classifier will be tested in regards with the testing dataset (Jain & Dandannavar, 2016). A Supervised learning classifier has been more widely used in SA tasks in contrast to unsupervised learning. Though the efficiency of the supervised classifier is severely dependant on the composition of appropriate algorithms, amid the more commonly applied algorithms such as: Naïve Bayes, Support Vector Machines, and Maximum Entropy (Agarwal & Mittal, 2015).



**Figure 1.1:** Approaches in establishing SA (Medhat et al., 2014)

Lexicon-based approach on the other hand works on detecting the opinion lexicon in conducive to calculating the sentiment of a given text (Jain & Dandannavar, 2016). This approach substantially relies on lexical resources encompassing the words and their associated sentiments in establishing the classification. It takes on calculating the overall number of positive and negative score in a text, as such:

- Overall positive sentiment > Overall negative sentiment (Positive sentiment score assigned)
- Overall positive sentiment < Overall negative sentiment (Negative sentiment score assigned)
- Overall positive sentiment = Overall negative sentiment (Neutral sentiment score assigned)

(Jain & Dandannavar, 2016)

To successfully establish in the lexicon approach, it requires a manual annotation in constructing the lexicons, which can be divided into two approaches: Dictionary-based and Corpus-based (Medhat et al., 2014). Dictionary-based approach is entirely dependent on existing available resources –*i.e.* *Wordnet*, *SentiWord*- in acquiring the opinion seed word. Corpus-based approach on the other hand, too is dependent on acquiring seed words through available resources, though it also gathers opinion seed words through a rather large domain-specific corpus (Feldman, 2013).

## **1.2 Problem Statement**

The importance of imposing Sentiment Analysis (SA) to extract the notion of public opinions has long been acknowledged by various industries. However, most conducted approach regarding SA was established in a monolingual manner, especially in English. This would cause complications in extracting public opinions from a multilingual community such as Malaysia, as they would tend to code-mix during their articulations. Establishing SA onto code-mixed data using a general monolingual models would simply serve it ineffective, resulting in an inaccurate classification of sentiments. Although the idea of implementing code-mixed SA is slowly gaining traction throughout different cultures and languages, there are however, still a lack of SA studies conducted in regards to the code-mixing nature of Malaysian's language (English – Malay). In summary, there is a need of a structured approach in imposing

SA towards Malaysian's code-mixed content, prompting and ensuring for a more accurate extraction of public sentiments in Malaysia.

### **1.3 Research Questions**

To address the problems discussed earlier, the following research questions are considered:

RQ1: How can we perform sentiment classification upon code-mixed datasets that results with a compelling outcome.

RQ2: Which Sentiment Analysis approach is optimal and more compelled to be utilised in classifying code-mixed datasets?

RQ3: To what extent of accuracy can we model and classify upon the code-mixed data using Sentiment Analysis approaches?

RQ4: How does code-mixed datasets influence upon the performance and accuracy of sentiment classifying in compared to monolingual data models?

### **1.4 Aim and Objectives**

This project and thesis yearns to apply appropriate model that imposes sentiment analysis upon Malaysian's code-mixing contents, specifically code mixed English-Malay contents on social media. The reasoning behind choosing both English and Malay language is due to the prominence mixing and usage of both throughout the Malaysian's social media community. This would help in aggregating the Malaysian's public sentiments in terms of certain topics, including political, conventional, and even controversial topics in a more coherent manner.

Thereby, in order to achieve the said aim, the research questions listed previously is in need to be answered, which can be attained through accomplishing the following objectives:

- Objective 1: To construct/collect a sufficient amount of code-mixed datasets (English – Malay) to be used for training and classification.
- Objective 2: Determine and implement optimal Sentiment Analysis models upon the collected datasets.

- Objective 3: Evaluate the performance of the determined model based on the metrics of Accuracy, Precision, Recall, and F1-score.
- Objective 4: Analyse for any inconsistencies of the model and refine it accordingly.

### **1.5 Project scope**

In spite of what is discussed earlier, most research regarding sentiment analysis is often bounded upon monolingual content, especially upon the English language. With the rate of globalisation, the widespread and usage of social media had been exponentially increased worldwide. Having a multiracial and bilingual country such as Malaysia, the community will tend to code-mix during their articulation of tweets or posts, and this phenomenon poses varying challenges and difficulties to the establishment of Natural Language Processing (NLP) and sentiment analysis (SA). The rationale for composing SA on these code-mixed text, is that the sentiment result may differ than what of prevalent monolingual English data. Sentiment analysis is mostly sensitive to the languages and domains which the training data is drawn, through code-mixing English-Malay data, the grammatical differences could result the general monolingual models ineffective, producing in an inept and inaccurate classification of sentiments. There is also some basis of SA's limitations upon code-mixing contents, which is due to the lack of code-mixed corpus and datasets (Yadav & Chakraborty, 2020). Not only that, a proportion of unseen issues would be due by the incorporation of lexicons and syntax from two or more different languages (Yadav & Chakraborty, 2020).

Due to a limited time constraint of the study, as well as having a finite knowledge upon this topic, this thesis project is purely confined under the scope of developing a workable prototype model. This thesis study emphasises on implementing Sentiment Analysis (SA) onto both code-mixed Malay and English language datasets only. The datasets to be utilized will be collected and extracted from pre-existing studies, researcher's open source datasets, or social media domains such as Twitter and Facebook. As it is observed on (Bakar et al., 2018), that the Malaysian community is relatively active in partaking and sharing their opinions on these platforms. This study intends to employs on Sentence-level Sentiment Analysis classification, achieving through with various machine learning approaches, which will be determined and evaluated on which model achieves the highest performing metrics. The ground-truth

datasets for machine learning expenditures will be collected from various lexicon sources such as SentiWordNet, or other existing lexical studies.

## **1.6 Contributions**

The outcome and findings of this current thesis project is intended to provide contributions towards analysis practitioners or the likes of scholar purposes. Unambiguously, this thesis project may benefit on practitioners that aims on obtaining the public opinions and sentiments from the Malaysian community towards any kinds of subject. To date, many use cases of Sentiment Analysis had been implemented towards products, politics, services, etc. However, there are inadequate approach that considers on the code-mixing norms within the Malaysian community, which could cause complications with the evaluation of sentiment classification. In stipulation for a fruitful result, the outcome of this thesis could contribute for a more accurate and precise sentiment aggregations for practitioners to utilize towards the Malaysian community.

Methodologically, this thesis may also benefit upon scholars or academicians that intended to work on similar subjects. A substantial contribution is on the novel approach of collecting and implementing the (Malay-English) code-mixed datasets, how machine learning approaches are applicable to them, and the evaluation of which machine learning approaches performs significantly compared to each other. The outcome of this thesis could also contribute towards the progression of multilingual/code-mixed Sentiment Analysis, where it attests that the negligence of code-mixing factor could influence upon the classification of conventional Sentiment Analysis model.

## **1.7 Thesis organisation**

This thesis will be organised into six chapters. Excluding upon this current chapter, the remainder of the thesis is organised as follows. Chapter 2 probes on the literature review, which provides a detailed description on the relevant background of the thesis topic, as well as providing a comprehensive review on related previous studies. The identified literature resources are enquired upon on its Sentiment Analysis approaches and the implementation techniques that fulfils to a successful Sentiment Analysis

model. Chapter 3 scrutinizes on the research methodology, which demonstrates on the detailed methodology that will be carried out in order to collect the datasets, determine and build the according Sentiment Analysis models, classify the corresponding dataset's sentiment, and aggregate the performance results. It also involves the technical aspect which describes the methods and environmental setup for the whole thesis project. Furthermore, Chapter 4 dissects on the experiment and implementation attempts of the research, where it covers the complete process flow of the implementation of the sentiment analysis modes. Chapter 5 then discusses in-depth upon the results obtained, illustrating the performance evaluation of the Sentiment Analysis models, and comparing between the results of the different models, and also to finally determine whether if the outcome of the project answers the initial research questions established. Lastly, Chapter 6 summarizes upon the thesis, providing insights on the challenges faced, and the area of improvements available. Additionally, it suggests on the future prospect that this project could have headed and how improvements on the proposed model can benefit the general consensus of Malaysia.

## Chapter 2

### Literature Review

This chapter will present a comprehensive literature review to all the relevant matter that is subjected to the research topic, which is sentiment analysis on code-mixed data, specifically on the combination of the English and Malay languages. Upon this chapter, similar and related published works will be reviewed thoroughly to garner the understandings and insights to the existing solutions affixed to the proposed topic. The intended topic, Sentiment analysis as a subfield of natural language processing (NLP), plays a vital role in extracting and understanding subjective information expressed in textual data (Bhattacharyya, 2013). It has numerous applications in various domains, including social media analysis, customer feedback analysis, and market research. However, in countries where bilingualism is prevalent, it introduces on the conundrum of code-mixing, where the universality of code-mixed data, with the account to its unstructured and uncontrollable nature of use, has made it a focused research topic throughout differing cultures and languages (S & Poornachandran, 2018). Even so, there is still minimal, and a lack of research findings towards the relation of sentiment analysis and the code-mixing nature of the Malaysian's community and its languages.

Sentiment Analysis researches upon code-mixing and bilingual contents in comparison to monolingual contents are still scarce in quantity. Code-mixed data introduces complexities that traditional sentiment analysis models designed for monolingual text may not effectively handle (S & Poornachandran, 2018). The unique linguistic and structural characteristics of code-mixed data necessitate specialized approaches and models that can accurately capture sentiment across different languages and their combinations. Thereby in this chapter, literature researches will be done through several scholarly publication's database and search engines –*Google scholar, IEEE, ACM, Researchgate, Springer-*, with the keyword search of “Sentiment Analysis, Multilingual, Code-mixing, Code-Switching, Natural Language Processing, Lexicon, Machine Learning”. The inclusion criteria for the literature research was set to be publications between 2018 to 2023, with no domain restriction on the topic. The concluding goal for this literature review is to thoroughly explore on the background of sentimental analysis, delve into the related approach that had been previously

attempted, and systematically compare the approaches to refer as the most optimal way in conducting this research. By examining on the previous existing studies, the gaps and limitations in the existing approaches may be indisputably identified, which ultimately guides the development of this study's very own research methodology. Thereby with a thorough research, five similar and relevant published papers were identified, analysed, summarized, then listed in a systematic manner in **Table 2.1** below.



**Table 2.1:** Related Works on Code-Mixing Sentiment Analysis

No.	Journal papers	Authors (Year)	Language-pairs	Pre-processing (NLP)	Methods/Approaches	Datasets	Performances
LR1	Aspect-based opinion mining for code-mixed restaurant reviews in Indonesia	(Suciati & Budi, 2019)	English - Indonesian	Emoticon processing, Lowercasing, Spelling correction & Abbreviation, URL removal, Punctuation removal, Stop words removal, Duplicate character remover, Stemming	Decision Tree, Random Forest, Logistic Regression, Extra Tree Classifier, Multinomial Naïve Bayes	2000 reviews collected from PergiKuliner, containing: <ul style="list-style-type: none"> <li>• English review</li> <li>• Indonesian review</li> <li>• Code-mixed review</li> </ul>	<ul style="list-style-type: none"> <li>• Decision Tree: 77.39% (F1)</li> <li>• Random Forest: 78.54 (F1)</li> <li>• Logistic Regression: 80.91 (F1)</li> <li>• Extra Tree: 78.21% (F1)</li> <li>• Naïve Bayes: 75.52% (F1)</li> </ul>
LR2	Sentiment Analysis on Multilingual Code-Mixed Kannada Language	(Dutta et al., 2021)	English - Kannada	URL removal, Emoji removal, Emoticons removal, Special symbol removal, Lowercasing.	Bidirectional Encoder Representations from Transformers (BERT)	<ul style="list-style-type: none"> <li>• 7671 code-mixed Kannada-English dataset.</li> </ul>	BERT with tensorflow: <ul style="list-style-type: none"> <li>• F1-score: 0.64</li> <li>• Precision: 0.64</li> <li>• Recall: 0.64</li> </ul>
LR3	Sentiment Analysis of Persian-English Code-mixed Texts	(Sabri et al., 2021)	Persian - English	Sub word tokenization, Word embedding (BERT)	Naïve Bayes, Random Forest, BERT, Bidirectional Long Short-term Memory (Bi-LSTM)	3,640 tweets labelled with polarity values	<ul style="list-style-type: none"> <li>• Naïve Bayes: 47.17 % (F1)</li> </ul>

							<ul style="list-style-type: none"> <li>• Random Forest: 57.67% (F1)</li> <li>• Bi-LSTM: 63.66% (F1)</li> </ul>
LR4	Sentiment Analysis of Mixed Code for The Transliterated Hindi and Marathi Texts	(Ansari & Govilkar, 2018)	English – Hindi - Marathi	Emoticon & slang processing	Naïve Bayes, Support Vector Machine (SVM)	1,200 Hindi and 300 Marathi documents are extracted from social medias, including: Tweets, Youtube comments	SVM: <ul style="list-style-type: none"> <li>• Hindi: 0.6 (F1)</li> <li>• Marathi: 0.59 (F1)</li> </ul> Naïve Bayes: <ul style="list-style-type: none"> <li>• Hindi: 0.6 (F1)</li> <li>• Marathi: 0.67 (F1)</li> </ul>
LR5	Sentiment Analysis of English-Punjabi Code Mixed Social Media Content for Agriculture Domain	(Singh et al., 2019)	English - Punjabi	Hashtag removal, URL removal, Punctuation removal, Repeated character removal,	Support Vector Machine (SVM), Naïve Bayes	Total of 95,800 comments were extracted from social medias. (Twitter, Youtube, Facebook)	<ul style="list-style-type: none"> <li>• SVM: 85% (Accuracy)</li> <li>• Naïve Bayes: 85.6% (Accuracy)</li> </ul>

## **2.1 Sentiment Analysis on Code-Mixed Data**

Sentiment analysis, also known as opinion mining, is the process of determining and classifying subjective information from textual data (Bhattacharyya, 2013). With the increasing prevalence of multilingual communication on social media platforms and online forums, code-mixed data, which involves the combination of multiple languages within a single sentence or a paragraph, has befitted as a challenging area for sentiment analysis. With this research in particular, the combination of English and Malay poses on unique difficulties due to their grammatical and syntactic differences between the two languages. In this section, we will compare the characteristics and nature of the code-mixed data found in each of the study listed above.

In LR1, Suciati & Budi (2019) conducted sentiment analysis on the English-Indonesian code-mixed data. The code-mixed data in their study consisted of restaurant reviews which was collected from a food review website called the “PergiKuliner”, where the users who uses the website often blended English and Indonesian language within their reviews. The code-mixing usage for the reviews could be primarily driven by language preferences, user context, and the user’s need to express their point effectively. The data in LR1 is separated into 3 instances, pure Indonesian reviews, pure English reviews, and code-mixed reviews, where there are many included informal language, abbreviations, and slangs that are commonly used on social media platforms.

In LR2, Dutta et al (2021) explored on the sentiment analysis on the code-mixed user-generated contents from YouTube, where they used a YouTube Comment Scraper to collect the code-mixed Kannada-English comments from 18 different videos. The code-mixed data in their study ranged from conversational exchanges to user comments on various topics, such as Movie trailers, India trending topics, India-China border issue, Mahabharata, and many others. The data exhibited variations in language switching patterns, such that the collected comments is either with Kannada grammar with English lexicon or English grammar with Kannada lexicon, some variations may also include comments in Kannada script but with English expressions in between.

For LR3 and LR4, Sabri et al., 2021 and Ansari & Govilkar, 2018 both focused on the sentiment analysis on code-mixed social media data. Their code-mixed data

consisted of social media posts which the users blends the use of English-Persian and also English-Hindi language respectively in their social media posts. In the case of Ansari & Govilkar, 2018, the data collected is found to consist phonetically-typed Hindi text, while also further extending to English words, which forms a perplexing dataset where there is different language usage in a single data text, however are all in the latin-alphabet form. On the other hand, Sabri et al., 2021 deduced that the Persian language is a low resource language, but they are fairly used in mixture with English amongst the Persian community. However, instead of code-mixing or code-switching with English words, the data in LR3's study is mostly English word transliterated into the Persian language, such that the English word is instead written in Persian, but having the closest pronunciation to match the original English wordings.

Lastly in LR5, Singh et al., 2019 focuses on exploring the sentiment analysis of code-mixed English-Punjabi social media contents that specifies on the agriculture domain. The code-mixed data in their study were collected from micro-blogging sites such as Twitter, Facebook, and Youtube for the span of 6 months, collecting up to a total 95800 comments. Their collected data consist of informal Punjabi-English mixture scripts, which also consist of some transliterate English to Punjabi wordings. Observing through the study, it is stated that approximately 2.57% of Punjab's total population expresses their opinions in an English-Punjabi code mixing manner on social media. It is acclaimed that the usage of English-Punjabi code-mixing was primarily driven by the local community's language preferences, and also their need to express their opinions efficiently.

## **2.2 Data Pre-processing**

Numerous techniques for pre-processing text have their origins in Natural Language Processing (NLP). In NLP, raw textual data is transformed into clean tokens, which serve as features for subsequent analysis (Anandarajan et al., 2018). These pre-processing methods aid in removing irrelevant information or correcting errors in the inputted text, preparing it for smoother analysis. The specific pre-processing steps can vary depending on the context of sentiment analysis (SA) applications. For instance, in the case of LR1, LR2, and LR5, URL and punctuation removal were included as pre-processing steps because these studies focused on social media, where such elements do not contribute additional information for the analysis. In LR1, spelling

and abbreviation correction were incorporated due to the informal nature of social media language, where words with incorrect spelling or abbreviations (e.g., tbh, afk, lol) were replaced with their corresponding correct forms. Additionally, LR2 and LR3 employed word tokenization to break the raw input texts into smaller chunks of words or tokens, aiming to enhance understanding and interpretation of the context. In LR2, the tokenized words were used as training inputs for the classification phase. While some pre-processing approaches employ different machine learning frameworks, LR3, for example, utilized the BERT framework to convert words into embeddings, representing the meaning of a word numerically. By converting words into embeddings, the model can recognize the semantic significance of a given word, enabling mathematical operations to be performed on it (Tanaka et al., 2020).

### **2.3 Feature Extraction Techniques**

The notion of feature extraction also plays a crucial role in establishing sentiment analysis, as these techniques help transform raw text data into numerical representations (Kalaivani et al., 2020). This is done so that the machine learning or deep learning models can effectively process the given data (Kalaivani et al., 2020). In the reviewed literature studies, there are various kinds of feature extraction techniques that were employed, ranging from more traditional approaches such as n-gram features, up to utilizing BERT embedding tokens for feature extractions.

In LR1, Suciati & Budi (2019) utilized something called the bigram term frequency technique to capture the sequential word combinations as features for sentiment classifications. The bigram term frequency is an approach that considers the pairs of consecutive words in a text and calculates the frequency of each unique bigram within the document. By incorporating these bigram features, their model can capture more specific linguistic patterns and dependencies between adjacent words, which can be valuable in performing sentiment analysis. Suciati & Budi (2019) also vectorised the word representations in their food review data, and in an attempt to increase the model's performance score, they also used a combination of stemming and stopwords steps onto the dataset. Overall, the use of bigram term frequency as a feature extraction method highlights the importance of capturing local contextual information in sentiment analysis on code-mixed data. By considering pairs of consecutive words, the

model can better understand the sentiment implications of specific word combinations and improve its ability to classify sentiment accurately.

In LR2, Dutta et al. (2021) utilizes on BERT embedding tokens as a mean of feature extraction. A Bidirectional Encoder Representations from Transformers (BERT) model, is considered a state-of-the-art language representation model that has achieved remarkable success in various kinds natural language processing tasks. In Dutta et al. (2021) study, they leverage the power of BERT by utilizing its pre-trained language representations as feature vectors for sentiment analysis. BERT tokenizes the input text data into subword units and assigns contextualized embedding vectors to each token. These embeddings encode the meanings of individual words as well as the contextual information derived from the surrounding words in the sentence or document. By using BERT embedding tokens as features, Dutta et al. (2021) was able to take advantage of the rich contextual information captured by BERT. This contextualization helps the sentiment analysis model understand the nuanced sentiments expressed in code-mixed data, which often involves complex linguistic variations and combinations. Overall, LR2 has demonstrated the effectiveness of utilizing BERT embedding tokens as a feature extraction method in sentiment analysis on code-mixed data. By incorporating BERT's contextualized representations, Dutta et al.'s model achieved improved performance in capturing sentiment nuances and accurately classifying sentiments in the complex code-mixed text.

Moving on to LR3, Sabri et al. (2021) also utilizes on a similar approach as the previous study, which is to employ on BERT's word embeddings as a method of feature extraction. There is a slight different aspect for this study, for Sabri et al., they employed on a multilingual variant of BERT that supports both English and Persian languages. Their selected variant of BERT has been pre-trained on a large corpus of text data containing diverse language pairs, which includes on their desired English and Persian language. Through leveraging on this pre-trained BERT model, they were able to create on high-quality word embeddings for the words in both languages. To create the embeddings for code-mixed words, Sabri et al. (2021) also followed a tokenization process where the input text data was divided into individual subword tokens. These tokens were then converted into BERT word embeddings using their chosen pre-trained multilingual BERT model. The resulting word embeddings captured the semantic information and contextual relationships between words in both

English and Persian, also as they stated, allowing for a better understanding of slangs or other non-common words.

With LR4, Ansari & Govilkar (2018) aims to analyse the sentiments in code-mixed text containing English and Hindi (Marahti) language. In their study, they employed on the N-gram features as an intent to capture the sequential word combinations and their contextual information. Ansari & Govilkar (2018) utilized on a sliding window approach where they consider the consecutive sequences of  $n$  words within their code-mixed text data, and treated them as individual features. By varying the value of  $n$ , they captured on the different levels of contextual information, ranging from unigrams up to a higher order  $n$ -grams, such that  $n=5$ .

Lastly with LR5, Singh et al. (2019) too also employed on the N-gram features as a mean for feature extraction. However, there is a slight different on the ways they do it, in which they employed a combination of Unigram, Bigram, and Trigram features to capture the different levels of contextual information and linguistic patterns of their data. Also similar to before, Singh et al. (2019) considers individual words as Unigrams, consecutive words as Bigrams, and lastly three consecutive words as Trigrams. By utilizing a combination of unigram, bigram, and trigram features, Singh et al. aimed to capture both local and global contexts in the code-mixed data. The utilization of Unigrams allowed their model to consider the sentiment-bearing words individually, while bigrams and trigrams enabled their model to capture the relationships and dependencies between adjacent words. The incorporation of  $n$ -gram features facilitated a more nuanced understanding of sentiment towards their code-mixed text data. Unigrams captured individual sentiment-carrying words, while bigrams and trigrams provided a broader context and captured meaningful word combinations that contribute to sentiment expression.

In summary, the five reviewed literature studies had employed on various feature extraction techniques. While some studies utilize on the  $n$ -gram features, there are also multiple studies that explored the use of BERT embedding tokens to leverage contextual information. The choice of feature extraction technique depends on the specific characteristics of the code-mixed data and the desired level of semantic understanding required for sentiment analysis.

## **2.4 Sentiment Analysis Approaches/Methods**

Within the reviewed studies list, it is observable that each studies had utilized on various different methods and models to employ as an approach for their code-mixed sentiment analysis. Thereby, this section will review on each of the methods used, and provide a detailed discussion of the models and approaches utilized in each study, confer on their specific applications, and their effectiveness in the context of code-mixed sentiment analysis. This is to fathom on the available models, and highlight on the models that appears to be most suitable for this thesis research topic.

### **2.4.1 LR1: Decision Tree, Random Forest, Logistic Regression, & Naïve Bayes**

For the case in LR1, Suciati & Budi (2019) employed and explored on the effectiveness of various machine learning models, including on Decision Tree, Random Forest, Logistic Regression, and lastly Naïve Bayes on classifying the sentiments of code-mixed English-Indonesian data. Each of their chosen model was applied to the classification task with the goal of determining the most suitable approach, and also to ascertain on which machine learning algorithm works best with their collected code-mixed dataset.

The first machine learning algorithm that they employed is the Decision Tree. It is known as a tree-based model that recursively partitions the data based on its feature value. How the algorithm works, is that it builds a tree structure where each internal node is represented as a feature, and each leaf node will correspond to a class label. For the algorithm to perform on sentiment classification, it will engage in a decision making process where it traverses the tree based on the feature values of the inputted data, and ultimately traverses onto the end of a leaf node, which assigns a sentiment label based on the traversal path (A. Jain & Dandannavar, 2016). As Suciati & Budi (2019) stated, the Decision Tree algorithm was selected based on its recommendation in solving multilabel and multiclass problems, this is because the Decision Tree approach has the advantage of being interpretable and being capable of capturing non-linear relationship between the different features.

Subsequently, LR1 also employed on the Random Forest machine learning algorithm. The Random Forest method can be considered as an advancement from the previous decision tree method, as the Random Forest model is considered as an



ensemble model, where it combines on multiple Decision Trees in order to improve on the overall performance (S. Singh & Sarraf, 2020). It also utilizes on a technique called Bagging, where each decision trees are trained on a random subset of the data. The final classification is done through by aggregating on the prediction results of the individual decision trees. As how Suciati & Budi (2019) stated in their literature, this model was also selected based on a recommendation for multiclass problems. But overall, the Random Forest algorithm does have the notorious implication on reducing overfitting problems, and also present a robust result by averaging the predictions from multiple decision trees.

Not only that, Suciati & Budi (2019) also utilized on another machine learning algorithm called the Logistic Regression. This was selected in their study due to their performance review on Chawla and Mehrotra (2018) study, where the Logistic Regression provides the best performance and scores throughout different scenarios and dataset. Logistic Regression itself is considered as a statistical model, it works on estimating the probability of a data belonging to a specific sentiment class, where it applies a logistic function to the input features, then transforming the output into a probability score (Alzubi et al., 2018). They are often used in sentiment analysis instances because Logistic Regression can easily handle both binary and multiclass classification problems by employing appropriate strategies such as one-vs-rest or softmax (Alzubi et al., 2018).

Lastly, Suciati & Budi (2019) likewise employed on another probabilistic machine learning model called the Naïve Bayes model. It is a relatively simple model which is usually known for its simplicity, efficiency, and also ability to handle on high-dimensional data. The model similarly works on estimating the probability of a data belonging to a specific sentiment class (A. Jain & Dandannavar, 2016), but instead of applying logistic functions as how the Logistic Regression model did, the Naïve Bayes model simply assumes the independency among the features, and calculates the probability based on the occurrence of words in the data (A. Jain & Dandannavar, 2016). While the assumption of feature independence from Naïve Bayes model may not hold true in all cases, but observing on Suciati's & Budi's literature, it has demonstrated some promising performance in sentiment analysis tasks, including their English-Indonesian code-mixed sentiment analysis.

Through the performance evaluations, Suciati & Budi (2019) did a comparison and analysed the strengths and weaknesses of each implemented models in validating with their collected English-Indonesian dataset. Based on their evaluation assessment, the Decision Tree and Random Forest models were found to capture complex relationships between features, both coming in second and third place in their performance score respectively. However, both these models seem to have the tendencies to overfit on the training data. The Naïve Bayes model on the other hand, was placed last in the performance, but despite its assumption of feature independence, it still demonstrates a conventional and competitive performance due to the model's simplicity and efficiency. Subsequently, the Logistic Regression model provided the highest accuracy and achieved optimal results in sentiment classification. Its interpretability and robustness were valuable in understanding and analysing the sentiment patterns in code-mixed data, allowing it to achieve the highest performance score in almost all scenarios.

#### **2.4.2 LR2: BERT Model**

Reviewing through LR2, it is recognisable that Dutta et al. (2021) chose to focus on the application of Bidirectional Encoder Representations from Transformers (BERT) models in their study. A BERT model is considered a state-of-the-art transformer-based model, and it has gained quite a significant attention throughout the Natural Language Processing (NLP) tasks, which also includes on sentiment analysis. The workings of BERT models have demonstrated its remarkable performance in capturing the contextual information and semantic meaning in text data like no other machine learning models do. The way that BERT models work, is that it leverages on a bidirectional transformer architecture, where it allows them to consider both left and right context when encoding word representations (Jacob Devlin et al., 2018). This bidirectional notion enables BERT models to capture the interdependencies of words within a sentence (Jacob Devlin et al., 2018), making it particularly suitable for code-mixed sentiment analysis. In code-mixed data, where language switches or language mix occurs, the sentiment can be influenced by the context and relationships between words, rendering it a remarkably ideal upbringing for the usage of BERT models.

Conversely, Dutta et al. (2021) attempted in employing two different BERT models to address on their code-mixed sentiment analysis concerns. Firstly, they trained a

BERT model from scratch using the TensorFlow framework. This involved in initializing the BERT model with pre-trained weights and fine-tuning it on the code-mixed sentiment classification dataset. Fine-tuning the BERT model enables the model to adapt its parameters to the specific sentiment analysis task, taking into account the nuances of code-mixed data. Additionally, they also trained on a second BERT model by utilizing on the Ktrain library, which simplifies the process of training and fine-tuning BERT models. Ktrain provides a high-level interface for TensorFlow and enables efficient implementation of BERT-based models. The researchers leveraged the Ktrain library to fine-tune a pre-trained BERT model on the code-mixed sentiment analysis dataset, further exploring its effectiveness in capturing sentiment patterns in code-mixed text.

The results presented in LR2 demonstrated the effectiveness of both implemented BERT models in establishing code-mixed sentiment analysis. Dutta et al. had also experimented the code-mixed data with several machine learning models, including on Logistic Regression, Random Forest, SVM, and Naïve Bayes. Through comparison, the BERT model trained from scratch using TensorFlow and the fine-tuned BERT model with Ktrain consistently outperformed the other machine learning models in terms of sentiment classification accuracy. It is perceptible that these two BERT models excelled better at capturing contextual information, including on language switches, and interdependencies between words, thereby enabling accurate sentiment analysis in code-mixed data.

#### **2.4.3 LR3: Bidirectional Long Short-Term Memory (Bi-LSTM)**

Although LR3 also utilized on an instance of BERT model, the researchers Sabri et al. (2021) implemented on it in quite a distinct way. Their literature study aims to leverage on the effectiveness of a pre-trained BERT model by using it purely for embeddings generation. The created embeddings is instead fed into an ensemble model that uses on three Bidirectional Long Short-Term Memory (Bi-LSTM) networks. Their study aims to leverage the potential of BERT embeddings and also their proposed ensemble approach to improve on the sentiment analysis accuracy in their code-mixed English-Persian data.

In their study, Sabri et al. (2021) employed a pre-trained BERT model simply to generate the embeddings for their code-mixed data. The BERT embeddings allows the

capture of the rich contextual information and semantic meaning by considering the bidirectional nature of the model. By leveraging on the pre-trained BERT model, Sabri et al is able to benefit from the learned representations of words and their contextual relationships, providing a strong foundation for sentiment analysis. The generated embeddings is then fed into their proposed ensemble model, which is made up of three different Bi-LSTM networks, each serving on a specific purpose. The first network was intended as a stacked Bi-LSTM network, designed to encode the general information available in the data sentences. This network captures the sequential patterns and dependencies within the code-mixed text, which is what enables the model to understand the overall sentiment context. The second network is incorporated as an attention mechanism, which allows the model to focus more on the most important words in a data sentence. By attending to these crucial words, the model could better capture the key sentiment-bearing terms, enhancing the accuracy of sentiment classification in code-mixed data. Finally, the third network in the ensemble model was organised as a pooling layer, which aim is to ensure that the long-distance dependencies were accounted for and that information was not lost. The pooling layer aggregated information from the entire sentence, capturing broader context and improving on the model's understanding of sentiment expression in code-mixed text.

Evaluating on the performance of the model, Sabri et al also implemented on two machine learning models, including Naïve Bayes and Random Forest to act as the baseline models to their research. The final result of their proposed ensemble model highly outperforms the two baseline models in terms of sentiment classification accuracy. Their generated BERT embeddings helps to capture the rich contextual information, while the ensemble approach with Bi-LSTM networks allowed for a deeper understanding of sentiment expression. This combination led to improved sentiment classification accuracy in code-mixed text. This literature had proposed on a fascinating approach in classifying the sentiments of code-mixed data, the incorporation of similar techniques can be considered, leveraging on pre-trained models and an ensemble architecture, enhancing the performance accuracy for sentiment analysis in code-mixed data.

#### **2.4.4 LR4: Support Vector Machine (SVM) and Naïve Bayes**

For the case of LR4, the researchers Ansari & Govilkar (2018) aimed to investigate the effectiveness of two particular machine learning models, including the Support

Vector Machine (SVM) model and the Naïve Bayes model for their English-Hindi code-mixed text data. Their study focused on comparing and analysing the performance of these two models to understand on which method is better suited to classify on the code-mixed sentiment analysis.

The first model they utilized on is the Support Vector Machine (SVM) model, it is quite a well-known model notorious for its ability to effectively classify data by finding the optimal hyperplane that maximally separates the different label class (A. Jain & Dandannavar, 2016). In the sentiment analysis circumstances, the SVM models aim to find a decision boundary that separates between the positive and negative class in the code-mixed text. SVM models leverage on a kernel function to transform the input features into a higher-dimensional space, where a hyperplane can be identified to separate the sentiment classes (A. Jain & Dandannavar, 2016). The second model that they utilized is once again, the Naïve Bayes model. The Naïve Bayes model is known for its simplicity and its computational efficiency, making it an attractive choice as a baseline model for sentiment analysis tasks, and despite the model's assumption of feature independence, it still is able to showcase on a competitive performance in capturing sentiment patterns. The performance of both model is peculiarly similar, but the Naïve Bayes model still slightly outperforms the SVM model in the sentiment classification accuracy, especially in classifying English-Marathi transliterated text data.

#### **2.4.5 LR5: Support Vector Machine and Naïve Bayes**

In LR5, the researchers Singh et al. (2019) also utilized on the SVM and Naïve Bayes model to experiment on their English-Punjabi code-mixed dataset. Surprisingly, the results of LR5 showed that the Naive Bayes model outperformed the SVM model in the context of code-mixed sentiment analysis. The Naive Bayes model exhibited a more superior performance in accurately classifying sentiment in code-mixed text. Despite its assumption of feature independence, the Naive Bayes model successfully captured the underlying sentiment patterns, leveraging the word occurrences in the code-mixed data. In contrast, the SVM model demonstrated a slightly lower performance in sentiment classification of the code-mixed data. While SVM models are known for their ability to handle complex decision boundaries and non-linear relationships, they somehow struggled to fully capture the intricacies of sentiment

expression in code-mixed text. The SVM model's performance might have been affected by the challenges posed by code-mixing phenomena, such as language switches, transliteration and the diverse word order patterns.

The findings of LR5 suggest that in the specific context of code-mixed sentiment analysis, the Naive Bayes model might offer on a better performance compared to the SVM model. The Naive Bayes model's simplicity and efficiency, coupled with its ability to effectively capture sentiment patterns in code-mixed data, makes it a suitable choice for sentiment classification tasks in code-mixed text.

## **2.5 Conclusion**

This section aims to elaborate on the determined research gaps and to identify on areas that have not been adequately addressed in previous studies. By examining the existing literature, this research gaps section will highlight on the limitations and shortcomings of the prior found research, paving the way for the current study to contribute on new insights.

The first gap to be identified is on the insufficient exploration of code-mixed data. Throughout each of the existing studies, it highlights the limited amount of research that specifically focuses on sentiment analysis of code-mixed data. It may emphasize that despite the increasing prevalence of code-mixing in various contexts, including social media and multilingual societies, there is still a scarcity of studies that delve deeply into sentiment analysis techniques tailored for code-mixed data. This research gap calls for the need to develop specialized methods to handle the unique challenges posed by code-mixing. Other than that, the limited consideration of specific language pairs also posed as a research gap in this study. Review through the different studies, it is identified that there is a lack of studies that investigate on the sentiment analysis on code-mixed data for specific language pairs. For instance, there might be a scarcity of research examining sentiment analysis for code-mixed data containing English and Malay languages. This research gap highlights the need for studies that address the specific challenges and linguistic characteristics of code-mixing between those languages. Lastly, it may also emphasize the lack of comprehensive evaluation of the different machine learning models for sentiment analysis on code-mixed data. It may highlight on the need for studies that rigorously compare the performance of various

models, including traditional approaches like Naive Bayes and newer techniques like deep learning models. The research gap calls for the identification of the most effective models for sentiment analysis on code-mixed data, considering factors such as accuracy, computational efficiency, and robustness to language variations.

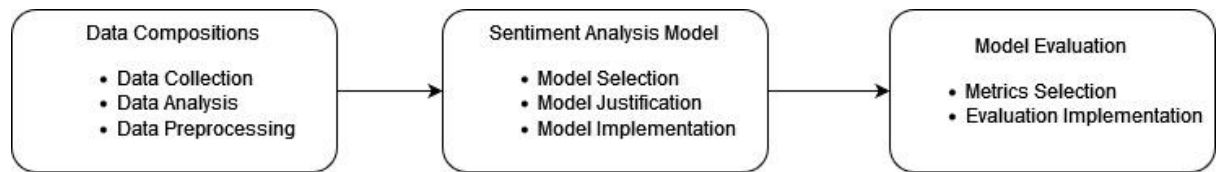
By highlighting on these research gaps, it is an aim to position the current study as a valuable contribution to the field. It demonstrates the need for a novel approach, techniques, and evaluations to overcome the limitations of prior research and advance the understanding and accuracy of sentiment analysis on code-mixed data.

## Chapter 3

### Methodology

This thesis contemplates to impose sentiment analysis onto the Malaysian's code-mixed data on social media. The aim towards this research problem is to gain a more in-depth understanding towards the sentiment nature of code-mixing contents through developing a code-mixed SA model, especially towards Malaysian's content that circulates throughout social medias. This would require for a quantitative research approach that constructs upon models that satisfies the classification of sentiments in regards to code mixing contents with significant precision and accuracy. The results and evaluations shall be presented in statistical model and numerical grading that will justify on the research's result accuracy and legitimacy.

#### 3.1 Research Methodology Framework



**Figure 3.1:** Established Research Methodology Framework

The research methodology framework in establishing this thesis project is divided onto multiple segments. The first segments comprise on the compositions of data; it discusses on the data collection process: why are datasets needed to be collected? What types of datasets are being collected? What are the source and integrity of the collected datasets? And also the justification of the collected datasets. After that, it analyses on the collected datasets, presenting on the distribution and cleanliness of each datasets. Finally, it discusses on data pre-processing, presenting the necessary process to clean the different types of dataset, and which types of pre-processing techniques are required.

Subsequently, the following segment comprises on the composition of different sentiment analysis models. It converses on the different types of sentiment analysis techniques utilized, such as the different type of machine learning models and deep



learning models. The following goal is to determine on which techniques and models has the most ideal/optimal performance implemented in this topic. The justification to the model selection shall then be presented precisely. Successively, it then presents on the necessary process to implement each of the selected models, and how the collected datasets work with them.

Thereafter, the succeeding segment comprises in evaluating the implemented model. The method of evaluation is listed and presented upon, along with the justification and reasoning in choosing the specific evaluation matrices. Each implemented models shall be then evaluated against the chosen metrics, thenceforth each model's ensuing results are compared against one another. Conclusively, based on each evaluation result, an improvement model is proposed in an attempt to further improve the classification accuracy.

### **3.2 Data compositions**

As being stated previously, the first segment consists of discussing the relevant datasets being used within this study. The intention to collect and utilize datasets within this study derived from the basis of machine learning/deep learning operation. Machine learning/deep learning models utilizes algorithms to constantly learn and improve itself, however a quality dataset will be required to be fed in order for the models to learn and improve efficiently. According to (Jain et al., 2020), the performance of a Machine learning or deep learning model is upper bounded by the quality of fed data, moreover the collected datasets will also be differentiated to fulfil various roles in establishing the model. Contingent upon this thesis project, establishing the Sentiment Analysis models would require three different types of dataset, one which is the training dataset to be fed into the learning algorithm. Ensuing on, a validation dataset, also known as the testing datasets will be needed to ensure that the trained model is classifying and interpreting the data correctly. Lastly, an annotated ground-truth dataset is utilized in the study for the established models to predict and classify on. It imitates for a real world usage for the models to execute on. It allows an interpretation and an evaluation for how accurately the model can actually work towards unseen data.

### **3.2.1 Data Collection**

By the reason that this current study emphasises on the mixture of two major languages – *English, Bahasa Melayu*-, the data collected is categorized into three different language types, which are:

- English Datasets
- Bahasa Melayu (Malay) Datasets
- Code-mixed (English – Malay) Datasets

#### **English Dataset**

The English dataset utilized in this study is mainly meant to train and establish an English Sentiment classifier. The collected dataset is derived from the corpus created by Go et al., (2009), named the Stanford Twitter Sentiment140-dataset, which it consists of a total of 1.6 Million annotated tweets entailing the labelled sentiments of Positive and Negative. Considering this current thesis study focuses on the domain of social media, a large sentiment dataset consisting of social media posts could thoroughly train a model to even classify phrases or words that conjugates as “Social Media lingo”. Additionally, a larger training dataset is always desirable for training a model, as uniting a larger dataset allows the improvement of the model’s feature engineering, where more raw-variables is available for feature extractions. Substantially, a better training process and also learning outcome the model is able to be achieved (Morrell, 2022).

#### **Bahasa Melayu (Malay) Dataset**

The Bahasa Melayu (Malay) dataset being utilized is also meant to establish and train a separate Malay Sentiment classifier. As the Malay language is utilized fairly little in the software and data analytic sector, the existing and available Malay datasets are rather scarce in compared to the English counterparts. As Fortunate in the recent years, contributors within the Malaysian community had kick-started and expanded a massive Malay corpus and Malay Natural Language Library. The collected Malay dataset is stemmed from the source of (Zolkepli, 2018a), which is the largest open-sourced Malay corpus made and meant for research purposes. The corpus range is still continuously being expanded overtime, as currently the collected Malay sentiment dataset has accumulated approximately 600,000+ labelled sentiment data. The collected Malay data derives from crawling and the extraction of Social Media –

*Twitter*-, which comprises on the domain of casual tweets, product-related talks and political discussions.

### **Code-mixed (English-Malay) Dataset**

The code-mixed English-Malay dataset collected is meant to act as the ground-truth dataset for the establish models to predict and validate from. The collected dataset derives from a code-mixed corpus created by (Romadhona et al., 2022), which they coined as being the first Malaysian code-mixed Sentiment dataset, called the SentiBahasaRojak. The dataset is manually crawled, constructed and annotated by five Malaysian experts, resulting in a total of 2,285 code-mixed sentiment data available. The crawled code-mixed data derives from public forums, and Social Media, in which the domain composes of financial talks, Movie reviews, and product discussions. This current dataset will act as a real-world data for the established Sentiment Analysis models to predict on, to evaluate on how accurate the models will perform with unseen code-mixed datasets.

### **3.2.2 Data Analysis**

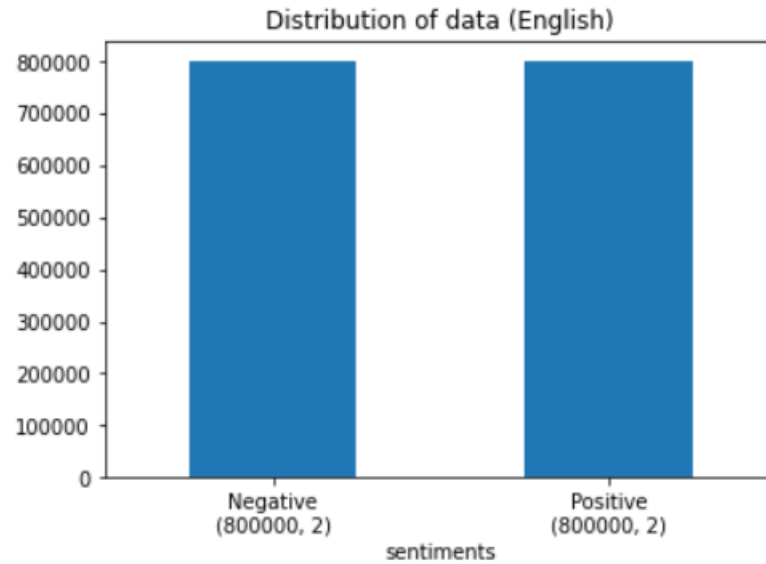
To analyse on the collected datasets, there are two analysing criteria that needs to be scrutinised upon. Firstly, which is to analyse and observe on the data distributions of each dataset, uncovering on any high-class imbalance problem within them. A high-class imbalance stems from datasets with unequal distribution of classes, or simply having classes that has significantly more samples than the other (Leevy et al., 2018). This is considered as a problem in data analytics and machine learning models in general, is because if a degree of class imbalance is extreme enough, then a classifier model will be yield being not practically useful, since the overall learning and prediction accuracy in most instances will favour upon the majority class (Leevy et al., 2018). Thereby, the analysis for data class imbalance is needed in order to devise on any down sampling approach, if the problem do occur.

The second criteria that needs to be scrutinised upon, is on the data cleanliness within each dataset. It is a crucial process to analyse and ensure that the datasets are devoid of any erroneous or faulty data. It is stated that, the prerequisite of developing a high performing and accurate Machine Learning model is in need to comprise the availability of high-quality data (Lee et al., 2021). Be that as it may, collected data is rarely ever clean by the reason either from noisy inputs from manual data collections,

or imperfections that curates from automatic data crawling. Thereby, the analysis for data cleanliness and imperfections are needed in order to devise on the approaches to process and clean the data.

### English Dataset Analysis

Below **Figure 3.2** presents the Sentiment class distribution – *Positive & Negative*- among the English dataset, with a totalling of 1.6 Million data:



**Figure 3.2:** S140 English dataset distribution

As observed above, the Stanford Twitter Sentiment140-dataset has distributed its sentiment class in a rather equal and balanced manner, with each sentiment class equally containing 800k data. Thereby, the predicament of a high-class imbalance problem occurring with this dataset is scarce, and no shifting and down sampling procedure is needed to be carried out in this dataset.

To follow through on the second analysis criteria, below **Table 3.1** presents the data samples where 10 random data were extracted from the English dataset:

**Table 3.1:** S140 English 10 Random Data Samples

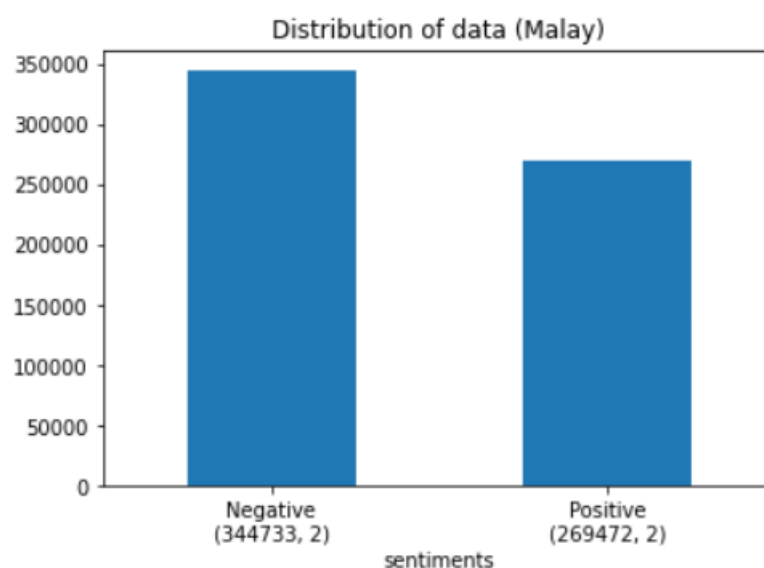
index	sentiments	tweets
514293	0	i miss nikki nu nu already shes always there when needed thank u xxx
142282	0	So I had a dream last night. I remember a sign which clearly told me where to get a job. I can't rememer what the sign said.
403727	0	@girlyghost ohh poor sickly you (((hugs))) hope you feel a little better soon

649503	0	it is raining again
610789	0	@MissKeriBaby wish I was in LA right now
67315	0	Nala Olowalu still has a full tummy from bread basket #2 <a href="http://apps.facebook.com/dogbook/profile/view/6285546">http://apps.facebook.com/dogbook/profile/view/6285546</a>
833521	1	@macintom site doesn't seem to want to load up, they must be getting a lot of hits
256032	0	time for some sleep- hav to actually do some work tmrw!!
657012	0	@supercoolkp In Oxford that month.
980587	1	..time for a cup of tea and fruit bagels, i'm going to turn the day around!

Analysing through the sampled data, it is inherently observable that the dataset is riddled with erroneous and noisy data. As the extracted dataset derives from Social Media, specifically Twitter, it contains many of web-based word and characters usage, such as alias tagging - @-, Hashtags - #-, URL formats – *https://*-, internet slang words, and acronyms. These noisy data typically serve no meaningful purpose in training the Sentiment Analysis model, and would rather plague the models with faulty insights, leading to accuracy depreciation. Thereby, a moderate data preprocessing procedure would be needed to be applied on this current dataset, triumphing for a cleaner, high-quality dataset.

### Bahasa Melayu (Malay) Dataset Analysis

**Figure 3.3** below illustrates the Sentiment class distribution – *Positive & Negative*- among the Bahasa Melayu (Malay) dataset, with a totalling of 614,205 data:



**Figure 3.3:** Malay dataset distribution

Observing on the data class distribution, it is apparent that there is a class imbalance within this current dataset. The Negative class is observed to be in majority, and exceeds significantly over in compared to the Positive class. The Negative class has a total of 344,733 data, whereas the Positive class is totalling on 269,472 data, bearing a disparity of approximately 90,000+ data class imbalance. The imbalance between the data classes would certainly be problematic, thereby a counter measure to balance on the data classes needs to be implemented.

Advancing on to the subsequent analysis criteria, **Table 3.2** below presents 10 random samples extracted from the Malay dataset.

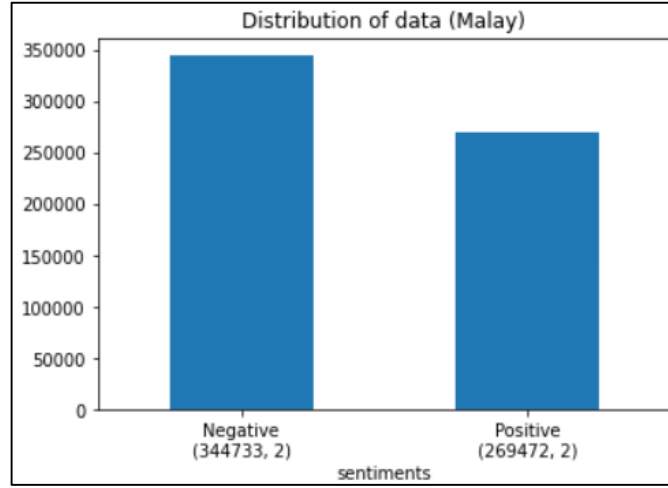
**Table 3.2:** Malay Dataset 10 Random Data Samples

index	sentiments	tweets
519839	1	perenggan kedua
113777	0	yang sangat bising di luar. . fiesta di sini itulah sebabnya. . hehehe. . saya terlepas john begitu banyak. .
522631	1	ketawa kuat. terima kasih! !
418292	1	anda harus pergi, kemudian beritahu kami apa yang anda fikirkan
19898	0	ugh seseorang hanya memanggil saya dan bangun saya
463120	1	heh, ketika im dengan sahabat saya, mereka adalah kejadian biasa.
270476	0	kembali ke program berjadual yang kerap! ketawa nikmat menikmati hun tayangan yang lain! berharap saya berada di sana bersama dengan kamu!
176667	0	ketawa kuat! tidak. . . tidak pasti bagaimana untuk mengatasinya dan tidak menjumpai sebagai btch! saya hanya tidak mengendalikan perasaan melepaskan diri dengan baik!
275479	0	suis adalah butang. . . dan penutup butang hilang dan lubang untuk sampai ke sana benar-benar kecil. ugh.
224873	0	bos membelikan saya makan tengah hari! bekerja pada pembelian macbook, jadi pertunjukan itu mungkin keluar dari udara sehingga minggu depan.

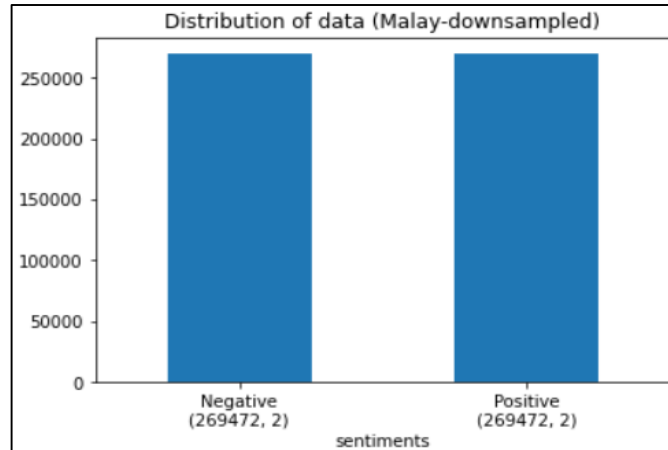
Examining through the sampled data, it is noticeable that this current dataset appears to be fairly clean, as it seemed to be cleaned and processed beforehand. However, it still consists of minimum noisy elements such as punctuations, and stop words. Thereby, a minimal pre-processing could be applied on this current dataset, further improving for a cleaner and high-quality dataset.

### 3.2.3 Data Pre-Processing

The notion of data preprocessing had ceaselessly take up as an important influence towards the generalization performance of a machine/deep learning algorithm (Alexandropoulos et al., 2019). As a continuation on the previous dataset analysis, this segment encompasses on discussing the required preprocessing techniques in cleaning and processing the collected datasets. For the Malay dataset, a down-sampling procedure is applied to address the high-class imbalance problem. The Negative sentiment class is down sampled to match an equal amount of data with the Positive sentiment class. Within the down sampling approach, samples are randomly selected from the Negative majority class to be reduced.



**Figure 3.4(a):** Malay Distribution of data before down-sampling



**Figure 3.4(b):** Malay Distribution of data after down-sampling

Subsequently, for all three of the datasets –*English, Malay, and code-mixed*–, the standard preprocessing technique for social media text cleaning has been applied, which are:

- Reformatting and relabelling
- Lower casing
- URL removal
- Non-alphanumeric character removal
- Stop words removal

### **Reformatting and relabelling**

With the use of multiple datasets from different sources; the format, columns, and labels tend to differ between one another. Thereby, in order to maintain a consistent layout on all the datasets, they are reformatted into having the same set of columns and sequence, which are:

**Table 3.3:** Dataframe layout for data consistency

Index	Sentiments	Tweets

Subsequently, the sentiment class labels for all the datasets were also relabelled into a standardized value, where the Positive class is relabelled as 1, and the Negative class is relabelled as 0. Through this, it would be easier for the learning algorithm, as well as it is for humans to comprehend between the classes.

### **Lower casing**

The lower casing preprocessing technique is being utilized simply for the sake of simplicity, it lower cases all the data within a dataset, which helps in maintaining the consistency of expected output. Not only that, it also deals in mitigating data with badly-cased words, which could cause on sparsity issues. Below illustrates the example of data lower casing:

- Ali did mention that he lives in Malaysia → ali did mention that he lives in malaysia

### **URL removal**

Dealing with social media data, there would be instances where URL links were included. These URLs do not provide any useful information as it does not carry any sentimental meanings, it may even provide false insights while feeding it towards a machine learning model, plaguing the performance of the model's output. Thereby,



the reasonable thing to do is to simply remove any indication of URLs from the datasets.

### **Non-alphanumeric character removal**

Again, while dealing with social media data, there would be occasions where symbols such as hash tags, aliases, or emoticons - #, @, :, *etc*- are included. Not only that, punctuations such as full stop, commas, question marks, or exclamation marks are also regarded as Non-alphanumeric characters. This preprocessing technique technically removes every non-alphanumeric characters from the dataset, as these symbols or punctuations will only add up noises that causes ambiguity while training the classification models.

### **Stop words removal**

Stop words are a collection of frequently used words in a language, which mostly is considered as a low information texts that generally don't carry much useful information or sentimental meanings. Below presents a few stop words example in both English and Malay language.

**Table 3.4:** Stopwords example for English and Malay

Language	Stopwords example
English	I, me, my, myself, we, our, ours, ourselves, you, your, yours, yourself, yourselves, he, him, his, himself, she, her, hers
Malay	Ada, agak, agar, akan, aku, itu, jadi, kalua, kami, kamu, ke, kecuali, kepada, kerana, ketika, sangat, sebab , sebagai

The intention to utilize stop words removal, is so that by removing the low information texts, the more important words can be focused upon, and the learning model is able to consider only the key words of a sentence. Thereby, a more efficient learning process will be achieved.

## **3.3 Sentiment Analysis Models**

As regarded on the earlier sections, there are numerous approach, techniques, and model utilization to establish a Sentiment Analysis model, each with their own advantages over one another. Thereby, this segment will consist on discussing the various models and approach selected to establish the Sentiment Analysis models.

Aforementioned, this thesis study will be focusing on two main approaches, which is machine learning approach, and deep learning approach. The model's learning process shall employ on supervised learning, where the learning algorithms are provided on both the inputs and desired outputs – *English, Malay, and Code-mixed datasets*- to produce an inferred function. It works for the learning models to generalise from an inputted training data, and predict upon unseen situations (Kansara & Sawant, 2020). In this manner, the rest of the section below will describe and justify upon the selected machine learning and deep learning models.

### **3.3.1 Model Selection**

In the recent years, the establishments of Sentiment Analysis had been flourishing with great innovations and advancements. There are varying machine learning algorithms that had proven effectiveness in classifying sentiments from given inputs. Moreover, the contemporary advancement on Sentiment Analysis has been leveraging on the deep learning approaches, which even renders machine learning to be the “traditional approach”. Thereby in this thesis study, the two paradigms approach for establishing Sentiment Analysis shall both be selected and utilized.

#### **Machine Learning Models**

To begin with, the notion of machine learning is comprised with their own categorization of model types. In this thesis project, three of the model types is being utilized, namely a probabilistic classifier, linear classifier and a statistical classifier. The different categorization of classifiers is each being utilized in an attempt to assess which classifiers offers the optimum accuracy and performance by working with code-mixed dataset. Without further ado, the segments below discuss and justifies upon the selection of the three models.

First of all, a probabilistic classifier works on the idea that predicts the probability distribution across a collection of classes, as opposed to merely presenting the likely occurring class that the provided sample might belong to (Kansara & Sawant, 2020). A more commonly used algorithm for this kind of classification is the Naïve Bayesian classifier, which is selected as one of the models to be implemented upon this thesis project. The concept of Naïve Bayes model is grounded from Bayes' theorem regarding the independence assumptions among the predictors (Berrar, 2019). The conception on how Bayes' Theorem works as a classifier, is that it calculates on

the subsequent probability of an event (A), provided with a preceding probability of event (B), which will represent as an equation as below.

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)}$$

Essentially, the Bayes' theorem equation is explained as such, given that:

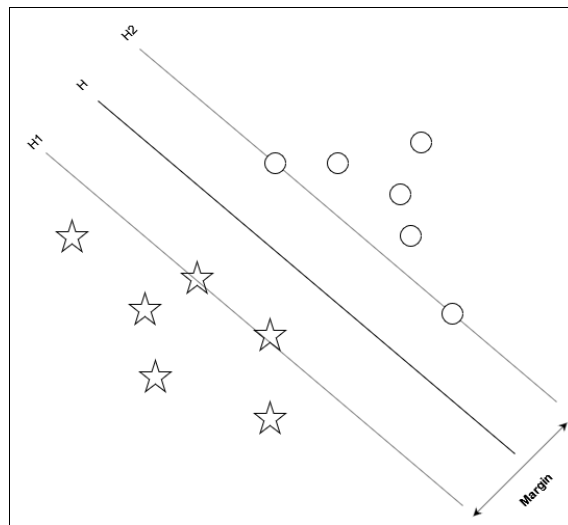
- A and B are both events
- P(A) and P(B) is the probabilities in discerning A and B, who are independent to one another.
- P(A | B) and P(B | A) are the conditional probabilities – *measure of the probability of an event occurring, given that a preceding event has already occurred-*, which in this case, P(A | B) is the probability of observing A, given that B is *true*. While P(B | A) is the probability of observing B, given that A is *true*.

By this means, the full equation states the probability of A, given that B is *true*, is equivalent to the probability of B, given that A is true, times the probability of A being *true*, divided by the probability of B being *true* (Lewis, 1998). In simpler terms, utilizing Bayes' Theorem allows a way of finding a probability, given that we know other probabilities. By utilizing on this equation, the Naïve Bayes model can calculate on the data or words against each other, computing on the probability of a data being in a Sentiment class – *Positive, Negative-*, given by the preceding training data inputted to the algorithm.

The Naïve Bayes model is regarded as a simple probabilistic classifier, which according to (Singh & Shahid Husain, 2014), is significantly easier to implement, and computes rather efficiently compared to other kinds of machine learning models. Rendering from (Ahmad et al., 2017), Naïve Bayes' classifier is considered the most efficient and effective inductive learning algorithm to be working through large amount of datasets. The selection of the Naïve Bayes model to be implemented as one of the Sentiment Analysis models, is frankly due to its simplistic nature, computing efficiency, and ease of implementation in general even for beginners within the machine learning field. The model here is being represented as an entry-level approach in the attempt to classify on the code-mixed datasets. The purpose and justification of

this model selection is simply to compare how an entry-level model performs in contrast to the subsequent more complex machine learning models.

Ensuing on to the linear classifier, it works on the idea where an operation that involves linear elements is performed to classify between the data classes. A commonly used technique in this matter, is a model called the Linear Support Vector Machine (LSVM), which is selected as the second model to be implemented in this thesis project. The notion on how LSVM works, is that it constructs hyper-planes within a multi-dimensional space that separates the different class boundaries (Ghosh et al., 2019). In the case where the data points are linearly separable, the concept is brewed from the structural risk minimization theory to locate the optimal separating hyperplane from the feature space (Liu et al., 2010). Thereby, the learning model is able to convene for a global optimization, and the estimated risk for the entire sample would be met with a certain upper bound. To simplify on the concept, below **Figure 3.5** illustrates the case of a linear SVM classification.



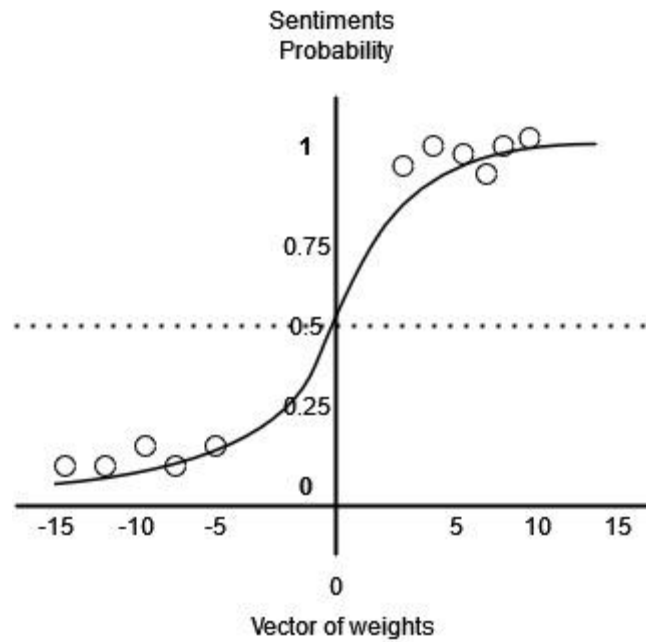
**Figure 3.5:** Concept of Linear Support Vector Machine

As shown above, the circles and the five-pointed stars each represents as two different types of data samples – *e.g: Positive & Negative*-.  $H$  is the classifying line/plane where it separates the two types of samples each onto their respective classes based on their features. Meanwhile,  $H1$  and  $H2$  are the lines that passes through the closest sample points it can find, which they are in parallel with the classifying line  $H$ . The two lines –  $H1$  and  $H2$ - are also known as the support vectors, which is utilized to compute the distance between each other and the classifying line,  $H$ . The distance

between support vectors is a parameter known as the margin, and the goal of SVM is to maximize the perpendicular distance of the margin, and constructing on an optimum hyperplane between the two classes, this is a concept known as the soft-margin formulation. Through the optimum hyperplane, it will minimize on the occurrence of generalization error (Ghosh et al., 2019). The attempt in SVM classification is to make a decision boundary – *the margin*- such that the separation between the two classes are as wide as possible, thereby the probability of a data sample being separated and classified within the incorrect class is as low as possible.

The Support Vector Machine (SVM) model is described as a supervised learning model that embodies excellent and efficient result in text categorization tasks, and it's even said to triumph above the Naïve Bayes model in performance and accuracy (Khairnar et al., 2013). According to (Ahmad et al., 2017), the extensions that SVM model offers are what makes it more efficient and adaptable to actual real-world usage. The soft-margin as mentioned prior is an example of the extension, which helps in ignoring any outliers and excessive noisy data. That is possible due to data at times being linearly visible on multi-dimensional problems, thereby they are able to be separate linearly (Khairnar et al., 2013). Either way, the LSVM model is selected due to its positive reputation in understanding and classifying the margin of dissociation between classes. This concept of LSVM would function relatively well in place such as Sentiment classification – *classify between two classes: Positive and Negative*-. This model is selected in an attempt to observe and rake a higher performance accuracy in classifying the code-mixed data.

The final machine learning model being utilized within this project, is on the statistical classifier. A statistical classifier conceptualizes on the approach that classifies unseen, and unlabelled data based on the relevance to prior known, labelled data. Logistic Regression is one of the statistical technique that analyses on the independent variables within a dataset that ultimately determine towards an outcome. It then probes for a best fitting model to depict on the relationship between the dependant and independent variables (Alzubi et al., 2018). To simplify, the depicted model will assign a probability scale to discrete the occurring outcomes between 1 and 0. Below **Figure 3.6** illustrates how the logistic regression algorithm fits the variables on a probability scaling.



**Figure 3.6:** Concept of Logistic Regression

To classify between Sentiments, the probability scaling can be moulded to depict point 1 as likely to be Positive, and point 0 to likely to be Negative. A cut-off at point 0.5 can be used to divide between the classes, where everything above point 0.5 is considered to be Positive, and everything under point 0.5 is considered to be Negative. Assuming that the model was pre-trained with vectorised words beforehand, where positive words carries a higher weight, and negative words carries a lower weight – *e.g: Positive +1, Negative -1* -. Thereby, if there were to have a total vectorised score of -13 from a sample sentence, it can be fit into the graph above, and be classified against based on the curve. The sample sentence will likely land on the Negative class.

The Logistic Regression model is also regarded as one of the simpler machine learning algorithms to be implemented, correspondingly has also proved to have great training efficiency in some cases. According to Alzubi et al (2018), logistic regression is dichotomous in nature – *Denoting suitability to only have two possible classes-*, which favours upon binary predictions such as predicting sentiments – *between 0s and 1s*. Additionally, rendering from Gladence et al., (2015) which states that Logistic Regression model will inherently give better result working on numerical data values. In that case, a vectorizer transformer – *ie: TF-IDF vectorizer-* could be utilized to transform the raw input data into numerical features, which theoretically will further improve on the performance and accuracy. Regardless on which, the Logistic

Regression model is selected appointed to the fact that it has positive reputations in predicting new and unseen data accurately, which is precisely what is needed to predict the “unseen” ground-truth code mixed datasets.

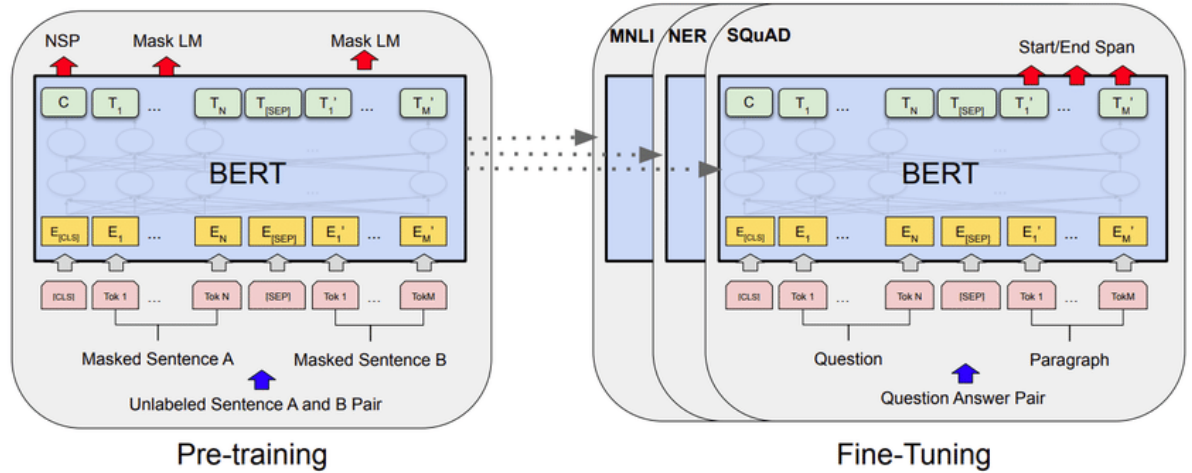
### **Deep learning model – BERT**

The notion of Deep learning is not an entirely new concept. Instead, it derives as being a more sophisticated and mathematically complex subset of machine learning (K. Jain & Kaushal, 2018). It rides off behind the success of Neural Network, where it utilized on its own layered structure of algorithms called the Artificial Neural Network (ANN) (Poomka et al., 2021). The contemporary advancement of deep learning had garnered a lot of attention towards it. That is because the design and utilization of Artificial Neural Network (ANN) had proved to be a learning process that has far more capabilities than any of the standard machine learning models prior.

Additionally, within the field of Natural Language Processing (NLP), deep learning has its own specific model called a language model, and what makes it exceptional, is that it allows the analysing of data to be in sequence form (Poomka et al., 2021). These kind of sequence based learning is rarely seen in a generic machine learning model, as it is designed to consider the context of each subsequent words in a sentence. In the case of this thesis project, a Bidirectional Encoder Representation from Transformer (BERT) model will be utilized. BERT is a pre-trained model developed by the Google AI team back in 2018. It is considered state-of-the-art, as the BERT intuition to prediction and classification is conceived to consider the word contextualization from all its surrounding – *left and right*- simultaneously (Jacob Devlin et al., 2018). This means that BERT will have an understanding access to the whole context to predict or classify, instead of only focusing on individual words. Thereby, the same word from BERT’s perspective could have different sentiments, based on its surrounding context – *e.g. This bread has gone bad; This bread taste not bad-*.

Applying BERT to fit for a specific use case is relatively straightforward. A pre-trained BERT model has already been trained on a huge unlabelled dataset. What is left to do, is to simply fine tune the model to match the desired use case, which in this case is Sentiment classification. The model can be fine-tuned simply by adding an additional classification output layer on top of the model (Jacob Devlin et al., 2018).

Also, by inputting labelled training data, so that it “transfers” the learning from the pre-trained unlabelled dataset, to the current labelled training dataset. Thereby, in order to apply BERT for this current thesis project, two BERT pre-trained models of different languages – *English & Malay*- is utilized. The pre-trained English BERT model is obtained from (Jacob Devlin et al., 2018), whereas the pre-trained Malay BERT model is obtained from (Zolkepli, 2018b).



**Figure 3.7:** Overall pre-training and fine-tuning procedure for BERT (Jacob Devlin et al., 2018).

### 3.3.2 Model Implementation

In conjunction to the prior segments, this section will denote on the implementation of the selected Sentiment Analysis models. It is intended to have a total of eight resulting Sentiment classifying models, which are as listed below **Table 3.5**.

**Table 3.5:** Models intended to be implemented

Language	Classifying models
English	Naïve_Bayes_Eng, LSVM_Eng, Logistic_Regression_Eng, BERT_Eng
Malay	Naïve_Bayes_BM, LSVM_BM, Logistic_Regression_BM, BERT_BM

Before affirming onto the implementation process, there are two important modules that is crucial to be stated beforehand, which are:

- TF-IDF Vectorizer



- Google Translation API module

As seen on the prior section, the chosen machine learning algorithms are all in-lined with mathematical elements – *Probability, Linear, Statistics*-. Which they would be expected to be dealing with numerical data such as 2-dimensional arrays, or numpy values. However, the problem with natural language data, is that they come in a form of raw text. Thereby, before applying the training data onto the selected machine learning algorithm, the data would first need to be represented in a numerical feature form, utilizing a process called Text vectorization. A common approach to this process is something called the Term frequency-inverse document frequency (TF-IDF) vectorizer, which is a text vectorizer that helps to transform raw text data into usable vectors.

The working notion for the TF-IDF vectorizer is to numerically measure the relevancy of words within a document (Addiga & Bagui, 2022). In which TF-IDF vectorizer combines on two concepts, as per their name states, it works on the term frequency and document frequency. The term frequency (TF) represents the number of times a specific term occurs in a document, which the score frequency given to a word by TF-IDF can be deduced on how important that word is for a document (Addiga & Bagui, 2022). The document frequency (DF) indicates on how common the specific term is throughout the document, adding Inverse to the mix, Inverse Document Frequency (IDF) acts to reduce the weight of common terms (Addiga & Bagui, 2022). Thereby, if a specific term is commonly scattered throughout a document, TF-IDF will deduce that the specific term has less weight, thereby being less important.

Moving on, the second important module is regarded on a Google translation API module named GoogleTrans. GoogleTrans is an open-source, free and unlimited python library that implemented on Google's own translation API, which the library is accessible through (Terryyz, 2020). The GoogleTrans module allows a direct call to the API to detect languages and translate them on any environment. This is necessary as within this thesis project, language detection is needed to detect the languages that exists within the code-mixed dataset. Subsequently, it also needs to translate the code-mixed dataset onto two separate languages –*English and Malay*- so that they can be fitted and classified onto their corresponding Sentiment Analysis models.

### 3.4 Evaluation Metrics

Evaluating the performance of a classification model is inherently an essential step within the machine learning and deep learning pipeline. It allows for an acknowledgement on the effectiveness and efficiency of a model, and it provides for more informed judgement on the model's applicability and reliability within a real-world scenario. The necessity of having evaluation metrics in this research is mainly to observe on how well a model is performing. These metrics can also allow the comparison between different models, which helps in identifying the strength and weaknesses of each model's performance. In this section, it is intended to discuss on the common evaluation metrics that is utilized, and why it is to be utilized in evaluating the proposed models. There are many different kinds of metrics that is used as the basis in evaluating a classification model. In this research's case, a totalling of five evaluation metrics will be discussed, and be considerate as the basis in evaluating the different models, namely: Confusion Matrix, Accuracy, Recall, Precision, and F1-Score.

To begin with, Confusion Matrix is a concept in which the name is derived that people get easily confused with this theory. However, it is regarded as one of the best evaluation metrics, which is also considered as the basis for all other metrics. The Confusion Matrix is able to provide a more insightful proposition not only on the performance on the model, but also able to envisage the classes that are being predicted correctly or incorrectly, and the type of errors that the model has made. To accentuate, a confusion matrix table helps summarizes the performance of a model in a tabled manner, where it displays the number of correctly and incorrectly predicted instances for each class (Hossin & Sulaiman, 2015). To put things into perspective, a typical confusion matrix is structured into a 2x2 four quadrants manner, which represents the True Positive (TP), True Negative (TN), False Positive (FP), and the False Negative (FN) (De Diego et al., 2022).

**Table 3.6:** Structure for Confusion Matrix (De Diego et al., 2022)

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

The True Positive (TP) notion represents the amount of instances where the model correctly predicts a positive outcome. This happens where the model says a data is positive, and that the data actual sentiment is actually positive. The False Positive (FP) cell represents the amount of instances where the model incorrectly predicts a positive outcome. This arises when the model mistakenly declares a data as positive, however the data actual sentiment is negative. Following on, the True Negative (TN) notion is also pretty self-explanatory, it is the amount of instances where the model correctly predicts a negative outcome, and where the model declares a data negative, and the data actual sentiment is negative. Finally, the False Negative (FN) is the amount of instances where the model incorrectly predicts a negative outcome. This arises when a model declares a data as negative, however the data true sentiment is actually positive.

The rationale of utilizing the confusion matrix as a means to evaluate the proposed models is mainly due to its strength and benefits. To start with, the confusion matrix is able to provide for a clear and concise summary of a model's performance, where it allows the identification on where the model is making mistakes, and which set of classes the model is struggling in predicting correctly (De Diego et al., 2022). Not only that, through analysing the outcome of the confusion matrix, it allows for the discovery on areas where the model will need improvements and fine-tuning on (Salmon et al., 2015). It also provides as a standardized measure to compare the performance between different classification models (Salmon et al., 2015), conceding the selection for the best model for a particular task. In conjunction to the previous mentioned notions (TP, TN, FP, FN), by applying these four notions, it can be further extended to various evaluation metrics that help provide insights to the performance of the proposed classification models. The more common metrics that derives from the confusion matrix include on Accuracy, Precision, Recall, and F1-score (De Diego et al., 2022).

To continue on, accuracy is regarded as one of the most commonly used measures to be utilized with classification problems (Hossin & Sulaiman, 2015). In this research's case, it is used as the measure of how well the models are classifying the sentiments, basically in defining the ratio or percentage of correctly classified instances over the total number of instances within a dataset (Hossin & Sulaiman, 2015). The measure of accuracy can be represented with the following formula.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

( 1 )

Being acknowledge as one of the most straightforward measures, accuracy can be understood as the number of correct predictions, divided by the total number of predictions. For instances, if a dataset consists of 1000 data, and a given model correctly classifies 900 of them, it is said that the model will have an accuracy of 90%. However, although the notion of accuracy sounds pretty straightforward, the perception of always having “accurate” prediction can sometimes be misleading. With the case in point, the accuracy measure can be useful when the target class is well balanced, but is not the case with unbalanced classes (Hossin & Sulaiman, 2015). For instance, if 90% of a dataset is annotated with Positive sentiment, having a model that always predicts positive will still be able to achieve the accuracy of 90%, even though it does not actually classify anything correctly. Hence, to be able to gain a full picture on a model’s evaluation, other metrics should also be put into perception.

Subsequently, precision is a metric that measures the percentage of the true positive predictions over the total amount of positive predictions made by a model (Hossin & Sulaiman, 2015). Precision as a metric exemplifies the ability of a model to correctly predict positive sentiments without falsely classifying negative sentiment data as positive. For instance, if a model predicts that a data is positive and the actual ground truth label is also positive, this is considered a true positive. If the model predicts a text is positive, but the actual ground truth label is negative, this is considered a false positive. Precision is calculated as the ratio of true positives to the sum of true positives and false positives (Hossin & Sulaiman, 2015), which can be structured with the following formula.

$$Precision = \frac{TP}{TP + FP}$$

( 2 )

Equivalently, recall is also used as a measure in a similar fashion. It is a metric that measures the percentage of true positive predictions over the total amount of positive samples in a dataset (Hossin & Sulaiman, 2015). It exemplifies the ability of the model to be able to correctly identify all the positive sentiments within a dataset (Hossin &

Sulaiman, 2015). For occasion, if there are 100 positive instances within a dataset, and a model is able to correctly identify 80 of them, then the recall of the model is stated to be 80%. Recall is calculated as the ratio of the true positives to the sum of true positive and false negative, which is structured as the following formula.

$$Recall = \frac{TP}{TP + FN}$$

( 3 )

Ultimately, the last metric to be put in the discussion is the F1-score. The F1-score is regarded as the harmonic mean that balances both the concern of precision and recall, providing a single score that evaluates a models performance (Hossin & Sulaiman, 2015). The F1-score can be devised with the following formula.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

( 4 )

A higher F1-score generally indicates that a model is better in correctly predicting positive sentiments while avoiding false positives (De Diego et al., 2022). Thereby, a good F1-score entails having low false positives and low false negatives, so a model will be precisely identifying real threats and not be bothered with false alarms (De Diego et al., 2022). Ideally, a perfect F1-score is considered to be 1, and a model will be considered a total failure when the F1-score is 0.

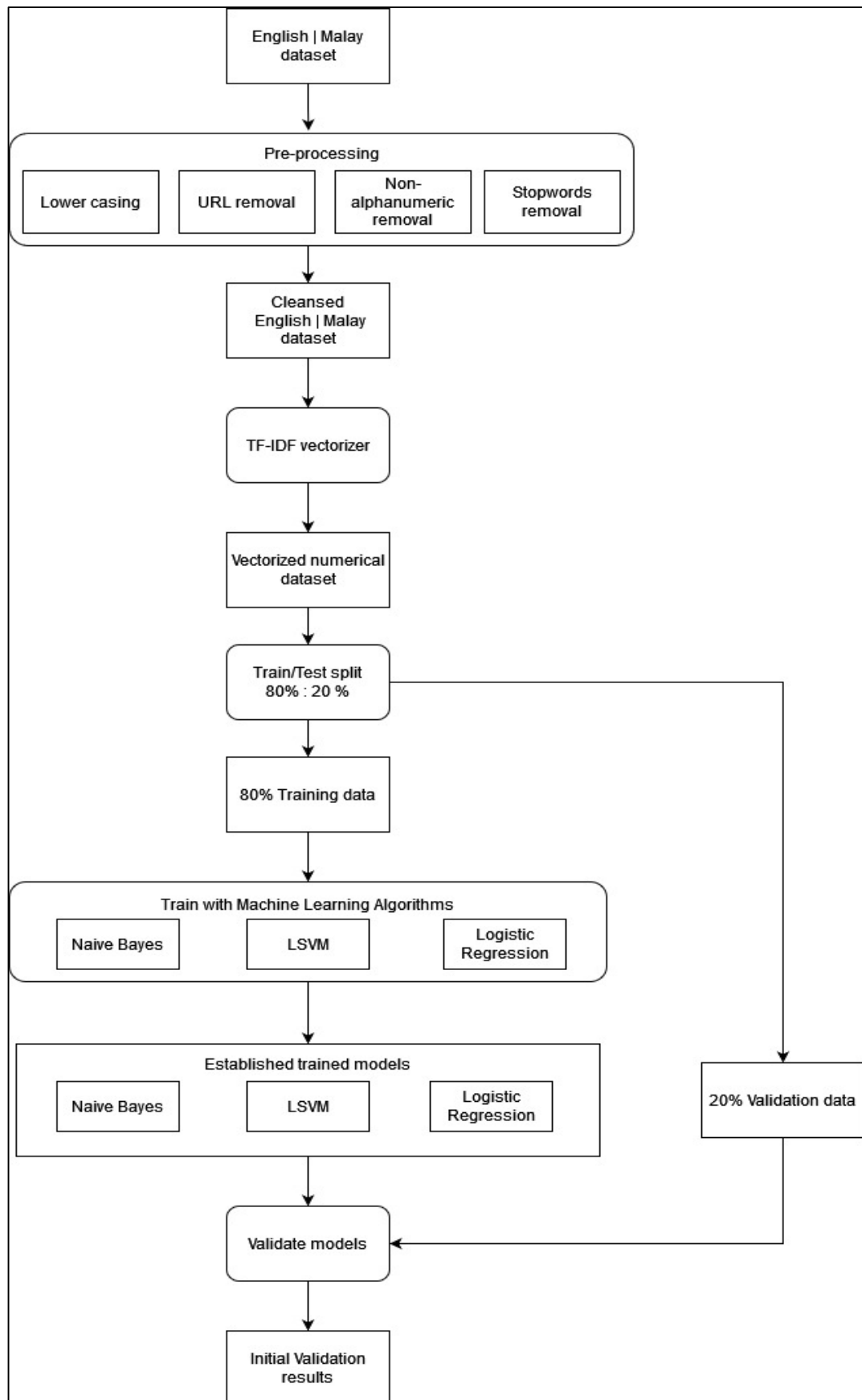
## Chapter 4

### Experiment and Implementations

This chapter presents the implementation details of the research in conducting the sentiment analysis model for code-mixed data. Throughout the chapter, it will focus on the implementation of various machine learning models, BERT models. Additionally, an improvement model which integrates both machine learning and BERT approach, including with a voting system approach will be discussed. This chapter will provide a detailed description of the implementation steps for each model, including data pre-processing, feature extraction, model training, and evaluation metrics. The outcomes of this chapter will serve as a foundation for the subsequent result and analysis chapters, where the performance and effectiveness of each implemented model will be assessed. The findings from these evaluations will contribute to our understanding of sentiment analysis on code-mixed data and provide valuable insights for further research and practical applications in multilingual sentiment analysis.

#### 4.1 Implement Machine Learning models

Apropos towards the machine learning model implementation, an overview of the machine learning model's framework can be ascertained as shown on below **Figure 4.1**.



**Figure 4.1:** Implementation Framework for machine learning models

As referred above, various supervised machine learning techniques are applied on the two different datasets, each comprising of differing languages (English & Malay). To summarize on the implementation framework, the two mentioned datasets are firstly pre-processed into a cleansed dataset. They are then transformed into a numerical vectors, and will be split in a ratio of 80:20 to act as training data and testing data respectively. The 80% split training vectors are then fed and processed by the different machine learning techniques to construct the respective ML classification models. Irrevocably, the performance of the produced ML models is assessed through different parameters, comprising on testing through the 20% split test dataset, and also through the ground-truth dataset. To further elaborate on the ML model implementing framework, a stepwise detailed elaboration is depicted on the latter segment.

### Step 1: Dataset preparation

- The prior mentioned dataset Sentiment-140 (English), and Sentiment-Bahasa (Malay) is prepared in advance. The Sentiment-140 dataset is a large dataset which consist of a total 1.6 million data, equally divided onto 800k positive data and negative data. The Sentiment-Bahasa is a relatively smaller dataset. They were both consisted into separate .csv files, and was read into a dataframe by utilizing on Python's pandas data analysis library.

```
[ ] data_eng = pd.read_csv(r'/content/drive/MyDrive/data/S140data.csv', encoding = 'latin', header=None) #import the english dataset
data_eng.columns = ['sentiments', 'id', 'dateTime', 'query', 'username', 'tweets'] #rename the column header
data_eng.head()
```

	sentiments	id	dateTime	query	username	tweets
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

```
[ ] data_bm = pd.read_csv(r'/content/drive/MyDrive/data/BM_data_Sentiments.csv', encoding = 'latin')
data_bm = data_bm[['sentiments', 'tweets']]
# data_bm.columns = ['sentiments', 'tweets']
# data_bm = data_bm.iloc[1:]
data_bm.head()
```

	sentiments	tweets
0	Negative	perlukan pelukan
1	Negative	que me muera
2	Negative	Tidak, mereka tidak memilikinya
3	Negative	bukan kru keseluruhan
4	Negative	seluruh tubuh saya berasa gatal dan seperti api

**Figure 4.2:** Loading dataset to panda's dataframe

- The order of the annotated dataset is randomized throughout the dataframe by using the DataFrame.sample() function. As the original dataset has the data categorized by their annotated sentiments, performing the randomization is to ensure that the latter training and testing set will consist of random selection of



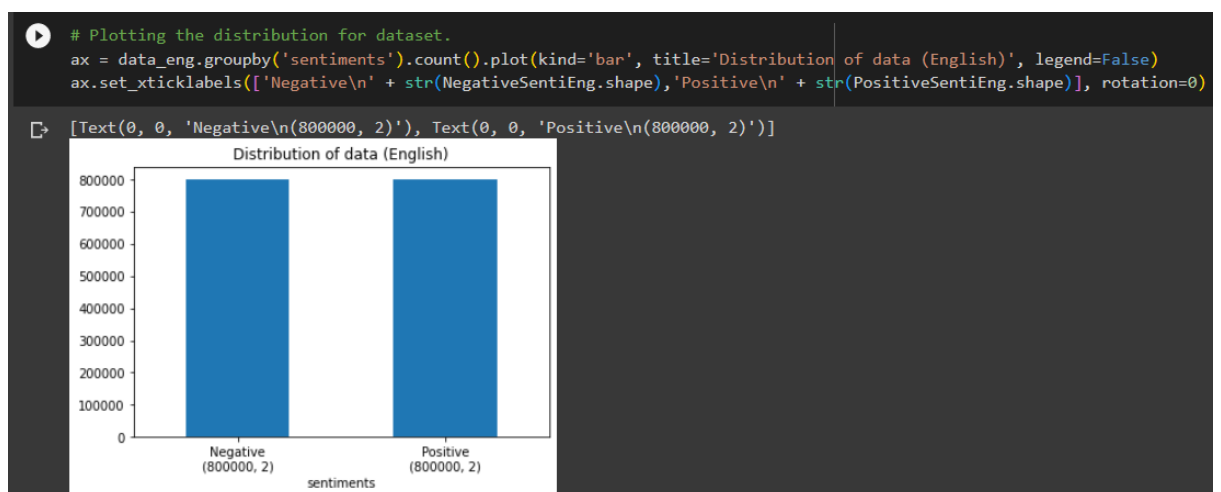
Positive and Negative data, and not having the same sentiments bundled together.

```
data_eng_rand = data_eng.sample(n=10, random_state = 1)
data_eng_rand
```

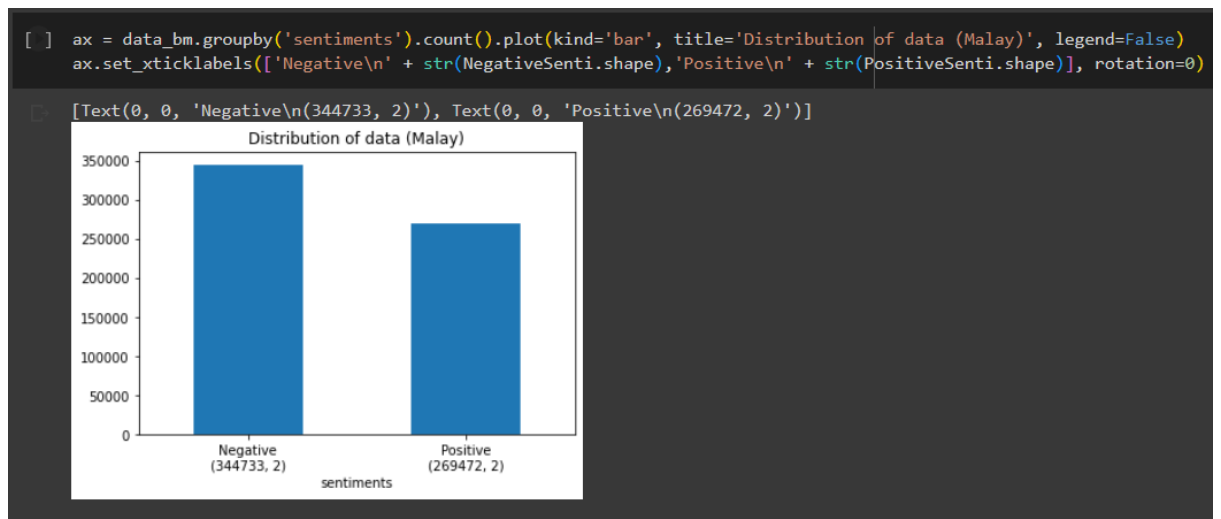
	sentiments	tweets
514293	0	i miss nikki nu nu already shes always there ...
142282	0	So I had a dream last night. I remember a sig...
403727	0	@girlyghost ohh poor sickly you (((hugs)) ho...
649503	0	it is raining again
610789	0	@MissKeriBaby wish I was in LA right now
67315	0	Nala Olowalu still has a full tummy from bread...
833521	1	@macintom site doesn't seem to want to load up...
256032	0	time for some sleep- hav to actually do some w...
657012	0	@supercoolkp In Oxford that month.
980587	1	..time for a cup of tea and fruit bagels, i'm ...

**Figure 4.3:** Randomize dataset with sample function

- The dataframes are then visualized into a barchart to examine on the data distribution based on the annotated sentiment class. A Python data visualisation library, matplotlib is utilized to plot the data from the dataframe into an observable bar chart. Obervation and analysis is then done on the chart to determine whether if there consist of any class imbalance. If there is a class imbalance on the dataset, a downsampling approach is done by utilizing the DataFrame.concat() function, where the oversampled class is reduced to match an equal number of sample size between classes.



**Figure 4.4(a):** Visualize english dataset class distribution



**Figure 4.4(b):** Visualize malay dataset class distribution

## Step 2: Dataset pre-processing

The prepared dataset on this step still consist of dirty and unclean data, which will be needed to be removed, clean, or altered before it is considered to be fed onto the machine learning algorithm. The pre-processing algorithms that is used for the cleaning the dataset is as such:

- Lower casing is a pre-processing technique that is commonly used in NLP to convert all the text within a dataset to lowercase. This is performed to ensure the consistency within a dataset, and also to reduce the amount of unique words that consists within a dataset. Thereby, less computing resource is needed while training the models. This technique is achieved through utilizing the built in Python `str.lower()` function, where it takes in a string input and returns the string with all its characters in lowercase.
- URL and Non-Alphanumeric removal is another technique that is utilized to remove any instance of URLs or non-alphanumeric characters – *Symbols, emojis, etc-* from the dataset. This is performed to reduce the instances of noisy data that lurks within the dataset, and is only applicable to this research’s case as URLs and non-alphanumeric characters are not relevant to the analysis. This technique can be achieved through Python’s regular expression (re) module, where it provides a means to examine and match patterns in strings. The ‘`re.sub()`’ function is then utilized to search and replace any instances of URLs or non-alphanumeric characters that consist within a text string. Any instances that matches the regular expression patterns ‘`((http://)[^ ]*(https://)[^ ]*( www\.)[^ ]*)`’ and ‘`^[a-zA-Z0-9]`’ is replaced

with an empty string, effectively removing any kinds of URLs or non-alphanumeric characters in a data.

- Stopword removal is also another pre-processing technique that is used to eliminate the commonly occurring words within a text data. Stopwords offers minimal to no impact towards the overall meaning of a sentence, and they can be removed without really affecting the overall accuracy of a model's analysis. The rationale of performing this technique is to reduce to size of a dataset, and in turn improving the efficiency of the training algorithm, and also enhancing the quality of the text analysis by removing noisy and irrelevant data.

For the English dataset, this technique can be achieved through utilizing the Python's built-in library, Natural Language Toolkit (NLTK). The NLTK library offers a module that consist a corpus of English stopwords, which allows for the matching between the dataset and stopwords corpus in order to filter and remove them completely. An algorithm is written that splits and iterates through each word within the dataset, and compares them to the stopwords corpus. If a stopwords is found within the dataset, the particular word is filtered out, and the processed text data will be appended into a new dataframe that is clear of stop words.

For the Malay dataset, this technique can be achieved through utilizing the "PySastrawi" library. This library offers modules that consist of Indonesian and Malay stopwords, which allows for the matching and filtering of stopwords within the Malay dataset. To remove the malay stopwords, a 'StopWordRemovalFactory' module needs to be imported from the PySastrawi library, and the object will be created to obtain the list of Malay stopwords. Further on, the data text is defined and stopwords are removed from it by utilizing a list comprehension. Finally, the remaining words are joined together and is appended into a new dataframe that is clear of Malay stopwords.

---

**ALGORITHM 4.1: DATA PREPROCESSING**

---

**Input:** Text data to be preprocessed (e.g. englishDF, malayDF)

**Output:** Pre-processed text data

---

- 1: Import regular expression (re)  
Import nltk  
From nltk import stopwords  
Import Sastrawi

---

From Sastrawi import StopWordRemovalFactory

```
2: Define URL pattern = “((http://)[^ ]*|(https://)[^ ]*( www\.)[^ ]*)”
3: Define non-alphanumeric pattern = "[^a-zA-Z0-9]"

4: For each data in input text data:
5:     Lowercase all characters of data, data.lower()
6:     Remove URL instance from data, re.sub(URL)
7:     Remove non-alphanumeric instance from data, re.sub(non-alpha)
8:     For each words in data:
9:         Filtered_text = join() the words that doesn't match in the stopwords
           corpus
10: Return Filtered_text
11: End
```

---

### Step 3: Split training and testing data

The split train-test technique is one of the common practices in a machine learning framework to establish the model, as well as to evaluate the model. This technique involves in splitting the dataset into two sets, which is the training set, and also the testing set. The training set is used for the training of the model, while the testing set is used to assess the performance of the model towards unseen data. On this research, a common split ratio is utilized, which is 80% for training and 20% for testing. The splitting of data is achieved through utilizing the Python's scikit-learn library, and importing on the `train_test_split()` module.

```
[ ] X_train_eng, X_test_eng, y_train_eng, y_test_eng = train_test_split(processedtext_eng, sentiments_eng, test_size = 0.20, random_state = 0) #split 80% : 20% train test dataset
[ ] X_train_bm, X_test_bm, y_train_bm, y_test_bm = train_test_split(processedtext_bm, sentiments_bm, test_size = 0.20, random_state = 0) #split 80% : 20% train test dataset
```

**Figure 4.5:** Splitting data to 80:20 train/test

### Step 4: Data Vectorisation:

TF-IDF (Term Frequency-Inverse Document Frequency) is another commonly used technique in NLP that represents text data into a numerical form. This is done so that the vectorised data can be easily processed by the machine learning algorithm. To achieve this technique, the `TfidfVectorizer` class from the scikit-learn library is imported and utilized. In this research's case, the preprocessed data in the respective dataframe is fit into the instance of the `TfidfVectorizer` class, and was transformed into

a TF-IDF weighted matrix. Subsequently, the transformed vectorised data is fit into a new dataframe and is ready to be utilized upon.

---

**ALGORITHM 4.2: TF-IDF VECTORIZER**

---

**Input:** training or testing data to be vectorised (e.g. `x_train_data`)

**Output:** vectorised data in a dataframe

---

```
1:  From sklearn.feature_extraction.text import TfidfVectorizer

2:  Vectorizer = TfidfVectorizer()
3:  Vectorizer.fit(x_train_data)
4:
5:  X_train_data = Vectorizer.transform(X_train_data)
6:  X_test_data = Vectorizer.transform(X_test_data)
```

---

### Step 5: Model training

In conjunction with step 4, once the required training data is transformed into a TF-IDF matrix, it can be proceeded to be fit into machine learning algorithms to train the desired classification models. The model's training algorithm can also be attained through the sklearn() python library, in this research case, the algorithm for Gaussian Naïve Bayes, Support Vector Machine, and Linear Regression is utilized. Below pseudocode shows the fitting and training of the dataset for the different machine learning algorithms.

---

**ALGORITHM 4.3: TRAIN MODELS**

---

**Input:** vectorised training data (e.g. `x_train_data`)

**Output:** trained model

---

```
1:  From sklearn.linear_model import GaussianNB
    From sklearn.linear_model import SVC
    From sklearn.linear_model import LinearRegression

2:  NBModel = GaussianNB()
3:  SVMMModel = SVC()
4:  LRModel = LinearRegression()
5:
6:  NBModel.fit(X_train_data, Y_train_data)
    SVMMModel.fit(X_train_data, Y_train_data)
    LRModel.fit(X_train_data, Y_train_data)
```

---

## Step 6: Validate models

Once the models are trained, the performance of the models can initially be evaluated through the split test set. This is done so to generalize on how the trained models interpret on new and unseen data. The validation results can be compute through the various evaluation metrics such as accuracy, precision, recall, and F1-score. This is again achieved through utilizing the sklearn library, where it provides the metrics() module to get access to the stated evaluation metrics. By utilizing the split test set, the model's performance can be computed through with the sklearn.predict() function and the evaluation functions – e.g. *accuracy\_score(y\_test, y\_pred)*, *precision\_score(y\_test, y\_pred)* -. The below tables are the results obtained through validating with the split test dataset.

---

### ALGORITHM 4.4: EVALUATION MODELS

---

**Input:** Model to be evaluated, test data, validation data

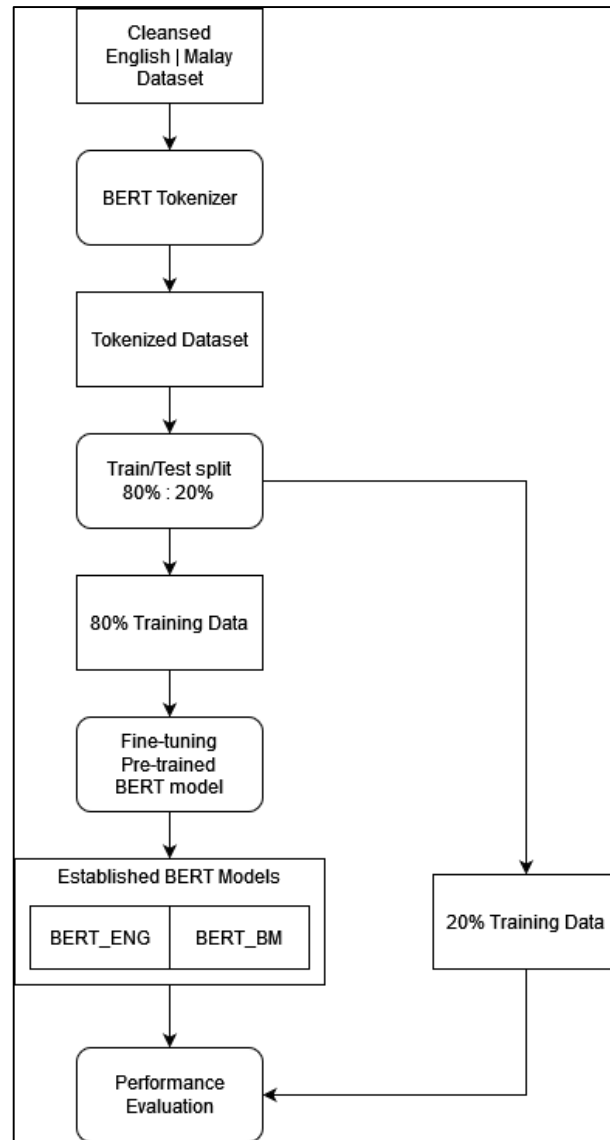
**Output:** Accuracy, Precision, Recall, Confusion Matrix

---

- 1: *y\_pred = model.predict(X\_test) # Predict values for Test dataset*
  - 2: *# Print the evaluation metrics for the dataset.*  
*print(classification\_report(y\_test, y\_pred))*
  - 3: *# Compute and plot the Confusion matrix*  
*cf\_matrix = confusion\_matrix(y\_test, y\_pred)*
  - 4: *# Calculate percentage values for each matrix cell*  
*group\_percentages = ['{0:.2%}'.format(value) for value in cf\_matrix.flatten() / np.sum(cf\_matrix)]*
  - 5: *# Plot the heatmap with labeled cells*  
*sns.heatmap(cf\_matrix, annot = labels, cmap = 'Blues', fmt = "",*  
*xticklabels = categories, yticklabels = categories)*
  - 6: *# Set labels and title for the plot*  
*plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad = 10)*  
*plt.ylabel("Actual values" , fontdict = {'size':14}, labelpad = 10)*  
*plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)*
- 

## 4.2 Implement BERT models

In connection with the previous framework, an overview of the BERT model's framework can be perceived as shown on the below **Figure 4.6**.



**Figure 4.6:** Implementation Framework for BERT Models

As presented above is the implementation framework outlined to prepare and train the Sentiment Classification BERT models. The above framework is delineated to fine-tune pre-trained models to make it perform as the research purpose intended, which is to establish BERT models for sentiment classification. In order to achieve this, the stepwise detailed elaboration is depicted on the below segments.

### **Step 1: Dataset preparation**

The data preparation phase is always recommendable to start with the data preprocessing technique. Both the BERT models are trained using the same dataset as the machine learning models, which is the Sentiment-140, and Sentiment-Bahasa. As the previous machine learning implementation had already implemented

preprocessing on the datasets, it can be utilized again directly even on this BERT implementation. To iterate, the preprocessing technique used on the datasets are Lowercasing, URL removal, Non-alphanumeric characters removal, and stopwords removal.

## **Step 2: BERT AutoTokenizing**

- In order to train or fine-tune a BERT model, it would require on tokenized inputs, where each words in a dataset is represented as a unique token. In order to achieve on this, a library called the Hugging Face Transformer library can be utilized upon. The Hugging Face library is represented by its wide and notable set of transformers library built for the various natural language processing applications, and utilizing on this library will allow for this research to import on the AutoTokenizer module to tokenize the required dataset with the pre-trained model.
- The BERT tokenizer is initialized together by the specific pre-trained BERT model this research is using on, which is ‘BERT-base-case’ and ‘BERT-base-bahasa-cased’. The tokenizer is utilized by calling the ‘BertTokenizer’ class from the ‘transformers’ huggingface library.
- To employ the tokenizer onto the input datasets, the function ‘tokenizer.encode\_plus()’ is utilized to convert any data into input IDs, Token type ID’s, and attention masks. The input IDs are the representation of the tokenised data, whereas the attention masks are indicators for which tokens that should be attended to. Lastly, the token type IDs are what helps differentiate between sentence pairs.
- Through the BERT tokenizer, there are typically two types of embeddings to be obtained out of it, which is the token embeddings, and special token embeddings. The token embeddings are representatives of the individual tokens from the input data, where these embeddings are pre-trained as part of the ‘BERT-base-case’ model and allows for the capture of meaning and context of each token within the input sequence. It is also needed for special tokens embeddings to the input sequences, such as the ‘[CLS]’ token and the ‘[SEP]’ token. Which are tokens associated with special classification embeddings, as well as sentence pair separation tokens for sentence classifications.



- The special tokens are utilized by adding them to the input sentences. The '[CLS]' token is added to the beginning of the input sentence, while the '[SEP]' token is added to the end of the input sentence. Subsequently, a padding special token called the '[PAD]' token is also added to the input sentence. The '[PAD]' token is utilized to ensure that all the input sentences will be the same length. Lastly, an unknown token called the '[UNK]' token is added to everything else that is not within the training set.

```
[ ] tokenizer = BertTokenizer.from_pretrained(PRE_TRAINED_MODEL_NAME)
```

```
tokens = tokenizer.tokenize(sample_txt)
token_ids = tokenizer.convert_tokens_to_ids(tokens)

print(f' Sentence: {sample_txt}')
print(f'   Tokens: {tokens}')
print(f'Token IDs: {token_ids}')

> Sentence: When was I last outside? I am stuck at home for 2 weeks.
   Tokens: ['When', 'was', 'I', 'last', 'outside', '?', 'I', 'am', 'stuck', 'at', 'home', 'for', '2', 'weeks', '.']
   Token IDs: [1332, 1108, 146, 1314, 1796, 136, 146, 1821, 5342, 1120, 1313, 1111, 123, 2277, 119]
```

#### ▼ Special Tokens

[SEP] - marker for ending of a sentence

```
[ ] tokenizer.sep_token, tokenizer.sep_token_id

(['SEP'], 102)
```

[CLS] - This token is added to the start of each sentence, so BERT knows we're doing classification

```
[ ] tokenizer.cls_token, tokenizer.cls_token_id

(['CLS'], 101)
```

There is also a special token for padding:

```
[ ] tokenizer.pad_token, tokenizer.pad_token_id

(['PAD'], 0)
```

BERT understands tokens that were in the training set. Everything else can be encoded using the [UNK] (unknown) token:

```
[ ] tokenizer.unk_token, tokenizer.unk_token_id

(['UNK'], 100)
```

**Figure 4.7:** BERT autoTokenizer and special tokens

### Step 3: Loading datasets

- In order to have the datasets to be effectively loaded and processed during training, data loaders are needed to be created, so that it provides for an interface that iterates over the dataset in batches, data shuffling, and also performing any necessary data transformations. To achieve this, a Pytorch library is utilized upon together with its data loaders module.
- To initialize on the Pytorch data loaders, a Pytorch dataset class is implemented that encapsulates the preprocessed datasets so that it can provide an interface that is accessible to individual samples.
- The data loaders are typically used in the pre-training phase and also the fine-tuning phase. For this case, the data loader is utilized to load and batch process the tokenized dataset. It performs as an iterator and also helps conserve memory throughout the training process, where it ensures that the fine-tuning data is processed efficiently in batches and passed to the BERT model for training or inference. It basically works on easing and improving the efficiency of fine-tuning the BERT model.

Split the data:



```
df_train, df_test = train_test_split(df, test_size=0.1, random_state=RANDOM_SEED)
df_val, df_test = train_test_split(df_test, test_size=0.5, random_state=RANDOM_SEED)
```

```
[ ] df_train.shape, df_val.shape, df_test.shape
```

```
((11250, 2), (625, 2), (625, 2))
```

```
Create a couple of data loaders:

def create_data_loader(df, tokenizer, max_len, batch_size):
    ds = TweetDataset(
        reviews=df.tweets.to_numpy(),
        targets=df.sentiments.to_numpy(),
        tokenizer=tokenizer,
        max_len=max_len
    )

    return DataLoader(
        ds,
        batch_size=batch_size,
        num_workers=4
    )

[ ] BATCH_SIZE = 16

train_data_loader = create_data_loader(df_train, tokenizer, MAX_LEN, BATCH_SIZE)
val_data_loader = create_data_loader(df_val, tokenizer, MAX_LEN, BATCH_SIZE)
test_data_loader = create_data_loader(df_test, tokenizer, MAX_LEN, BATCH_SIZE)
```

**Figure 4.8:** BERT data split and dataloaders

#### Step 4: Fine-tune BERT model

- In order to fine tune the BERT model, a pre-trained BERT model would need to be loaded and trained with specific sentiment analysis tasks using annotated data. Through fine-tuning, it will allow the model to learn the tasks specific representations, and also improve the model's performance on sentiment classifications.
- A pre-trained BERT model can be loaded through utilizing the hugging-face transformer library. In this research's case, the pre-trained model 'BERT-base-case' and 'BERT-base-bahasa-cased' is utilized upon.
- An optimiser is also needed to be chosen to update the BERT model's parameters, based through the computed gradients during the fine-tune training. A commonly used optimizer called the Adam (Adaptive Moment Estimation) Optimizer is chosen for this research, which has the benefits of adaptive learning rate and also momentum. It is optimal for this research as it tends to work well with a wide range of tasks, such as sentiment classification for instance. This optimiser is also obtained from the hugging face library.

- After the optimiser is loaded, a linear learning rate scheduler is utilized through using the `get_linear_schedule_with_warmup()` function from the transformer package. The epochs for the training is set to 3, which is the total number of iterations of all the training data in one cycle for training the BERT model. The learning rate is then set to a relatively low value of  $3e-5$ , this is due to fine-tuning a BERT model typically requires lower learning rates compared to training from scratch since BERT models have already learned on some rich representations. Utilizing a smaller learning rates can help prevent some catastrophic forgetting of pre-trained knowledge.
- A function is then written to evaluate and record the model's validation loss and validation accuracy upon each trained EPOCHs. The best state of the model upon each trained EPOCHs is then stored onto the local drive as a '.bin' file, which can be later reuse to classify data.

---

**ALGORITHM 4.5: FINE TUNE BERT MODEL**


---

**Input:** BERT base model, data loader,

**Output:** fine-tuned BERT model

---

- 1:** *#Load the pre-trained 'bert-base-case' model and tokenizer.*  
`tokenizer = BertTokenizer.from_pretrained('bert-base-case')`  
`model = BertForSequenceClassification.from_pretrained('bert-base-case', num_labels=2)`
- 2:** *Prepare the training and validation data through loading dataframe.*
- 3:** *#Set batch size and create data loaders for training and validation.*  
`batch_size = 16`  
`train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)`  
`val_loader = DataLoader(val_dataset, batch_size=batch_size)`
- 4:** *#Set up the optimizer and learning rate scheduler.*  
`optimizer = AdamW(model.parameters(), lr=2e-5)`
- 5:** *#Move the model to the appropriate device (GPU if available).*

```

device = torch.device('cuda') if torch.cuda.is_available() else
torch.device('cpu')
model.to(device)

6:  #Start the training loop for a specified number of epochs.
    num_epochs = 3
    for epoch in range(num_epochs):
7:      #Set the model to training mode.
        model.train()

8:      #Iterate over batches of training data.
        for batch in train_loader:
9:          input_ids = batch['input_ids'].to(device)
10:         attention_mask = batch['attention_mask'].to(device)
11:         labels = batch['labels'].to(device)
12:         optimizer.zero_grad()
13:         outputs = model(input_ids, attention_mask=attention_mask,
                           labels=labels)
14:         loss = outputs.loss
15:         loss.backward()
16:         optimizer.step()
17:  #Save the fine-tuned model for future use.
    torch.save(model.state_dict(), 'fine_tuned_bert_model.pt')

```

---

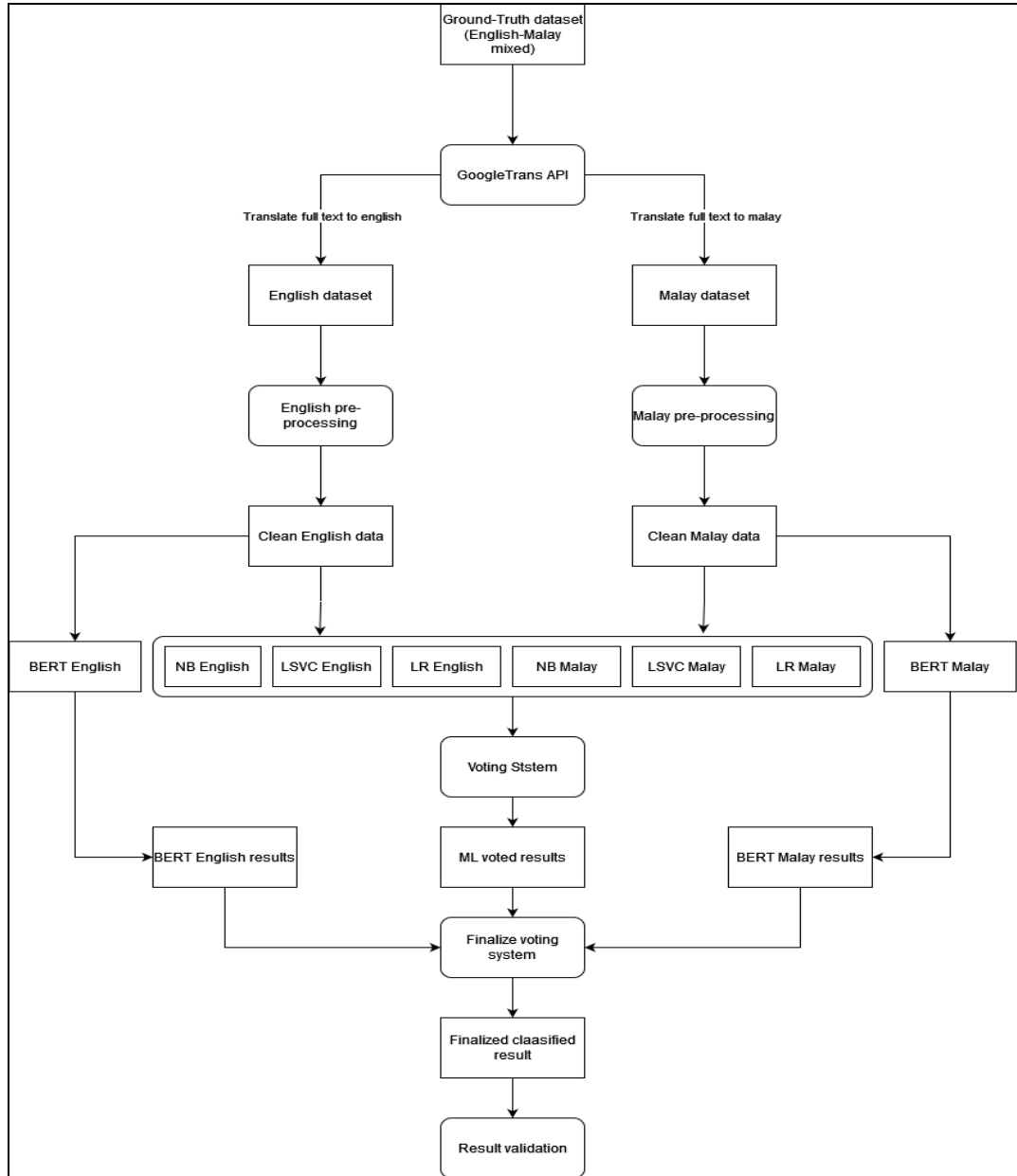
## Step 5: Validate BERT Models

Just as how the machine learning models are, once the BERT models are trained, the model's performance will be initially evaluated through the split test set. The validation results can be compute through the various evaluation metrics such as accuracy, precision, recall, and F1-score. This is again achieved through utilizing the sklearn library, where it provides the metrics() module to get access to the stated evaluation metrics. By utilizing the split test set, the model's performance can be computed through with the sklearn.predict() function and the evaluation functions – e.g. `accuracy_score(y_test, y_pred)`, `precision_score(y_test, y_pred)` -.

### 4.3 Proposed Code-mixed Sentiment Classification Model

In the previous segment, this research explored on the implementation of the different machine learning models and the BERT models for Sentiment Analysis classification task. While the implemented models individually exhibit on promising results, this

research true target was to enhance the sentiment classification accuracy and robustness through an integrated improvement model. This section introduces an approach that integrates the previous implemented machine learning models and BERT models utilizing a voting system. The voting system is an approach that leverages the collective decision-making power of the different models to arrive at a more reliable sentiment classification result. The below diagram presents the framework of the proposed improvement model.



**Figure 4.9:** Implementation Framework of the Ensemble/Integration Model

While the machine learning models and BERT models have demonstrated their effectiveness in classifying the sentiments, they would still possess on certain limitations when they are utilized in isolation. Take the machine learning models for instance, it can be observable that these models often struggle to capture intricate linguistic nuances and context dependencies, which are often times seen on social media, such as sarcastic sentences which sometimes could be negative in nature but reads like a positive sentiment sentence. Meanwhile with BERT models, although they are powerful, they are computationally expensive to train and fine-tune (Taneja & Vashishtha, 2022). Thereby, they often times are trained on a smaller-sized dataset which would cause the model to not be exposed to wider corpus and contextual dependencies between wordings, making it to perform not as optimally and might struggle with domain-specific sentiment analysis tasks due to limited domain adaptation during pre-training (Taneja & Vashishtha, 2022). By this means, this research is trying to address these limitations by combining these models, and proposes a model that combines the strengths of both the Machine Learning and BERT approaches. Through leveraging the interpretability of traditional machine learning models and the strong contextual understanding of BERT models, we can create a significantly robust sentiment analysis system. By implementing on a voting system, it will serve as a mechanism to aggregate the individual predictions of each model and determine the final sentiment classification, garnering a collective decision between the models, achieving an improved sentiment classification outcome. The following segments presents the stepwise elaboration in implementing the proposed model.

### **Part 1: Preparation of dataset**

The dataset used on this implementation is slightly different than what is utilized before. This dataset consists a collective of sentences that are code-mixed in English and Malay. The collected dataset derives from a code-mixed corpus created by (Romadhona et al., 2022), where they coined as being the first Malaysian code-mixed Sentiment dataset, called the SentiBahasaRojak. It consisted a total of 2,285 code-mixed data, and these data are loaded onto a dataframe as a preparation for further utilization.

### **Part 2: Data Translation**

To address on the challenges posed by the code-mixed datasets, this research will be utilizing on a translation technique to convert the code-mixed data onto two specific

languages, which is English and Malay. By translating the data into the two languages, it will allow to leverage the previously implemented language specific models – *Linear Regression English*, *BERT English*, *BERT BM*, etc- and obtain a more accurate sentiment prediction from these models. To achieve on this, this research will be utilizing a Python translation library called GoogleTrans. GoogleTrans is a free and unlimited Python library that implements on Google Translate API, allowing to make call methods such as language detection and language translation.

To begin with the translation, a single text translation function is written that applies the Translator() module from the GoogleTrans library. Then by utilizing on the translate() function, it will allow for the specification on the input language, and destination language to be translated to. After the function is established, the SentiBahasaRojak dataframe is utilized and iterated over each column to apply the translation function. The function will translate each value in every column of the DataFrame to the specified destination language, which in this research will be specifying on the ‘en’ and ‘ms’ destination languages, which stands for English and Malay respectively. Both the translated datasets will then be saved onto separate .csv files for further utilizations.

---

**ALGORITHM 4.6: TRANSLATION**

---

**Input:** Code-mixed user input sentence

**Output:** English and Malay translation

---

- 1: *# Import the necessary library*  
*from googletrans import Translator*
  
- 2: *# Create an instance of the Translator class*  
*translator = Translator()*
  
- 3: *# Define the code-mixed text to be translated*  
*code\_mixed\_text = "Enter your code-mixed text here."*
  
- 4: *# Function to translate the code-mixed text to English and Malay*  
*def translate\_code\_mixed\_text(text):*  
    *# Translate to English*  
    *translation\_en = translator.translate(text, dest='en')*  
    *english\_translation = translation\_en.text*  
  
    *# Translate to Malay*  
    *translation\_my = translator.translate(text, dest='ms')*  
    *malay\_translation = translation\_my.text*



```

    | # Return the translations
    | return english_translation, malay_translation
5: # Call the function with the code-mixed text
    english_translation, malay_translation =
    translate_code_mixed_text(code_mixed_text)

```

---

### Part 3: Data Pre-processing

Subsequently, after the dataset is translated onto two different language sets, both of the newly formed dataset needs to be pre-processed based on their language group. The pre-processing implementation is conducted onto two separate datasets, one for English and another for Malay, and the outcome is to obtain a clean dataset to input them onto the classification models.

Both the translated English and Malay dataset implemented on similar pre-processing technique. They both utilizes on Lowercasing, URL removal, and Non-alphanumeric removal. The only difference is on the stopwords removal set, and also the tokenization encoder used for both datasets. For the English dataset, a NLTK module that consist a corpus of English stopwords were utilized. Whilst the Malay dataset utilizes on the PySastrawi module that consist of Indonesian and Malay stopwords. Other than that, for both the BERT English and BERT Malay models, the different dataset utilizes on different tokenizing encoders to convert the data into token forms for the established BERT models to process. The English dataset utilizes on a tokenizer obtained from the English pre-trained model ‘bert-base-cased’, meanwhile the Malay dataset utilizes on the tokenizer obtained from the pre-trained model ‘malay-huggingface/bert-base-bahasa-cased’.

### Part 4: Model Predictions

After the datasets are pre-processed accordingly, and the clean datasets are obtained, the consequent steps are to pass these data onto the corresponding Machine Learning Models, and BERT Models for prediction. As the previous implementation shown, a diverse set of machine learning models are established for both the language set, resulting with a total of six machine learning models – *Naïve Bayes English*, *Linear SVC English*, *Linear Regression English*, *Naïve Bayes Malay*, *Linear SVC Malay*, and *Linear Regression Malay* – and BERT models – *BERT English* and *BERT Malay*- to predict the incoming data. Both the translated and cleaned dataset are inputted to their

corresponding language specific ML models for prediction, and the resulting predictions are saved for each of the models. The same instance goes for the BERT models; both the dataset is inputted onto the corresponding language specific BERT model for predictions, and the predicted results are saved separately for each of the models. Each model independently analyses the input text and produces its own sentiment prediction. Once the individual predictions for each classification models are obtained, they are served as individual votes in the voting system.

### **Part 5: Voting System**

The voting system implemented is an approach where the sentiment predictions from each models is being aggregated. Upon predicting the sentiment of a data, the sentiment label that receives the highest votes among the varying classification models will be taken as the final prediction result. However, there are certain weights that corresponds to the different prediction votes that affects the final collective decision. For this case, all the machine learning model's vote are assigned with the same weightage, whilst the BERT model's vote is assigned with a higher weight due to its superior performance in capturing contextual understandings and information.

There will be two voting phases implemented throughout the framework. The first voting phase is to garner the collective decision between the different machine learning models. This is why the ML models are assigned with a lower voting weightage, only after a conclusive prediction result is determined, only then the determined result from all the ML models will be weighted the same as the BERT models. The second voting phase is renowned as the finalization voting phase, this is where the BERT English prediction, BERT Malay prediction, and the collective ML prediction is once again voted together to determine the final prediction result. In the final phase, the sentiment label that receives the highest votes amongst the three model will be appointed as the final prediction result. This two-phased voting approach is design purposely in case of a tie-outcome, having three models voting against each other will always result in a defined and concise result where there won't be a stalemate scenario on both the Positive and Negative sentiment labels. Instead, there would only either be a two vote majority on the Positive label, a two vote majority on the Negative label, or a complete tie across three votes.

---

**ALGORITHM 4.7: VOTING SYSTEM**

---

**Input:** Results from each implemented model

**Output:** A final collective result

---

- 1: *# Initialize empty dataframes*  
    `dfinal = empty_dataframe()`  
    `dMLEng = empty_dataframe()`  
    `dBert = empty_dataframe()`
  - 2: *# Initialize counter variable*  
    `i = 0`  
  
    *# Define list of dataframes*  
    `pdList = [dEng1, dEng2, dEng3, dBm1, dBm2, dBm3]`  
    `pdList2 = [dEng4, dBm4]`
  - 3: *# Function to improve accuracy*  
    `function improveAcc(predSet, dataframe)`  
        `global i`  
        `insert_column(dataframe, i, 'sentiment' + convert_to_string(i),`  
        `get_list_of_sentiments(predSet))`  
        `i = i + 1`
  - 4: *# Iterate over pdList and call improveAcc function*  
    *for items in pdList*  
        `improveAcc(items, dMLEng)`  
    *# Reset counter variable*  
    `i = 0`
  - 5: *# Iterate over pdList2 and call improveAcc function*  
    *for x in pdList2*  
        `improveAcc(x, dBert)`
  - 6: *# Create a list to store the concluded sentiment*  
    `concludeSentiment = []`
  - 7: *# Iterate over rows in dMLEng dataframe*  
    *for index, row in iterate\_over\_rows(dMLEng)*  
        `concludeSentiment.append(dMLEng.loc[index].value_counts().idxmax())`
  - 8: *# Insert the concluded sentiment as a column in dBert dataframe*  
    `dBert.insert(0, 'sentimentFin', concludeSentiment)`
  - 9: *# Clear the concludeSentiment list*  
    `concludeSentiment.clear()`
  - 10: *# Iterate over rows in dBert dataframe and append the concluded result*  
    *for index, row in iterate\_over\_rows(dBert)*  
        `concludeSentiment.append(dBert.loc[index].value_counts().idxmax())`
-

## **Part 6: Validation**

The last phase of this framework is to assess the effectiveness of this proposed integration model. The well-established evaluation metrics such as Accuracy, Precision, Recall, and F1-score is utilized to validate the results obtained. Subsequently, the performance of the individual machine learning models, BERT models, and the integrated model is assessed and compared to ensure that there are actual improvements in the performance of the proposed integrated model. The experimental results from this research should demonstrate that the integrated improvement model will outperform the individual models, producing more accurate sentiment classifications. The incorporation of voting system should effectively harness the diversity of the models, minimizing errors and improving the overall performance.

### **4.4 Conclusion**

This chapter had elaborated concisely on the implementation details for the varying machine learning models, the BERT models, as well as the proposed integrated model for sentiment analysis on code-mixed data, specifically focusing on English and Malay. Through the implementation process, the framework included on conducting extensive data pre-processing, which involved cleaning and tokenizing the code-mixed texts. It also included on vectorizations to represent the textual data in a format suitable for the respective machine learning models. Not only that, this chapter also elaborated on the implementation of BERT models, discussing on the notion of BERT tokenizers, data loaders, and BERT fine-tuning. The implementation of BERT models proved to be beneficial in capturing contextual language representations, leading to improved sentiment analysis results. By leveraging both the pre-trained BERT English and Malay models and fine-tuning them on a language specific data, it allows for the harness of the power of state-of-the-art language models for enhanced sentiment classification.

Furthermore, this chapter also elaborated on the proposed integration model, combining both the machine learning models and BERT models, adding with the incorporation with a novel voting system that garners the collective decision of the different models. The aggregation of individual model predictions will surely help mitigate biases and uncertainties, leading to more reliable sentiment predictions. The

outline of this whole chapter lays the foundation for the following result and analysis chapter. The findings obtained from the implementation of different models and techniques will be further evaluated, compared and analysed to assess their performance, strengths, and limitations.

## Chapter 5

### Results and Discussion

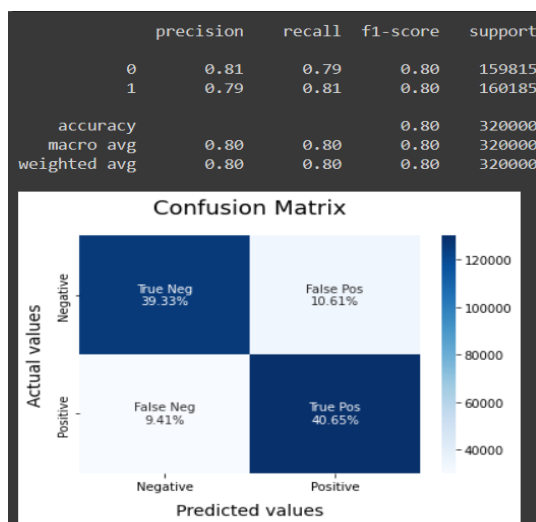
This chapter will present on the results obtained throughout this research, devoting various sentiment analysis models onto data consisting English, Malay, and code-mixed instances. This research has employed on the various machine learning models, including on Gaussian Naïve Bayes model, Linear Support Vector Classification model, and Linear Regression model. It has also implemented on BERT models, and also proposed on an integrated model assorted with a voting system. Conclusively, this chapter will also stipulate on the discussion of the findings and implications that is derived from the results obtained.

In the following segments, the results obtained from the previous implementations of the machine learning models, BERT models, and integrated model will be presented. The results will be showcased by the evaluation parameters of Confusion Matrix, Precision, Recall, F1-Measure, and Accuracy. Not only that, there will be two set of result evaluations presented for the Machine Learning and BERT models, as they had been evaluated on twice, once by the test split dataset, and another by the ground-truth dataset. The purpose and objective of this result analysis is to evaluate the performance of each model, and inherently compare the performance between them. Only the proposed integrated model will be evaluated once with the ground-truth dataset, and compared with the performance of the previous individual models. This is to validate on how the proposed model actually works on real and unknown data, and whether it really improves the performance in classifying the sentiments. Subsequently, the comparison will be used as a constituent factor on determining the overall success of this research.

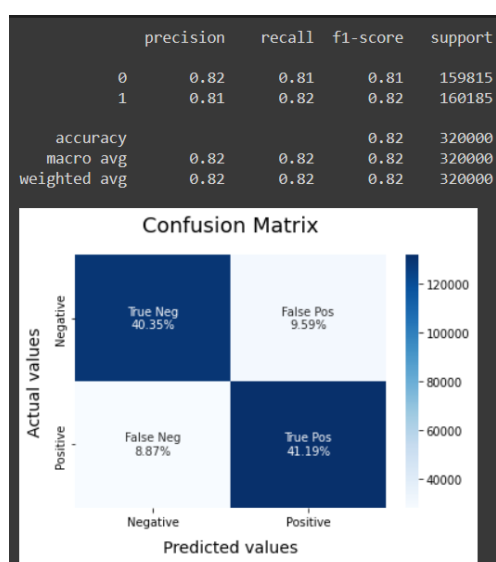
#### 5.1 Results of Machine Learning Models

This segment will present on the result obtained from the implementation of the various machine learning models, evaluating the performance of the three individual models: Gaussian Naive Bayes, LSVC, and Linear Regression on performing sentiment analysis task. The below **Figure 5.1 – 5.3** and **Table 5.1** presents the

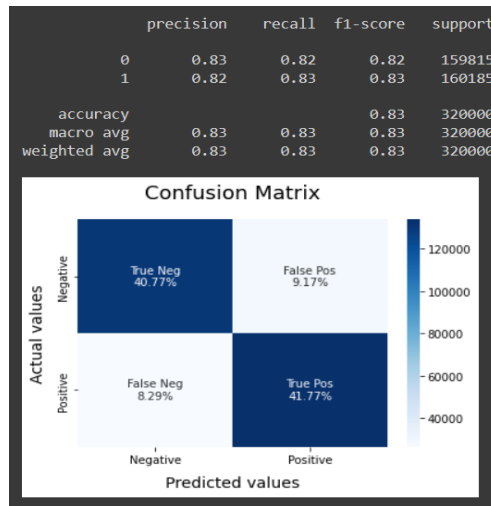
performance metrics achieved by the English machine learning models on the split test data.



**Figure 5.1:** Split Test data validation result on English Gaussian Naive Bayes



**Figure 5.2:** Split test data validation result on English LSVC



**Figure 5.3:** Split test data validation result on English Logistic Regression

**Table 5.1:** Summary of Eng machine learning model’s split test validation result

Algorithm	Confusion Matrix			Evaluation Parameter			Accuracy
Naïve Bayes English		Positive	Negative	Precision	Recall	F1-score	0.80
	Positive	39.33%	10.61%	0.81	0.79	0.80	
	Negative	9.41%	40.65%	0.79	0.81	0.81	
Linear SVC English		Positive	Negative	Precision	Recall	F1-score	0.82
	Positive	40.35%	9.59%	0.82	0.81	0.81	
	Negative	8.87%	41.19%	0.81	0.82	0.82	
Linear Regression English		Positive	Negative	Precision	Recall	F1-score	0.83
	Positive	40.77%	9.17%	0.83	0.82	0.82	
	Negative	8.29%	41.77%	0.82	0.83	0.83	

The result above is the result obtained through validating the English Machine Learning Models with the split test data. Through observing the table, it is apparent that all three models received a reasonably well performance in predicting the sentiments of the split test data. The Linear Regression (LR) model exhibits the highest accuracy score amongst the machine learning models, achieving a score of 83%. It also

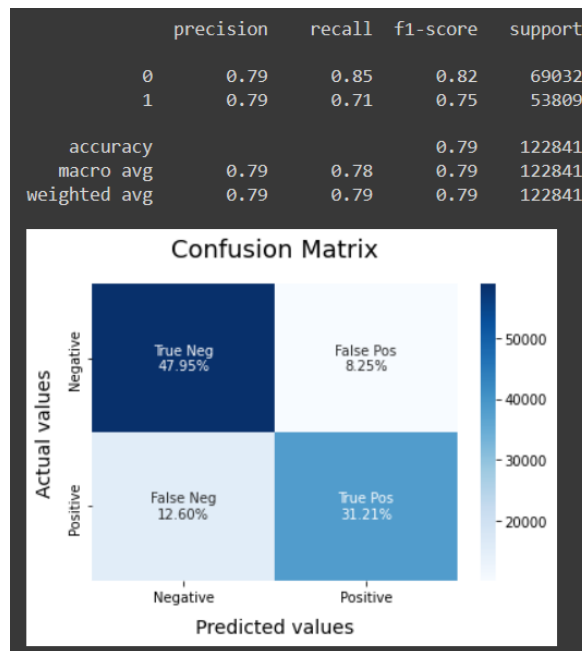


presents the highest Precision, Recall, and F1-score, signifying the model's ability to correctly classify both positive and negative sentiments in different instances.

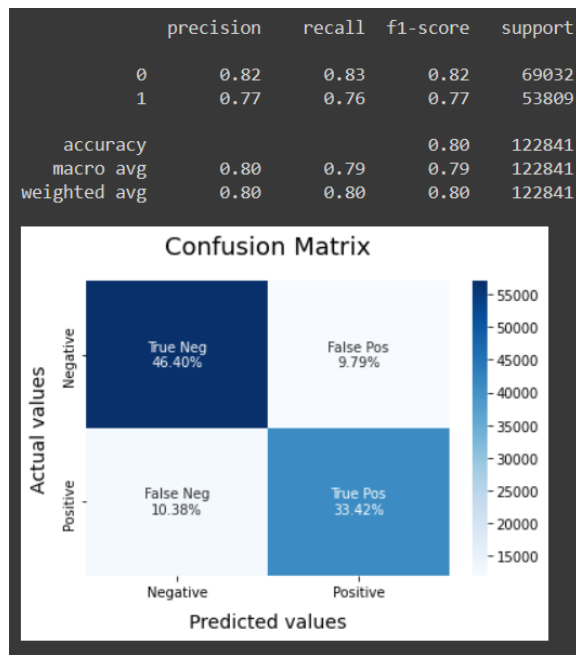
The Linear SVC model also performed exceptionally well, coming in second with an accuracy score of 82%. It has a slightly lower precision, recall, and F1-score compared to the Linear Regression (LR) model, but it still demonstrated a robust performance in classifying the sentiments of the test data, displaying a consistent precision and recall scores across both sentiment classes.

The Gaussian Naïve Bayes Model, while achieving an accuracy score of 80%, exhibits the lowest performance in compared to the other two model. The Gaussian Naive Bayes model is known for its simplicity and assumes independence among the features, which makes it a computationally efficient choice for sentiment analysis. However, it may not capture the complex relationships present in sentiment analysis tasks as effectively as the other two linear models like the Linear SVC and Linear Regression model.

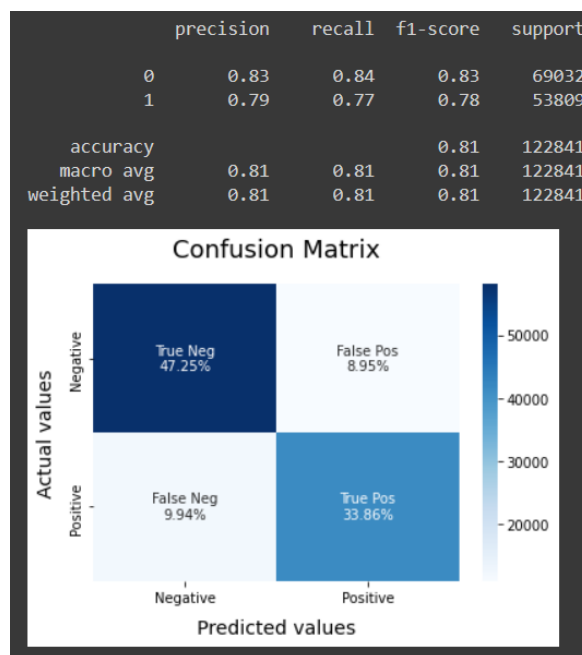
Moving on, after the English Machine Learning models are done with their validation on the split-test dataset, the following phase is to continue on with the Malay Machine Learning models. In doing so, the following **Figure 5.4 – 5.6** and **Table 5.2** presents the performance metrics achieved by the Malay machine learning models on the split test data.



**Figure 5.4:** Split Test data validation result on Malay Gaussian Naive Bayes



**Figure 5.5:** Split Test data validation result on Malay LSVC



**Figure 5.6:** Split Test data validation result on Malay Logistic Regression

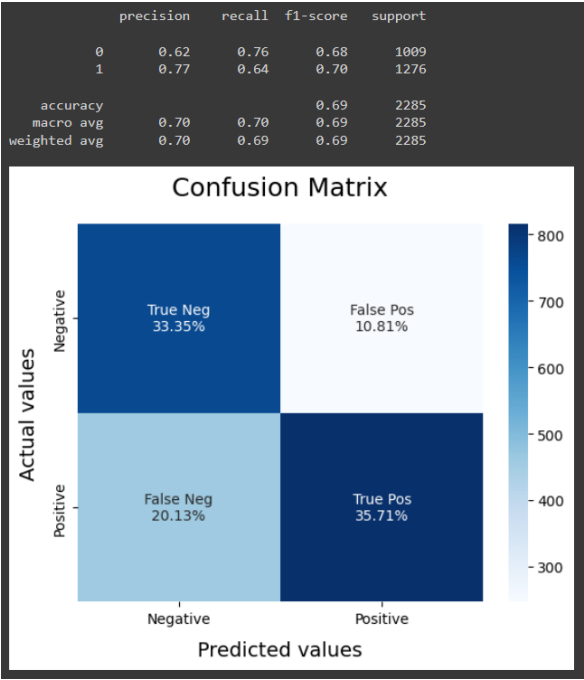
**Table 5.2:** Summary of BM machine learning model's split test validation result

Algorithm	Confusion Matrix			Evaluation Parameter			Accuracy
Naïve Bayes BM		Positive	Negative	Precision	Recall	F1- score	0.79
	Positive	47.95%	8.25%	0.79	0.85	0.82	
	Negative	12.60%	31.21%	0.79	0.71	0.75	
Linear SVC Malay		Positive	Negative	Precision	Recall	F1- score	0.80
	Positive	46.40%	9.70%	0.82	0.83	0.82	
	Negative	10.38%	33.42%	0.77	0.76	0.77	
Linear Regression Malay		Positive	Negative	Precision	Recall	F1- score	0.81
	Positive	47.25%	8.95%	0.84	0.83	0.83	
	Negative	9.94%	33.86	0.79	0.77	0.78	

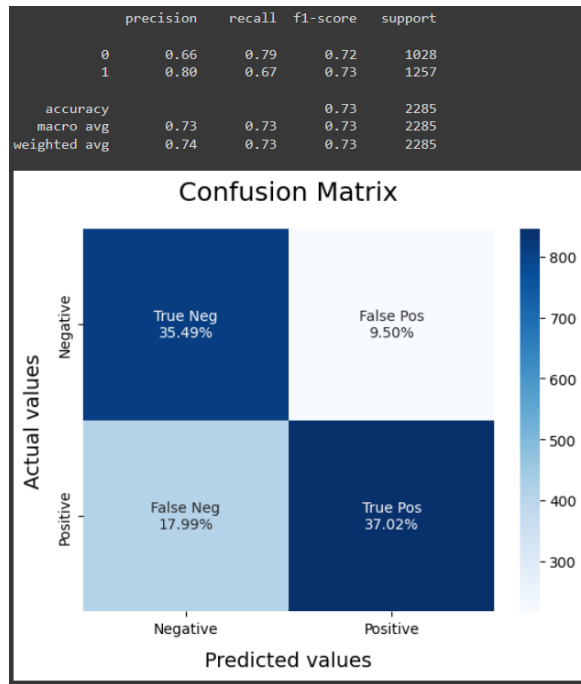
The machine learning models that trains on the Malay datasets came out with similar results compared to its English counterpart. The Linear Regression (LR) model achieved the highest accuracy score of 81%. It presents a high Precision, Recall, and F1-score for the positive classes, while presenting a slightly lower score for the negative classes. The Linear SVC model comes in second achieving an accuracy score of 80%, also with a higher precision, Recall, and F1-score classifying on the positive class, with a slight disparity on the negative class. Lastly the Gaussian Naïve Bayes model comes in last with an accuracy score of 79%. This model presents a large disparity between classifying the positive class and the negative class. It exhibits a fairly well Precision, Recall, and F1-score on the positive class, but achieves an underperforming score on classifying the negative class.

After both the implemented machine learning models are validated through the split test data, it also needs to be validated through the ground-truth dataset in order to assess the model's performance in classifying unknown data. The ground-truth dataset consists of text sentence that is code-mixed with English and Malay language, as per the implementation framework from before, the ground-truth dataset is translated onto two different language set – *English and Malay*- and inputted for each of the

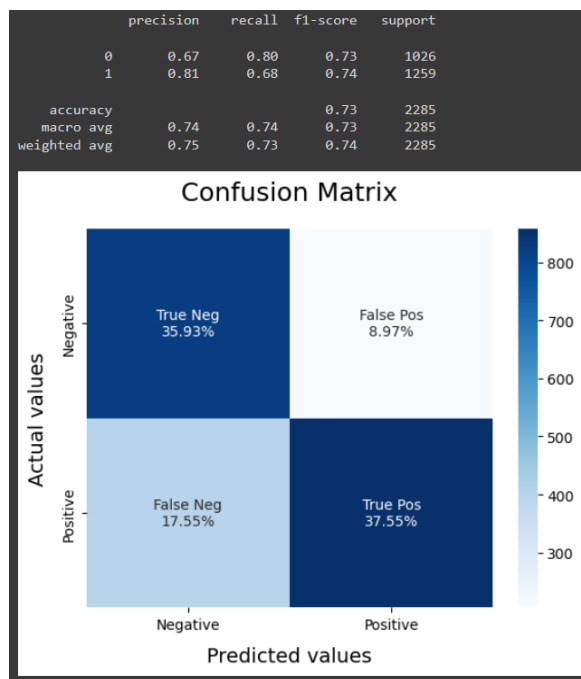
corresponding machine learning models to predict. The predicted result is then validated with the original annotated sentiment labels to test on the instances that the models had classify the data correctly. The below **Figure 5.7 – 5.9** and **Table 5.3** presents the performance metrics achieved by the English machine learning models on the ground-truth dataset.



**Figure 5.7:** Ground-truth data validation result on English Gaussian Naive Bayes



**Figure 5.8:** Ground-truth data validation result on English LSVC



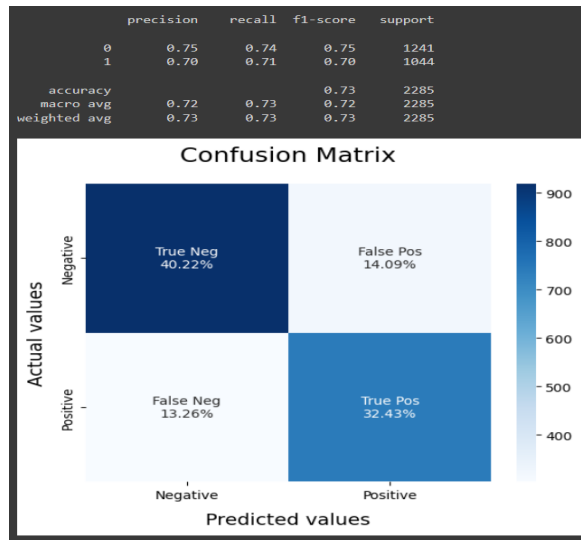
**Figure 5.9:** Ground-truth data validation result on English Logistic Regression

**Table 5.3:** Summary of Eng machine learning model's ground-truth validation result

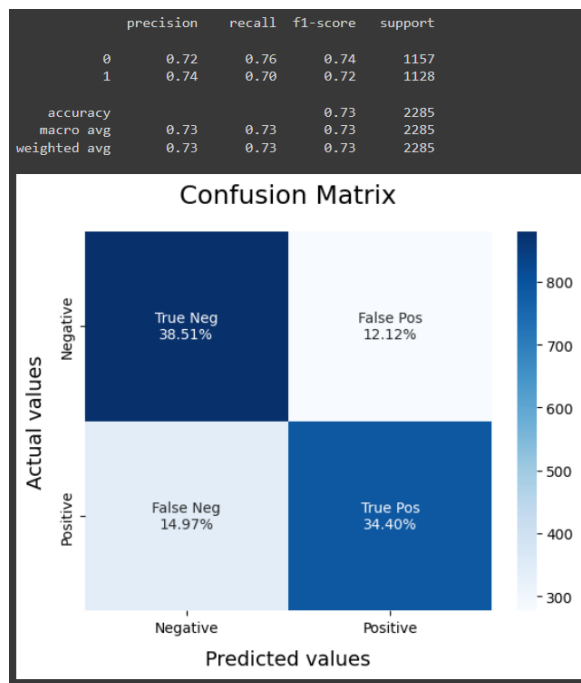
Algorithm	Confusion Matrix			Evaluation Parameter			Accuracy
Naïve Bayes English		Positive	Negative	Precision	Recall	F1- score	0.69
	Positive	33.35%	10.81%	0.62	0.76	0.68	
	Negative	20.13%	35.75%	0.77	0.64	0.70	
Linear SVC English		Positive	Negative	Precision	Recall	F1- score	0.73
	Positive	35.49%	9.50%	0.66	0.79	0.72	
	Negative	17.99%	37.02%	0.80	0.67	0.73	
Linear Regression English		Positive	Negative	Precision	Recall	F1- score	0.73
	Positive	35.93%	8.97%	0.67	0.80	0.73	
	Negative	17.55%	37.55%	0.81	0.68	0.74	

The result above is the result obtained through validating the English Machine Learning Models with the ground-truth dataset. It is observable that there is a slight performance decrease across all three models when they are validating through the ground-truth dataset. Nevertheless, the performance sequence of the models is still similar to what is shown before. The Naïve Bayes model is again the lowest in performance amongst all three models, it achieves an accuracy score of 0.69, with its precision, recall, and F1-score varying across 0.62 – 0.70. The LSVC model on the other hand, performed slightly better with an accuracy score of 0.73, and an improved precision, recall and F1-score of approximately 0.66 – 0.79. Finally, the Logistic Regression model achieves the same accuracy score as the LSVC model of 0.73. However, it does achieve on a slightly improved Precision, Recall, and F1-score all around 0.67 – 0.80.

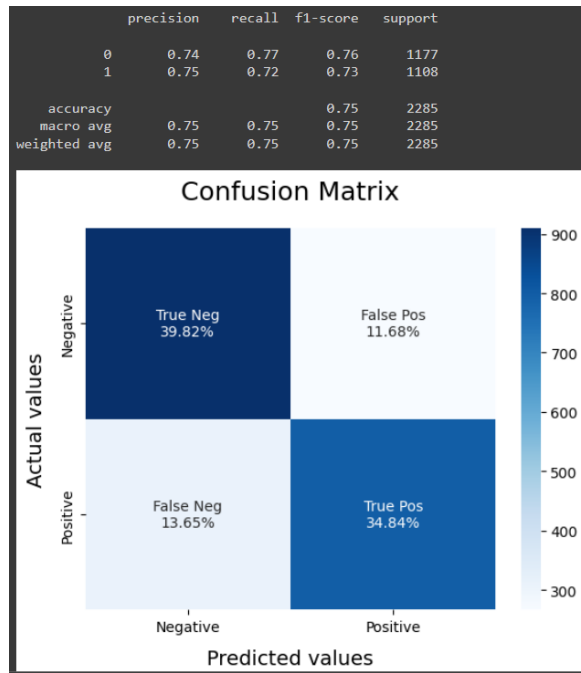
A similar instance can also be observed through the Malay machine learning models, which is as shown below on **Figure 5.10 – 5.12** and **Table 5.4**.



**Figure 5.10:** Ground-truth data validation result on Malay Gaussian Naive Bayes



**Figure 5.11:** Ground-truth data validation result on Malay LSVC



**Figure 5.12:** Ground-truth data validation result on Malay Logistic Regression

**Table 5.4:** Summary of BM machine learning model's ground-truth validation result

Algorithm	Confusion Matrix			Evaluation Parameter			Accuracy
Naïve		Positive	Negative	Precision	Recall	F1-score	0.73
Bayes							
Malay	Positive	40.22%	14.09%	0.75	0.74	0.75	
	Negative	13.26%	32.43%	0.70	0.71	0.70	
Linear		Positive	Negative	Precision	Recall	F1-score	0.73
SVC							
Malay	Positive	38.51%	12.12%	0.72	0.76	0.74	
	Negative	14.97%	34.40%	0.74	0.70	0.72	
Linear		Positive	Negative	Precision	Recall	F1-score	0.75
Regression							
Malay	Positive	39.82%	11.68%	0.74	0.77	0.76	
	Negative	13.65%	34.84%	0.75	0.72	0.73	

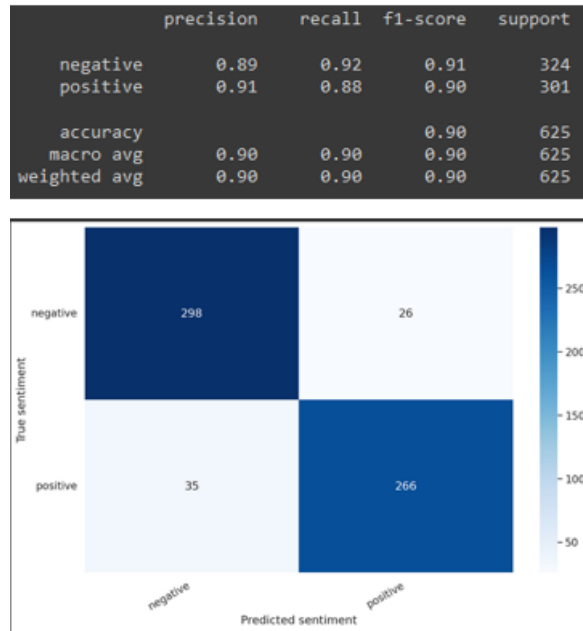
As observed, the Malay Machine learning models also had Logistic Regression model as the best performance models amongst the three. However, in this instance the Malay models surprisingly had all three of the models performing better on validating the



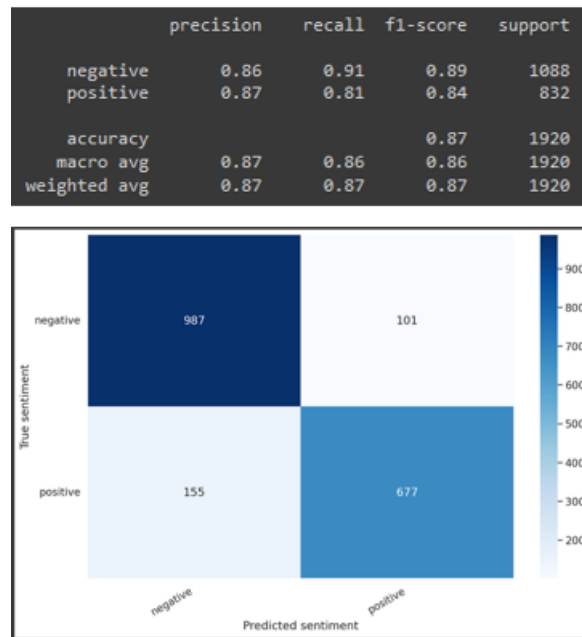
ground-truth dataset in compared to its English model counterpart, having each model leading a 1~2% higher accuracy score than the English models. Other than that, the Malay models had similar instances where the Naïve Bayes and LSVC models equally achieves an accuracy score of 0.73, with the LSVC model leading a slightly better Precision, Recall, and F1-score.

## 5.2 Result of BERT models

This segment will present on the result obtained from the implementation of the BERT models, which are distinguished between two instance, BERT English and BERT Malay. The below **Figure 5.13 – 5.14** and **Table 5.5** presents the performance metrics achieved by the BERT models on the split test data.



**Figure 5.13:** BERT English validation result on split test data



**Figure 5.14:** BERT Malay validation result on split test data

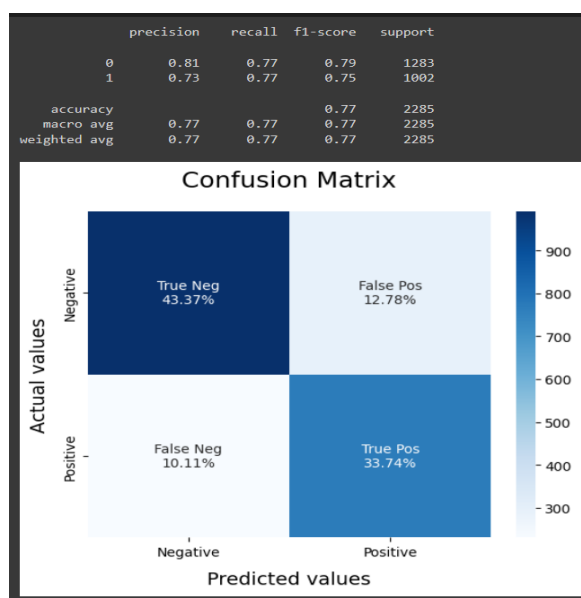
**Table 5.5:** Summary of both BERT model's split test data validation result

Algorithm	Confusion Matrix			Evaluation Parameter			Accuracy
BERT English		Positive	Negative	Precision	Recall	F1-score	0.90
	Positive	47.68%	4.16%	0.89	0.92	0.91	
	Negative	5.60%	42.56%	0.91	0.88	0.90	
BERT BM		Positive	Negative	Precision	Recall	F1-score	0.87
	Positive	51.41%	5.26%	0.86	0.91	0.89	
	Negative	8.07%	35.25%	0.87	0.81	0.84	

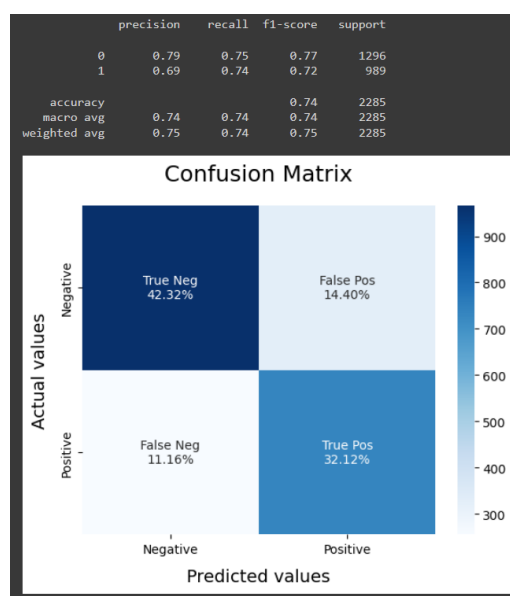
The above results are the BERT models validated through with the split test data. As observed above, both the BERT English and BERT BM achieved an impressively high result in classifying sentiments on unknown data, attaining an accuracy score of 0.90 and 0.87 respectively. If the BERT models are compared with the previous machine learning models, it is observable that both the BERT model outperforms all the machine learning models, presenting an overall higher accuracy, precision, recall, and F1-score of approximate 0.89-0.92 and 0.81 – 0.91 respectively. This highly suggest that leveraging the contextual word embeddings of the BERT model, and utilizing on

the transformer-based architecture can significantly enhance the sentimental analysis performance.

After validating the BERT models through the split test dataset, it is also validated through the ground-truth dataset. The below **Figure 5.15 – 5.16** and **Table 5.6** presents the performance metrics achieved by the BERT models on the ground-truth dataset.



**Figure 5.15:** BERT English validation result on ground-truth data



**Figure 5.16:** BERT Malay validation result on ground-truth data

**Table 5.6:** Summary of both BERT model's ground-truth data validation result

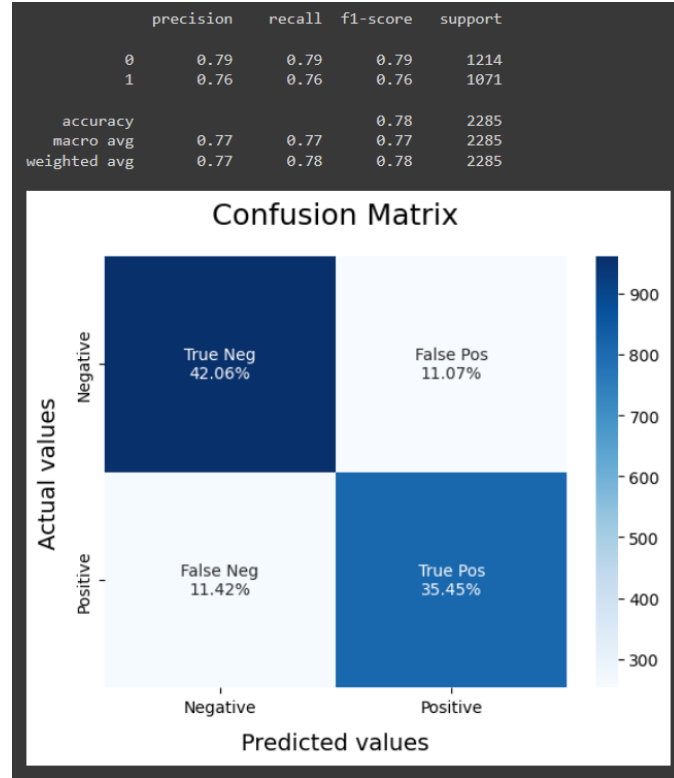
Algorithm	Confusion Matrix			Evaluation Parameter			Accuracy
BERT English		Positive	Negative	Precision	Recall	F1-score	0.77
	Positive	43.37 %	12.78%	0.81	0.77	0.79	
	Negative	10.11%	33.74%	0.73	0.77	0.75	
BERT BM		Positive	Negative	Precision	Recall	F1-score	0.74
	Positive	42.32%	14.40%	0.79	0.75	0.77	
	Negative	11.16%	32.12%	0.69	0.74	0.72	

As observed above, the BERT model still exhibits a fairly moderate performance on the ground-truth dataset, which is reasonably expected due to it being an unknown data to the models. The BERT English model achieved a fairly decent accuracy of 0.77, with an all around Precision, Recall, and F1-score between 0.73-0.81. Meanwhile, the BERT Malay model achieves a lower accuracy score of 0.74, which is still a fairly adequate result, and also with an all around Precision, Recall, and F1-Score of roughly 0.69 – 0.79. The result of the BERT Malay model was certainly astonishing, because it achieve an accuracy score slightly than the Logistic Regression Malay model by 1%, which is unexpected considering the strong contextual understanding ability of a BERT model, where one would anticipate a BERT model to perform better compared to a machine learning model. Meanwhile, the BERT English model do outperform all the other models in quite a hefty margin, leading with a approximate 2~5 % accuracy score comparing with the lowest performing model to the second runner up model.

### 5.3 Results of proposed Ensemble/Integration model

As a model that leverages on the strength of all the individual models, the proposed integrated model supposedly will achieve a performance result that surpasses each of the individual models. The implemented voting system considered on the majority sentiment predicted by the models to make a final, collective prediction. This approach assuredly will improve on the performance and mitigate the weaknesses of the

individual models. The following **Figure 5.17** and **Table 5.7** depicts the performance metrics achieved by the integrated model on the ground-truth dataset.



**Figure 5.17:** Ensemble/Integration model validation result on ground-truth data

**Table 5.7:** Summary of Ensemble model's ground-truth data validation result

Algorithm	Confusion Matrix			Evaluation Parameter			Accuracy
Integrated Model		Positive	Negative	Precision	Recall	F1-score	0.78
	Positive	35.45%	11.42%	0.79	0.79	0.79	
	Negative	11.07%	42.06%	0.76	0.76	0.76	

As it can be seen above, the integrated model achieved a remarkable result, demonstrated superior performance compared to both the individual models and the BERT models in predicting the ground-truth dataset. It achieves an accuracy score of 0.78, with an improvement across the evaluation parameters too, such that the Precision, Recall, and F1-Score all achieving 0.79 and 0.76, showcasing a substantial improvement over the individual models. One of the notable improvements achieved by the integrated model is in accuracy score. The integrated model outperformed all the individual models, including each of the language specific models of Gaussian

Naive Bayes, LSVC, and Linear Regression, as well as the BERT models. This improvement can be attributed to the combination of predictions from multiple language-specific models through the voting system. By considering the majority vote, the integrated model leverages the strengths of each individual model, thereby achieving a higher overall accuracy. Comparing the metrics with those from the individual models, we can observe a substantial improvement in performance, with an increase of 1~10% in accuracy score if compared to the lowest performing model, and the 2<sup>nd</sup> runner up performing model.

Additionally, the integrated model exhibits improvements in precision, recall, and F1-score. Precision refers to the proportion of correctly predicted positive instances out of the total predicted positive instances. Recall measures the proportion of correctly predicted positive instances out of the total actual positive instances. F1-score combines precision and recall into a single metric that balances both aspects. The integrated model demonstrated higher precision, recall, and F1-score compared to the individual models, indicating improved performance in accurately identifying positive instances and reducing false positives.

The integrated model's improved performance can certainly be attributed to its ability in capturing both the English and Malay language-specific patterns and nuances. By considering predictions from separate language-specific models, the integrated model benefits from the expertise and knowledge of each individual model in their respective languages. This enables the integrated model to effectively handle the intricacies of code-mixed data, leading to enhanced sentiment analysis results. Furthermore, the integrated model showcases the potential of combining machine learning techniques from multiple languages to address the challenges posed by code-mixed data. This approach can be applied to other multilingual sentiment analysis tasks, where code-mixing occurs, and can pave the way for improved understanding and analysis of sentiments expressed in diverse linguistic contexts.

## **5.4 Discussions**

In this segment of the chapter, it is to present a comprehensive discussion of the results obtained from each of the implemented models from before. The comparison of the performance of various machine learning models, including Gaussian Naive Bayes, LSVC, Linear Regression, BERT, and the proposed integrated model shall be

conducted. Subsequently, a comparison table of the accuracy scores achieved by each model shall also be provided in order to facilitate a clear understanding of their relative performance.

#### 5.4.1 Comparison of individual models

To compare between the individual models, the below table presents a comparison of the accuracy score achieved by each model on validating the ground-truth dataset.

**Table 5.8:** Sentiment Accuracy of each implemented model

Model	Accuracy
Gaussian Naïve Bayes (English)	0.69
Gaussian Naïve Bayes (Malay)	0.73
Linear Support Vector Machine (English)	0.73
Linear Support Vector Machine (Malay)	0.73
Logistic Regression (English)	0.73
Logistic Regression (Malay)	0.75
BERT (English)	0.77
BERT (Malay)	0.74
Integrated Model	0.78

From the **Table 5.8**, it is observable that the highest achieving accuracy model goes to the English BERT model, which consistently outperforms the all the individual models across both English and Malay complements. The BERT English model achieves an accuracy score of 0.77, which triumphs over the other individual models with a 2~8% improvement. These results could indicate the effectiveness of BERT models in capturing the complex linguistic dynamics of code-mixed data. Furthermore, amongst the machine learning models, they all achieve on a similar performance, averaging on an accuracy score of 0.73. The only exception is on the Gaussian Naïve Bayes English model, as well as the Logistic Regression Malay model.

The Gaussian Naïve Bayes English model achieves the lowest performance amongst the individual models in predicting the ground-truth dataset, achieving merely an accuracy score of 0.69. This lower performance can be attributed to the model's

assumption of independence among features, which may not hold true for code-mixed data. Code-mixed data often exhibits complex language dynamics, and the simplistic assumptions made by Gaussian Naive Bayes limit its ability to capture the nuances of sentiment expressed in a mixed-language context. It could also be attributed to many other factors, such as the pre-processing of the code-mixed data or simply the translation. Code-mixed text often contains noise, such as spelling variations, grammatical inconsistencies, and language-specific slang. Inaccurate or inadequate pre-processing techniques may introduce noise into the data, which can adversely impact the performance of the model, particularly those relying on lexical features. For the translation factor, Code-mixed data often contains phrases or sentences that are partially or entirely in one language. Accurate translation of these language-specific segments is crucial for capturing the sentiment expressed within them. Thereby, having an inaccurate or incomplete translations may lead to misinterpretations and impact the overall sentiment analysis performance of this particular model.

On the contrary, the Logistic Regression Malay model surprisingly achieves the highest performance amongst the individual machine learning models, triumphing a result higher than its English counterpart, and in an instance even higher than the BERT Malay model with an accuracy score of 0.75. There could be many speculations on the factors that made this so, firstly the selection of logistic regression as the model itself could be the factor for the triumphing result. Despite being a linear model, logistic regression can capture non-linear relationships between features and the target variable by applying non-linear transformations to the input features. Through these transformations, logistic regression can effectively model complex patterns and capture the intricate sentiment expressions present in code-mixed data. Not only that, Logistic regression is well-suited for handling sparse data since it assigns separate weights to each feature. Meanwhile, Code-mixed data often results in a sparse feature representation due to the presence of multiple languages and their unique linguistic characteristics, and this property allows logistic regression to focus on relevant features and disregard irrelevant ones, leading to improved sentiment analysis performance on code-mixed data. Aside factors from the model characteristic, it could be a language-specific characteristic that allow for a higher prediction result. Each language has its unique linguistic characteristics, such as grammar, vocabulary, and sentence structure. The sentiment analysis model may be more adept at capturing the



sentiment patterns and nuances present in the Malay language compared to English. The Malay language may exhibit more explicit sentiment indicators or have more distinguishable sentiment patterns, making it easier for the model to learn and classify sentiments accurately.

The integrated model unquestionably achieves the highest performing score in compared to all the other individual models, achieving an accuracy score of 0.78. The factor of improvement could be well due to the integrated model leveraging the collective strength of multiple language-specific models. By considering the majority vote from individual models, the integrated model captures a broader range of language-specific patterns and nuances, leading to improved sentiment analysis results on code-mixed data. It benefits from the diversity of approaches and expertise embedded within each language-specific model, enhancing its ability to handle the intricacies of code-mixed data, and thus enhances the model's overall accuracy and robustness.

## **5.5 Conclusion**

The results obtained from both the split test dataset and the ground-truth dataset provide valuable insights into the performance of the implemented code-mixed sentiment analysis models. It is evident that the BERT models consistently outperformed the traditional machine learning models in many instances. BERT models, with their contextual understanding capabilities, excel in capturing the nuances and complexities of code-mixed text. Additionally, the integrated model with a voting system showcased improved performance compared to individual models in most cases. The combination of BERT models with other machine learning models leveraged the strengths of each model, resulting in enhanced accuracy, precision, recall, and F1-scores.

However, it is important to note that the performance of the models varied between the split test dataset and the ground-truth dataset. This discrepancy may be attributed to the inherent differences in data distribution and the manual labelling process of the ground-truth dataset. The models might encounter challenges in generalizing well to unseen code-mixed data. Overall, the results highlight the potential of BERT models in classifying sentiments on code-mixed data. The

integrated model with a voting system also shows promise in improving model performance. Further research and experimentation can focus on fine-tuning the models, exploring additional feature engineering techniques, and expanding the dataset to enhance the models' robustness and generalization capabilities.

## **Chapter 6**

### **Research Conclusion and Future Work**

This chapter presents the research conclusion and future work of this thesis, which focuses on the sentiment analysis on code-mixed data in English and Malay languages. The chapter highlights the main findings of the study, including the effectiveness of various machine learning models, deep learning models and the development of an ensemble model with a voting system that optimizes the sentiment classification accuracy. Additionally, the chapter outlines the implications of the research for the general consensus on natural language processing (NLP) tasks and proposes the future directions that this research can advance on the notion of sentiment analysis in code-mixed environments.

The research conducted in this thesis recognizes the growing significance of language code-mixing, where multiple languages are spoken, written, or typed within a single sentence or a full document in multilingual societies. Code-mixed data presents unique challenges for sentiment analysis due to linguistic variations and language mixing and switching. Therefore, this chapter aims to provide a comprehensive conclusion on the attempted sentiment analysis on code-mixed data and explore on any potential solutions or future prospects that can improve on the practical accuracy and performance. This chapter will begin with a research conclusion, summarizing the key findings that is concluded throughout the study. It will discuss on the performance of the different machine learning models and deep learning models, including on Naive Bayes, LSVC, logistic regression, and BERT models, when applied to sentiment analysis in code-mixed texts. Furthermore, it emphasizes the importance of language-specific models and the integration of deep contextual representations for accurate sentiment classification. Following the research conclusion, the chapter outlines the future work that can be undertaken based on the research findings. A proposed program for detecting sentiment in code-mixed input is discussed, highlighting the need for a comprehensive system that accounts for the nuances of both English and Malay languages in code-mixed texts. Additionally, the

chapter presents a current working prototype of the sentiment detection system, showcasing the progress made towards realizing the future work.

By providing a conclusive summary of the research findings and proposing future directions, this chapter aims to contribute to the broader understanding of sentiment analysis in code-mixed data and inspire further advancements in NLP tasks. The outcomes of this research have practical implications for various domains, including social media analysis, customer feedback analysis, and opinion mining in multilingual contexts. The chapter ultimately serves as a foundation for future studies that aim to enhance sentiment analysis techniques and develop comprehensive solutions for code-mixed language environments.

## **6.1 Research conclusion**

This research study aimed to explore the effectiveness of various machine learning and deep learning models, including Naive Bayes, LSVC (Linear Support Vector Classifier), logistic regression, and BERT models, in accurately classifying sentiments in code-mixed English-Malay texts. Additionally, this study also proposed an ensemble model that integrated a voting system to leverage the collective decision-making of each model's output, thereby enhancing sentiment classification accuracy.

The research's findings demonstrated that the machine learning models implemented with both English and Malay languages achieved promising results in sentiment analysis of code-mixed data. The Naive Bayes, LSVC, and logistic regression models exhibited competitive performance when compared individually. However, the BERT models, with their contextual understanding capabilities, outperformed the traditional models, showcasing the importance of leveraging deep contextual representations for sentiment analysis in code-mixed data.

By combining the outputs of these models through the ensemble/integrated model with a voting system, the research is able to achieve a further improvement in sentiment classification accuracy. The ensemble model demonstrated an enhanced generalization and robustness by aggregating diverse perspectives from individual models. This approach not only optimized sentiment analysis performance but also addressed the inherent challenges associated with code-mixed data, such as linguistic variations and language switching. The research outcomes provided a valuable insight

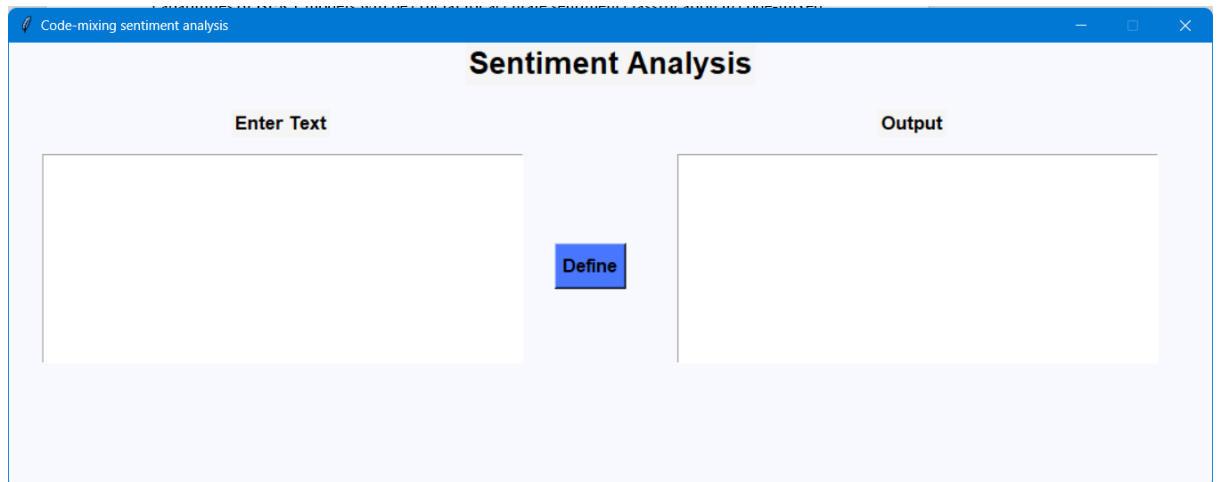
into sentiment analysis on code-mixed data and contribute to the general consensus on natural language processing (NLP) tasks. The findings emphasize the significance of accounting for code-mixing phenomena in NLP research and applications, as multilingual environments become increasingly prevalent. Moreover, the study highlights the effectiveness of incorporating language-specific models and leveraging deep contextual representations to enhance sentiment classification accuracy.

## **6.2 Future work**

Grounded across the research being conducted, there are several pathways for future works to emerge to, with this research as the laying foundation. This study will allow for the further advancements of sentiment analysis in code-mixed data, especially in the language pair of English and Malay. One promising direction that transpires from this research, is to develop a program that is capable of detecting the sentiments of a code-mixed user input sentence. By combining the research findings with additional resources and algorithms, a complete sentiment analysis system can be created, catering specifically to code-mixed language environments. The proposed sentiment detection program should be designed to handle both English and Malay code-mixed data, effectively capturing the nuances and sentiment expressions of these languages within a single text. The system should utilize the proposed ensemble/integrated model approach, leveraging the voting system to aggregate the decisions of multiple language-specific models. Additionally, incorporating the contextual understanding capabilities of BERT models will be crucial for accurate sentiment classification in code-mixed texts.

## **6.3 Current Working Prototype**

To showcase on the progress that has been made towards realizing the future work, this section will present a developed working prototype of the sentiment detection system for code-mixed input. The prototype integrates the ensemble model with the voting system and utilizes language-specific models implemented with both English and Malay languages. The aspect and the operation of the working prototype can be seen as below.



**Figure 6.1:** Visual representation of the working prototype

The system takes in a user English-Malay code-mixed input as its primary input on the left input box. It then employs translation and pre-processing techniques to handle language switching and linguistic variations commonly found in code-mixed texts. The text is then passed through each language-specific model, generating sentiment predictions for each language. These predictions are subsequently combined using the ensemble model's voting system, which provides the final sentiment classification output on the right output box.

To demonstrate the capabilities of the working prototype, below presents several demo inputs and their corresponding sentiment classification results:

1. Demo Input: "I really suka the movie, it was amazing!"  
Sentiment Classification: Positive
2. Demo Input: "This app is so susah to use, pening kepala!"  
Sentiment Classification: Negative
3. Demo Input: "The weather today is so nice, terasa damai and peaceful."  
Sentiment Classification: Positive
4. Demo Input: "Tadi I pergi shopping mall, but terkejut lah with the harga!"  
Sentiment Classification: Negative

These demo inputs will then be inserted onto the working prototype, and showcase the current prototype's ability to predict the sentiments accurately of the code-mixed

English-Malay data. As shown on the below figures are the outputs of the prototype predicting on the demo inputs:

The screenshot shows a web application titled "Code-mixing sentiment analysis" with a "Sentiment Analysis" header. On the left, under "Enter Text", the input is "I really suka the movie, it was amazing!". A blue "Define" button is in the center. On the right, under "Output", a table lists sentiment scores for various models, all of which are "Positive". Below the table, it states "Concluded sentiment: Positive".

Model	Sentiment
Naive Bayes English Sentiment	Positive
Naive Bayes BM Sentiment	Positive
Support Vector Machine English Sentiment	Positive
Support Vector Machine BM Sentiment	Positive
Linear Regression Eng Sentiment	Positive
Linear Regression BM Sentiment	Positive
BERT English Sentiment	Positive
BERT BM Sentiment	Positive

Concluded sentiment: Positive

**Figure 6.2(a):** Output result for demo input 1

The screenshot shows the same web application with a different input. Under "Enter Text", the input is "This app is so susah to use, pening kepala!". The "Define" button is still present. In the "Output" section, the table shows that all models have predicted a "Negative" sentiment. Below the table, it states "Concluded sentiment: Negative".

Model	Sentiment
Naive Bayes English Sentiment	Negative
Naive Bayes BM Sentiment	Negative
Support Vector Machine English Sentiment	Negative
Support Vector Machine BM Sentiment	Negative
Linear Regression Eng Sentiment	Negative
Linear Regression BM Sentiment	Negative
BERT English Sentiment	Negative
BERT BM Sentiment	Negative

Concluded sentiment: Negative

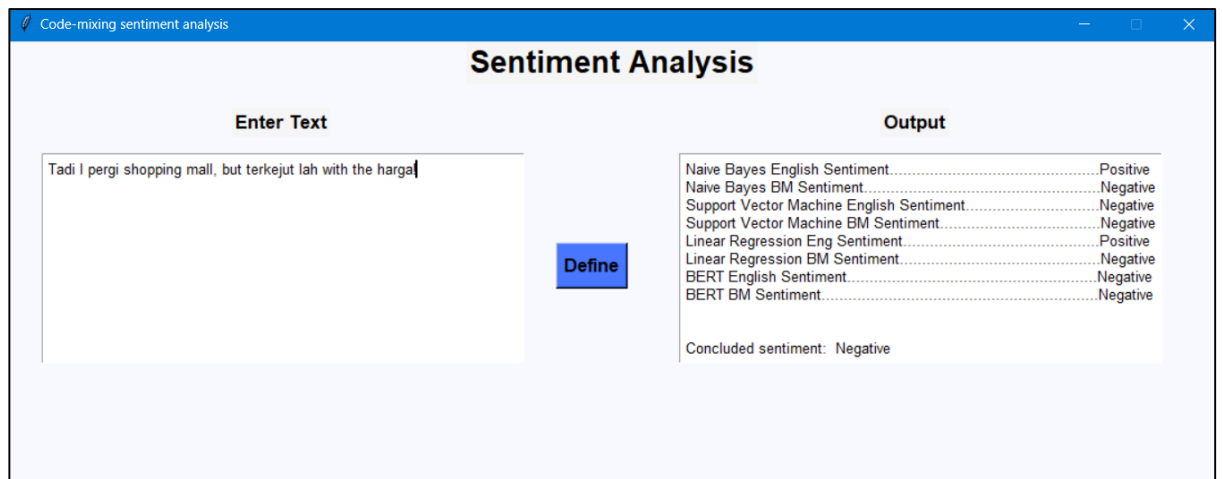
**Figure 6.2(b):** Output result for demo input 2

The screenshot shows the web application with a third input. Under "Enter Text", the input is "The weather today is so nice, terasa damai and peaceful". The "Define" button is present. The "Output" section shows that all models have predicted a "Positive" sentiment. Below the table, it states "Concluded sentiment: Positive".

Model	Sentiment
Naive Bayes English Sentiment	Positive
Naive Bayes BM Sentiment	Positive
Support Vector Machine English Sentiment	Positive
Support Vector Machine BM Sentiment	Positive
Linear Regression Eng Sentiment	Positive
Linear Regression BM Sentiment	Positive
BERT English Sentiment	Positive
BERT BM Sentiment	Positive

Concluded sentiment: Positive

**Figure 6.2(c):** Output result for demo input 3



**Figure 6.2(d):** Output result for demo input 4

These demo inputs showcase the prototype's ability to handle code-mixed text with the likes of daily conversations, where it accurately capturing sentiments expressed in both English and Malay languages. The sentiment detection system effectively processes the input, extracts relevant linguistic cues, and combines the outputs of the language-specific models using the ensemble model's voting system. The final sentiment classification accurately reflects the overall sentiment expressed in the code-mixed text.

To make things more challenging and taxing for the developed system, another list of code-mixed inputs will be tested against the system. On this instance, the input data will be incorporated with false-positive terms to challenge and strain on the system, observing how well the system can predict on nuances and sentences with unclear sentiments. The below list presents the demo input that is incorporated with false positive terms for the test:

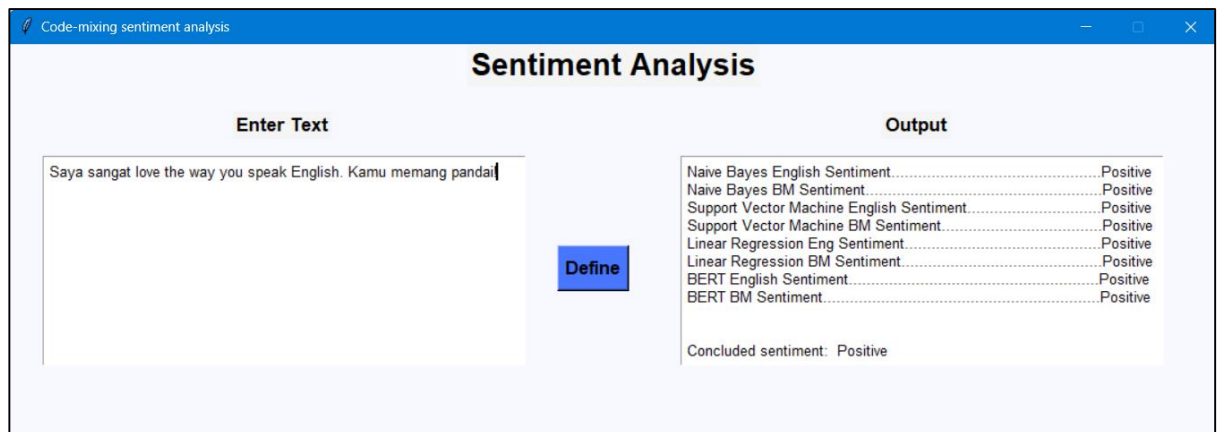
#### **Positive Sentiments:**

1. "Saya sangat love the way you speak English. Kamu memang pandai!"
2. "I rasa sangat tired hari ni, tetapi mood saya memang best!"
3. "Saya really appreciate your bantuan. Kamu memang hebat!"

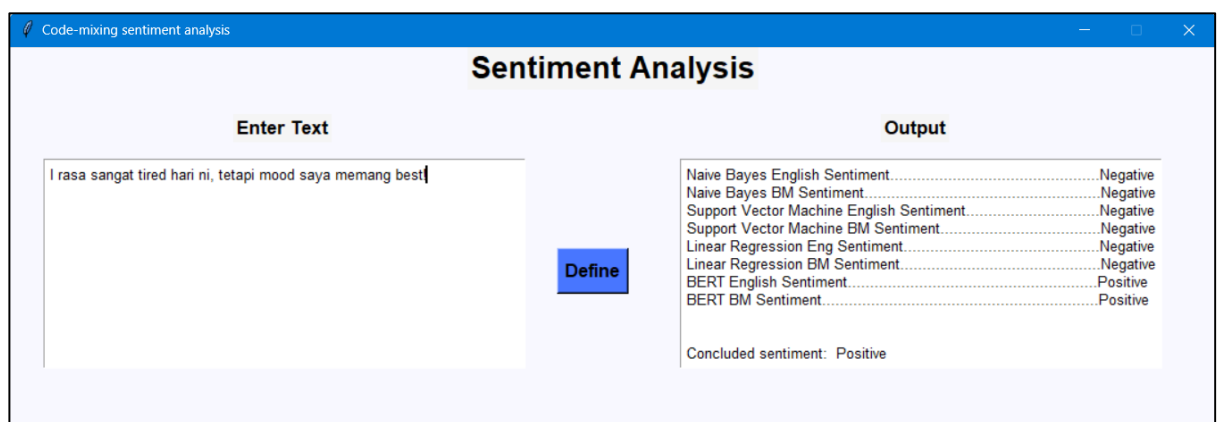
#### **Negative Sentiments:**

4. "Your presentation tadi was sangat bagus, but content dia memang boring!"
5. "The harga rumah-rumah di sini memang murah, tetapi lokasinya sangat teruk!"
6. "This movie memang best, tetapi storyline dia sangat confusing!"

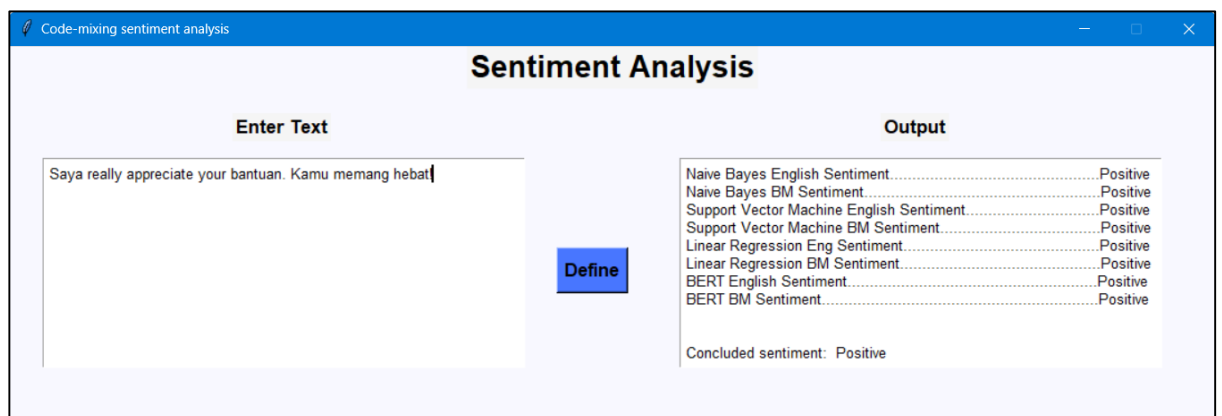




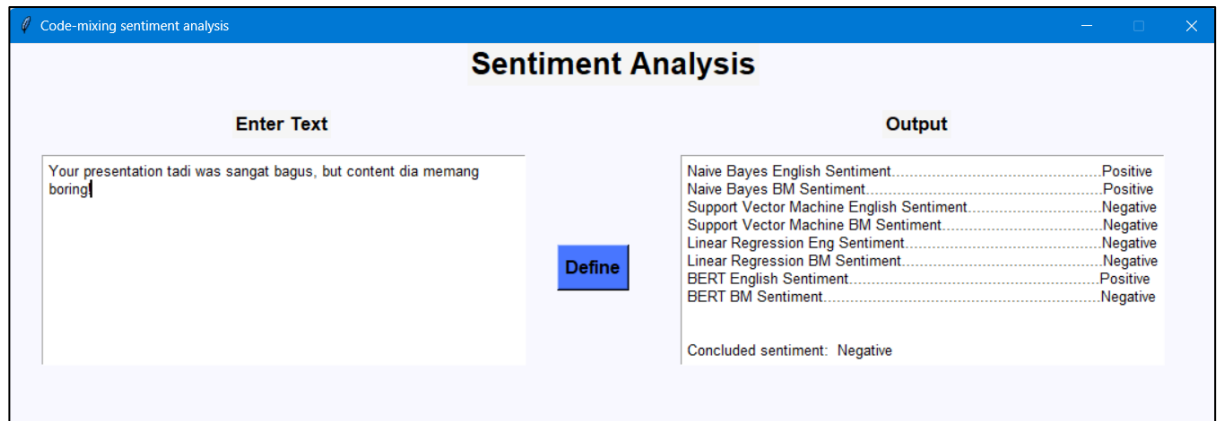
**Figure 6.3(a):** Output result for positive input 1



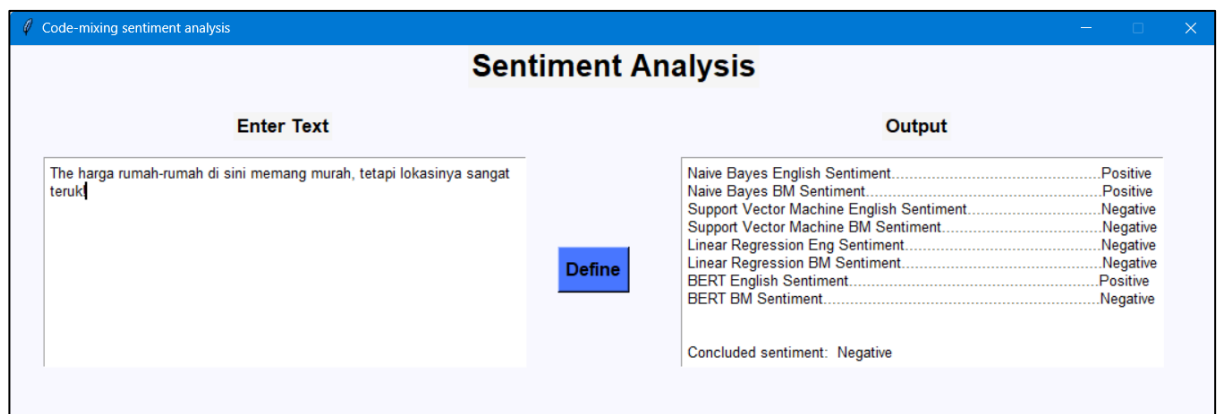
**Figure 6.3(b):** Output result for positive input 2



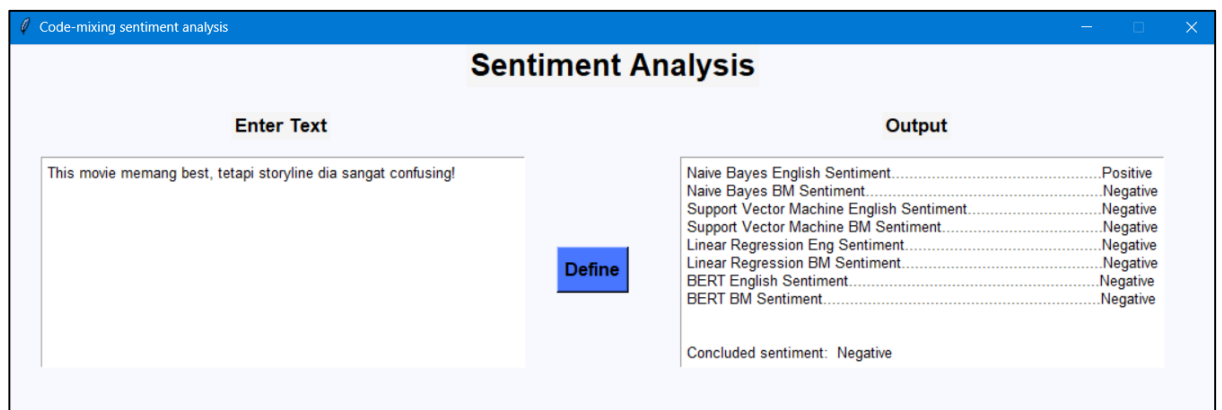
**Figure 6.3(c):** Output result for positive input 3



**Figure 6.3(d):** Output result for negative input 4



**Figure 6.3(e):** Output result for negative input 5



**Figure 2.3(f):** Output result for negative input 6

These code-mixed sentences are intentionally constructed with false positive terms, where the sentiment expressed in one part of the sentence contradicts the sentiment expressed in another part. This will provide a challenging scenario for the developed system's ability to accurately interpret and classify sentiments in such complex language mixtures. However, as shown on the above figures, the prototype system still managed to accurately predict the sentiments of each listed inputs. While the current working prototype demonstrates the feasibility of sentiment analysis on code-mixed

data, it is important to note that further refinement and optimization are required. The system's performance can be enhanced by incorporating larger and more diverse training datasets, fine-tuning the language-specific models, and addressing specific challenges related to code-mixing, such as language variations and informal expressions.

In summary, the presented demo inputs and their sentiment classification results validates the potential of the sentiment detection system for code-mixed input. The developed prototype serves as a tangible demonstration of the research findings and lays the foundation for developing a comprehensive sentiment analysis system catering to multilingual environments, and overall contributing to the broader advancements in the field of natural language processing.

## **6.4 Limitations**

While the research being conducted in this thesis is acclaimed to be successful and provides a valuable insight onto code-mixed sentiment analysis, it is still essential to acknowledge the limitation that is realized upon from this study, which might highlight on the areas where further investigation and improvements are necessary for any future research endeavours on this topic.

The first limitation that has been realized upon conducting this study, is on the data availability and quality. One of the primary limitations of this research is regarded on the availability and quality of code-mixed English-Malay datasets, specifically on code-mixed data that had been annotated for sentiment analysis. Upon researching and looking for the datasets, it is realized that the English-Malay Code-mixed data that are complemented with sentiment annotations are relatively scarce, making it challenging to train and validate sentiment analysis models that generalize well across different domains and contexts. The limited availability of the found code-mixed datasets seems to pose some difficulties in capturing the diverse range of code-mixing phenomena and sentiment expressions. Additionally, the quality of existing datasets may vary, with inconsistencies in annotation guidelines, and subjective interpretations. These limitations can and may impact the model's ability to accurately capture the nuances of sentiment in code-mixed data.

The second limitation that has been realized upon this study, is on the different code-mixing variations that the English and Malay languages can be construct upon. The notion of code-mixing can manifest in many different forms, including intra-sentential code-switch and inter-sentential mixing. This research mainly focuses on exploring sentiment analysis in code-mixed sentences, but it may not fully capture the complexities arising from code-switching at the word or phrase level. Expanding the language coverage in code-mixed sentiment analysis is crucial for a comprehensive understanding of sentiment expression in mixed-language texts. Future research on this topic should explore on sentiment analysis models that encompass a broader consideration of linguistic variations, cultural nuances, and sentiment markers specific to each language involved in the code-mixing phenomenon.

Lastly, another limitation on this research, is on the lack of focus and emphasis on the informal expressions and slang of each of the language pairs. Code-mixing often involves the usage of informal expressions, slang, or non-standard language forms. These expressions play quite a significant role in sentiment expression within specific communities or social groups. However, this research may not fully account for the unique sentiment cues and connotations associated with informal expressions and slang in code-mixed data, where they can have context-specific meanings and sentiment associations that may differ from their literal translations. These linguistic elements can greatly impact sentiment analysis results, as their interpretation requires a deep understanding of cultural references, idiomatic expressions, and local sentiments. Addressing the challenge of informal expressions and slang in code-mixed sentiment analysis could require on specialized attention. Future research on these topics could explore on the techniques to capture and analyse informal language forms, including the use of domain-specific lexicons, sentiment dictionaries, or linguistic resources that cover colloquial and slang expressions.

## **6.5 Research Achievements**

Reaching to the end of the thesis, this section is written in the aim to reiterate and address the initial research questions posed and provide insights into the findings obtained. In the following subsections, we will discuss each research question in detail, presenting the findings and implications derived from the research conducted in this thesis. These insights contribute to our understanding of sentiment analysis in code-

mixed language environments and shed light on the potential applications and challenges in this domain.

**RQ1: How can we perform sentiment classification upon code-mixed datasets that result in a compelling outcome?**

The research conducted in this thesis demonstrates that sentiment classification on code-mixed datasets can yield compelling outcomes by employing a combination of machine learning models and leveraging language-specific approaches. The ensemble model with a voting system proposed in this study has shown promising results in enhancing sentiment classification accuracy. By integrating the collective decisions of multiple models, the ensemble model optimizes sentiment classification performance on code-mixed data and achieves compelling outcomes in sentiment analysis tasks.

**RQ2: Which Sentiment Analysis approach is optimal and more compelling to be utilized in classifying code-mixed datasets?**

The research explored various Sentiment Analysis approaches, including Naive Bayes, LSVC, logistic regression, and BERT models. Each approach was implemented with both English and Malay languages, providing language-specific models for sentiment classification on code-mixed data. The findings indicate that the BERT models, with their ability to capture contextual information, outperformed the traditional machine learning models in terms of accuracy and effectiveness. Therefore, BERT models are considered more optimal and compelling for classifying code-mixed datasets in sentiment analysis tasks. But if it were to be chosen between the machine learning models, Logistic Regression model seems to be the most compelling out of the three models, and were performing rather stable across validating through the different datasets.

**RQ3: To what extent of accuracy can we model and classify the code-mixed data using Sentiment Analysis approaches?**

The research demonstrates that sentiment classification models trained on code-mixed datasets can achieve a substantial level of accuracy. The ensemble model, which combines the outputs of multiple language-specific models, further enhances the sentiment classification accuracy achieving the highest sentiment accuracy of 0.78. The specific accuracy levels attained may vary depending on the dataset, language

pairs, and the complexity of code-mixing patterns. However, the research findings indicate that by leveraging machine learning models, BERT models and adopting language-specific approaches, it is possible to achieve a higher degree of accuracy in sentiment classification on code-mixed data.

**RQ4: How does code-mixed datasets influence the performance and accuracy of sentiment classifying compared to monolingual data models?**

Code-mixed datasets present unique challenges in sentiment classification compared to monolingual models. The notion of mixed languages, code-mixing/code-switching occurrences, and specific cultural nuances within code-mixed data can highly impact on the performance and sentiment accuracy of classification models. The research findings indicate that sentiment classification on code-mixed datasets often requires some kind of specialized approaches that considers on the linguistic variations, code-switching patterns, and sentiment expressions that is specific to each language involved. While sentiment classification accuracy on code-mixed data can seem high, it may differ from the accuracy achieved on monolingual datasets due to the added complexity introduced by code-mixing phenomena.

## References

- Addiga, A., & Bagui, S. (2022). Sentiment Analysis on Twitter Data Using Term Frequency-Inverse Document Frequency. *Journal of Computer and Communications*, 10(08), 117–128. <https://doi.org/10.4236/jcc.2022.108008>
- Ahmad, M., Aftab, S., Muhammad, S. S., & Ahmad, S. (2017). Machine learning techniques for sentiment analysis: A review. *Int. J. Multidiscip. Sci. Eng*, 8(3), 27.
- Alexandropoulos, S. A. N., Kotsiantis, S. B., & Vrahatis, M. N. (2019). Data preprocessing in predictive data mining. *The Knowledge Engineering Review*, 34. <https://doi.org/10.1017/s026988891800036x>
- Alzubi, J., Nayyar, A., & Kumar, A. (2018). Machine Learning from Theory to Algorithms: An Overview. *Journal of Physics: Conference Series*, 1142, 012012. <https://doi.org/10.1088/1742-6596/1142/1/012012>
- Ansari, M. T., & Govilkar, S. (2018). Sentiment Analysis of Mixed Code for The Transliterated Hindi and Marathi Texts. *International Journal on Natural Language Computing*, 7(2), 15–28. <https://doi.org/10.5121/ijnlc.2018.7202>
- Bakar, M. A. A., Ariff, N. M., & Hui, E. X. (2018). Exploratory data analysis of Twitter's rhythm in Malaysia. *AIP Conference Proceedings*. <https://doi.org/10.1063/1.5054255>
- Berrar, D. (2019). Bayes' Theorem and Naive Bayes Classifier. *Encyclopedia of Bioinformatics and Computational Biology*, 403–412. <https://doi.org/10.1016/b978-0-12-809633-8.20473-1>
- Bhattacharyya, P. (2013). *Sentiment Analysis*. <https://doi.org/10.1109/icetacs.2013.6691379>
- Chawla, S., & Mehrotra, M. (2018). *An Ensemble-Classifier Based Approach for Multiclass Emotion Classification of Short Text*. <https://doi.org/10.1109/icrito.2018.8748757>
- De Diego, I. M., Redondo, A., Fernández, R. D., Navarro, J. a. R., & Moguerza, J. M. (2022). General Performance Score for classification problems. *Applied Intelligence*, 52(10), 12049–12063. <https://doi.org/10.1007/s10489-021-03041-7>
- Devika, M., Sunitha, C., & Ganesh, A. (2016). Sentiment Analysis: A Comparative Study on Different Approaches. *Procedia Computer Science*, 87, 44–49. <https://doi.org/10.1016/j.procs.2016.05.124>
- Dutta, S., Agrawal, H., & Roy, P.K. (2021). Sentiment Analysis on Multilingual Code-Mixed Kannada Language. Fire.

- Ghosh, S., Dasgupta, A., & Swetapadma, A. (2019). A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification. *2019 International Conference on Intelligent Sustainable Systems (ICISS)*. <https://doi.org/10.1109/iss1.2019.8908018>
- Gladence, L. M., Karthi, M., & Anu, V. M. (2015). A statistical comparison of logistic regression and different Bayes classification methods for machine learning. *ARPJ Journal of Engineering and Applied Sciences*, 10(14), 5947-5953.
- Go, A., Bhayani, R. & Huang, L. (2009). Twitter Sentiment Classification using Distant Supervision. *Processing, CS224N Project Report, Stanford*.
- Hossin, M., & Sulaiman, M. R. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2), 01–11. <https://doi.org/10.5121/ijdkp.2015.5201>
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, & Kristina Toutanova. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv: Computation and Language*.
- Jain, A., & Dandannavar, P. (2016). *Application of machine learning techniques to sentiment analysis*. <https://doi.org/10.1109/icatcct.2016.7912076>
- Jain, A., Patel, H., Nagalapatti, L., Gupta, N., Mehta, S., Guttula, S., Mujumdar, S., Afzal, S., Sharma Mittal, R., & Munigala, V. (2020). Overview and Importance of Data Quality for Machine Learning Tasks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. <https://doi.org/10.1145/3394486.3406477>
- Jain, K., & Kaushal, S. (2018). A Comparative Study of Machine Learning and Deep Learning Techniques for Sentiment Analysis. *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. <https://doi.org/10.1109/icrito.2018.8748793>
- Kalaivani, K. S., Uma, S., & Selvi, C. S. K. (2020). *A Review on Feature Extraction Techniques for Sentiment Classification*. <https://doi.org/10.1109/iccmc48092.2020.iccmc-000126>
- Kansara, D., & Sawant, V. (2020). Comparison of Traditional Machine Learning and Deep Learning Approaches for Sentiment Analysis. *Algorithms for Intelligent Systems*, 365–377. [https://doi.org/10.1007/978-981-15-3242-9\\_35](https://doi.org/10.1007/978-981-15-3242-9_35)
- Khairnar, J., & Kinikar, M. (2013). Machine learning algorithms for opinion mining and sentiment classification. *International Journal of Scientific and Research Publications*, 3(6), 1-6.



- Lee, G. Y., Alzamil, L., Doskenov, B., & Termehchy, A. (2021). A Survey on Data Cleaning Methods for Improved Machine Learning Model Performance. *arXiv preprint arXiv:2109.07127*.
- Leevy, J. L., Khoshgoftaar, T. M., Bauder, R. A., & Seliya, N. (2018). A survey on addressing high-class imbalance in big data. *Journal of Big Data*, 5(1). <https://doi.org/10.1186/s40537-018-0151-6>
- Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. *Machine Learning: ECML-98*, 4–15. <https://doi.org/10.1007/bfb0026666>
- Liu, Z., Lv, X., Liu, K., & Shi, S. (2010). Study on SVM Compared with the other Text Classification Methods. *2010 Second International Workshop on Education Technology and Computer Science*. <https://doi.org/10.1109/etcs.2010.248>
- Morrell, J. (2022, June 2). *Does More Data Equal Better Analytics?* Datameer. <https://www.datameer.com/blog/does-more-data-equal-better-analytics/>
- Poomka, P., Kerdprasop, N., & Kerdprasop, K. (2021). Machine Learning Versus Deep Learning Performances on the Sentiment Analysis of Product Reviews. *International Journal of Machine Learning and Computing*, 11(2), 103–109. <https://doi.org/10.18178/ijmlc.2021.11.2.1021>
- Romadhona, N. P., Sin-En, L., Bo-Han, L., & Tzong-Han, R. T. (2022). BRCC and SentiBahasaRojak: The First Bahasa Rojak Corpus for Pretraining and Sentiment Analysis Dataset. *Journal of the International Committee on Computational Linguistics, Proceedings of the 29th International Conference on Computational Linguistics*, 2022.coling-1.389. <https://aclanthology.org/2022.coling-1.389>
- S, T., & Poornachandran, P. (2018). *Code-Mixing: A Brief Survey*. <https://doi.org/10.1109/icacci.2018.8554413>
- Sabri, N., Edalat, A., & Bahrak, B. (2021). Sentiment Analysis of Persian-English Code-mixed Texts. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2102.12700>
- Salmon, B. P., Kleynhans, W., Schwegmann, C. P., & Olivier, J. C. (2015). *Proper comparison among methods using a confusion matrix*. <https://doi.org/10.1109/igarss.2015.7326461>
- Singh, M., Goyal, V., & Raj, S. (2019). *Sentiment Analysis of English-Punjabi Code Mixed Social Media Content for Agriculture Domain*. <https://doi.org/10.1109/iscon47742.2019.9036204>
- Singh, P. K., & Shahid Husain, M. (2014). Methodological Study of Opinion Mining And Sentiment Analysis Techniques. *International Journal on Soft Computing*, 5(1), 11–21. <https://doi.org/10.5121/ijsc.2014.5102>

- Singh, S., & Sarraf, T. (2020). *Sentiment Analysis of a Product based on User Reviews using Random Forests Algorithm*.  
<https://doi.org/10.1109/confluence47617.2020.9058128>
- Suciati, A., & Budi, I. (2019). *Aspect-based Opinion Mining for Code-Mixed Restaurant Reviews in Indonesia*.  
<https://doi.org/10.1109/ialp48816.2019.9037689>
- Taneja, K. K., & Vashishtha, J. (2022). Comparison of Transfer Learning and Traditional Machine Learning Approach for Text Classification. In *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*. <https://doi.org/10.23919/indiacom54597.2022.9763279>
- Terryyz. (2020). Googletrans [Library]. In *py-googletrans*. Github. <https://py-googletrans.readthedocs.io/en/latest/>
- Zolkepli, H. (2018a). Supervised Sentiment for Bahasa Twitter [Dataset]. In *Malay-Dataset*. Github. <https://github.com/huseinzol05/malay-dataset/tree/master/sentiment/supervised-twitter>
- Zolkepli, H. (2018b). BERT-Bahasa, Natural-Language-Toolkit library for bahasa Malaysia, powered by Deep Learning Tensorflow [Model]. In *Malaya*. Github. <https://github.com/huseinzol05/Malaya/tree/master/pretrained-model/bert#citation>