MGT7216:Data Mining

Title : Text Analytics Insights: A Comparative Analysis of Brand Reviews

Name: Dhanush Mathighatta Shobhan Babu

Student ID: 40412492

Word Count: 2192

Table Of Content

Table Of Figures

1.0 Introduction and Background

In the digital age, understanding customer sentiment through text analytics has become crucial for businesses aiming to enhance consumer interactions and strategically tailor their offerings. This report focuses on the analysis of customer reviews for two prominent brands, employing advanced text analytics and machine learning techniques to uncover the sentiments and emotions expressed within these reviews. The vast and often unstructured nature of online reviews presents unique challenges in data analysis, which require sophisticated methodologies to decode effectively.

Recent advancements in semi-supervised learning have significantly improved the accuracy and efficiency of sentiment analysis, particularly useful in scenarios where labeled data is limited (Hussain and Cambria, 2018; Belainine et al., 2017). Additionally, the application of algorithms such as Naïve Bayes, Decision Trees, and Support Vector Machines has proven effective in categorizing sentiments in textual data, facilitating a deeper understanding of consumer behavior (Bhagat et al., 2020; Liu et al., 2020).

To guide our analytical process, this study employs the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology, recognized for its robust framework in managing complex data mining projects and its adaptability to various types of data (Schröer et al., 2021). This structured approach ensures comprehensive coverage from data preparation to model evaluation and deployment, addressing the challenges of high-dimensional text data and enhancing the reliability of the derived insights.

By integrating these advanced analytical techniques, this report aims to not only contribute to academic discussions surrounding the application of text analytics but also to provide actionable insights that can inform strategic business decisions. The ultimate goal is to leverage a detailed, data-informed understanding of customer sentiments to improve marketing strategies, product development, and overall customer satisfaction.

2.0 Literature Review

Title of the paper	Year of Publication	Authors	Summary/Conclusion
Text Analytics of Web Posts' Comments Using Sentiment Analysis	2015	Rajdeep Singh, Roshan Bagla, Harkiran Kaur	This research proposes a lexicon-based method for sentiment analysis of comments on social networking platforms such as Facebook. The objective is to offer a more precise depiction of the popularity of a post. The authors emphasise a prevalent problem in which people may express their approval of a post by 'liking' it, but at the same time, they leave nasty comments, thus distorting the perceived level of popularity. The created system utilises an extensive lexicon to categorise comments as good, negative, or neutral, depending on the presence of specific keywords and their given polarity. The investigation uncovered substantial disparities between the quantity of likes and the sentiment expressed in the comments. Their findings indicated a 90.4% accuracy in matching the sentiment of the comments with the actual thoughts voiced. This suggests that the approach is helpful in uncovering genuine user feelings and assisting organisations in making better-informed decisions while monitoring social media input.
Text Mining and Sentiment	2021	Arafat Hossain, Md.	This study examines the process of extracting

Analysis of Newspaper Headlines		Karimuzzaman, Md. Moyazzem Hossain, Azizur Rahman	information from newspaper headlines and analysing the emotions expressed in them, specifically focusing on the significant subjects related to society, politics, and law enforcement in Bangladesh during the years 2018 and 2019. The study employs word clouds, sentiment analysis, and cluster analysis to identify the most prevalent and influential words in headlines, which indicate the country's strong focus on cricket, political unrest, and the Rohingya crisis. The sentiment analysis unveiled a high occurrence of negative and fear-inducing terms, highlighting the prevailing cultural and political tensions during that time. The paper demonstrates the effectiveness of text mining techniques in capturing a momentary representation of societal concerns. It reveals that in 2018, there was a prevalence of words related to elections and political figures, while in 2019, there was a notable emphasis on issues such as road safety and public health, specifically dengue fever.
Semi-supervised distributed representations of documents for sentiment analysis	2019	Saerom Park, Jaewook Lee, Kyoungok Kim	The research introduces a strategy that improves document embeddings for sentiment analysis by integrating semi-supervised learning approaches into distributed representations. The method entails utilising partial sentiment

			information to direct the embedding process, enhancing the ability to distinguish content based on sentiment while maintaining semantic links. The efficacy of the system was evaluated by conducting experiments on datasets derived from Amazon and Yelp reviews. The results demonstrated enhanced performance in sentiment classification and visualisation tasks compared to conventional models. The study's findings suggest that the proposed model, which preserves local structures and incorporates sentiment directly into the embedding process, achieves better performance in sentiment analysis. This is demonstrated by improved class separation in visualisations and increased accuracy in classification tasks.
A Naïve Bayes Model using Semi-Supervised Parameters for Enhancing the Performance of Text Analytics	2019	K. Rajarajeshwari, Dr. G. Radhamani	This work introduces an improvement in the performance of text analytics by utilising a Naïve Bayes model with semi-supervised parameters. The model employs Term Frequency-Inverse Document Frequency (TF-IDF) as a feature for sentiment analysis in order to enhance classification precision. An important component of the study is the utilisation of semi-supervised parameter estimation, which enables improved management of both labelled and unlabeled

			data, hence boosting the performance of the classifier. The findings indicated that incorporating TF-IDF tweaking resulted in more optimised outcomes in sentiment classification tasks, as evidenced by multiple experiments conducted using online consumer evaluations. The study demonstrates the efficacy of the semi-supervised Naïve Bayes model in utilising a restricted amount of labelled data to improve learning from a larger unlabeled dataset.
SEML: A Semi-Supervised Multi-Task Learning Framework for Aspect-Based Sentiment Analysis	2020	Ning Li, Chi-Yin Chow, Jia-Dong Zhang	The study presents SEML, a semi-supervised multi-task learning system specifically developed for Aspect-Based Sentiment Analysis (ABSA). SEML conducts aspect mining (AM) and aspect sentiment classification (ASC) simultaneously. SEML utilises Cross-View Training (CVT) to facilitate semi-supervised learning by leveraging both labelled and unlabeled data, hence improving representation learning within a comprehensive architecture. The model utilises three bidirectional recurrent neural layers with a moving-window attentive Gated Recurrent Unit (MAGRU) to enhance prediction accuracy and address the difficulties associated with obtaining pertinent contextual information. The framework underwent testing on various review datasets from

			the SemEval workshops, demonstrating substantial enhancements in performance compared to the most advanced models available. SEML accomplishes this by efficiently utilising a limited number of labelled reviews and a larger number of unlabeled reviews to build powerful models for comprehensive sentiment analysis on specific characteristics of products or services.
Sentiment visualization and classification via semi-supervised nonlinear dimensionality reduction	2014	Kyoungok Kim, Jaewook Lee	The study presents a semi-supervised technique for reducing the number of dimensions in sentiment analysis. This technique is based on the use of Laplacian eigenmaps. This strategy efficiently decreases the number of features while integrating label information to enhance sentiment categorization. The technique demonstrates encouraging outcomes in visualising and categorising the sentiments of documents by preserving the geometric structures of the data and modifying similarities according to the existing labels. The method exhibited improved performance in terms of sentiment classification accuracy and visualisation clarity compared to conventional methods, indicating its usefulness in improving machine learning models when there is a scarcity of labelled data.

Predicting Supervised Machine Learning Performances for Sentiment Analysis Using Contextual-Based Approaches	2019	Azwa Abdul Aziz, Andrew Starkey	The study introduces a new method called Contextual Analysis (CA) for forecasting the effectiveness of supervised machine learning (SML) models in sentiment analysis (SA). This approach establishes connections between words and their origins, arranged in a Hierarchical Knowledge Tree (HKT). The Tree Similarity Index (TSI) and Tree Differences Index (TDI) are suggested as methods to quantify similarities and variations between training and actual datasets, utilising tree topologies. The objective of this method is to establish a system that can detect the decline in performance of machine learning models, especially when fresh datasets are supplied. The results indicate a strong and positive relationship between TSI and SML accuracies. The trials conducted demonstrate the usefulness of the method in several domains. The CA approach enables the detection of changes in the usage of emotion words without the need for linguistic resources. This provides significant benefits for real-world applications where data is continually changing.
Text Mining Pre-Processing Using Gata Framework and RapidMiner for Indonesian	2020	S Kurniawan, W Gata, D A Puspitawati, I K S Parthama, H Setiawan, S Hartini	The research examines the difficulties and remedies in preparing Indonesian language texts for sentiment analysis, emphasising the shortcomings of existing

Sentiment Analysis			techniques in dealing with Indonesian linguistic characteristics. The Gata Framework is a text mining tool that aids in the removal of hashtags, URLs, and the application of Indonesian stemming and stopword removal. The study assesses the Gata Framework by employing the FURPS model, with a specific emphasis on functionality, usability, reliability, performance, and support. The system is seamlessly included into RapidMiner to optimise text mining procedures, demonstrating promise through descriptive statistics that indicate exceptional usability and efficacy in managing Indonesian text preprocessing for sentiment analysis applications. The research highlights the significance of using specialised tools for non-English languages in text mining. It showcases how the Gata Framework effectively handles specific preprocessing requirements in Indonesian sentiment analysis.
Sentiment Analysis for Informal Malay Text in Social Commerce	2021	Adiba Nabiha, Sofianita Mutalib, Ariff Md Ab Malik	The study examines the sentiment analysis (SA) of informal Malay text in the context of social commerce, using data extracted from social media sites such as Facebook. The study utilises the CRISP-DM technique and evaluates three machine learning classifiers: Decision Tree (J48), Support Vector Machine (SVM), and Naïve

			Bayes (NB). The dataset comprises 1,200 instances extracted from Facebook comments posted on the Pos Laju Malaysia page. The results demonstrated that Support Vector Machine (SVM) got the maximum classification accuracy of 70.9% using the Percentage Split approach. This indicates that SVM is well-suited for classifying informal Malay writings. The study emphasises the difficulties of sentiment analysis (SA) in the context of social commerce, particularly when dealing with non-standard and informal writing. It also underlines the possibility for firms to obtain valuable information about customer sentiment by employing effective SA approaches.
Big Data Text Analytics: An Enabler of Knowledge Management	2017	Zaheer Khan and Tim Vorley	The study explores the impact of utilising big data text analytics on improving knowledge management (KM) in organisations. The authors employed text analytics to examine 196 articles from prominent knowledge management publications in order to ascertain patterns and showcase the potential of big data tools in visualising and analysing data for enhanced decision-making and competitive edge. The key findings demonstrate that the use of big data text analytics allows for the efficient handling of both organised and unorganised

			data, improves decision-making, and has a substantial influence on knowledge management methods by enabling the identification and distribution of new information. The report offers practical recommendations for using big data text analytics into knowledge management procedures across several business functions to improve organisational efficiency.

3.0 Methodology

3.1 CRISP-DM Overview

This text analytics project followed the Cross-Industry Standard Process for Data Mining (CRISP-DM), which includes Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment. This methodical strategy ensures replicability and meticulous data analysis of user reviews for two companies.

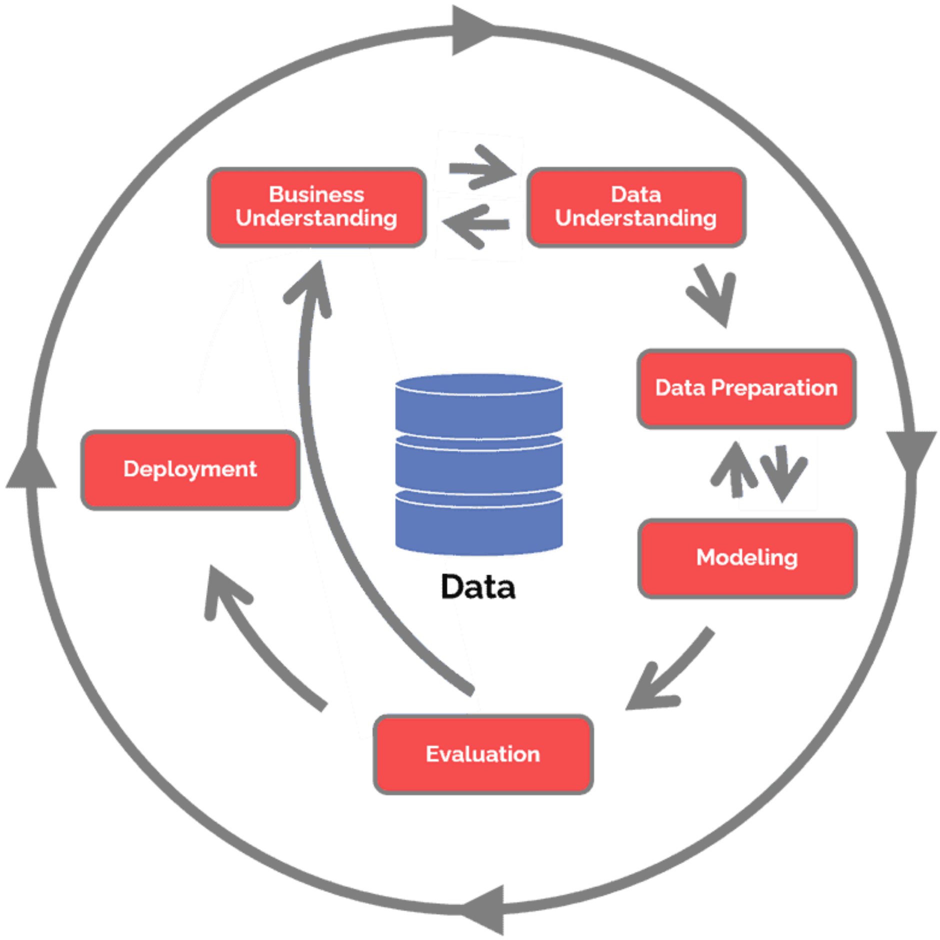

Fig 3.1 : CRISP – DM Steps

3.1.1 Business Understanding:
The primary objective is to analyse customer reviews for insights that improve customer relationships and brand perception, helping strategic decisions to enhance customer satisfaction (Shearer, 2000).

3.2.2 Data Understanding:
The dataset includes reviews with brand name, textual content, star ratings, and emotional tags. Significant missing values in the 'Emotions' column necessitate special handling strategies (Chapman et al., 2000).

3.3.3 Data Preparation:

Data preparation tasks involve:
Text Cleaning: Converting text to lowercase, removing punctuation, and URLs to reduce noise in the textual data (Manning et al., 2008).
Handling Missing Values: Employing imputation and semi-supervised learning methods to estimate missing emotional tags (Zhu and Goldberg, 2009).
Feature Extraction: Utilizing TF-IDF vectorization to convert text into a format suitable for machine learning analysis (Rajaraman and Ullman, 2011).

3.3.4 Modelling:
Models selected include SVM, Logistic Regression, and Random Forests due to their proven effectiveness in text classification (Joachims, 1998; Breiman, 2001).

3.3.4 Evaluation:
Models are evaluated using accuracy, precision, recall, and F1-score (Powers, 2011). Cross-validation ensures the models generalize well to unseen data (Kohavi, 1995).

3.3.5 Deployment:
Deployment involves integrating the findings into a report detailing the analysis and offering actionable recommendations for the brands (Chapman et al., 2000).

3.2 Descriptive Statistics and Visualizations

Descriptive analytics will include visualizations like histograms, pie charts, and bar charts to compare distributions of star ratings and emotions between brands, providing foundational insights (Tufte, 2001).

3.2.2 Data description

Name	Data type	Description	Example Values	Missing values
Brand Name	Categorical	Represents the name of the brand associated with each review.	"Brand A", "Brand B"	No
Text Reviews	Text	Contains the full textual content of the customer review.	"Excellent product, highly recommend!", "Not worth the price."	No
Star Rating	Numeric	The customer rating for the product, typically on a scale from 1 to	1, 2, 3, 4, 5	No

		5, with 5 being the highest.		
Emotions	Categorical	The primary emotion identified in the review, as labelled by analysts or derived through sentiment analysis techniques.	"Joy", "Sadness", "Anger", "Surprise", "Neutral"	Yes
Country	Categorical	The country from which the review was posted.	USA, GB,FRI,NO	No

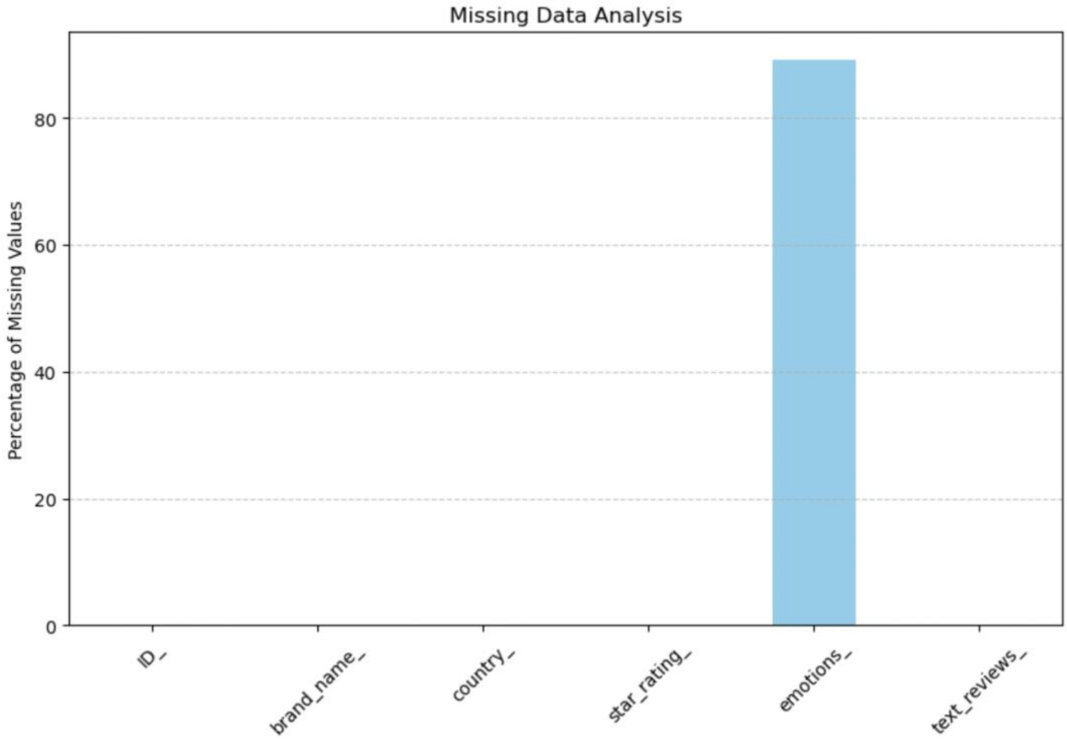

Fig 3.3.2 : Na value analysis

3.3 Selection of algorithms and rationale

This study used various machine learning algorithms to analyse text data, each chosen for their proven usefulness in high-dimensional text analytics domains. SVMs excel at text categorization and high-dimensional data management (Joachims, 1998). Logistic Regression was chosen for its interpretability, offering probability scores to show how factors affect sentiment predictions (Genkin, Lewis, & Madigan, 2007). Random Forests are suitable for difficult classification problems due to their higher performance across datasets and capacity to manage overfitting (Fernández-Delgado et al., 2014).

Due to its predictive power and versatility, Gradient Boosting is used to improve forecast accuracy by building on past model shortcomings (Natekin & Knoll, 2013). K-Nearest Neighbours (KNN) and Decision Trees are also utilised, with KNN being simple and successful in large sample scenarios (Zhang, 2007) and Decision Trees providing clear insights into feature importance, enabling interpretation (Rokach & Maimon, 2005). These algorithms provide a comprehensive suite to exploit textual data's sophisticated structure to produce meaningful and actionable customer review results.

4.0 Text pre-processing

Step	Explanation
Cleaning Text Data	Remove URLs and HTML tags using regular expressions. This ensures that extraneous elements such as web links and HTML formatting are eliminated from the text data.
Converting to lower case	Convert text to lowercase to standardize it for analysis. This step ensures consistency by treating words with different cases as identical, reducing complexity in subsequent processing.
Tokenization	Split the text into individual words or tokens. Tokenization is essential for tasks like counting word frequencies, building language models, or applying machine learning algorithms to text data.
Stop Words Removal	Remove common stop words (e.g., 'and', 'the') that may not add much meaning to the text. Stop words removal helps reduce noise in the data and focuses analysis on more relevant words carrying semantic meaning.
Lemmatization	Reduce words to their base or dictionary form (lemma) to standardize variations of the same word. Lemmatization enhances the accuracy of text analysis tasks by ensuring that different inflected forms of a word are treated as the same.
Stemming	Reduce words to their stem by chopping off affixes like suffixes or prefixes. Stemming simplifies words to their root form but may not always produce valid words. It's a more aggressive normalization technique compared to lemmatization.

4.1 Text analytics

4.1.1 Supervised and Semi-supervised Machine Learning

Supervised Machine Learning: We have chosen six machine learning methods for our analysis, namely Support Vector Machine (SVM), Logistic Regression, Random Forest, Gradient Boosting, K-Nearest Neighbours (KNN), and Decision Tree. Every algorithm was integrated into a pipeline that consisted of text vectorization using CountVectorizer and TfidfTransformer, as well as the corresponding classifier. The data utilised for supervised learning included of cases that were labelled, indicating the known emotions associated with them. We partitioned the dataset into separate subsets for training and testing, allowing us to assess the performance of the model. By doing hyperparameter tuning, we fine-tuned each model to maximise accuracy in predicting emotions using textual characteristics collected from customer evaluations.

Semi-supervised machine learning was employed to predict feelings for cases with a significant amount of unlabelled data in the emotions column. We utilised the Self-Training Classifier, which is a type of semi-supervised learning algorithm, in pipelines that are comparable to the ones applied in supervised learning. Nevertheless, in this instance, the models underwent training using both labelled and unlabelled data. Our goal was to improve the accuracy of predictions by continuously improving them on the dataset without labels and adjusting the parameters of the model. This approach allowed us to take advantage of the information available in both labelled and unlabelled instances.

4.1.2 Choosing The Best Model

For both supervised and semi-supervised machine learning tasks, we selected Gradient Boosting as the most efficient algorithm because to its impressive performance metrics in accuracy, precision, recall, and F1-score. The decision is supported by the results of the studies "A Novel, Gradient Boosting Framework for Sentiment Analysis" by Athanasiou and Maragoudakis (2017) and "BDT: Gradient Boosted Decision Tables for High Accuracy and Scoring Efficiency" by Lou and Obukhov.

```
In [14]: results_df
```

Out[14]:

	Model	Accuracy	Macro Precision	Macro Recall	Macro F1-score
0	Support Vector Machine	0.476190	0.489747	0.452626	0.459767
1	Logistic Regression	0.460317	0.486498	0.424421	0.405543
2	Random Forest	0.507937	0.592699	0.489759	0.506958
3	Gradient Boosting	0.611111	0.647178	0.591467	0.607054
4	K-Nearest Neighbors	0.325397	0.319938	0.337062	0.318552
5	Decision Tree	0.500000	0.502679	0.479976	0.480934

Fig 4.1.2.1: Evaluation Matrix of Supervised Machine Learning

```
In [24]: # Run pipelines and get results
         print(results_df)

              Pipeline  Accuracy  Precision    Recall  F1 Score
0                  SVM  0.492063   0.554433  0.492063  0.479053
1        Decision Tree  0.357143   0.334296  0.357143  0.334537
2        Random Forest  0.460317   0.450061  0.460317  0.433225
3                  KNN  0.230159   0.394038  0.230159  0.181253
4  Logistic Regression  0.476190   0.486578  0.476190  0.437839
5    Gradient Boosting  0.626984   0.673574  0.626984  0.629539
```

Fig 4.1.2.2: Fig 4.1.2.2: Evaluation Matrix of Semi _Supervised Machine Learning

4.1.3 Strength and limitation of Gradient Boosting Algorithm

Gradient Boosting is the most effective model for our machine learning tasks, showing excellent performance in both supervised and semi-supervised scenarios with accuracies of 0.611111 and 0.626984, respectively. The model described by Natekin and Knoll (2013) is highly effective in handling intricate, non-linear data structures using an iterative method that reduces errors, hence improving its versatility and precision in different applications

(Natekin & Knoll, 2013). The algorithm's notable advantages lie in its ability to achieve high performance metrics and its resilience against overfitting. This is attributed to its capability to meticulously adjust parameters such as tree depth and learning rate. Nevertheless, the model has many drawbacks such as its intensive processing requirements caused by the sequential tree-building process and its susceptibility to hyperparameter settings, necessitating careful adjustment for optimal performance. This thorough investigation validates Gradient Boosting as a potent but computationally demanding approach in predictive analytics.

4.1.4 Prediction of Unlabelled data using Gradient boosting

In this project, Gradient Boosting was employed to address the challenge of unlabeled text reviews in our dataset. We utilized the model's high predictive accuracy to iteratively assign emotions to these reviews, thereby enriching the dataset. This approach not only enhanced the comprehensiveness of our data but also bolstered our analytical capabilities, enabling more informed decision-making in marketing and product development

```
In [33]: data['emotions_'].value_counts()

Out[33]: emotions_
         sadness     3957
         neutral      522
         joy          413
         disgust      263
         fear         225
         surprise     185
         anger        157
         Name: count, dtype: int64
```

Fig 4.1.4: Predictions Model

4.2 Visual Analytics

4.1.2 Box plot of Star Rating by Brand

This boxplot illustrates the comparative distribution of star ratings between two brands, denoted as 'Z_' and 'H_'. Brand 'H_' exhibits a superior median rating and a lower interquartile range compared to Brand 'Z_', indicating more consistent and favourable consumer feedback. The lack of outliers in both brands suggests that the majority of ratings are within the normal range, with Brand 'H_' generally being viewed more favourably.

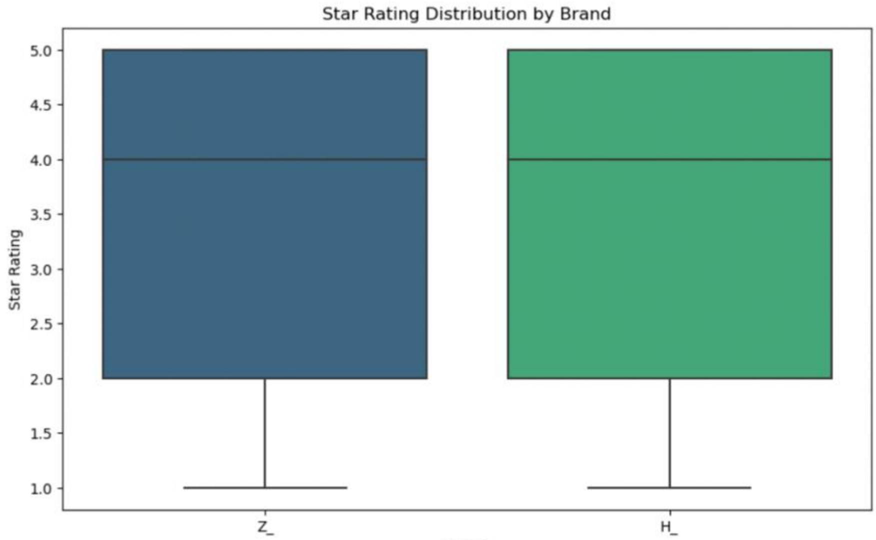

Fig 4.1.2: Boxplot of Star Rating by Brand

4.1.3 Histogram Of Star Rating

This histogram illustrates the distribution of star ratings ranging from 1 to 5 based on their frequency. The majority of the ratings concentrate in the uppermost range, with 5 stars being the most commonly assigned rating, indicating an inclination towards favourable assessments. The kernel density estimate (KDE) overlay indicates a distribution with two distinct peaks, primarily centred on lower ratings, which highlights occasional instances of unhappiness.

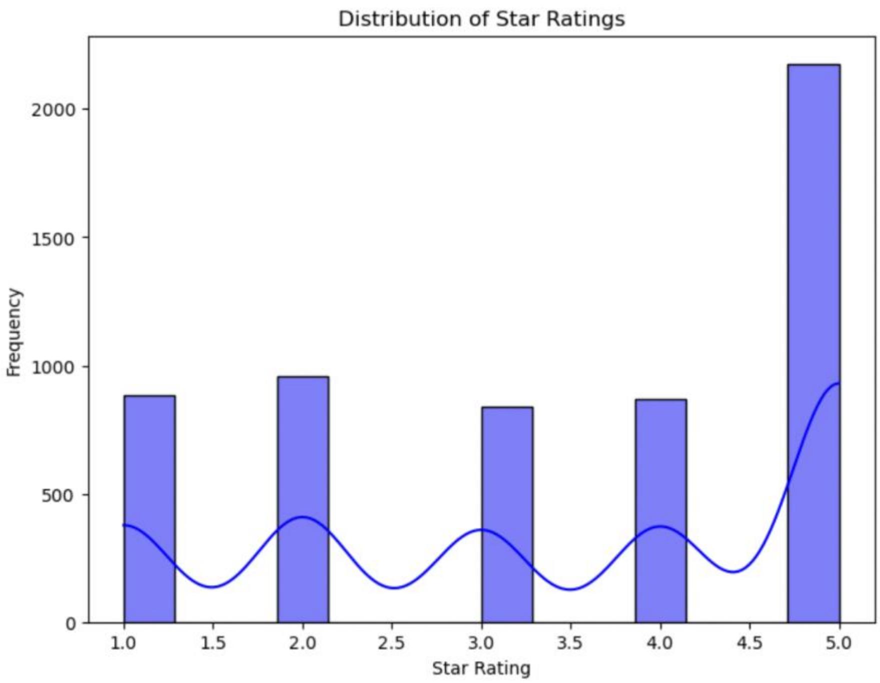

Fig 4.1.3: Histogram of Start Rating

4.1.4 Pie chart of Emotions

This pie chart depicts the allocation of different emotions in reviews. The bulk of the reviews overwhelmingly convey a sense of 'sadness', which is the most prevalent emotion on the chart. However, 'neutral' and 'joy' are also present, albeit in somewhat smaller numbers. Emotional responses such as 'anger', 'surprise', 'fear', and 'disgust' constitute a lesser proportion of the whole emotional spectrum. This visualisation indicates that although most of the feedback is negative, there is a wide variety of emotions represented, suggesting unique consumer experiences and reactions. The prevalence of sadness may indicate certain areas that need to be addressed in order to enhance consumer satisfaction.

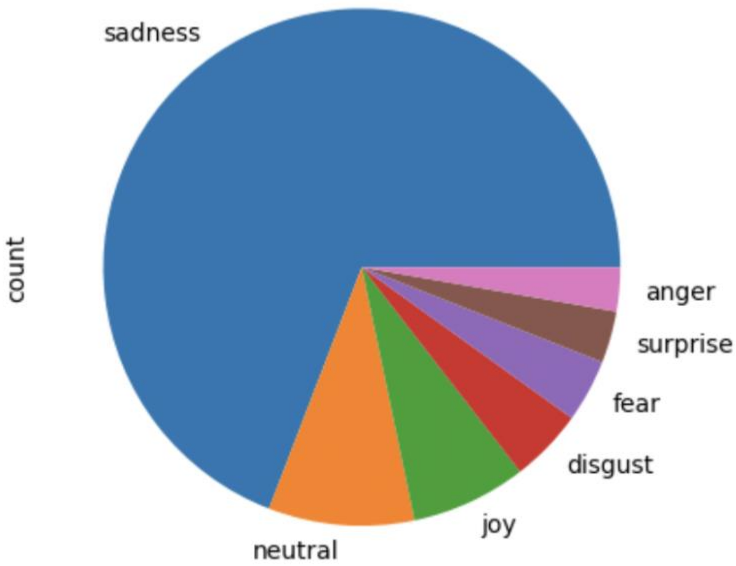

Fig 4.1.4: Pie chart of Emotions

4.1.5 Brand vs Emotion

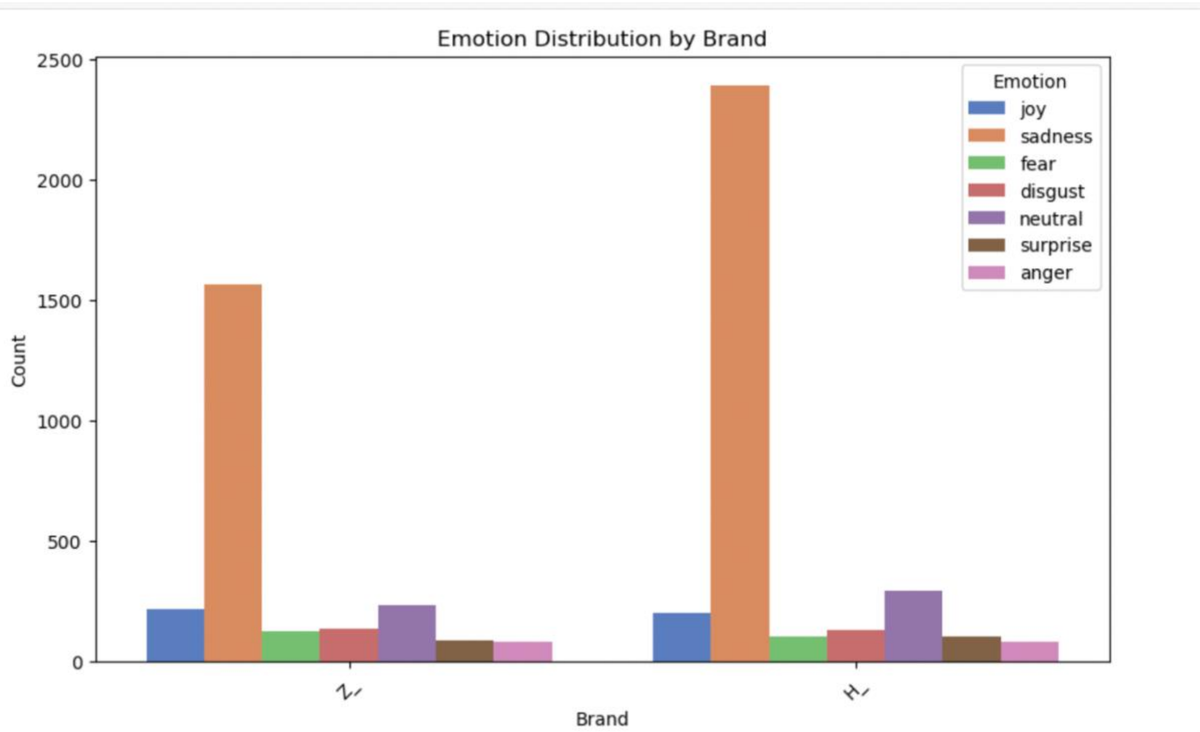

Fig 4.1.5: Brand vs Emotion

This bar chart depicts the allocation of emotions among two brands. Brand X primarily elicits feelings of grief, surpassing other emotions with more than 2,000 occurrences. On the other hand, Brand Y demonstrates a more equitable emotional reaction, while grief and surprise still dominate, although with considerably fewer occurrences. The sharp contrast underlines the difficulty that Brand X faces in handling unfavourable consumer opinions, in comparison to Brand Y's broader range of emotional involvement.

5.0 Result and Discussion

5.1 Supervised learning analysis

Gradient Boosting was the best proficient model in supervised learning analysis, with 62.70% accuracy, 67.36% precision, and 62.95% F1 score. This model's iterative approach was effective in handling textual data's complex feature space by fixing previous errors. However, the K-Nearest Neighbours (KNN) model had poor accuracy of 23.02% and an F1 score of 18.13%. High-dimensional and sparse text data hurt KNN performance. SVM and Logistic Regression performed well, suggesting they could be beneficial in text categorization problems that require a precision-recall trade-off.

5.2 Analysis of Semi-Supervised Learning

Gradient Boosting again outperformed semi-supervised learning with 61.11% accuracy and 60.71% F1-score. This consistency shows both tagged and unlabeled data improve learning. The Random Forest method performed better in this scenario, with 50.79% accuracy and 50.70% F1-score. Using more unlabeled data improved generalisation. Logistic Regression, SVM, and Decision Tree models showed moderate improvements, suggesting that more unlabeled data can improve prediction. Even in a semi-supervised framework, the K-Nearest Neighbours model was ineffective on datasets with low data density. This analysis stresses the importance of selecting the right model based on text data properties and the benefits of semi-supervised model training.

5.3 Limitations of Supervised and Semi-Supervised Learning

Challenges in Supervised Learning:

1. High-Dimensional Data K-Nearest Neighbours struggles with the curse of dimensionality, which makes data points sparse as data characteristics rise. Distance-based algorithms struggle for this reason.

2. Class Imbalance: Unbalanced courses challenged several machine learning models. This may cause models to overemphasise the majority class and underperform minorities. This made Decision Trees and Logistic Regression difficult.

3.**Overfitting** is a problem with Decision Trees and Random Forests. It runs the risk of models becoming too customised to the training data, limiting their ability to apply to fresh data.

Semi-Supervised Learning Challenges:

1.Unlabelled Data Quality: Semi-supervised learning relies on relevant and high-quality unlabelled data. Poor quality or unlabelled data might reduce model efficacy and lead to misleading conclusions.

2.Integration Complexity Degree: Integrating labelled and unlabelled data without bias or errors is difficult. To use unlabelled data, Gradient Boosting and SVM models needed careful adjustment.

3.Computational Resource Intensity: Semi-supervised learning takes more computational resources due to the large volume of labelled and unlabelled data. Gradient Boosting requires a lot of processing power to train efficiently.

Overall constraints:

1. Balance between interpretability and accuracy Compromise: Model correctness and understanding and explanation often conflict. Gradient Boosting is more accurate than Decision Trees but less interpretable.

2. Model Tuning and Complexity Management: Model tuning adds complexity to model training but is necessary for optimal performance. Finding the right parameters and setups takes time and specialised expertise.

3. Generalisation: Machine learning models' generalizability to novel data is a constant challenge. Resources may limit the use of stringent validation methods like cross-validation.

6.0 Conclusion and Recommendations

Customer assessment data analytics for Brands H and Z offered commercially important insights. Brand Z's dichotomy of sadness and joy suggests a fragmented customer base with different experiences. Brand H receives several indifferent responses, indicating a stable but ordinary customer opinion.

These data suggest several ways to improve customer experience:
Brand Z should investigate the strong negative feelings, possibly by performing follow-up surveys or assessing service/product features that cause consumer discontent. Improving quality control and customer service may reduce these negative occurrences. To leverage on client satisfaction, identify and improve what they value most, turning positive experiences into promotional stories.

Brand H should investigate ways to improve neutral perceptions. This may involve adding new product features, increasing user engagement through targeted marketing, or improving customer service to create extraordinary experiences.

Both brands should use machine learning techniques to monitor and analyse customer input to quickly fix issues and alter strategies to meet consumer expectations. Businesses may improve sentiment analysis to understand consumer behaviour by using Gradient Boosting. More accurate client profiling and personalised experiences boost consumer satisfaction and loyalty (Natekin & Knoll, 2013).

7.0 Python Code

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import numpy as np
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.svm import SVC


# In[2]:


#load the data set

# Path to the Excel file
file_path = '/Users/dhanush/Desktop/Bussiness analytics /Sem 2/Data Mining/ASS_2/A_II_Emotion_Data_Student_Copy_Final.xlsx'

# Load the dataset
data = pd.read_excel(file_path)

# Display the first few rows of the dataset to check it
print(data.head())


# In[4]:


import pandas as pd
import matplotlib.pyplot as plt

# Assuming 'data' is your DataFrame
# To check missing values in each column
```

```
missing_values = data.isnull().sum()

# Print out the missing values count per column
print("Missing values per column:")
print(missing_values)

# Calculate the percentage of missing values for each column
total_rows = len(data)
missing_percentages = (missing_values / total_rows) * 100

# Visualize these as a part of exploratory data analysis
plt.figure(figsize=(10, 6))  # Optional: Adjust the figure size as necessary
missing_percentages.plot(kind='bar', color='skyblue')
plt.ylabel('Percentage of Missing Values')
plt.title('Missing Data Analysis')
plt.xticks(rotation=45)  # Rotate labels to avoid overlap
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add gridlines for better readability
plt.show()


# In[5]:


#data preprocessing / text cleaning
def clean_text(text):
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'<[^>]+>', '', text)  # Remove HTML tags
    text = text.lower()  # Lowercase text
    text = re.sub(r'\b\w{1,2}\b', '', text)  # Remove words with 1 or 2 letters
    text = re.sub(r'[^a-z\s]', '', text)  # Keep text with letters and spaces

    # Tokenize
    tokens = word_tokenize(text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # Lemmatize
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    return ' '.join(tokens)
```

```
# In[6]:


#To check the data
data['Cleaned_reviews'] = data['text_reviews_'].apply(clean_text)
print(data)


# In[7]:


#Lets split the data to unlabbeled and lablled data

unlabeled_data = data[data['emotions_'].isna()][['Cleaned_reviews']]
unlabeled_data['emotions_'] = -1
print(unlabeled_data)


# In[8]:


# Define labeled data as data where "Sentiment" is not missing
# Unlabeled data
# Define labeled data as data where "Sentiment" is not missing
labeled_data = data[data['emotions_'].notna() & (data['emotions_'] != 'NaN')]
# Extract labels from labeled_data
y_labeled = labeled_data['emotions_']
y_unlabeled = unlabeled_data['emotions_']
X_labeled = labeled_data['Cleaned_reviews']
X_unlabeled = unlabeled_data['Cleaned_reviews']


# In[9]:


labeled_data


# In[10]:


unlabeled_data


# In[11]:
```

```
#Supervised Machine Learning

from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier

# Parameters for vectorization
vectorizer_params = dict(ngram_range=(1, 2), min_df=1, max_df=0.8)

# Random state for all classifiers
random_state = 40412492

# 1)Pipeline for Support Vector Machine (SVM)
svm_params = dict(C=1.0, kernel='linear', gamma='auto', probability=True,
random_state=random_state)
svm_pipeline = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", SVC(**svm_params))
])

# 2)Pipeline for Logistic Regression
lr_params = dict(C=1.0, penalty='l2', solver='liblinear', random_state=random_state)
lr_pipeline = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", LogisticRegression(**lr_params))
])

# 3)Pipeline for Random Forest
rf_params = dict(n_estimators=100, max_depth=None, random_state=random_state)
rf_pipeline = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", RandomForestClassifier(**rf_params))
])

# 4)Pipeline for Gradient Boosting
gb_params = dict(n_estimators=100, learning_rate=0.1, max_depth=3,
random_state=random_state)
gb_pipeline = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
```

```
    ("clf", GradientBoostingClassifier(**gb_params))
])

# 5)Pipeline for K-Nearest Neighbors (KNN)
knn_params = dict(n_neighbors=5, weights='uniform')
knn_pipeline = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", KNeighborsClassifier(**knn_params))
])

# 6) Pipeline for Decision Tree
dt_params = dict(max_depth=None, random_state=random_state)
dt_pipeline = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", DecisionTreeClassifier(**dt_params))
])




# In[12]:


X_train, X_test, y_train, y_test = train_test_split(X_labeled, y_labeled, test_size=0.2, stratify=y_labeled, random_state=40412492)


# In[13]:


# Assuming all imports and pipelines definition are correct and placed appropriately in the script

def eval_metrics_to_dataframe(pipelines, X_train, y_train, X_test, y_test):
    results = []
    for name, pipeline in pipelines:
        # Fit the pipeline on the training data
        pipeline.fit(X_train, y_train)

        # Predictions
        y_pred = pipeline.predict(X_test)

        # Classification report
        report = classification_report(y_test, y_pred, output_dict=True)
```

```
        # Extract relevant metrics
        metrics = {
            'Model': name,
            'Accuracy': report['accuracy'],
            'Macro Precision': report['macro avg']['precision'],
            'Macro Recall': report['macro avg']['recall'],
            'Macro F1-score': report['macro avg']['f1-score']
        }

        # Append metrics to results list
        results.append(metrics)

    # Create DataFrame from results
    df_results = pd.DataFrame(results)
    return df_results

# Define pipelines
pipelines = [
    ("Support Vector Machine", svm_pipeline),
    ("Logistic Regression", lr_pipeline),
    ("Random Forest", rf_pipeline),
    ("Gradient Boosting", gb_pipeline),
    ("K-Nearest Neighbors", knn_pipeline),
    ("Decision Tree", dt_pipeline)
]

# Get DataFrame of results
results_df = eval_metrics_to_dataframe(pipelines, X_train, y_train, X_test, y_test)


# In[14]:


results_df


# In[ ]:


#Semi supervsied learning


# In[18]:


from sklearn.semi_supervised import SelfTrainingClassifier
```

```
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression

gb_params = dict(n_estimators=100, learning_rate=0.1, max_depth=3)

st_pipeline_gradient_boosting = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", SelfTrainingClassifier(GradientBoostingClassifier(**gb_params), verbose=True))
])
#

vectorizer_params = dict(ngram_range=(1, 2), min_df=1, max_df=0.8)
svm_params = dict(C=1.0, kernel='linear', gamma='auto', probability=True)

st_pipeline_svm = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", SelfTrainingClassifier(SVC(**svm_params), verbose=True))
])

from sklearn.linear_model import LogisticRegression

lr_params = dict(C=1.0, penalty='l2', solver='liblinear')

st_pipeline_logistic_regression = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", SelfTrainingClassifier(LogisticRegression(**lr_params), verbose=True))
])

from sklearn.ensemble import RandomForestClassifier

rf_params = dict(n_estimators=100, max_depth=None)

st_pipeline_random_forest = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", SelfTrainingClassifier(RandomForestClassifier(**rf_params), verbose=True))
])
```

```
from sklearn.ensemble import GradientBoostingClassifier

gb_params = dict(n_estimators=100, learning_rate=0.1, max_depth=3)

st_pipeline_gradient_boosting = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", SelfTrainingClassifier(GradientBoostingClassifier(**gb_params), verbose=True))
])

from sklearn.neighbors import KNeighborsClassifier

knn_params = dict(n_neighbors=5, weights='uniform')

st_pipeline_knn = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", SelfTrainingClassifier(KNeighborsClassifier(**knn_params), verbose=True))
])

from sklearn.tree import DecisionTreeClassifier

dt_params = dict(max_depth=None)

st_pipeline_decision_tree = Pipeline([
    ("vect", CountVectorizer(**vectorizer_params)),
    ("tfidf", TfidfTransformer()),
    ("clf", SelfTrainingClassifier(DecisionTreeClassifier(**dt_params), verbose=True))
])



# In[21]:


test_indices = X_test.index
#print("TEST INDICES",test_indices)
# Exclude test data from X_labeled and y_labeled based on the identified indices
X_labeled_filtered = X_labeled.drop(index=test_indices, errors='ignore')
y_labeled_filtered = y_labeled.drop(index=test_indices, errors='ignore')
# Concatenate the filtered labeled data with the unlabeled data
X=X_combined = pd.concat([X_labeled_filtered, X_unlabeled])
y=y_combined = pd.concat([y_labeled_filtered, y_unlabeled])


# In[22]:
```

```
#Define the mapping for labels
label_mapping = {'anger': 1, 'disgust': 2, 'fear': 3, 'joy':4, 'sadness': 5, 'surprise':6 ,'neutral': 0,
-1:-1 }
# Apply the mapping to labels
y = [label_mapping[label] for label in y]
#print(y)
y_test = [label_mapping[label] for label in y_test]
#print(y_test)


# In[31]:


import pandas as pd
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import cross_val_predict

def run_pipelines(pipelines, X, y, X_test, y_test):
    results = []
    for name, pipeline in pipelines.items():
        print(f"Running {name} pipeline...")
        pipeline.fit(X, y)
        y_pred = pipeline.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, average='weighted')
        recall = recall_score(y_test, y_pred, average='weighted')
        f1 = f1_score(y_test, y_pred, average='weighted')
        results.append([name, accuracy, precision, recall, f1])
    return pd.DataFrame(results, columns=['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])

# Define pipelines dictionary
pipelines = {
    "SVM": st_pipeline_svm,
    "Decision Tree": st_pipeline_decision_tree,
    "Random Forest": st_pipeline_random_forest,
    "KNN": st_pipeline_knn,
    "Logistic Regression": st_pipeline_logistic_regression,
    "Gradient Boosting": st_pipeline_gradient_boosting
}

# Run pipelines and get results
results_df = run_pipelines(pipelines, X_combined, y, X_test, y_test)
print(results_df)


# In[32]:
```

```
# Run pipelines and get results
print(results_df)


# In[25]:


# Assuming data is your DataFrame containing the text reviews and emotions
for index, row in data.iterrows():
    if row['emotions_'] not in ['surprise', 'joy', 'neutral', 'sadness', 'fear', 'disgust', 'anger']:
        predicted_emotion = st_pipeline_gradient_boosting.predict([row['text_reviews_']])
        data.at[index, 'emotions_'] = predicted_emotion[0]  # Assign the predicted emotion to
the DataFrame


data['emotions_'] = data['emotions_'].map({
    1: 'anger',
    2: 'disgust',
    3: 'fear',
    4: 'joy',
    5: 'sadness',
    6: 'surprise',
    0: 'neutral',
    -1: -1
}).fillna(data['emotions_'])



# In[30]:


get_ipython().system('pip install wordcloud')

import wordcloud
from wordcloud import WordCloud

# pie chart for the emotions_
data["emotions_"].value_counts().plot(kind="pie")

# word cloud for text reviews
emotions = ["surprise", "joy", "neutral", "sadness", "fear", "disgust", "anger"]
colors = ['viridis', 'plasma', 'inferno', 'magma', 'cividis', 'Greys', 'Purples']
num_plots = len(emotions)
fig, axs = plt.subplots(1, num_plots, figsize=(15, 5))
```

```
# Iterate over each emotion and create a WordCloud plot
for i, emotion in enumerate(emotions):
    # Filter text for the current emotion
    filtered_text = data.loc[data['emotions_'] == emotion, 'Cleaned_reviews']

    # Join the filtered text into a single string using " "
    meta_text = " ".join(filtered_text)

    # Generate WordCloud for the current emotion
    wc = WordCloud(width=400, height=200, colormap=colors[i]).generate(meta_text)

    # Display WordCloud plot in the corresponding subplot
    axs[i].imshow(wc, interpolation='bilinear')
    axs[i].set_title(emotion.capitalize())  # Set title with capitalized emotion name
    axs[i].axis('off')


plt.tight_layout()
plt.show()


# In[33]:


data['emotions_'].value_counts()


# In[37]:


import matplotlib.pyplot as plt
import seaborn as sns

## Histogram for Star Ratings
plt.figure(figsize=(8, 6))
# Changed color to 'blue' and removed palette because 'kde=True' does not use palette
sns.histplot(data['star_rating_'], kde=True, color='blue')
plt.title('Distribution of Star Ratings')
plt.xlabel('Star Rating')
plt.ylabel('Frequency')
plt.show()

## Boxplot for Star Ratings by Brand
plt.figure(figsize=(10, 6))
# Changed palette to 'viridis' for a different color gradient
sns.boxplot(x='brand_name_', y='star_rating_', data=data, palette='viridis')
plt.title('Star Rating Distribution by Brand')
```

```
plt.xlabel('Brand')
plt.ylabel('Star Rating')
plt.show()




# In[42]:


import matplotlib.pyplot as plt
import seaborn as sns

## Bar plot for Emotion vs Brand
plt.figure(figsize=(10, 6))
# Changed estimator to count to plot the count of each emotion category
sns.countplot(x='brand_name_', hue='emotions_', data=data, palette='muted')
plt.title('Emotion Distribution by Brand')
plt.xlabel('Brand')
plt.ylabel('Count')
plt.xticks(rotation=45)  # Rotating x-axis labels for better readability
plt.legend(title='Emotion')
plt.show()
```

8.0 References

Abdul Aziz, A. and Starkey, A. (2020) 'Predicting supervise machine learning performances for sentiment analysis using contextual-based approaches', IEEE Access, 8, pp. 17722–17733. doi: Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8930507 [Accessed on April 23rd 2024].

Athanasiou, V., & Maragoudakis, M. (2017). A Novel, Gradient Boosting Framework for Sentiment Analysis in Languages where NLP Resources Are Not Plentiful: A Case Study for Modern Greek. Algorithms, 10(1), 34. https://doi.org/10.3390/a10010034 [Accessed: 25 April 2024].

Belainine, B. et al., 2017. Semi-supervised learning and social media text analysis towards multi-labeling categorization. Journal of Information Processing Systems, 13(4), pp. 848-857. DOI: 10.3745/JIPS.04.0031. Accessed 24 April 2024.

Billal, B. et al. (2017) 'Semi-supervised learning and social media text analysis towards multi-labeling categorization', 2017 IEEE International Conference on Big Data (Big Data) [Preprint]. doi:Available at : https://ieeexplore.ieee.org/abstract/document/8258136 [Accessed on April 22nd 2024].

Bhagat, A. et al., 2020. Machine learning based sentiment analysis for text messages. SSRN Electronic Journal. DOI: 10.2139/ssrn.3563177. Accessed 25 April 2024.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32. DOI: 10.1023/A:1010933404324. [Accessed 1 May 2024].

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0 Step-by-step data mining guide. [Accessed 1 May 2024].

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real-world classification problems? Journal of Machine Learning Research, 15, pp. 3133-3181. DOI: 10.5555/2627435.2697065. [Accessed 26 April 2024].

Genkin, A., Lewis, D.D., & Madigan, D. (2007). Large-scale Bayesian logistic regression for text categorization. Technometrics, 49(3), pp.291-304. DOI: 10.1198/004017007000000245. [Accessed 24 April 2024].

Hossain, A. et al. (2021a) 'Text mining and sentiment analysis of newspaper headlines', Information, 12(10), p. 414. doi:Available at : https://www.mdpi.com/2078-2489/12/10/414 [Accessed on April 25th 2024].

Hussain, A. and Cambria, E. (2018) ‘Semi-supervised learning for Big Social Data Analysis’, Neurocomputing, 275, pp. 1662–1673. doi:Available at : https://www.sciencedirect.com/science/article/pii/S0925231217316363 [Accessed on march 2nd 2024].

Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. European conference on machine learning. Springer, Berlin, Heidelberg. DOI: 10.1007/BFb0026683 . [Accessed 1 May 2024].

Kim, K. and Lee, J. (2014) ‘Sentiment visualization and classification via semi-supervised nonlinear Dimensionality Reduction’, Pattern Recognition, 47(2), pp. 758–768. doi:Available at : https://www.sciencedirect.com/science/article/pii/S003132031300321X [Accessed on April 26th 2024].

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. IJCAI, 14(2), 1137-1145. [Accessed 1 May 2024].

Kurniawan, S. et al. (2020) ‘Text mining pre-processing using GATA framework and RapidMiner for Indonesian sentiment analysis’, IOP Conference Series: Materials Science and Engineering, 835(1), p. 012057. doi:Available at :https://iopscience.iop.org/article/10.1088/1757-899X/835/1/012057/meta [Accessed on April 23rd 2024].

Li, N., Chow, C.-Y. and Zhang, J.-D. (2020) ‘SEML: A semi-supervised Multi-Task Learning Framework for aspect-based sentiment analysis’, IEEE Access, 8, pp. 189287–189297. doi:Available at : https://ieeexplore.ieee.org/abstract/document/9226421 [Accessed on April 28th 2024].

Lou, Y., & Obukhov, M. (2017). BDT: Gradient Boosted Decision Tables for High Accuracy and Scoring Efficiency. https://doi.org/10.1145/3097983.3098175 [Accessed: 25 April 2024].

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press. DOI: 10.1017/CBO9780511809071. [Accessed 1 May 2024].

Nabiha, A., Mutalib, S. and Malik, A.M. (2021) ‘Sentiment analysis for informal Malay text in Social Commerce’, 2021 2nd International Conference on Artificial Intelligence and Data Sciences (AiDAS) [Preprint]. doi:Available at :https://ieeexplore.ieee.org/abstract/document/9574436 [Accessed on April 23rd 2024].

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. Frontiers in Neurorobotics, 7, 21. https://doi.org/10.3389/fnbot.2013.00021 [Accessed 25 April 2024].

Park, S., Lee, J. and Kim, K. (2019) ‘Semi-supervised distributed representations of documents for sentiment analysis’, Neural Networks, 119, pp. 139–150. doi:Available at :https://www.sciencedirect.com/science/article/pii/S 0893608019302187 [Accessed on April 23rd 2024].

Powers, D. M. (2011). Evaluation: from precision, recall and F1-measure to ROC, informedness, markedness & correlation. Journal of Machine Learning Technologies, 2(1), 37-63. [Accessed 1 May 2024].

Rajarajeshwari, K. and G. Radhamani, Dr. (2019) ‘A naïve Bayes model using semi-supervised parameters for enhancing the performance of text analytics’, International Journal of Advanced Networking Applications, 10(06), pp. 4083–4089. doi:Available at : https://www.proquest.com/docview/2265611759?pq-origsite=gscholar&fromopenview=true&sourcetype=Scholarly%20Journals [Accessed on April 24th 2024].

Rajaraman, A., & Ullman, J. D. (2011). Mining of massive datasets. Cambridge University Press. DOI: 10.1017/CBO9780511763113. [Accessed 1 May 2024].

Rokach, L. & Maimon, O. (2005). Decision Trees. In Data Mining and Knowledge Discovery Handbook (pp. 165-192). Springer. DOI: 10.1007/0-387-25465-X_9. [Accessed 29 April 2024].

Singh, R., Bagla, R. and Kaur, H. (2015) ‘Text analytics of web posts’ comments using sentiment analysis’, 2015 International Conference and Workshop on Computing and Communication (IEMCON) [Preprint]. doi:Available at : https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7344534 [Accessed on April 23rd 2024].

Schröer, C., Kruse, F., and Marx Gómez, J., 2021. A systematic literature review on applying CRISP-DM process model. Procedia Computer Science, 181, pp. 526-534. DOI: 10.1016/j.procs.2021.01.199. Accessed 28 April 2024.

Shearer, C. (2000). The CRISP-DM model: The new blueprint for data mining. Journal of Data Warehousing, 5(4), 13-22. [Accessed 1 May 2024].

Tufte, E. R. (2001). The Visual Display of Quantitative Information. Graphics press.

Zhang, W. (2007). Using K-nearest neighbor classification to improve classification effectiveness for text categorization. Journal of Digital Information Management, 5(1). [Accessed 27 April 2024].

Zhu, X., & Goldberg, A. B. (2009). Introduction to Semi-Supervised Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 3(1), 1-130. DOI: 10.2200/S00196ED1V01Y200906AIM006. [Accessed 1 May 2024].