TweetPulse: A Power BI-Driven Interactive Sentiment Analysis Dashboard for Brand Monitoring

The Project Phase 2 report submitted in partial fulfillment

of the requirements for the award of the Degree of

Bachelor of Technology

in

Computer Science and Engineering

under the

APJ Abdul Kalam Technological University

by

Ajith M Joshy

MGP21UCS017

Anitta Mary Jose

MGP21UCS033

Jerry Narakathara Thomas

MGP21UCS071

SAINTGITS COLLEGE OF ENGINEERING(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KOTTAYAM, KERALA (INDIA)

May 2025

SAINTGITS COLLEGE OF ENGINEERING(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KOTTAYAM, KERALA (INDIA)

BONAFIDE CERTIFICATE

This is to certify that the project entitled "**TweetPulse: A Power BI-Driven Interactive Sentiment Analysis Dashboard for Brand Monitoring**" submitted by **Ajith M Joshy (MGP21UCS017), Anitta Mary Jose (MGP21UCS033), Jerry Narakathara Thomas (MGP21UCS071)** for the award of the **Bachelor of Technology** in **Computer Science and Engineering** is a bonafide record of the Project Phase 2 carried out by them under my guidance and supervision at "**Saintgits College of Engineering (Autonomous)**".

Er. Talit Sara George	**Dr. Jubilant J Kizhakkethottam**	**Dr. Arun Madhu**
Assistant Professor	Professor	Head of Department
(Project Advisor)	(Project Coordinator)	

Er. Sanoj C Chacko

Assistant Professor

(Project Coordinator)

SAINTGITS COLLEGE OF ENGINEERING(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KOTTAYAM, KERALA (INDIA)

DECLARATION

We, **Ajith M Joshy(MGP21UCS017)**, **Anitta Mary Jose(MGP21UCS033)**, **Jerry Narakathara Thomas(MGP21UCS071)** hereby declare that this thesis entitled "**Tweet-Pulse: A Power BI-Driven Interactive Sentiment Analysis Dashboard for Brand Monitoring**" is the record of the original work done by us under the guidance of **Er.Talit Sara George**, Assistant Professor, Department of Computer Science and Engineering, Saintgits College of Engineering. To the best of our knowledge, this work has not formed the basis for the award of any degree to any candidate in any University.

Place: Pathamuttom
Date:

Er. Talit Sara George
Assistant Professor
(Project Advisor)

Dr. Jubilant J. Kizhakkethotam
Professor
(Project Coordinator)

Er. Sanoj C. Chacko
Assistant Professor
(Project Coordinator)

Contents

Acknowledgements

We express our gratitude to our principal, **Dr. Sudha T.** , Professor and Principal, Saintgits College of Engineering (Autonomous) for providing us with excellent ambiance that laid potentially strong foundation for our work.

We express our heartfelt thanks to **Dr. Arun Madhu**, Associate Professor and Head of the Department of Computer Science and Engineering, Saintgits College of Engineering (Autonomous) who has been a constant support in every step of our project and the source of strength in completing this project.

We express our sincere thanks to **Dr. Jubilant J Kizhakkethottam and Er. Sanoj C Chacko** (Project Coordinators) providing us all the facilities, valuable and timely suggestions and constant supervision for the successful completion of our project.

We also thank **Er. Talit Sara George**, Assistant Professor, Computer Science and Engineering Department for her supervison, encouragement, help and support. We are highly indebted to all the faculties of the department for their valuable guidance and instant help and for being with us. We extend our heartfelt thanks to our parents, friends and well wishers for their support and timely help.

The authors would like to acknowledge the assistance of large language models in revising this report. However all original content and editing were conducted by the authors during the report preparation.

Last but not the least we thank Almighty God for helping us in successfully completing this project.

Ajith M Joshy
Anitta Mary Jose
Jerry Narakathara Thomas

List of Figures

List of Tables

Abstract

Today we live in a world where everything has become data and is keeps on growing and at a fast rate and it has become necessary in various industries to collect data and is use it in various applications. Brands when they release a product or before releasing the product want to know public opinions on how satisfied are the customers with their brand or their products. Sentiment analysis has become one of the most important and crucial tool and it analyse data from surces where public data is available for understanding public sentiments and it has been the fast growing area where companies and industries use it for brand monitoring. Social media platforms has become a rich source of data where companies use it for doing sentiment analysis. This paper reviews recent adavancements in sentiment analysis techniques, approaches and comparing different models used. It also review a proposed real-time system that collects recent data and classifies sentiments and then view various trends and insights in an interactive dashboard, which can help companies to monitor their brand and do various analytics and take timely decisions based on the insights.

Chapter 1

Introduction

1.1 Objective

The objective of this project is to develop a real-time system that analyzes Twitter data using the hashtag #Apple to assess public sentiment toward the Apple brand. This system aims to provide valuable insights into sentiment trends, helping Apple with reputation management and the strategic alignment of marketing campaigns through continuous monitoring and analysis of social media sentiment.

1.2 Problem Statement and Motivation

Understanding public sentiment in real time poses a significant challenge for businesses, organizations, and policymakers. With the massive volume of social media data generated daily, extracting actionable insights quickly enough to inform decisions can be overwhelming. Traditional sentiment analysis methods often struggle with the speed and scalability required for real-time applications, resulting in delayed or less accurate insights.

The motivation behind the "Real-Time Sentiment Analysis on Social Media" project stems from the increasing need to understand public sentiment in today's fast-paced, digital world. Social media has become a primary platform where individuals express opinions, share experiences, and discuss trends, making it an invaluable source of data. By analyzing this data in real time, businesses, policymakers, and organizations can make better-informed decisions and respond quickly to emerging trends.

1.3 Proposed System

We propose to build a system that overcomes the limitations of traditional sentiment analysis approaches by implementing a real-time sentiment analysis pipeline specifically designed for continuous monitoring of Twitter data related to the Apple brand. This system streams live tweets using the hashtag #Apple, enabling immediate analysis and insights from current public sentiment. After data collection, the tweets undergo comprehensive preprocessing (e.g., tokenization, normalization) and feature extraction stages using techniques such as TF-IDF. These processed inputs are then fed into multiple machine learning models (e.g., Logistic Regression, XGBoost), which are evaluated using standard performance metrics. The most accurate model is selected to classify sentiments into positive, negative, or neutral categories, ensuring a reliable analysis pipeline..

Additionally, the proposed system incorporates an advanced visualization component, featuring a real-time, interactive dashboard that presents sentiment trends, insights, and key metrics. This dashboard provides stakeholders, such as Apple's brand managers and analysts, with real-time access to public sentiment and trend detection, enabling the early identification of emerging patterns. This functionality enhances data-driven decision-making, making the proposed system a responsive and powerful tool for social media sentiment analysis.

1.3.1 Features

- **Real-Time Analysis**: This feature enables the platform to perform sentiment analysis instantly as new data streams in, providing insights on customer sentiment, brand perception, and emerging issues without delay. With real-time analysis, businesses can stay informed of sudden shifts in public opinion or feedback trends, allowing them to respond promptly to crises, capitalize on positive developments, or adjust strategies proactively to maintain a favorable brand image.

- **Continuous Trend Detection**: This feature continuously monitors the flow of data, identifying trends, recurring themes, and sentiment shifts as they evolve. By keeping a close watch on both established and emerging topics, the platform can alert teams to changes in customer sentiment early on, enabling swift action before trends fully materialize. This continuous tracking allows for proactive rather than reactive management, helping organizations stay ahead in fast-paced markets.

- **Dynamic Visualization**: Equipped with an interactive dashboard, the platform transforms raw data into visual insights that are easy to interpret and act upon. Users can explore trends, compare sentiment across time periods, and adjust views to focus on specific areas or demographics. This dynamic visualization makes it simpler for stakeholders to understand complex data at a glance, helping them to quickly identify key takeaways and make data-driven decisions.

- **Enhanced Decision-Making Support**: By providing up-to-date insights, the platform becomes a critical tool for decision-making in brand management, marketing, and public relations. With timely and accurate sentiment analysis, leaders gain a deeper understanding of customer needs and market dynamics, helping them develop strategies that are well-informed and responsive. The platform ultimately supports more agile, informed decision-making, which can improve customer satisfaction, protect brand reputation, and optimize marketing effectiveness.

- **Time-based sentiment analysis**: This feature analyzes sentiment across specific time intervals (e.g., hourly, daily, weekly) to track fluctuations in public opinion over time. By identifying sentiment trends around key events or product releases, it helps Apple's brand managers understand how public sentiment evolves and respond proactively. This time-based approach allows for more targeted strategies and quicker adjustments based on real-time public reactions.

Chapter 2

Literature Review

The sentiment analysis can help in product improvement, marketing strategy development and brand image management. It helps brands like Huawei monitor and analyze customer sentiment on social media platforms in an automated way [1]. Support Vector Machine is used in classifying sentiment of user comments related to 'Huawei Mate60' as the results indicate SVM is more effective in obtaining an accurate sentiment classification compared to other four models, Logistic Regression, K-Nearest Neighbors and Naive Bayes and XGBoost.

Brand reputation management and marketing strategy has become more and more important in the digital age [2]. The brands can comprehend and respond to changing public sentiment by using information retrieval techniques, such as calculating, textual similarity and using machine learning methods to analyze product similarity for competitive analysis. The objective is not only to monitor public sentiment towards a brand, but also to provide valuable insights by comparing it with similar brands in the market. Through user-friendly visualizations and tools, the brands stay abreast of public opinion and make informed decisions amid the dynamic landscape of consumer sentiment and competition, providing practical solution for brand reputation management and marketing strategy.

In social media platforms there are many hate speeches posted by the users. This hate speech can be detected using the system [3] that streams real-time Twitter data of a trending topic and classifies it into two classes: hate speech and non-hate speech. The model used in the system is based on Long Short Term Memory (LSTM) model, using term frequency inverse document frequency (TF-IDF) vectorization, and compares its performance with other machine learning and deep learning models such as Support Vector Machine (SVM), Naive Bayes (NB), Logistic Regression (LR), XGBoost (XGB), Random Forest (RF), K-Nearest Neighbor (kNN), Artificial Neural Network (ANN), and Bidirectional Encoder Representations from Transformers (BERT). The LSTM model achieved an accuracy of 97% in detecting hate speech, outperforming the other models.

During COVID-19 pandemic, a total of 999,978 data related to Weibo posts from January

1 to February 18, 2020 was collected to understand public sentiment and concerns in China during eary stages of the pandemic [4]. A fine-tuned Bert model was used to classify the sentiment of the posts as positive, negative and neutral, achieving an accuracy of 75.65%. Then topic modelling was applied using TF-IDF to extract key themes discussed in negative sentiment posts. The analysis revealed that people were concerned about the origin of virus, symptoms, impact on work and school, and the public health control measures. These findings could help governments make effective public health decisions during a crisis.

The field of sentiment analysis can be improved significantly by introducing AI-powered approach to this field. It showed that this approach [5] navigates the complexities of sentiment identification and classification in the digital age and also improves decision-making and quality assurance. It also integrates traditional statistical methodologies, such as machine learning models, with advanced deep learning architectures to capture the intricacies of sentiment in a dynamic, multilingual, and culturally diverse digital environment.

There are many techniques to extract topics from the dataset. In this study, [6], MABED and OLDA are the two techniques used for this purpose along with LR and SVM models to address if the topic has positive or negative impact on the event. The overall best score for sentiment analysis of events is obtained when combined MABED with LR.

Naive Bayes algorithm is used in the proposed system, which has got an accuracy of 83%. This system [7] allows users to input keyword, and the system extracts data related to the keyword from Twitter. These data are then classified as positive and negative, and the results are displayed in a pie chart for the users.

E-Commerce has become an important source for user data and an application for sentiment analysis in which users post their opinions about the products [8]. A web based E-commerce application was developed that collects data from Amazon.com and Support Vector Machine(SVM) model was chosen for sentiment classification. The model was trained using NLTK corpus library that contain sentences having negative and positive polarity scores. Flask RESTful API, a web framework is used for making the application RESTful. The application allows the users to submit their review about the product and the submitted reviews are summarized which is then classified as positive or negative polarity in the form of 1 and 0 respectively. Using sentiment analysis in online shopping websites can help customers make educated decision about the product.

Paper Reference	Research Focus	Dataset\Source	Methodologies	Results/Efficiencies	Limitations
Ingole et al., 2024	Brand sentiment analysis for competitive surveillance	Social Media	Sentiment analysis model (specific techniques not detailed)	Increased monitoring accuracy for brand sentiment	Limited to competitive brand contents only
Nguyen, 2024	Enhancing social media sentiment analysis with DL	Social media	Advanced deep learning techniques	Higher accuracy than traditional methods	High computational cost due to deep learning
Xiaohong He, 2024	Sentiment classification of social media comments	Social media	SVM models	87% accuracy in sentiment classification	limited scalability with large datasets.
Rani et al., 2024	Survey of sentiment analysis tools and techniques	Various social media datasets	Comparative analysis of multiple techniques	Highlights pros and cons of each technique	Does not offer solutions for integration issues
Omar, 2022	Opinion mining of telecom services during COVID-19	Telecom feedback	Discourse-based opinion mining	Insights on customer experience during the crisis	Context-specific model limits generalizability
Suhaiman et al., 2023	Sentiment analysis in public security	Social media	Taxonomy, trend analysis	Identified trends and public concerns in security	Limited applicability beyond public security
Kausar et al., 2021	Twitter sentiment analysis during covid-19	Twitter	Public sentiment analysis	Revealed shifts in public opinion during the outbreak	Focus on covid-19 limits general applicability
Zineb et al., 2021	Data analysis e-commerce sentiment	E-commerce reviews	Intelligent approach in big data	Improved decision-making through insights	Limited to e-commerce domain
Anupama B S, 2020	Real-time Twitter sentiment analysis	Twitter	NLP techniques	Effective real-time monitoring	Limited by API rate restrictions
Radaideh & Dweiri, 2023	Sentiment prediction in digital media	Digital media content	NLP and machine learning	High prediction accuracy in media sentiment	Requires domain specific adjustments
D. P. & Ahmed, 2016	Survey on big data analytics	Big data (general)	Comparative study of tools and challenges	Overview of big data challenges and tools	Outdated with new tools and technologies
Mahajan & Mansotra, 2021	Crime correlation with social media sentiment	Social media (crime-related)	Semantic sentiment analysis	Detects patterns linking sentiment to crime	Limited to crime-related data
Bangera & K. N., 2021	Progressive sampling model for big data analytics	Big data (general)	Fortune-sensitive sampling model	Enhanced sampling efficiency	High setup complexity
Wankhade et al., 2022	Survey on sentiment analysis methods and challenges	Various social media sources	Comparative survey	Highlights applications and challenges	Limited practical implementation details
Alattar & Shaalan, 2021	Opinion reason mining and sentiment variation	Social media	Opinion reason mining	Effective in interpreting sentiment shifts	Limited support for real-time processing
Motz et al., 2022	Live sentiment analysis with machine learning	Social media (various)	Multiple ML and text processing algorithms	Real-time sentiment tracking	Processing lag in high-volume scenarios
J. Park, 2020	Customer satisfaction evaluation for cosmetics brands	Twitter	Sentiment driven framework	Enhanced customer satisfaction insights	Focused on cosmetics; may not generalize
K. Park et al., 2024	Fine-grained product review analysis for product design	Product reviews	Contextual meaning-based approach	Improved detail in product review sentiment	Context-specific; lacks general sentiment analysis
Shayaa et al., 2018	Survey on sentiment analysis in big data	Big data	Comparative analysis	Identified open challenges in big data sentiment	Limited to foundational challenges
Roy et al., 2024	Real-time hateful sentiment detection in tweets	Twitter	LSTM-based approach	High accuracy in real-time hate speech detection	High computational cost due to LSTM

Chapter 3

System Design

The architecture of TweetPulse: A Power BI-Driven Interactive Sentiment Analysis Dashboard for Brand Monitoring is shown in Fig 3.1 is structured into three phases, Back-End processing, API integration and Front-End Visualization.

The first phase shows the back-end of the system and it consists of various modules. The first module is data collection, which collects data using Twikit scraper that interacts with Twitter API without authentication keys. It fetches tweets with hashtag #Apple in order to collect. The second module is data preprocessing, used for cleaning, transforming and transforming raw data before feeding into machine learning models, since raw data is often noisy, inconsistent and incomplete which makes the models performance and accuracy low. The third module is feature engineering, which creates new features from raw data to improve model performance and these features are in numerical form, which can be understood by machine learning models. The fourth module is sentiment prediction used for predicting data collected using model. This model represents the best model chosen from four machine learning models.

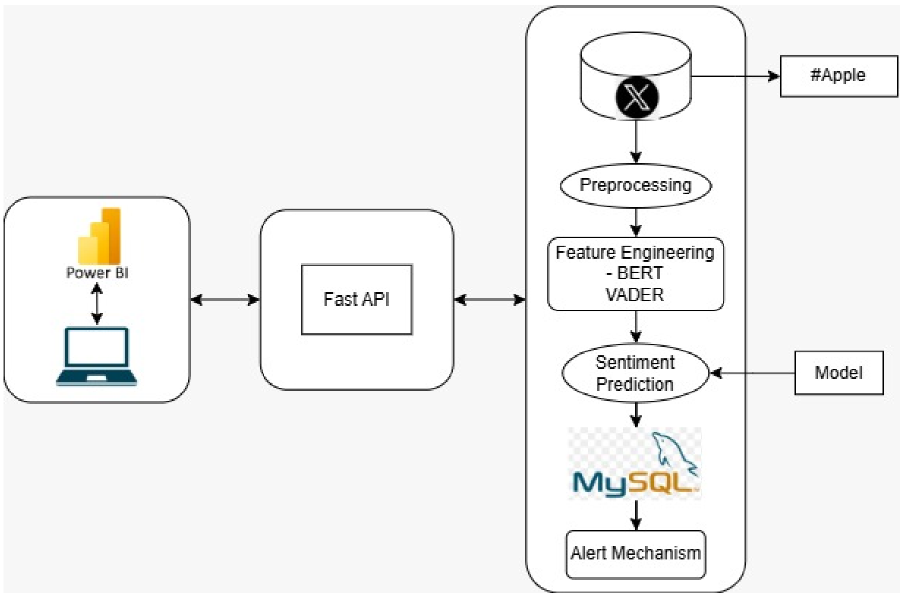
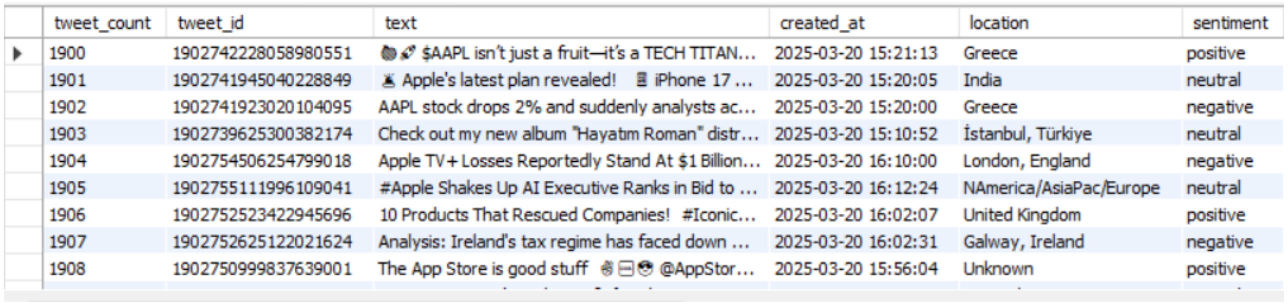


Figure 3.1: System Architecture

The fifth module is Mysql database as shown in Fig 3.2. It is a relational database management system used for storing and managing structured data efficiently. It stores live tweets and sentiment results efficiently and provides fast retrieval of data. The Power BI dashboard allows the database to be connected to it, which allows to use the data for various analytics and visualizations and it can be easily integrated with python.



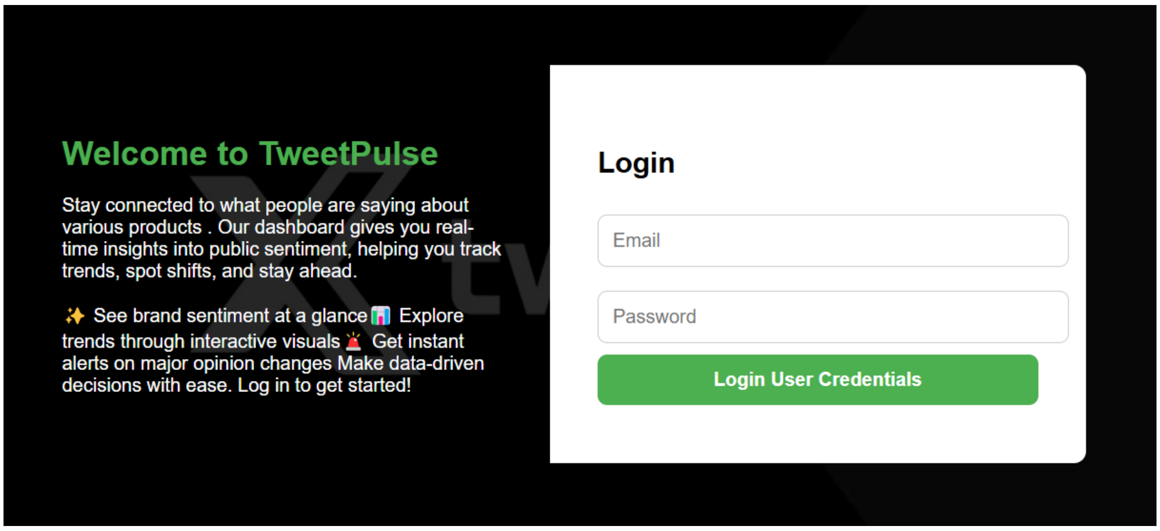
	tweet_count	tweet_id	text	created_at	location	sentiment
▶	1900	1902742228058980551	🍏🚀 $AAPL isn't just a fruit—it's a TECH TITAN...	2025-03-20 15:21:13	Greece	positive
	1901	1902741945040228849	🚨 Apple's latest plan revealed! 📱 iPhone 17 ...	2025-03-20 15:20:05	India	neutral
	1902	1902741923020104095	AAPL stock drops 2% and suddenly analysts ac...	2025-03-20 15:20:00	Greece	negative
	1903	1902739625300382174	Check out my new album "Hayatım Roman" distr...	2025-03-20 15:10:52	İstanbul, Türkiye	neutral
	1904	1902754506254799018	Apple TV+ Losses Reportedly Stand At $1 Billion...	2025-03-20 16:10:00	London, England	negative
	1905	1902755111996109041	#Apple Shakes Up AI Executive Ranks in Bid to ...	2025-03-20 16:12:24	NAmerica/AsiaPac/Europe	neutral
	1906	1902752523422945696	10 Products That Rescued Companies! #Iconic...	2025-03-20 16:02:07	United Kingdom	positive
	1907	1902752625122021624	Analysis: Ireland's tax regime has faced down ...	2025-03-20 16:02:31	Galway, Ireland	negative
	1908	1902750999837639001	The App Store is good stuff 👍📱😎 @AppStor...	2025-03-20 15:56:04	Unknown	positive

Figure 3.2: Mysql Database

The sixth module is an alert mechanism. This mechanism helps in detecting and alerting trends like positive or negative by providing an alert notification to the email of the specified user when the trend crosses a certain threshold. This is done by calculating the data required belonging to specific time period from the database. It uses gmail smtp to send emails and SQL Alchemy to connect to database. This allows the users to take timely decisions based on the ongoing trend.

The second phase shows the integration of front-end and back-end using Fast API. It is a modern, high-performance web framework for building APIs with python, known for its speed, automatic documentation, and ease of use. It helps in handling requests between front-end and back-end for data retrieval and processing, also it supports asynchronous operations. When a user interacts with Power BI dashboard via a button click, it sends the request to Fast API, which triggers the back-end processing. Power BI provides a button feature which holds this Fast API url.

The third phase shows the front-end of the system. The front-end consists of an authentication page as shown in Fig 3.3 to allow only the eligible users to access the dashboard. Here, only login option is available and no new users can register. It is because the system is designed for specific users or stakeholders of Apple brand, as our system is used for brand monnitoring, the case study used here is Apple brand. After the authentication page is successful, it directly opens the Power BI dashboard in Microsoft Fabric, which is a cloud-hosted version of Power BI instead of Power BI desktop as shown in Fig 3.4. The Power BI desktop is a standalone application and does not support web-based embedding or remote access links. This dashboard displays various visuals, sentiment trends and supports various analytics to be performed to get more insights from these collected data.


Welcome to TweetPulse

Stay connected to what people are saying about various products . Our dashboard gives you real-time insights into public sentiment, helping you track trends, spot shifts, and stay ahead.

✨ See brand sentiment at a glance 📊 Explore trends through interactive visuals 🚨 Get instant alerts on major opinion changes Make data-driven decisions with ease. Log in to get started!

Login

Email

Password

Login User Credentials

Figure 3.3: User Interface

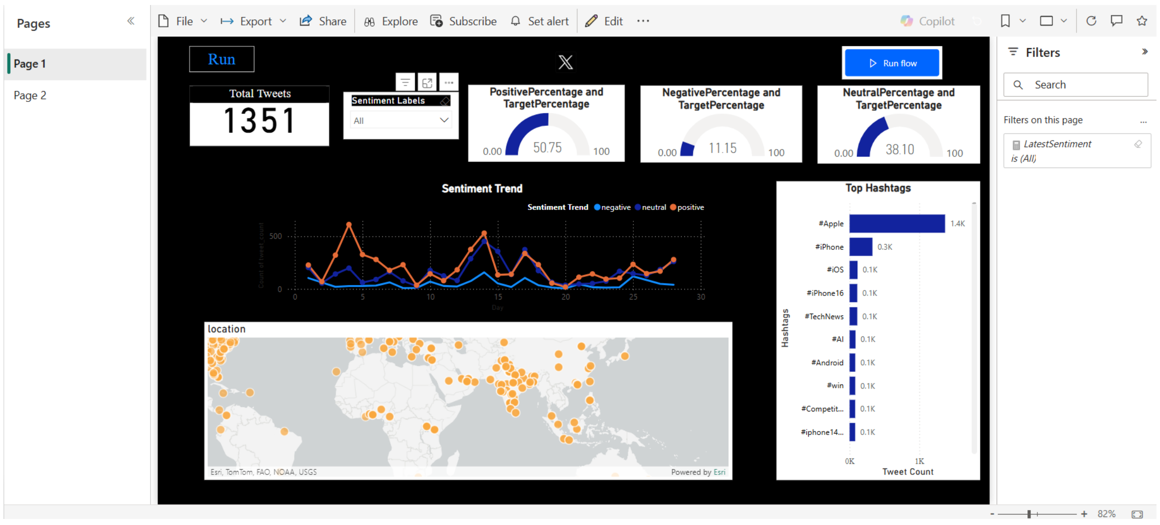


Figure 3.4: Brand Monitoring Dashboard

Chapter 4

System Implementation

4.1 Techonology Stack

The technology stack used in this project integrates several tools across each phase, leveraging Python as the primary language. Data collection is handled using the Twikit library in VS Code, enabling Twitter scraping. All subsequent analysis, including exploratory data analysis, preprocessing, feature engineering, and model training, is conducted in Google Colab. Key libraries for preprocessing include NLTK for tokenization and stopword removal, Emoji for emoji filtering, and re for pattern removal. Feature engineering employs BERT embeddings from Hugging Face's transformers library and Vader from NLTK library. For model training, Scikitlearn's Logistic Regression, SVM, Random Forest, and XGBoost classifiers are used. Additionally, the VADER sentiment analysis tool provides an initial sentiment label to data before model training. This stack enables efficient, real-time sentiment analysis with comprehensive NLP techniques and machine learning models.

4.2 Implementation

- **Data Collection**: Fetches recent tweets using hashtag #Apple via Twikit, a python library for interacting with Twitter API. The scraper is configured to authenticate through the Twitter account and saved the cookies once, and then load the saved cookies on subsequent data collection for preventing account ban. Along with tweet, other attributes of each tweet are also fetched such as created_at(timestamp of tweet), location(location of tweet), tweet_id(tweet id), hashtags and keywords. It also filters the tweets by collecting only tweets in english language and discards duplicate tweets, retweets and replies.

  In intial implementation, during a span of 3 months, a total of 4125 tweets was collected. This data for used for training and testing ML models. After the implementation of the back-end and integration of UI and back-end, tweets were collected again for testing the back-end and the system as a whole. The sentiment of tweets were predicted using the best performed model and then visualizing those predicted tweets in the dashboard using different visuals, Power Bi application provides. During the initial implementation, csv files were used for storing tweets.

- **Data Preprocessing**: Before feeding the tweets into machine learning models, the raw data must be cleaned and normalzed. Social media text is often noisy, containing urls,hashtags, emojis, special characters and inconsistent formats. To ensure accurate sentiment analysis, we apply a series of text preprocessing steps to convert raw tweets into a clean and structured format. The preprocessing module consists of the following steps:
  - Remove urls, digits, special characters(only hashtags are retained as they are extracted for analysis).
  - Tokenization: Splits cleaned text into individual words using NLTK.
  - Stopwords removal: Removes commonly used words that do not contribute meaningfully to sentiment.
  - Lemmatization: Converts words to their base forms (eg, "running" is converted to "run") using WordNetLemmatizer.

- **Feature Engineering** Feature Engineering is a crucial step in natural language processing that transforms raw text into numerical representations that machine learning models can process. It helps in catching meaningful patterns from tweets. Two techniques applied here are, Vader and Bert.

  The first technique is VADER. It is a lexicon based tool optimized for social media texts. It assigns sentiment scores based on predefined word polarity values and context modifiers. First, it initializes the VADER sentiment analyzer. The get_vader_sentiment function applies vader to each text, classifying sentiment as 'positive', 'negative', or 'neutral' based on the compound score threshold. This is then applied to the processed_text, creating sentiment labels against each processed text. For model compatibility, the sentiment labels are encoded into numeric values in the sentiment_encoded column.

  The second is BERT embeddings. It provides context-aware representation of text. Instead of using predefined scores, it learns from data to understand word meanings based on their context in a sentence. It initializes a BERT tokenizer and model (bertbase-uncased) to process batches of text data, extracting BERT-based feature vectors. The extract_bert_features_batch function tokenizes and feeds batches of text into BERT, capturing the CLS token embeddings, which represent each text's semantic features. These features are then stored in the DataFrame and standardized using StandardScaler for consistency in model training. The standardized BERT features are saved to a file, allowing for easy reuse.

- **Class Distribution** Checks the distribution of sentiment classes in the dataset to understand class balance and then visualizes the distribution using a Seaborn count plot, displaying the frequency of each sentiment class. There was class imbalance in the dataset where majority class was positive and minority class was negative. This does not improve our models, so resampling technique is applied for balancing the dataset. This step is only used during training of machine learning models and is not used in the main pipeline.

- **Splitting of dataset into Training and Testing** For any ML models two inputs are required for training. Here supervised ML models are used, so a label is required in order for models to learn. So the first input, that is X input, is the feature set by combining features from vader and bert. Then the second input is the labels, that is, y. First, it concatenates these feature sets from Vader and Bert embeddings into a single matrix feature set, X, and scales them for consistency. y is the target variable(label) for training machine learning models. Both Vader and Bert embeddings are in different ranges, so StandardScaler is used in order to standardize them. This ensures all features are on the same scale preventing any single feature from dominating the learning process due to its larger magnitude. Then both inputs are splitted as train and test data. Here test_size=0.2 mean 80% of the dataset is used for training and 20% is used for testing.

- **Resampling the dataset** This addresses class imbalance in the training data by using oversampling and undersampling techniques. First, it uses SMOTE (Synthetic Minority Over-sampling Technique) to increase the minority class samples. Then, it applies RandomUnderSampler to reduce the majority class samples, ensuring all classes are balanced, having same number of samples.

- **Training Machine Learning models** Trained four ML models, Logistic Regression, Support Vector Machine, Random Forest and XGBoost on the resampled training data. After training, the model's performance is evaluated using performance metrics (accuracy, precision, recall, and F1score for each sentiment class) and a confusion matrix to assess its performance across the sentiment categories (negative, neutral, and positive).

- **Data Storage**: A Mysql database is used for storing data. The data is stored in structured format, that is, in a table. The table creation, insertion of data and retrieval of data is done using SQL queries. The collected live tweets, predicted sentiment results and other attributes of tweets such as created_at(time tweet was created), location(location of tweet, text(tweet), tweet_id(tweet id), tweet_count(Count of tweets) and extracted keywords and extracted hashtags hashtags from the tweets are stored in the table of MySQL database.

- **Dashboard**: The Power BI dashboard is a web application already available to download for every computer. It is used as user-interface for the user and its main purpose is to get insights from the data collected by visualizing the data using different visualizations it provides. The visuals can be build on the existing data and also on new measures that can be created using existing data. The new measures are created by performing various calculations. The calculations performed here are:

  - Sentiment percentages: It is calculated by dividing the total number of tweet by total number of negative tweets.
  - Latest Text and its sentiment: The latest text is calculated by finding the max tweet count and taking its corresponding text and sentiment.

- **API Integration**: First, Fast API application is started using uvicorn, the server that runs Fast API app. The Fast API app will be live on local machine via on its server link, http://127.0.0.1:8000 and can recieve HTTP requests from client. When the user submits their login credentials, it checks with the credentials stored in the database. If the credentials matches correctly, it processes the login page and redirects the user to Power BI dashboard which is hosted on Microsoft Fabric, a cloud-hosted version of Power BI via the dashboard link.

  After login, API will continue listening for requests. When the user clicks the Power BI button that contains the API url, it sends a GET request to the the API server. On recieving the request, it provides the response back by giving POST request, which triggers the back-end pipeline. The POST initiates the data collected and processed in the back-end to be transferred to the front-end.

- **Trend Detection**: Different trends can be visualized in dashboard such as sentiment(positive, negative and neutral), hashtags and keywords trends. These trends are detected and notified using an alert mechanism when the particular trend crosses certain threshold. This is done by calculating the data required belonging to specific time period from the database. It uses SQL Alchemy to connect to database and Gmail SMTP to send emails. These thresholds are set in the back-end pipeline and are not able for the user in the user interface.

Chapter 5

Evaluation and Results

This section evaluates the performance of various machine learning models applied to sentiment analysis of Twitter data. The main objectives are to assess model accuracy, interpret performance across different sentiment classes, and understand the impact of various techniques performed on data for classification accuracy.

5.1 Evaluation Metrics

To assess model performance, we employed following metrics:

- Accuracy: Measures the overall percentage of correctly classified instances by comparing the number of correct predictions to the total number of predictions. It provides a general overview on how well the model is performing across all sentiment classes. A high accuracy alone could be misleading and should be used alongside precision, recall and f1-score.

- Precision: Indicates the accuracy of positive predictions for each sentiment class by measuring the proportion of correctly predicted positive instances out of all instances predicted as positive. Higher precision means that the model is making fewer false positive errors.

- Recall: Reflects the model's ability to correctly identify all instances of a given sentiment class. It measures the proportion of true positives retrieved out of all actual positives. A high recall indicates that the model is effective at detecting most of the relevant instances, meaning it rarely misses any sentiment examples that should be identified.

- F1 Score: Represents the harmonic mean of precision and recall. It is especially useful when dealing with imbalanced datasets, when high precision and recall alone could be misleading. A high F1 score indicates that the model has a good balance between accurately predicting sentiments (precision) and capturing all relevant instances (recall).

- Confusion Matrix: Provides a detailed breakdown of true versus predicted classifications for each sentiment, helping us understand common misclassification patterns.

5.2 Model Performance and Comparison

Table 5.1 show that Random Forest and XGBoost performed better among the four models. The model's can be further understood using confusion matrix as shown in Fig 5.1, Fig 5.2, Fig 5.3, Fig 5.4. The models are then further evaluated by applying cross validation technique and found that the results are consistent, so XGBoost is selected as the best performing model.

Table 5.1: Performance Comparison of ML Models

Algorithm	**Accuracy**	**Precision**			**Recall**			**F1-Score**		
		Pos	**Neg**	**Neu**	**Pos**	**Neg**	**Neu**	**Pos**	**Neg**	**Neu**
Logistic Regression	0.93	0.78	0.83	0.80	0.91	0.92	0.91	0.99	0.96	0.98
Support Vector Machine	0.94	0.87	0.93	0.90	0.93	0.93	0.93	0.98	0.96	0.97
Random Forest	0.95	0.91	0.93	0.92	0.96	0.92	0.94	0.96	0.99	0.97
XGBoost	0.99	0.95	0.98	0.96	0.99	0.99	0.99	1.00	1.00	1.00
Weighted Avg	-	0.60	0.57	0.58	0.68	0.65	0.66	0.63	0.60	0.62

5.3 Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's performance by comparing the actual versus predicted labels. It helps identify how well the model is performing for each class, as well as revealing any misclassifications. In a multi-class classification task, the confusion matrix becomes a square grid where rows represent the actual classes and columns represent the predicted classes. Each element of the confusion matrix corresponds to the following:

- True class labels(True Positives and True Negatives): Instances that were correctly predicted that are presented along the diagonal of the matrix.
- False class labels(True Positives and True Negatives): Instances that were incorrectly predicted that are present in off-diagonal elements.

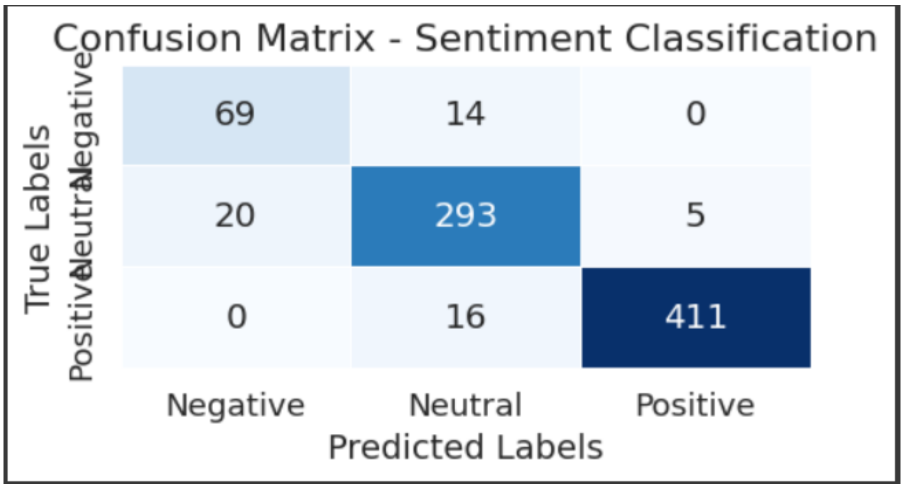


Figure 5.1: Confusion Matrix of Logistic Regression

The Figure 5.1 shows Logistic Regression correctly identified 69 Negative tweets. However, 14 Negative tweets were misclassified as Neutral, which could indicate difficulty in distinguishing between negative and neutral sentiments. 293 Neutral tweets were correctly classified, showing decent accuracy. However, 20 Neutral tweets were misclassified as Negative, and 5 were misclassified as Positive, indicating some confusion in sentiment differentiation. 411 Positive tweets were correctly classified, which is a strong performance. However, 16 Positive tweets were misclassified as Neutral, which is slightly higher than other models like XGBoost.

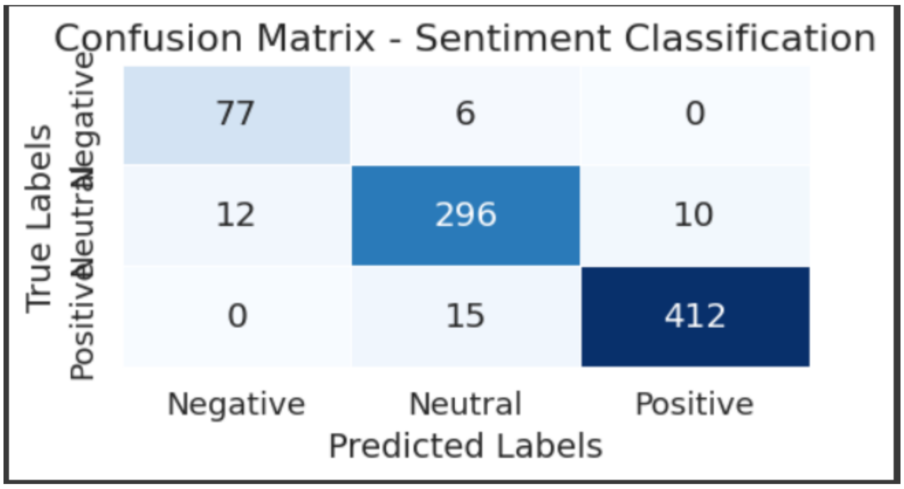


Figure 5.2: Confusion Matrix of Support Vector Machine

The Figure 5.2 shows SVM model performs well in classifying Positive sentiment, with 412 correct classifications and only 15 misclassified as Neutral. However, Neutral sentiment has some misclassifications, where 22 tweets (12+10) were incorrectly predicted. The Negative category has low misclassification, showing high precision for detecting Negative sentiment.

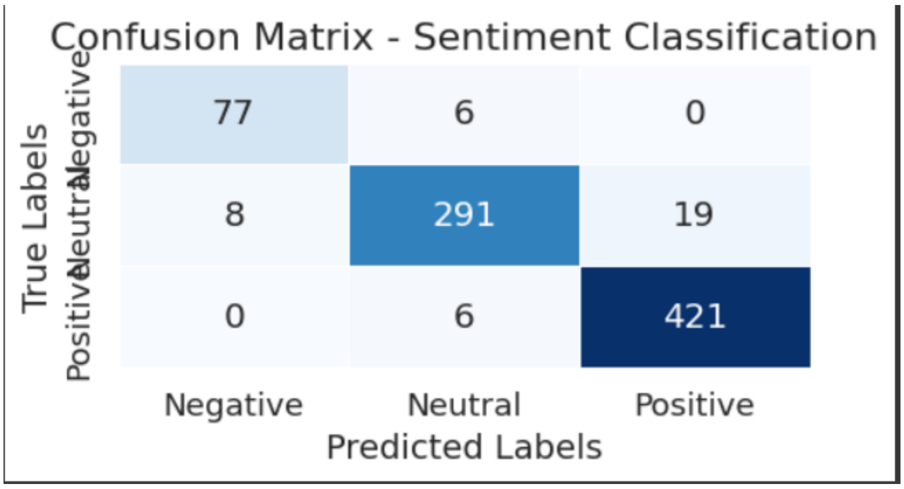


Figure 5.3: Confusion Matrix of Random Forest

The Figure 5.3 shows random forest model performs better in classifying Positive tweets. The model correctly classified 421 Positive tweets, an improvement over SVM. Only 6 Positive tweets were misclassified as Neutral, showing better accuracy for this category. 19 Neutral tweets were misclassified as Positive, which is higher than SVM's 10 misclassifications. The model struggles to distinguish between Neutral and Positive sentiment, leading to more errors. Slight improvement in Negative classification as fewer Neutral tweets were misclassified as Negative (8 vs. 12 in SVM), showing better precision for Negative sentiment detection.

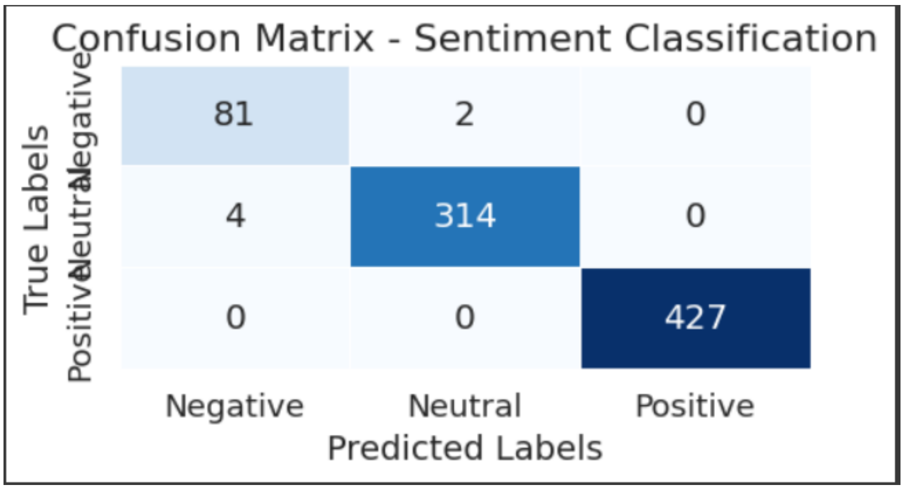


Figure 5.4: Confusion Matrix of XGBoost

The Figure 5.4 shows XGBoost model achieved perfect classification for Positive sentiment (427/427 correct). The Neutral category had the least misclassification errors (only 4 Neutral misclassified as Negative). The Negative category also had the highest accuracy (81 correct, only 2 misclassified as Neutral). Significant improvement over SVM and Random Forest as XGBoost has the lowest misclassification rates among all three models. The ability to separate Neutral from Positive and Negative tweets is much better than both SVM and Random Forest.

Chapter 6

Results and Discussion

In this section, the insights obtained from different visuals used in dashboard, are explained.

The positive, negative and neutral sentiment percentages of the total collected tweet is shown in Fig 6.1. It helps to understand in overall how customers are reacting toward the brand, if are they satisfied or dissatisfied. These percentages are new measures that are calculated using existing data by performing calculations. The visual used is gauge.

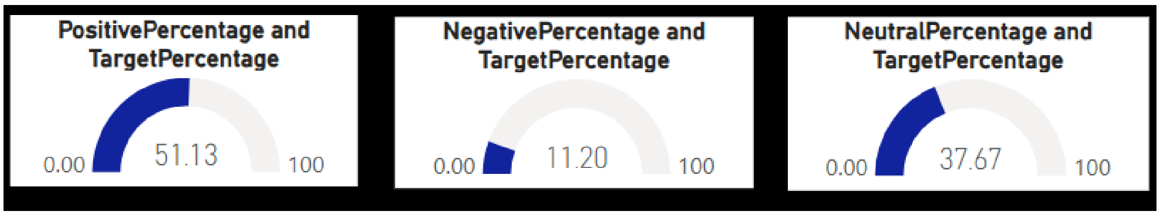


Figure 6.1: Percentage of Positive, negative and neutral sentiments

The trend of different sentiment categories is shown in Fig 6.2. It shows how different sentiments vary throughout the timeline. The x-axis represents each day and y-axis represents the count of collected data. The sentiment variation also depends on how much data is collected each day, as tweets are not collected the same amount everyday. The amount of data collected depends on how much the data is collected during each run and the system is started manually by the user. The visual used here is Line chart.

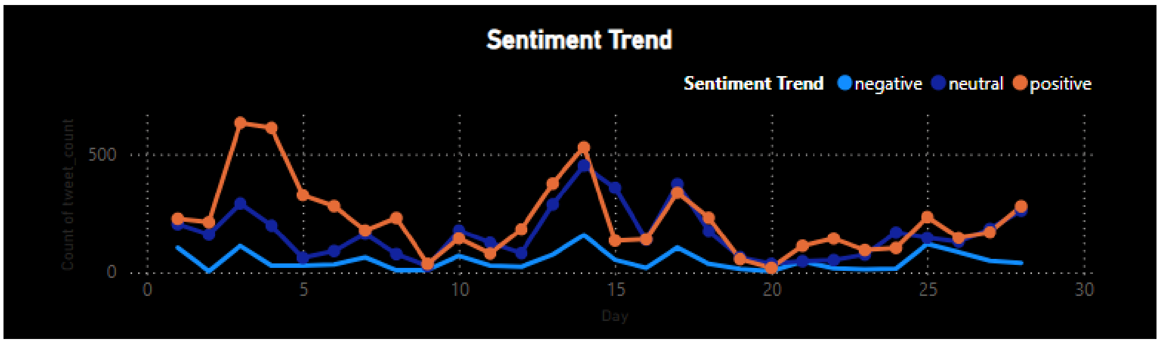


Figure 6.2: Sentiment Trend

The visual Top hashtags shown in Fig 6.3, reveals the hashtags that have been used by users the most. The count of hashtags are presented in the decreasing order from the top. This visual gives an idea to the user on how to reach their brand to maximum people using these hashtags. This is useful during product campaign. The visual used is clustered bar chart.

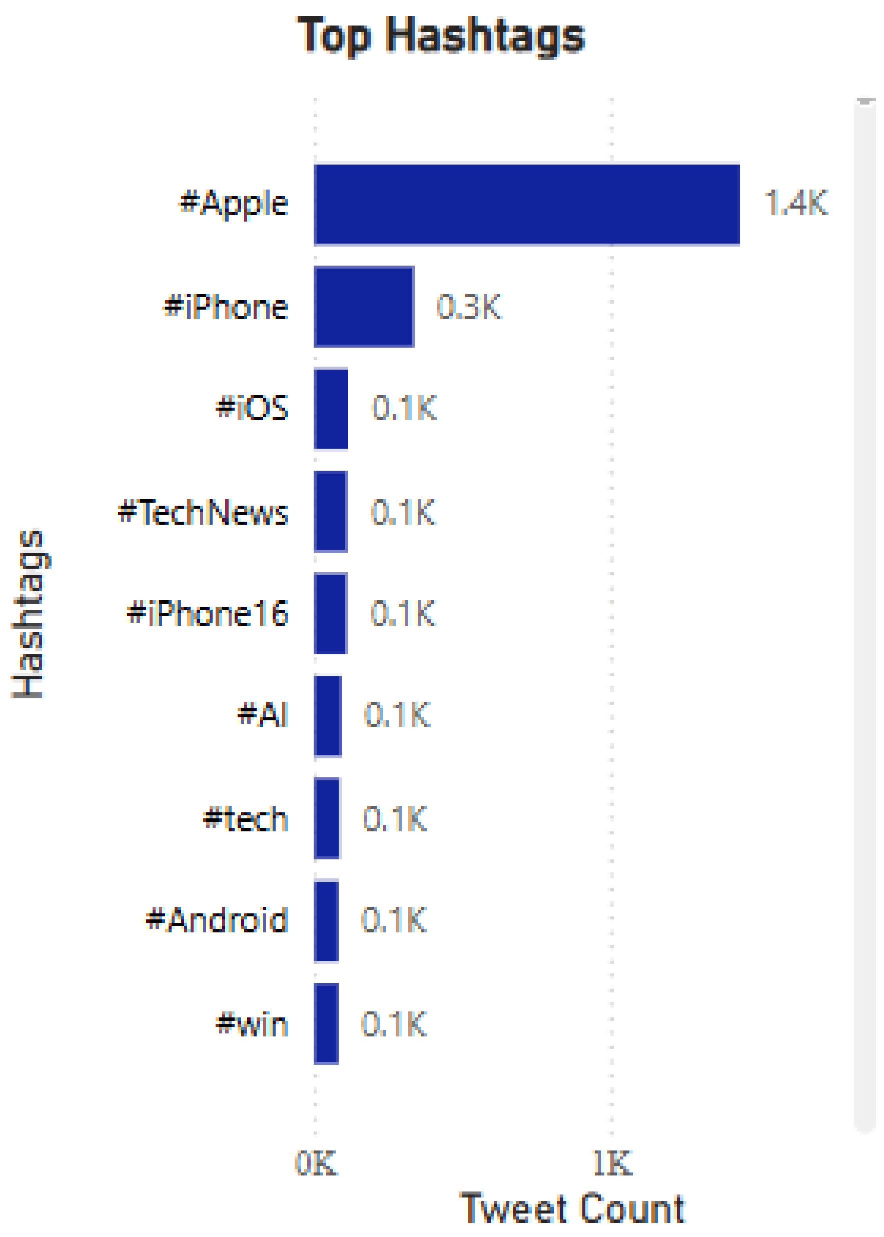


Figure 6.3: Trending Hashtags

The location of the users whose tweets are collected is shown in Fig 6.4. Sometimes, it displays the name of the country or state and does not go deep into the address of the users. This is ArcGIS visual and is provided by the dashboard.

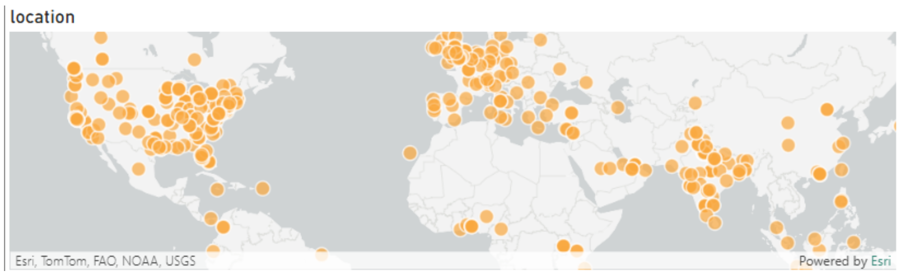


Figure 6.4: Location of Tweets

A visual providing the keywords that are used the most by users is shown in Fig6.5. The keywords that are in large size show the keyword that has been used the most by users and the keywords that are least in size show that they are used the least by users. This visual helps the brands to quickly identify what topics are trending and being repeated in conversations.

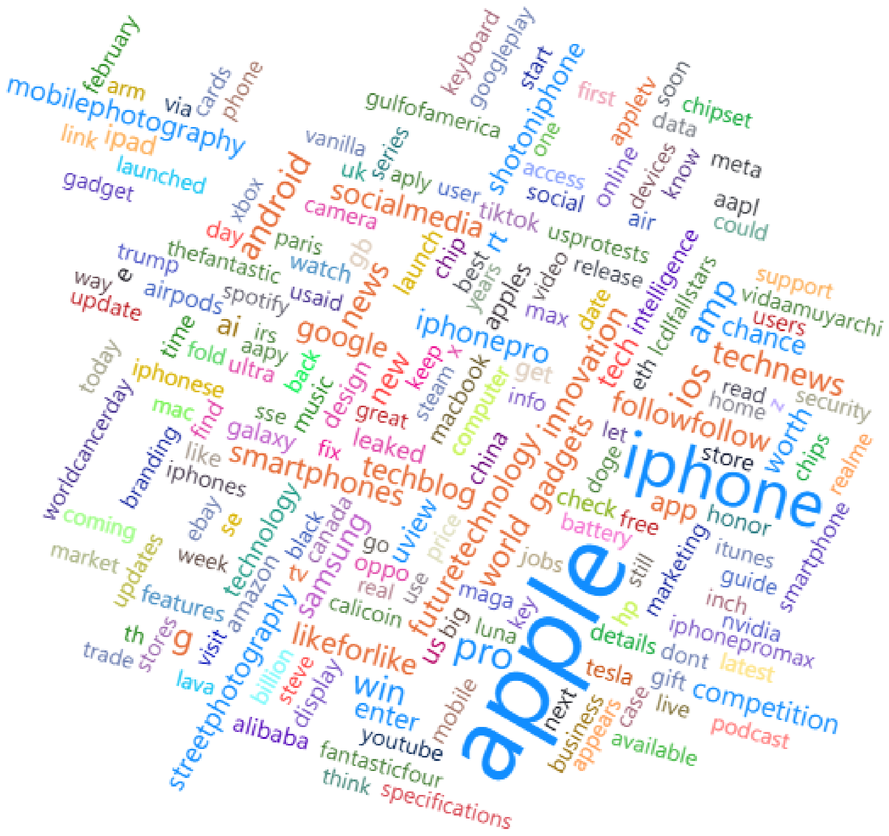
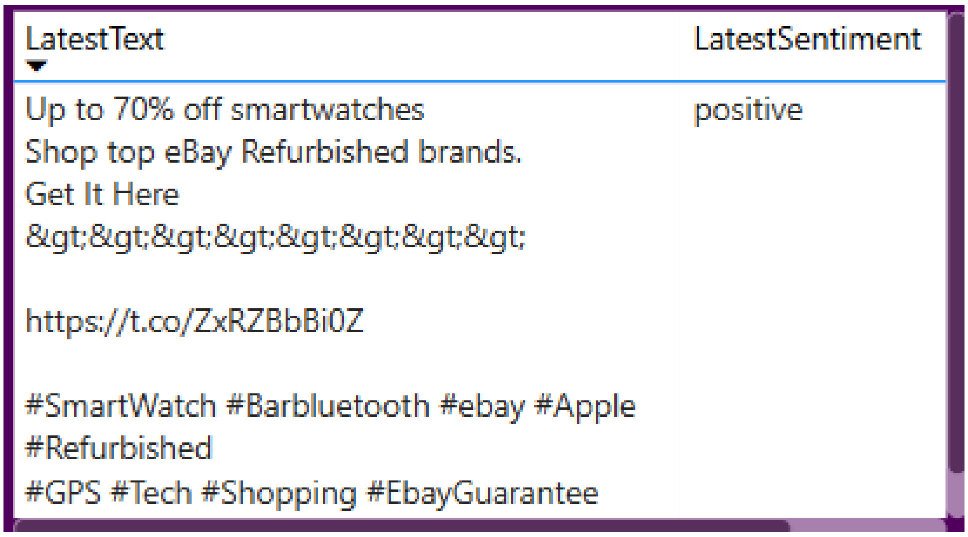


Figure 6.5: Word Cloud

The visual shown in Figure6.6 reveals the last collected real-time tweet and its corresponding sentiment. This helps in understanding the trending conversations among audience, which helps the brands to take decisions based on public discussions.



LatestText	LatestSentiment
Up to 70% off smartwatches Shop top eBay Refurbished brands. Get It Here &gt;&gt;&gt;&gt;&gt;&gt;&gt;&gt; https://t.co/ZxRZBbBi0Z #SmartWatch #Barbluetooth #ebay #Apple #Refurbished #GPS #Tech #Shopping #EbayGuarantee	positive

Figure 6.6: Latest Text and Sentiment

6.1 Limitations

In addition, the threshold for the alert mechanism is also implemented in the back-end and one cannot modify the threshold. Resolving these limitations can significantly improve the system.

- In Data collection, the Twikit scraper itsef has limitations in data collection. In 15 minutes, it can collect upto 50 tweets and the rate limit resets every 15 minutes.

- The system does not run automatically and needs to run manually.

- The system does not allow one to input other hashtags and if needed to collect data related to other hashtags then one needs to modify the python code with which we have implemented data collection.

- In addition, the threshold for the alert mechanism is also implemented in the back-end and one cannot modify the threshold. Resolving these limitations can significantly improve the system.

Chapter 7

Formulation of Project Plan

7.1 Phase 1

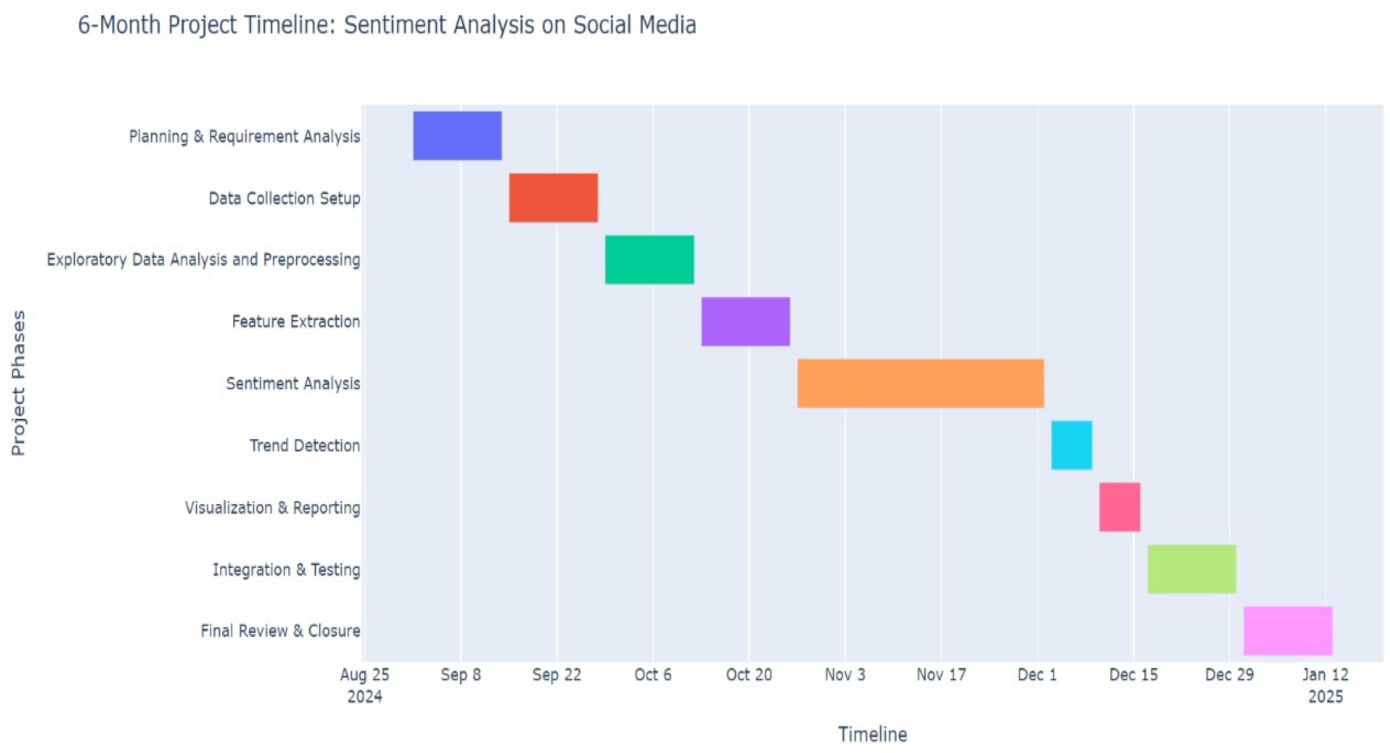


Figure 7.1: Gnatt Chart of Phase 1

Our goal is to complete our project phase-1 within six months from August to January. Therefore, we chose the domain of Social Media Monitoring tool and began our area identification process in August. We started looking into the area in August in preparation for choosing our project topic, and we decided three topics. We finished with the abstract of three topics and presented it. From the three topics, we finalized the topic on “REAL-TIME SENTIMENT ANALYSIS ON SOCIAL MEDIA”.

To do our literature study, we began investigating our topic, current apps, etc. in September 2022. We eventually came to the conclusion that our idea has been implemented but found certain gaps and limitations which we aimed to overcome. So, we designed our Flow Graph for getting ideas about the working of our system. But then we faced a challenge of using Twitter API for collecting data from Twitter because of it's cost so we had to look for other solutions. Then we were able to finish designing our script in order to decide and finalize the solution for collecting tweets. From October 2024 till the present, we have been developing our backend design.

7.2 Phase 2

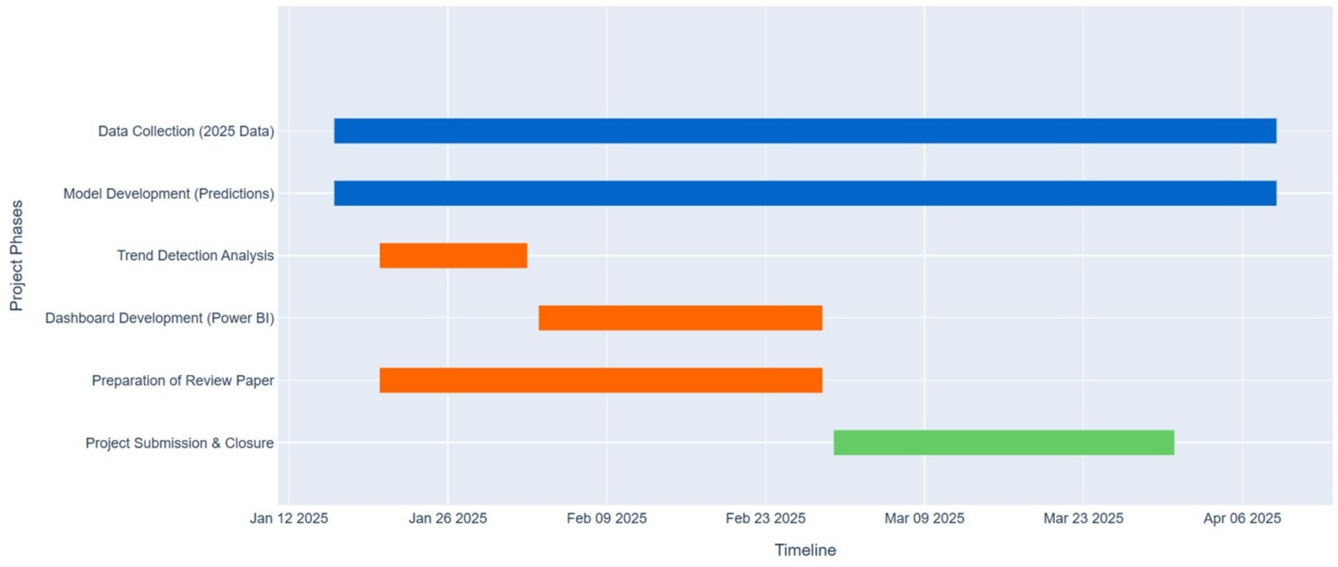


Figure 7.2: Gnatt Chart of Phase 2

Our primary goal in phase 2 was to improve our sentiment analysis system by continuing our data collection, refining preproceesing, feature engineering techniques and machine learning models. Starting from mid-January we started collecting fresh Twitter data and stored it in Mysql database. With this data, we started predicting sentiments using XGBoost model, which is our best performing model. After that, we implementd trend detection system to detect and alert trends that showed significant shifts. By February, our focus shifted to enhancing the Power BI dashboard, by adding visuals and creating new measures that can provide us more insights such as keyword and hashtag extraction, different sentiment percentages. After developing our front-end and back-end, we integratd them by using Fast API and tested our whole system. From mid-January, we started preparing our review paper and as we progressed into March, we were able to complete our paper. Finally, in April, we completed our final testing and submission of our project.

Chapter 8

Conclusion and Future Scope

In this project, we successfully developed a real-time sentiment analysis pipeline to evaluate real-time data on Twitter concerning the Apple brand. The tweets were analyzed using multiple machine learning models as well as feature engineering methods like BERT and VADER, enabling us to label them as positive, neutral, or negative with a fair amount of precision.

In Phase 1, we focused on setting up the foundation by implementing data collection using the Twikit library, which allowed us to gather tweets without authentication keys. We then worked on cleaning and preprocessing the data, applying feature engineering techniques like VADER and BERT, and training multiple machine learning models to determine the best one for sentiment prediction. Along the way, we tackled challenges like data imbalance and high-dimensional feature spaces by using resampling techniques and dimensionality reduction to improve model performance.

In Phase 2, we shifted our focus to refining and enhancing the system. We developed a real-time data streaming and processing pipeline to automate live data collection and analysis. To make the insights more accessible, we developed an interactive dashboard using PowerBIthat visually represents sentiment trends and added an authentication system for secure user access. We also integrated and tested both the front-end and back-end components to ensure the system runs smoothly and efficiently.

Overall, this project successfully demonstrated how real-time sentiment analysis can provide valuable insights into public opinion. The system we built lays a strong foundation for future improvements, such as incorporating more data sources, improving sentiment detection accuracy, and optimizing real-time processing for even better performance.

List of Publications

Papers Accepted

1. Mr. Ajith M Joshy, Ms. Anitta Mary Jose, Mr. Jerry Narakathara Thomas and Er. Talit Sara George, “TweetPulse:Interactive Sentiment Dashboard with Advanced Analytics“ accepted in 7th International Conference on Computational Intelligence and Digital Technology (ICCIDT 2025) (Accepted in April. 2025).
2. Mr. Ajith M Joshy, Ms. Anitta Mary Jose, Mr. Jerry Narakathara Thomas, and Er. Talit Sara George, “TweetPulse: A Power BI-Driven Interactive Sentiment Analysis Dashboard for Brand Monitoring“ accepted in 8th International Conference on Circuit, Power and Computing technologies 2025 (ICCPCT 2025) (Accepted in April. 2025).

Appendix

Back-End Pipeline

```
from _future_ import annotations
import os
import re
import asyncio
import csv
import pickle
import numpy as np
import nltk
import torch
import mysql.connector
import spacy
import smtplib
import pandas as pd
import pymysql
import emoji
import logging
import time
from time import sleep
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from mysql.connector import Error
from typing import NoReturn
from twikit import Client
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, TweetTokenizer
from nltk.stem import WordNetLemmatizer
from transformers import BertTokenizer, BertModel
from nltk.sentiment import SentimentIntensityAnalyzer
from sklearn.preprocessing import StandardScaler
from datetime import datetime
from sqlalchemy import create_engine
from sqlalchemy import text

# Config file (Path for models and scalers)

MODELS_DIR = r'H:\My Drive\Twitter 2024\Models'

# Scaler and model file paths
BERT_SCALER_PATH = f"{MODELS_DIR}/bert_scaler.pkl"
SCALER_PATH = f"{MODELS_DIR}/scaler.pkl"
XGB_MODEL_PATH = f"{MODELS_DIR}/xgb_bert.pkl"


# Import configurations
from config import (BERT_SCALER_PATH, SCALER_PATH, XGB_MODEL_PATH)

# Initialize NLTK resources
#nltk.download('stopwords')
#nltk.download('punkt')
#nltk.download('wordnet')
#nltk.download('vader_lexicon')
#nltk.download('punkt_tab')

# Load pre-trained tools
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
sia = SentimentIntensityAnalyzer()

# Global Objects
tweet_tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True,
reduce_len=True)
STOPWORDS = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

# Load spaCy's English model
nlp = spacy.load("en_core_web_sm")

# Load Scalers and Models
try:
    with open(BERT_SCALER_PATH, 'rb') as f:
        bert_scaler = pickle.load(f)
    with open(SCALER_PATH, 'rb') as f:
        scaler = pickle.load(f)
    with open(XGB_MODEL_PATH, 'rb') as f:
        loaded_rf_model = pickle.load(f)
    print("Models and scalers loaded successfully.")
except FileNotFoundError as e:
    print(f"Error loading files: {e}")
    raise

    # Twitter Client Initialization
client = Client('en-US')

import mysql.connector
from mysql.connector import Error

def get_db_connection():
    try:
        connection = mysql.connector.connect(
            host='localhost',  # Replace with your MySQL host
            user='root',  # Replace with your MySQL username
            password='Root@12345',  # Replace with your MySQL password
            database='ajith'  # Replace with your MySQL database name
        )
        if connection.is_connected():
            print("Connected to MySQL database")
        return connection
    except Error as e:
        print(f"Error connecting to MySQL: {e}")
        raise

    def is_tweet_in_database(connection, tweet_id: int) -> bool:
    query = "SELECT EXISTS(SELECT 1 FROM raw_tweets WHERE tweet_id = %s)"
    cursor = connection.cursor()
    cursor.execute(query, (tweet_id,))
    exists = cursor.fetchone()[0]
    cursor.close()
    return bool(exists)

# Text Preprocessing
def clean_text(text: str) -> str:
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'[^a-zA-Z0-9\s#]', '', text)
    # Remove non-alphanumeric characters except hashtags
    text = re.sub(r'\d+', '', text)  # Remove digits
    text = text.lower().strip()
    return text

def preprocess_text(text: str) -> str:
    text = clean_text(text)
    tokens = nltk.word_tokenize(text)
    tokens = [lemmatizer.lemmatize(word) for word in tokens
    if word not in STOPWORDS and not word.startswith('#')]
    return ' '.join(tokens)

# Extract Hashtags
def extract_hashtags(text: str) -> str:
    hashtags = re.findall(r'#\w+', text)
    return ", ".join(hashtags)

# Extract Keywords
def extract_keywords(text: str) -> str:
    clean = clean_text(text)
    tokens = tweet_tokenizer.tokenize(clean)
    keywords = [word for word in tokens if word not in
    STOPWORDS and not word.startswith('#')]
    return ", ".join(keywords)

# Extract BERT Features
def extract_bert_features_batch(texts, batch_size=32) -> np.ndarray:
    all_features = []
    for i in range(0, len(texts), batch_size):
        batch = texts[i:i + batch_size]
        inputs = tokenizer(batch, return_tensors="pt", padding=True,
        truncation=True, max_length=512)
        with torch.no_grad():
            outputs = model(**inputs)
        batch_features = outputs.last_hidden_state[:, 0, :].detach().numpy()
        all_features.extend(batch_features)
    return np.array(all_features)

# Sentiment Analysis
def get_vader_sentiment(text: str) -> str:
    sentiment_score = sia.polarity_scores(text)
    if sentiment_score['compound'] >= 0.2:
        return 'positive'
    elif sentiment_score['compound'] <= -0.2:
        return 'negative'
    else:
        return 'neutral'

# Store Raw Tweets
def store_raw_tweet(connection, tweet_count, tweet) -> None:

    location = tweet.user.location if tweet.user and tweet.user.location else
    'Unknown'
    tweet_id = int(tweet.id)

    # Convert created_at to MySQL-compatible format
    created_at = datetime.strptime(tweet.created_at, "%a %b %d %H:%M:%S %z %Y")
    created_at = created_at.strftime("%Y-%m-%d %H:%M:%S")

    query = """
    INSERT IGNORE INTO raw_tweets (tweet_count, tweet_id, text, created_at, location)
    VALUES (%s, %s, %s, %s, %s)
    """
    data = (tweet_count, tweet.id, tweet.text, created_at, location)
    cursor = connection.cursor()
    cursor.execute(query, data)
    connection.commit()
    cursor.close()

# Store Processed Tweets with Predictions
def store_processed_tweet(connection, tweet_count, tweet, sentiment,
sentiment_encoded, sentiment_confidence,
hashtags, keywords) -> None:

    location = tweet.user.location if tweet.user and tweet.user.location
    else 'Unknown'

    tweet_id = int(tweet.id)
    sentiment_encoded = int(sentiment_encoded)

    # Convert created_at to MySQL-compatible format
    created_at = datetime.strptime(tweet.created_at, "%a %b %d %H:%M:%S %z %Y")
    created_at = created_at.strftime("%Y-%m-%d %H:%M:%S")
    query = """
    INSERT INTO processed_tweets
    (tweet_count, tweet_id, text, created_at, sentiment, sentiment_encoded,
    sentiment_confidence, location,
    hashtags, keywords)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
    """
    data = (tweet_count, tweet.id, tweet.text, created_at, sentiment,
    sentiment_encoded,
    sentiment_confidence, location,
    hashtags, keywords)

    cursor = connection.cursor()
    #print("Inserting data:", data)
    cursor.execute(query, data)
    connection.commit()
    cursor.close()

# Main Twitter Pipeline
async def main() -> NoReturn:
    client.load_cookies('cookies.json')
    tweets = await client.search_tweet('#Apple', 'Latest')

    # Connect to MySQL
    connection = get_db_connection()

    try:

        # Track existing tweets to avoid duplicates
        raw_existing_ids = set()
        cursor = connection.cursor()
        cursor.execute("SELECT tweet_id FROM raw_tweets")
        raw_existing_ids = {row[0] for row in cursor.fetchall()}
        cursor.close()



        start_count = len(raw_existing_ids) + 1
        tweet_count = start_count - 1

        for tweet in tweets:
```

```
if tweet.lang == 'en' and tweet.id not in raw_existing_ids:
    if tweet.retweeted_tweet is None and tweet.in_reply_to is None:
        if not is_tweet_in_database(connection, tweet.id):
        # Check dynamically
            tweet_count += 1
            print(tweet.id, tweet.text, tweet.created_at)

            # Store raw tweet in MySQL
            store_raw_tweet(connection, tweet_count, tweet)

            # Extract hashtags
            hashtags = extract_hashtags(tweet.text)

            # Preprocess Text
            processed_text = preprocess_text(tweet.text)

            #Extract keywords
            keywords = extract_keywords(tweet.text)

            # Extract BERT Features
            bert_features = extract_bert_features_batch([processed_text])
            bert_features_scaled = bert_scaler.transform(bert_features)

            # Sentiment Analysis
            sentiment = get_vader_sentiment(processed_text)
            vader_features = [
                sia.polarity_scores(processed_text)['pos'],
                sia.polarity_scores(processed_text)['neg'],
                sia.polarity_scores(processed_text)['neu'],
                sia.polarity_scores(processed_text)['compound']
            ]
            X_combined = np.hstack
            ((bert_features_scaled, np.array(vader_features)
            .reshape(1, -1)))
            X_combined_scaled = scaler.transform(X_combined)

            # Predict Sentiment
            sentiment_encoded = loaded_rf_model
            .predict(X_combined_scaled)[0]
```

```
                        print(f"Processed tweet: {processed_text}")
                        print(f"Predicted Sentiment: {sentiment_encoded}")

                        # Store processed tweet in MySQL
                        store_processed_tweet(connection, tweet_count,
                        tweet, sentiment,
                        sentiment_encoded,
                        hashtags, keywords,
                        brand)

                        if tweet_count >= start_count + 19:
                            break

                        await asyncio.sleep(20)

    finally:
        if connection.is_connected():
            connection.close()
            print("MySQL connection closed.")

# Fetch sentiment data from MySQL and calculate negative percentage
def check_sentiment():

    engine = create_engine("mysql+mysqlconnector:
    //root:Root%4012345@localhost/ajith")

    # Query to fetch sentiment data from the last 1 hour
    query = text("SELECT created_at, sentiment FROM processed_tweets
    WHERE created_at >= NOW() - INTERVAL 1 HOUR")
    try:
        df = pd.read_sql(query, engine)
    except Exception as e:
        print(f"Error fetching data from MySQL: {e}")
        return

    negative_percentage = 0

    try:

        # Calculate negative sentiment percentage
        if not df.empty:
            negative_count = df[df['sentiment'] == 'negative'].shape[0]
            total_count = len(df)
            if total_count > 0:
                negative_percentage = (negative_count / total_count) * 100
                if total_count > 0 else 0

        # Send alert if negative sentiment is more than 20%
        if negative_percentage >= 10:
            send_email_alert(negative_percentage)
    except Exception as e:
        logging.error(f"Database error: {e}")

    finally:

        engine.dispose()  # Close the connection
        print("MySQL connection closed.")

# Email Alerting Function
def send_email_alert(negative_percentage):
    subject = "Alert: Sudden Increase in Negative Tweets!"
    body = f"Negative sentiment has risen to {negative_percentage:.2f}%
    in the last hour.\nCheck the dashboard for details."

    sender_email = ""
    receiver_email = ""
    password = ""

    msg = MIMEMultipart()
    msg["From"] = sender_email
    msg["To"] = receiver_email
    msg["Subject"] = subject
    msg.attach(MIMEText(body, "plain"))

    retries = 3  # Retry up to 3 times
    for attempt in range(retries):

        try:
            server = smtplib.SMTP("smtp.gmail.com", 587)
            server.starttls()
            server.login(sender_email, password)
            server.sendmail(sender_email, receiver_email, msg.as_string())
            server.quit()
            print("Email Alert Sent!")
            return
        except Exception as e:
            logging.error(f"Attempt {attempt+1} - Error sending email: {e}")
            time.sleep(2 ** attempt)

    # Main pipeline to handle both tasks
async def run_pipeline():

    try:

        await main()
        # Execute the sentiment check
        check_sentiment()

    except Exception as e:
        print(f"Error in pipeline: {e}")

# Run the Script
asyncio.run(run_pipeline())
```

Fast API

```
from fastapi import FastAPI, Depends, HTTPException, Request, Form
from fastapi.responses import HTMLResponse, RedirectResponse
from fastapi.templating import Jinja2Templates
from fastapi.staticfiles import StaticFiles
import subprocess
import mysql.connector
import bcrypt

  app = FastAPI()

# Load HTML templates
templates = Jinja2Templates(directory="templates")

POWER_BI_DASHBOARD_URL = "https://app.powerbi.com/links/zu4K47Vcyb?ctid=
2d78bb54-b810-411b-b716-d0e5a53f10a8&pbi_source=linkShare"

# Serve static files (CSS, images, etc.)
app.mount("/static", StaticFiles(directory="static"), name="static")
```

```
# Function to get a new database connection
def get_db_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="Root@12345",
        database="ajith"
    )

# Function to hash passwords securely
def hash_password(password: str) -> str:
    return bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt()).decode('utf-8')

# Function to verify hashed password
def verify_password(plain_password: str, hashed_password: str) -> bool:
    return bcrypt.checkpw(plain_password.encode
    ('utf-8'), hashed_password.encode('utf-8'))

def authenticate_user(username: str, password: str):
    db = get_db_connection()  # Get a new DB connection
    cursor = db.cursor()

    cursor.execute("SELECT password_hash FROM users WHERE email = %s", (username,))
    user = cursor.fetchone()

    cursor.close()
    db.close()  # Close the connection after query

    # Compare entered password with hashed password
    if user and verify_password(password, user[0]):
        return True
    return False

# Route for login page
@app.get("/", response_class=HTMLResponse)
async def login_page(request: Request):
    return templates.TemplateResponse("login.html", {"request": request})

# Route to process login
@app.post("/login")
```

```
async def login(request: Request, username:
str = Form(...), password: str = Form(...)):
if authenticate_user(username, password):
    return RedirectResponse(url=POWER_BI_DASHBOARD_URL, status_code=303)
else:
    raise HTTPException(status_code=401, detail="Invalid credentials")

# Route to trigger sentiment analysis pipeline
@app.api_route("/run-script", methods=["GET", "POST"])
async def run_script(request: Request):
    try:
        # data collection script
        subprocess.Popen(["python", "sentiment_pipeline_reprocess1.py"])
        return {"message": "Pipeline started successfully"}
    except Exception as e:
        return {"error": str(e)}
```

References

[1] X. He, “Sentiment Classification of Social Media User Comments Using SVM Models,” *2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pp. 1755–1759, Mar. 2024, doi: 10.1109/ainit61980.2024.10581547.

[2] A. Ingole, P. Khude, S. Kittad, V. Parmar, and A. Ghotkar, “Strategic Brand Sentiment Analysis for Competitive Surveillance,” 2024.

[3] S. S. Roy et al., “Hateful Sentiment Detection in Real-Time Tweets: An LSTM-Based Comparative Approach,” *IEEE Transactions on Computational Social Systems*, vol. 11, no. 4, pp. 5028–5037, Aug. 2024, doi: 10.1109/tcss.2023.3260217.

[4] T. Wang, K. Lu, K. P. Chow, and Q. Zhu, “Covid-19 sensing: Negative sentiment analysis on social media in China via BERT model,” *IEEE Access*, vol. 8, pp. 138162–138169, 2020.

[5] A. Radaideh and F. Dweiri, “Sentiment Analysis Predictions in Digital Media Content using NLP Techniques,” *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 11, 2023, doi: 10.14569/ijacsa.2023.0141128.

[6] A. Petrescu, C.-O. Truic˘a, and E.-S. Apostol, “Sentiment analysis of events in social media,” in 2019 *IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 143–149, 2019.

[7] A. S, “Real time twitter sentiment analysis using natural language processing,” *International Journal of Engineering Research* and, vol. V9, 07 2020.

[8] J. Jabbar, I. Urooj, W. JunSheng, and N. Azeem, “Real-time sentiment analysis on e-commerce application,” in 2019 *IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 391–396, 2019. 41

[9] M. S. Md Suhaimin et al., “Social media sentiment analysis and opinion mining in public security: Taxonomy, trend analysis, issues and future directions,” *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 9, p. 101776, Oct. 2023, doi: 10.1016/j.jksuci.2023.101776.

[10] M. A. Kausar, A. Soosaimanickam, and M. Nasar, “Public Sentiment Analysis on Twitter Data during COVID-19 Outbreak,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 2, 2021, doi: 10.14569/ijacsa.2021.0120252.

[11] E. F. Zineb, R. Najat, and A. Jaafar, “An Intelligent Approach for Data Analysis and Decision Making in Big Data: A Case Study on E-commerce Industry,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 7, 2021, doi: 10.14569/ijacsa.2021.0120783.

[12] Anupama B S, “Real Time Twitter Sentiment Analysis using Natural Language Processing,” *International Journal of Engineering Research and Technology*, vol. 9, no. 07, Jul. 2020, doi: 10.17577/ijertv9is070406.

[13] A. Omar, “Discourse-based Opinion Mining of Customer Responses to Telecommunications Services in Saudi Arabia during the COVID-19 Crisis,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 6, 2022, doi: 10.14569/ijacsa.2022.0130642.

[14] D. P. and K. Ahmed, “A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools,” *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, 2016, doi: 10.14569/ijacsa.2016.070267.

[15] R. Mahajan and V. Mansotra, “Correlating Crime and Social Media: Using Semantic Sentiment Analysis,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 3, 2021, doi: 10.14569/ijacsa.2021.0120338.

[16] Bangera and K. N., “Machine Learning Driven Feature Sensitive Progressive Sampling Model for BigData Analytics,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 11, 2021, doi: 10.14569/ijacsa.2021.0121138.

[17] M. Wankhade, A. C. S. Rao, and C. Kulkarni, “A survey on sentiment analysis methods, applications, and challenges,” *Artificial Intelligence Review*, vol. 55, no. 7, pp. 5731–5780, Feb. 2022, doi: 10.1007/s10462-022-10144-1.

[18] F. Alattar and K. Shaalan, “A Survey on Opinion Reason Mining and Interpreting Sentiment Variations,” *IEEE Access*, vol. 9, pp. 39636–39655, 2021, doi: 10.1109/access.2021.3063921.

[19] A. Motz et al., “Live Sentiment Analysis Using Multiple Machine Learning and Text Processing Algorithms,” *Procedia Computer Science*, vol. 203, pp. 165–172, 2022, doi: 10.1016/j.procs.2022.07.023.

[20] J. Park, “Framework for Sentiment-Driven Evaluation of Customer Satisfaction With Cosmetics Brands,” *IEEE Access*, vol. 8, pp. 98526–98538, 2020, doi: 10.1109/access.2020.2997522.

[21] S. Rani, N. Singh, and P. Gulia, “Survey of Tools and Techniques for Sentiment Analysis of Social Networking Data,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 4, 2021, doi: 10.14569/ijacsa.2021.0120430.

[22] S. Shayaa et al., “Sentiment Analysis of Big Data: Methods, Applications, and Open Challenges,” *IEEE Access*, vol. 6, pp. 37807–37827, 2018, doi: 10.1109/access.2018.2851311.

[23] H.-H. Nguyen, “Enhancing Sentiment Analysis on Social Media Data with Advanced Deep Learning Techniques,” *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 5, 2024, doi: 10.14569/ijacsa.2024.0150598.

[24] K. Park, S. Park, and J. Joung, “Contextual Meaning-Based Approach to Fine-Grained Online Product Review Analysis for Product Design,” *IEEE Access*, vol. 12, pp. 4225–4238, 2024, doi: 10.1109/access.2023.3343501.