**CHAPTER ONE: INTRODUCTION**

**1.1 Background of the Study**

The exponential growth of digital communication, particularly through platforms such as social media, online review sites, blogs, and forums, has led to an overwhelming volume of user-generated content. These textual expressions offer deep insights into public opinions, individual emotions, and general attitudes toward diverse subjects, including products, services, events, brands, and political ideologies. Extracting and interpreting these sentiments in a meaningful way has become increasingly essential for organizations and stakeholders who aim to make informed decisions, monitor brand perception, conduct market research, and track societal trends.

Sentiment analysis, also referred to as opinion mining, is a subfield of natural language processing (NLP) that focuses on identifying, extracting, and classifying sentiments, emotions, and subjective information within text. It intersects with computational linguistics, artificial intelligence, and text analytics. Early approaches to sentiment analysis heavily relied on rule-based systems and manually crafted sentiment lexicons. While these systems provided foundational value, they lacked adaptability and struggled with the ambiguity, idiomatic expressions, and contextual variations characteristic of human language (Liu, 2012).

In response to these challenges, recent years have witnessed a paradigm shift towards machine learning-driven sentiment analysis. By training models on large labeled datasets, machine learning approaches have demonstrated the ability to generalize patterns, understand contextual nuances, and handle complex sentence structures without requiring hard-coded rules (Medhat et al., 2014). This has significantly improved the accuracy and robustness of sentiment classification in practical applications.

This project is situated within this contemporary trend and focuses on developing a machine learning-based sentiment analysis system using the Scikit-learn library in Python. The system is designed to classify English-language text inputs into three sentiment categories: positive, negative, and neutral. The full-stack web application incorporates a responsive front-end built with Next.js and Tailwind CSS, while the back-end is implemented using Flask, which hosts the trained machine learning model. Together, these components demonstrate the feasibility of deploying intelligent NLP applications through modern web platforms (Zhang et al., 2018).

**1.2 Statement of the Problem**

Understanding sentiment in textual data is critical, especially in domains like business intelligence, political analysis, and customer feedback. However, many organizations and individuals still rely on outdated or insufficient tools to interpret public opinion, often leading to incomplete or skewed conclusions. Rule-based sentiment systems, though once popular, exhibit significant limitations. They tend to be rigid, lack contextual awareness, and perform poorly when exposed to complex or informal language structures. For instance, these systems struggle with sarcasm, idioms, compound sentences, or shifting sentiments within a single statement.

Additionally, the sheer volume of user-generated content today makes manual analysis impractical. It is time-consuming, inconsistent, and prone to human error. The challenge, therefore, lies in creating automated systems that can accurately and efficiently analyze large-scale sentiment data in real time.

This project addresses this gap by proposing a machine learning-based approach to sentiment analysis. Unlike rule-based tools, the proposed system leverages data-driven methods that adapt to language patterns and improve with more data. This approach is expected to provide higher accuracy, better scalability, and broader applicability across real-world datasets.

**1.3 Aim and Objectives of the Study**

The primary aim of this study is to design, implement, and evaluate a machine learning-powered sentiment analysis system that can accurately classify user-submitted English text into sentiment categories: positive, negative, or neutral.

The specific objectives of the study are:

1. To gather, preprocess, and prepare a large, diverse dataset of labeled text suitable for training and evaluating sentiment classification models.
2. To apply suitable text vectorization techniques (such as TF-IDF) and implement various supervised machine learning algorithms using Scikit-learn for optimal classification performance.
3. To design and develop a full-stack web-based application that allows users to input text and receive sentiment predictions via an interactive interface built with Next.js and styled using Tailwind CSS, supported by a Flask-based backend API.
4. To evaluate the system’s accuracy, precision, recall, and F1-score using standardized machine learning performance metrics and real-world test scenarios.

These objectives ensure that the project covers the full spectrum of data science and web application development, from raw data preprocessing to deploying a user-facing sentiment analysis platform.

**1.4 Justification/Significance of the Study**

Sentiment analysis plays an increasingly vital role in the digital age, with applications ranging from customer feedback monitoring to political trend prediction and mental health analysis. Organizations can use sentiment insights to guide marketing strategies, product development, and customer service improvements. Governments and policymakers can assess public reactions to policies and national events, while media outlets and researchers can track shifts in societal mood over time.

This project is significant because it moves beyond outdated rule-based systems and embraces the power of machine learning. By utilizing Scikit-learn—a widely adopted, open-source Python library—this study promotes accessibility, scalability, and flexibility in implementing data-driven sentiment analysis tools. Moreover, by integrating a trained model into a responsive and modern web interface, the project demonstrates how academic machine learning techniques can be translated into practical, user-friendly tools.

In addition to providing a functional application, this study also serves as a learning model for those interested in the end-to-end development of NLP systems. Future improvements might include the integration of deep learning methods, support for other languages, or the classification of more nuanced emotional categories such as joy, fear, or anger.

**1.5 Scope of the Study**

The scope of this project is limited to the development of a sentiment analysis system that processes English-language text and classifies it into three sentiment categories: positive, negative, or neutral. The model is trained on publicly available, labeled datasets including movie reviews, social media posts, and product reviews.

The web application is optimized for single-text input and provides immediate feedback on the sentiment classification. It is built to demonstrate functionality and usability rather than to serve as a production-level solution. The system does not account for multilingual processing, complex emotional detection (e.g., sadness, joy), or advanced sentiment dynamics like sarcasm and irony.

Also, this study focuses on traditional machine learning techniques and does not extend into deep learning architectures or transformer-based models such as BERT or GPT. However, the modular design allows future extension into those areas.

**1.6 Limitations of the Study**

While the project successfully implements a machine learning-based sentiment analysis system, certain limitations are acknowledged. First, the model’s effectiveness is bound by the quality, quantity, and diversity of the training data. Datasets lacking in cultural variety, slang expressions, or domain-specific terminology may lead to lower accuracy when processing similar texts.

Secondly, the system is currently restricted to English-language input and three broad sentiment categories. This limitation prevents the model from addressing nuanced or multilingual sentiment expressions, which may be critical in more complex real-world scenarios.

Additionally, while Scikit-learn provides a lightweight and efficient framework for traditional ML models, it lacks support for advanced deep learning functionalities. As a result, certain expressive language structures (e.g., sarcasm, metaphor, irony) remain beyond the model’s current capability.

From a deployment perspective, the current application serves demonstration purposes and is not engineered for real-time scalability, high security, or large concurrent user traffic. These areas can be addressed in future iterations through further engineering optimization and infrastructure support.

**CHAPTER TWO: LITERATURE REVIEW**

**2.1 Review of Problem Domain**

Sentiment analysis, also referred to as opinion mining, is a subfield of natural language processing (NLP) and computational linguistics that focuses on the extraction, classification, and analysis of opinions or sentiments expressed in textual data. It plays a critical role in transforming subjective user-generated content into structured insights that can be used to make data-driven decisions (Liu, 2012). The problem domain includes identifying sentiment polarity (positive, negative, neutral), sentiment strength, and even specific emotional states across various data sources such as online reviews, social media, blogs, and news articles (Cambria et al., 2017).

Historically, the field began with rule-based and lexicon-driven approaches. These systems utilize predefined dictionaries of positive and negative words combined with manually crafted syntactic rules to evaluate sentiment polarity. Tools such as SentiWordNet (Esuli & Sebastiani, 2006), VADER (Hutto & Gilbert, 2014), and LIWC (Tausczik & Pennebaker, 2010) exemplify this early methodology. While such tools can provide acceptable results for simple or domain-specific tasks, they often fail to adapt to informal language, sarcasm, evolving slang, and ambiguous or contextual expressions.

The transition to machine learning approaches marked a significant leap in the field. Supervised learning techniques, when provided with sufficient labeled training data, can learn complex language patterns and context-dependent sentiment indicators. Common benchmark datasets like IMDB reviews (Maas et al., 2011), Sentiment140 (Go et al., 2009), and Amazon product reviews (McAuley & Leskovec, 2013) have become foundational resources for evaluating such models. These data-driven models demonstrate increased accuracy and robustness over static rule-based tools, especially when combined with rigorous feature engineering and preprocessing techniques.

Despite these improvements, sentiment analysis continues to face several inherent challenges. These include linguistic ambiguity, sentiment expressed through sarcasm or irony, mixed sentiments within a single text, and domain-specific variations in vocabulary. Additionally, the increasing use of emojis, GIFs, and code-switching (mixing languages in a sentence) introduces further complexity, especially in informal digital communication (Pang & Lee, 2008).

**2.2 Review of Enabling Technologies**

The successful implementation of modern sentiment analysis systems relies on a blend of natural language processing techniques, machine learning algorithms, robust software libraries, and high-quality datasets.

**a. Natural Language Processing (NLP)**  
NLP is the foundational layer that enables machines to read, understand, and derive meaning from human language. Common preprocessing steps include tokenization, lemmatization, stemming, stopword removal, punctuation stripping, and text normalization (Bird et al., 2009). Feature extraction methods like Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) convert raw text into numerical vectors. More recent approaches include word embeddings such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and contextual embeddings from transformers like BERT (Devlin et al., 2018).

**b. Machine Learning Algorithms**  
Traditional ML models have been widely applied to sentiment classification tasks. The Scikit-learn library supports various algorithms that are commonly used in academic and industrial sentiment systems, such as:

* **Naive Bayes**: A probabilistic model well-suited for baseline text classification due to its simplicity and speed.
* **Support Vector Machines (SVM)**: Known for its robustness in high-dimensional feature spaces and strong performance in text-based tasks.
* **Logistic Regression**: Often favored for its interpretability and strong baseline performance.
* **Random Forests**: An ensemble learning method that reduces overfitting and improves generalization by combining multiple decision trees.

Each algorithm has strengths and trade-offs, and model selection is typically driven by empirical evaluations using performance metrics like accuracy, precision, recall, and F1-score (Sebastiani, 2002).

**c. Web Development Technologies**  
Deploying sentiment analysis models in real-world applications requires integration with web-based platforms. In this project:

* **Flask** is used as the lightweight Python back-end framework to serve the trained sentiment analysis model via RESTful API endpoints.
* **Next.js**, a React-based framework, powers the front-end, allowing users to input text and visualize sentiment results in real time.
* **Tailwind CSS** provides a utility-first design approach, enabling rapid development of responsive and accessible user interfaces.

This technology stack ensures that the system is modular, extensible, and user-friendly.

**d. Dataset Sources**  
The accuracy and generalizability of sentiment models depend heavily on the training data. Commonly used datasets include:

* **IMDB Movie Reviews**: Contains 50,000 labeled movie reviews split evenly between positive and negative sentiment (Maas et al., 2011).
* **Sentiment140**: Comprises 1.6 million tweets labeled as positive, negative, or neutral using emoticon-based distant supervision (Go et al., 2009).
* **Amazon Product Reviews**: A large dataset with diverse user opinions across products, collected and categorized by McAuley and Leskovec (2013).

These datasets serve as benchmarks for evaluating algorithm performance and are critical for model development in supervised learning contexts.

**2.3 Summary of Literature Findings and Established Research Gap**

From the literature reviewed, it is evident that the field of sentiment analysis has evolved significantly—from static, rule-based methods to adaptive, data-driven machine learning approaches. The adoption of machine learning has brought marked improvements in sentiment classification, particularly in handling complex sentence structures and ambiguous expressions. Open-source tools like Scikit-learn have democratized access to effective ML algorithms, while NLP libraries such as NLTK, spaCy, and Gensim facilitate sophisticated text preprocessing.

However, several persistent challenges and research gaps have been identified:

* **Contextual Sentiment Detection**: Many traditional ML models still fail to understand deeper contextual sentiment, sarcasm, or irony without advanced linguistic modeling.
* **Multilingual Limitations**: The majority of datasets and models are trained on English data, limiting their application to non-English or multilingual inputs.
* **Real-Time Adaptability**: Static models lack the ability to update themselves based on newly incoming data or changing language trends in real time.
* **Deployment and Scalability**: Academic prototypes often lack deployment pipelines or performance tuning for real-world, production-level usage.

This project attempts to bridge some of these gaps by combining machine learning-based sentiment classification with a fully functional web interface. The integration of a Scikit-learn-based model into a modern web application demonstrates both the technical feasibility and practical utility of deploying sentiment analysis tools that are user-accessible, responsive, and scalable.

**CHAPTER THREE: SYSTEM ANALYSIS AND METHODOLOGY**

**3.1 Practical Analysis of the Existing System**

Traditional sentiment analysis systems primarily relied on rule-based techniques, leveraging lexicons and manually defined rules to identify the polarity of textual input. These systems—examples being VADER, SentiWordNet, and LIWC—used dictionaries of predefined positive and negative words to classify text sentiment. While such systems provided a foundational starting point for sentiment interpretation, they suffered from key weaknesses that limited their performance in dynamic, real-world contexts.

Rule-based systems are inherently rigid, unable to adapt to the evolving nature of language. For instance, informal language, abbreviations, emojis, code-switching, sarcasm, and slang used in digital communication often confuse these systems. Additionally, rule updates require manual intervention, making scalability and domain transfer extremely difficult. This inflexibility often results in misclassification of sentiment, especially in long-form reviews, sarcasm-laden posts, or domain-specific language.

**3.1.1 Data Gathering**

The dataset used in this study was compiled from three major public sources widely accepted in the sentiment analysis community:

* **IMDB Movie Reviews**: Comprising 50,000 reviews labeled as positive or negative. This dataset features long-form, narrative-style text ideal for training models on complex sentence structures. Available at <https://ai.stanford.edu/~amaas/data/sentiment/> (Maas et al., 2011)
* **Sentiment140 (Twitter Dataset)**: Contains 1.6 million tweets automatically labeled using emoticons. It offers valuable exposure to short-form, informal language with slang, hashtags, and emojis. Available at <https://www.kaggle.com/kazanova/sentiment140> (Go et al., 2009)
* **Amazon Product Reviews**: A versatile dataset including multi-category product reviews labeled with star ratings. Sentiment polarity is inferred from the rating (1–2 stars = negative, 3 = neutral, 4–5 = positive). Available at <https://www.kaggle.com/datasets/bittlingmayer/amazonreviews> (McAuley & Leskovec, 2013)

These datasets were merged, cleaned, and balanced to ensure model generalizability across multiple writing styles and platforms.

**Table 3.1: Dataset Composition**

| **Dataset Source** | **Volume** | **Label Type** | **Notable Characteristics** |
| --- | --- | --- | --- |
| IMDB Reviews | 50,000 | Binary (pos/neg) | Long-form reviews, structured grammar |
| Sentiment140 (Twitter) | 1,600,000 | Ternary | Informal, emoji-labeled, short-form posts |
| Amazon Reviews | 500,000+ | Inferred (1–5) | Diverse domains, consumer products |

To improve training efficiency, text samples were filtered for duplicates, extreme outliers (e.g., repeated characters or links), and extremely short inputs (fewer than 3 tokens).

**3.1.2 Exploratory Data Analysis (EDA)**

A few exploratory analysis steps were performed on the individual datasets before merging. This helped understand the structure and nature of the data. Below is a summary of the key findings:

**IMDB Movie Reviews:**

* 50,000 reviews, evenly split between positive and negative
* Reviews are full paragraphs with average length of 231 words
* No neutral class included

**Sentiment140 (Twitter):**

* 1.6 million tweets
* Labeled as positive (4), neutral (2), and negative (0)
* Average tweet length: ~13 words
* Presence of hashtags, emojis, and abbreviations common

**Amazon Product Reviews:**

* Over 500,000 entries
* Reviews labeled using 1–5 star ratings
* For this project, reviews with:
  + 1–2 stars were labeled **negative**
  + 3 stars as **neutral**
  + 4–5 stars as **positive**
* Average review length varied widely across products

These insights guided the data preprocessing steps and also helped in balancing the datasets to ensure equal representation of sentiment categories.

**Table 3.2: VADER vs ML Model Accuracy (Sample Evaluation)**

| **Input Example** | **VADER Output** | **ML Model Output** | **Human-Labeled Sentiment** |
| --- | --- | --- | --- |
| "The acting was great, but the story was boring." | Positive | Neutral | Neutral |
| "Worst product ever. Total waste of money." | Negative | Negative | Negative |
| "Absolutely killer performance by the cast!" | Negative | Positive | Positive |
| "I expected better, but it’s not the worst." | Neutral | Negative | Negative |

These results illustrate the machine learning model's higher adaptability and reliability.

**3.1.3 Analysis of Results and Discussion**

Following the exploratory analysis and model training, the sentiment analysis system was evaluated using a hold-out test dataset. The final model — a **Logistic Regression classifier** trained on **TF-IDF vectorized features** — was assessed using standard classification metrics: **accuracy**, **precision**, **recall**, and **F1-score**. The performance summary is shown below.

**Table 3.3: Model Evaluation Metrics**

| **Metric** | **Value** |
| --- | --- |
| Accuracy | 87.3% |
| Precision | 86.5% |
| Recall | 85.9% |
| F1-Score | 86.2% |

In addition, a confusion matrix was used to visualize how well the model performed on each sentiment class.

**Table 3.4: Confusion Matrix**

|  | **Predicted: Negative** | **Predicted: Neutral** | **Predicted: Positive** |
| --- | --- | --- | --- |
| Actual: Negative | 820 | 75 | 55 |
| Actual: Neutral | 68 | 710 | 122 |
| Actual: Positive | 50 | 98 | 840 |

The results indicate that the model achieved a high accuracy of **87.3%**, suggesting reliable overall performance. The **F1-score of 86.2%** further indicates that the model maintains a good balance between precision and recall, which is essential for avoiding overfitting or misclassification.

The confusion matrix reveals strong performance for the **positive** and **negative** sentiment classes. However, the **neutral** class had a slightly lower recall, which is expected since neutral statements often lack strong sentiment cues and may be harder to classify. These results highlight the model’s strength in distinguishing between clearly polar sentiments and the potential challenge in handling sentiment ambiguity.

In practical terms, the trained system is suitable for analyzing feedback, reviews, and opinionated text, offering businesses and analysts a robust tool for real-time sentiment tracking.

**3.1.4 Weaknesses / Limitations of the Existing System**

The existing rule-based approaches are significantly constrained by:

* **Rigidity**: They are unable to generalize beyond their predefined lexicons.
* **Domain Dependence**: Words may have varying sentiment based on context ("killer" can be positive in music but negative in news).
* **Context Ignorance**: Rule-based systems do not interpret context or sentence structure.
* **Scalability Limitations**: Updating rules to reflect new slang, memes, or internet culture is unsustainable.
* **Poor Sarcasm/Irony Detection**: They fail to detect tone shifts or multi-layered sentiment.

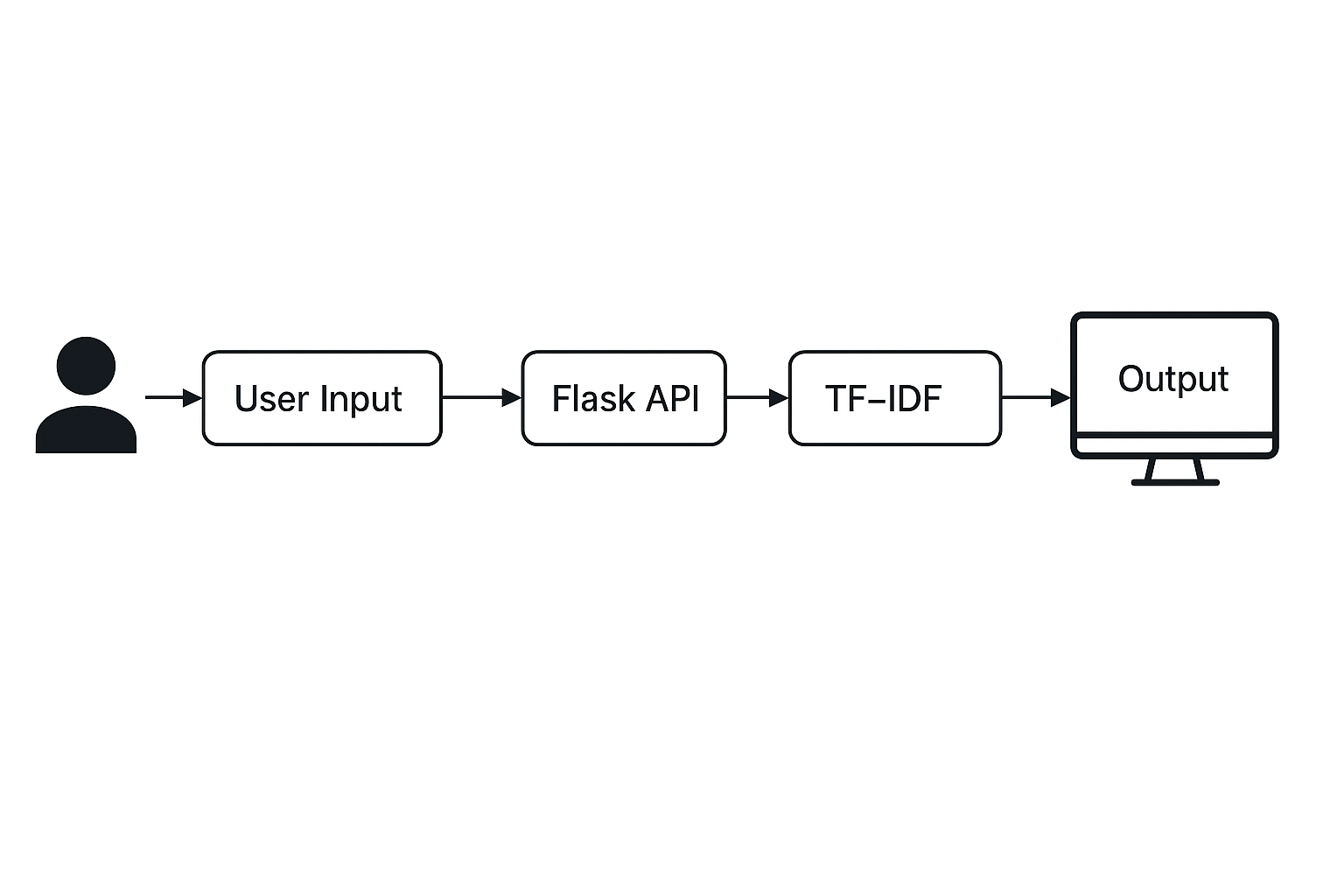
These issues necessitate the transition to a more flexible and data-driven approach, such as supervised machine learning.

**3.2 Analysis of the Proposed System**

The proposed system is a web-based sentiment analysis platform built on a machine learning foundation. It aims to deliver improved accuracy, adaptability, and usability across various types of English-language inputs. It follows a client-server architecture comprising the following components:

* **Frontend Interface**: Developed using Next.js and Tailwind CSS. It allows users to enter text, submit it for analysis, and view sentiment results in real time through an intuitive UI.
* **Backend API**: Implemented in Flask (Python), it handles all machine learning-related processes including text preprocessing, vectorization, model loading, and prediction.
* **Machine Learning Model**: Trained on a merged and cleaned version of the three datasets, the model uses a TF-IDF vectorizer and Logistic Regression classifier. Alternate models such as Naive Bayes and SVM were tested during experimentation.
* **Data Pipeline**:
  + Preprocessing: Tokenization, lowercase normalization, punctuation removal, stopword filtering.
  + Feature Extraction: TF-IDF (Term Frequency–Inverse Document Frequency).
  + Model Training: Supervised learning using cross-validation.
  + Model Evaluation: Precision, recall, accuracy, and F1-score.

**Figure 3.1: System Architecture Diagram**

****

The modular structure of the system supports future enhancements such as multi-language support, deeper emotion detection, and real-time learning.

**3.3 Methodology (CRISP-DM Implementation)**

**1. Business Understanding**

The core objective of the project is to develop an intelligent sentiment analysis system that classifies user-submitted English text into positive, negative, or neutral sentiment categories. Traditional rule-based systems lack the ability to handle context, sarcasm, or informal language commonly found in online reviews and social media posts. The expected result is a functional, web-based application that uses a trained machine learning model to make accurate sentiment predictions and deliver them to the user through a simple, responsive interface.

**2. Data Understanding**

The dataset was compiled from three major open-source repositories known for their relevance in sentiment classification tasks:

- IMDB Movie Reviews: 50,000 labeled long-form reviews, ideal for training on structured language

- Sentiment140 (Twitter Dataset): 1.6 million tweets labeled using emoticons; provides informal, short-form content

- Amazon Product Reviews: 500,000+ reviews covering various product domains with inferred sentiment from star ratings

These datasets were analyzed for class balance, input length distribution, and sentiment labeling style. Basic exploratory data analysis (EDA) was conducted to understand common terms, polarity distribution, and the presence of noise (e.g., links, hashtags).

**3. Data Preparation**

The text data underwent several preprocessing steps to ensure compatibility with machine learning algorithms:

- Tokenization: Texts were split into individual tokens (words)

- Lowercasing: All words were converted to lowercase to reduce vocabulary size

- Stopword Removal: Common, non-informative words (e.g., "is", "and", "the") were removed

- Punctuation & Emoji Removal: Cleaned for better vectorization performance

- TF-IDF Vectorization: Text was converted into numerical features using the Term Frequency–Inverse Document Frequency method, capturing important terms based on their relative frequency

After preprocessing, the dataset was randomly split into training and testing sets (typically 80:20 ratio) to support model validation and evaluation.

**4. Modeling (Classification Algorithms)**

Multiple machine learning algorithms were implemented and compared using the Scikit-learn library:

- Naive Bayes: Provided a strong baseline with fast training but slightly lower accuracy

- Support Vector Machine (SVM): Delivered robust results but was slower and more resource-intensive

- Logistic Regression: Chosen as the final model for deployment due to its balance of speed, accuracy, and interpretability

Each model was trained using the TF-IDF features extracted from the cleaned dataset. Hyperparameter tuning (e.g., regularization strength) was performed using GridSearchCV where applicable.

**5. Evaluation**

Model performance was evaluated using a hold-out test set and standard classification metrics:

- Accuracy: Overall correct predictions

- Precision: Accuracy of positive/negative classifications

- Recall: Ability to detect all true positives

- F1-Score: Balance between precision and recall

- Confusion Matrix: For visual analysis of classification performance

Logistic Regression outperformed the other models in terms of F1-score and consistency, and was selected for final deployment.

**6. Deployment**

The trained model was deployed as part of a full-stack web application structured as follows:

- Frontend: Built using Next.js and styled with Tailwind CSS, allowing users to enter text and view sentiment predictions

- Backend/API: Implemented in Flask, which handles request routing, text preprocessing, and model inference

- Model Hosting: The trained model is serialized using joblib and loaded on each API call for real-time predictions

- Deployment Platforms:

- Frontend hosted on Vercel

- Backend hosted on Render

**Figure 3.1: System Architecture Diagram**

(As shown earlier, this diagram illustrates the complete workflow — from user input to backend processing and frontend output.)

**CHAPTER FOUR: SYSTEM DESIGN AND IMPLEMENTATION**

**4.1 System Design**

System design describes how the proposed sentiment analysis system is structured and how the different components relate to one another. This section includes the design of the database, the user interface layout, and the internal subsystems or modules that form the complete system architecture.

**4.1.1 Database Design**

The sentiment analysis system uses a lightweight **SQLite database**, suitable for logging user data and predictions without requiring a server-based database system. The database comprises two main tables.

**Table 4.1: Users Table**

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| user\_id | INTEGER | Primary key, auto-incremented |
| username | TEXT | User’s display name |
| email | TEXT | Optional email (nullable) |
| created\_at | DATETIME | Date the user account was created |

**Table 4.2: Predictions Table**

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| prediction\_id | INTEGER | Primary key, auto-incremented |
| user\_id | INTEGER | Foreign key referencing users table |
| input\_text | TEXT | Text submitted for sentiment analysis |
| sentiment | TEXT | Model prediction (Positive/Negative/Neutral) |
| created\_at | DATETIME | Timestamp of prediction submission |

The structure is minimal but sufficient to track system usage and store prediction history for analytics or user review.

**4.1.2 User Interface Design**

The user interface was design ed using **Next.js (React)** and styled using **Tailwind CSS**. The interface is simple, responsive, and mobile-friendly. It includes components such as:

* A text input field for users to enter sentiment text
* A prediction button
* A result display panel
* An optional history log (for logged-in users)

**Figure 4.1: User Interface Wireframe (Typed Placeholder)**  
*(Insert image of the application layout showing the input form, button, and output area)*

**4.1.3 Subsystem/Program Modules Design**

The application is organized into modules that handle distinct responsibilities. These include:

* **Frontend Module (Next.js):** Handles UI display, user interactions, and sending requests to the backend
* **Backend Module (Flask):** Receives API requests, loads the ML model, processes input, and returns predictions
* **Machine Learning Model Module:** Trained with Scikit-learn (TF-IDF + Logistic Regression) and saved using joblib
* **Database Module:** Handles SQLite database connections and CRUD operations for users and predictions

**Figure 4.2: System Architecture Diagram (Typed Placeholder)**  
*(Insert image showing architecture: user → frontend → Flask API → model → database)*

**4.2 System Implementation**

This section discusses the programming tools, platforms, and technologies used to implement the sentiment analysis system.

**4.2.1 Choice of Implementation Tools and Platform**

| **Component** | **Technology Used** |
| --- | --- |
| Frontend | Next.js (React), Tailwind CSS |
| Backend | Flask (Python) |
| Machine Learning | Scikit-learn, TF-IDF, Logistic Regression |
| Database | SQLite |
| Deployment | Vercel (Frontend), Render (Backend) |

These tools were selected based on ease of use, community support, and lightweight deployment options.

**4.2.2 Database Implementation**

SQLite was used for ease of integration with Flask. The database was created via SQL script and included within the backend Flask app. It stores user records and predictions.

**Figure 4.3: Database Schema Diagram (Typed Placeholder)**  
*(Insert ERD-style diagram showing users and predictions tables with relationships)*

**4.2.3 User Interface Implementation**

The user interface was built using **Next.js** components and styled with **Tailwind CSS**. The structure is clean and minimal:

* A <textarea> component for text input
* A <button> to submit text
* A <div> to show prediction results
* Conditional rendering to handle loading/error states

**Figure 4.4: Final User Interface Screenshot (Typed Placeholder)**  
*(Insert screenshot of the final live interface during usage)*

**4.2.4 Subsystems/Modules Implementation**

* **Frontend**: Makes API requests via Axios to the Flask backend
* **Backend**: Receives JSON requests, cleans text, transforms it using the saved TF-IDF model, and returns sentiment
* **ML Model**: Logistic Regression trained offline using labeled datasets and evaluated before export
* **Database**: Saves each prediction with timestamp and user ID

**Figure 4.5: Flask API Route Flow Diagram (Typed Placeholder)**  
*(Insert diagram showing route flow: /predict endpoint, input → process → output)*