Sentiment Analysis of Social Media Data

Mrs.Sangita Mishra[1], G. Jahnavi[2], G.Sailu[3], S.Rushikesh[4],L.Manikanta[5]

Department of Computer Science & Engineering (Data Science)

Avanthi Institute of Engineering & Technology (Autonomous), Vizianagaram, India

Sangita.mishra451@gmail.com[1],gadadasu.jahnavi@gmail.com[2], garikinasailu6@gmail.com[3], rushikeshseera2@gmail.com[4], manikanta2052005@gmail.com[5]

Abstract

The proliferation of social media platforms has generated unprecedented volumes of user-generated textual content, making automated sentiment analysis an essential tool for extracting actionable insights from unstructured data. This paper presents a web-based sentiment analysis system targeting Twitter data, employing a Support Vector Classifier (SVC) trained on Term Frequency–Inverse Document Frequency (TF-IDF) feature representations. Raw tweets undergo a structured Natural Language Processing (NLP) pipeline comprising tokenization, stopword removal, stemming, and noise elimination before being vectorized and classified into positive, negative, or neutral sentiment categories. The trained model is serialized via Pickle and deployed within a lightweight Flask web framework, enabling real-time prediction through both a web interface and a REST API endpoint. Experimental evaluations demonstrate that the SVC model achieves competitive accuracy compared with traditional classifiers such as Naïve Bayes and Logistic Regression, particularly when operating on high-dimensional sparse text vectors. The system handles edge cases including abbreviated language, informal expressions, and empty inputs gracefully. The paper describes the end-to-end architecture, implementation details, evaluation methodology, and identifies directions for future enhancement using transformer-based language models and multilingual support.

Index Terms—Sentiment Analysis, Support Vector Classifier, Natural Language Processing, TF-IDF, Flask, Social Media Mining.

I. INTRODUCTION

Social media platforms such as Twitter generate millions of short-form posts daily, embedding opinions, reactions, and attitudes toward a vast spectrum of topics ranging from commercial products to geopolitical events [1]. The capacity to mine this data for sentiment—determining whether a given post expresses a positive, negative, or neutral viewpoint—has significant commercial and research value. Businesses leverage sentiment signals for brand monitoring and customer feedback analysis; public health agencies utilize them for tracking societal mood during crises; and political analysts employ them to gauge electorate sentiment [2].

Manual annotation of social media corpora is both time-intensive and economically impractical at scale. Consequently, automated machine learning methods have become the standard approach to sentiment classification. Classical algorithms such as Naïve Bayes and Logistic Regression established early baselines; however, Support Vector Machines (SVMs), and specifically Support Vector Classifiers (SVCs), have consistently demonstrated superior generalization performance on text classification tasks involving high-dimensional, sparse feature spaces [3].

This work presents a complete pipeline that (i) preprocesses raw Twitter data through an NLP workflow, (ii) extracts numerical features using TF-IDF vectorization, (iii) trains and serializes an SVC model, and (iv) deploys the model within a Flask-based web application capable of real-time inference. The remainder of this paper is structured as follows: Section II reviews related work; Section III describes the methodology and system design; Section IV presents results; Section V concludes and outlines future directions.

II. RELATED WORK

Sentiment analysis has a rich history in NLP. Pang et al. [4] pioneered the application of machine learning to movie review polarity classification, demonstrating that SVM outperforms Naïve Bayes and Maximum Entropy classifiers when combined with unigram features. Their findings established the foundation for supervised sentiment classification.

Go et al. [5] extended this paradigm to Twitter by employing distant supervision, using emoticons as noisy labels to train polarity classifiers. Their study confirmed that SVM with unigram and bigram features achieved the highest accuracy among the evaluated models. Pak and Paroubek [6] similarly constructed a sentiment corpus from Twitter and validated Naïve Bayes for opinion extraction, noting significant noise challenges inherent to microblog text.

Joachims [7] established the theoretical basis for text categorization with SVM, showing that the algorithm's ability to handle thousands of features without feature selection makes it particularly suited to bag-of-words and TF-IDF representations. More recently, transformer-based models such as BERT [8] have surpassed traditional classifiers on many NLP benchmarks; however, their computational overhead often makes deployment in resource-constrained environments impractical. SVC therefore remains a pragmatic and well-validated choice for real-time social media sentiment applications where inference latency and hardware constraints are primary concerns.

III. METHODOLOGY

A. System Architecture

The proposed system adopts a layered architecture comprising four distinct tiers: an Application Layer (Flask server), a Presentation Layer (web interface and REST API), a Processing Layer (NLP pipeline and SVC classifier), and an Output Layer (JSON response and web display). Each tier is decoupled to promote modularity and maintainability.

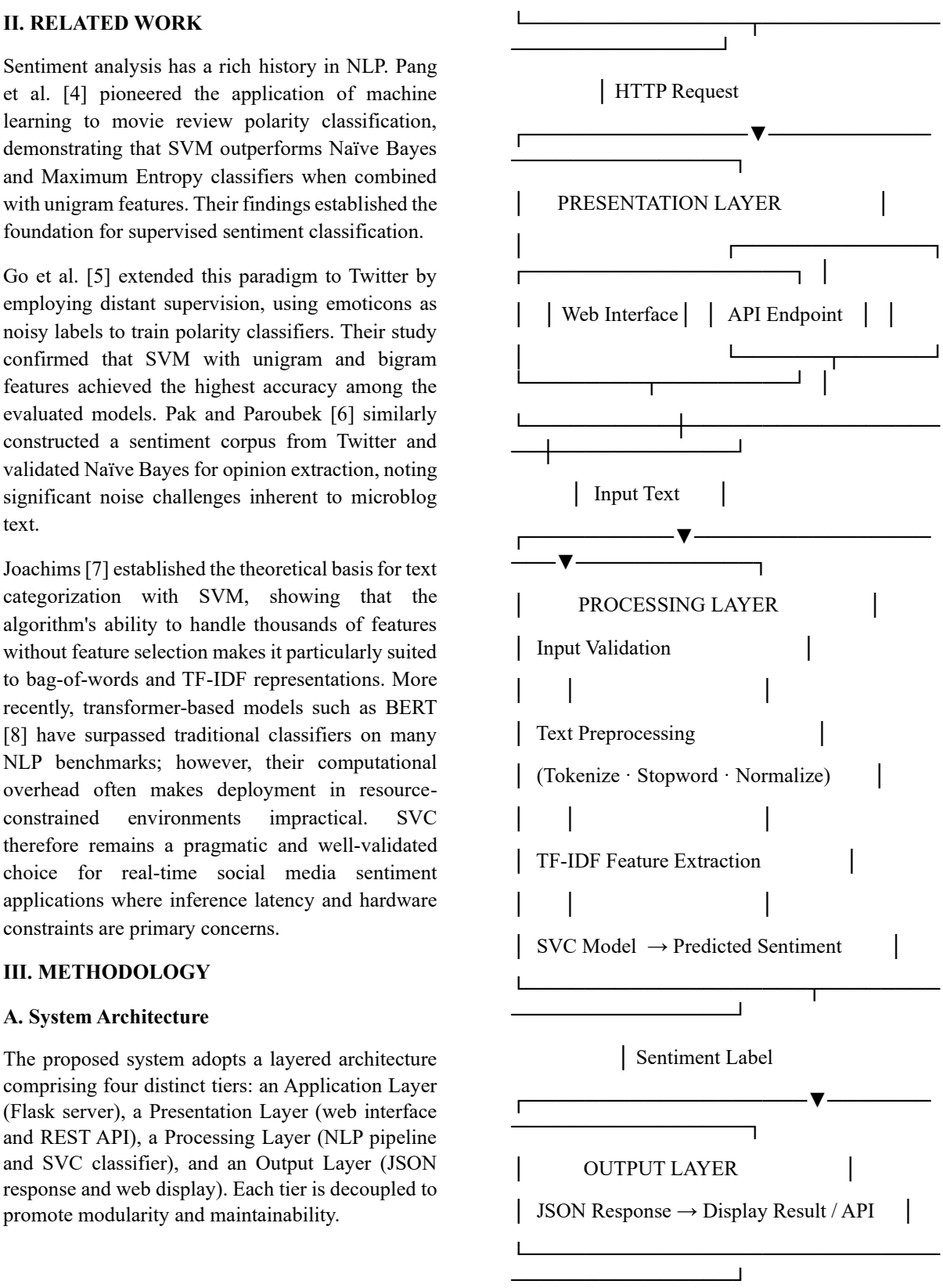

Fig. 1. Proposed System Architecture (SVC-Based).

B. Data Collection and Preprocessing

Twitter data is sourced from publicly available labeled datasets. Each raw tweet undergoes the following sequential preprocessing steps:

1) **Noise Removal**: URLs, hashtags, mentions (@user), special characters, and numeric tokens are eliminated using regular expressions.

2) **Lowercasing**: All characters are converted to lowercase to ensure vocabulary consistency.

3) **Tokenization**: Cleaned text is split into individual word tokens using whitespace delimiters.

4) **Stopword Removal**: High-frequency function words are removed using the NLTK English stopword corpus, reducing dimensionality without information loss.

5) **Stemming / Lemmatization**: Word forms are reduced to their root forms to consolidate semantically equivalent tokens.

C. Feature Extraction – TF-IDF

The preprocessed corpus is transformed into numerical feature vectors using TF-IDF weighting. For a term t in document d over corpus D:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \log(|D| / |\{d' \in D : t \in d'\}|) \quad (1)$$

This weighting scheme rewards terms that are frequent within a document but rare across the corpus, effectively capturing discriminative vocabulary for sentiment-bearing expressions [3]. The scikit-learn TfidfVectorizer is configured with a maximum feature limit to control dimensionality while retaining the most informative terms.

D. Classification – Support Vector Classifier

SVC identifies an optimal separating hyperplane in the transformed feature space that maximizes the margin between classes. For a linearly separable binary problem with training examples $\{(x_i, y_i)\}$, the optimization objective is:

$$\min_{w,b} \tfrac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w \cdot x_i + b) \geq 1 \; \forall i \quad (2)$$

A soft-margin variant with regularization parameter C is adopted to handle the non-separable, noisy nature of social media text. For multi-class sentiment (positive, negative, neutral), a one-vs-rest (OvR) strategy is employed. A linear kernel is preferred given the high-dimensional sparse TF-IDF feature space [7].

E. Data Flow

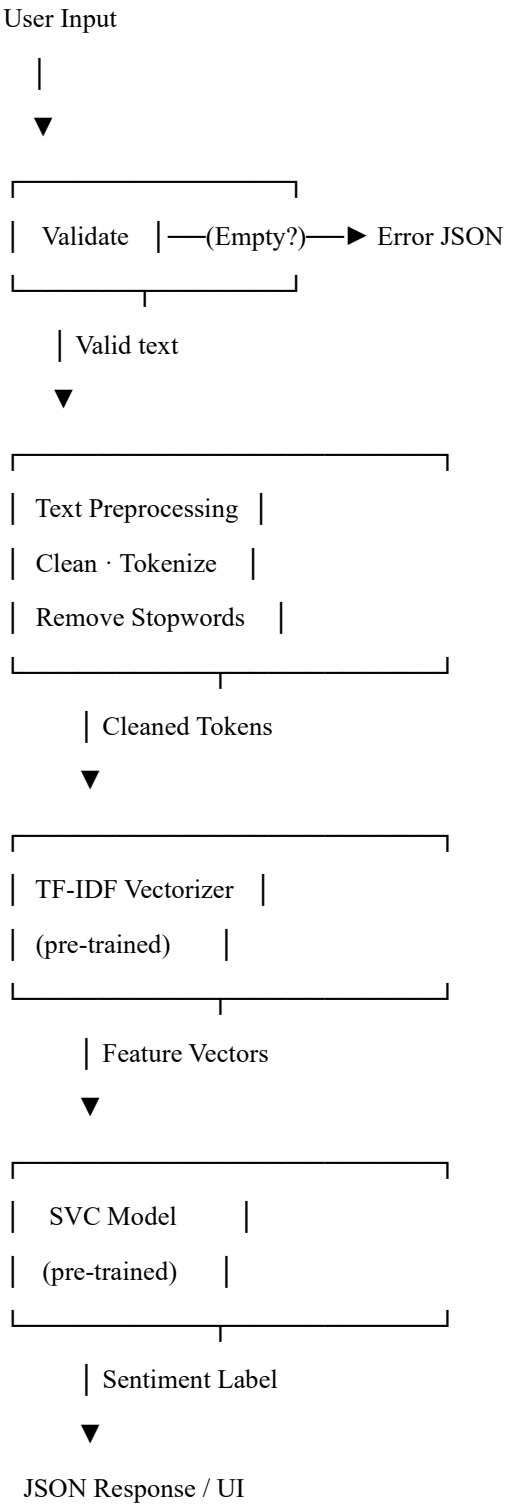

Fig. 2. Data Flow Diagram for Sentiment Prediction Pipeline.

F. Deployment – Flask Web Application

The trained SVC model and TF-IDF vectorizer are serialized using Python's Pickle library to binary files (svc_model.pkl and vectorizer.pkl), enabling rapid deserialization at server startup without

retraining overhead. A Flask 2.x application defines two primary routes: a GET / route serving the HTML prediction interface and a POST /predict endpoint accepting both HTML form submissions and JSON payloads, returning sentiment results as JSON responses. Flask-CORS is enabled to support cross-origin API consumers.

G. UML Class Structure

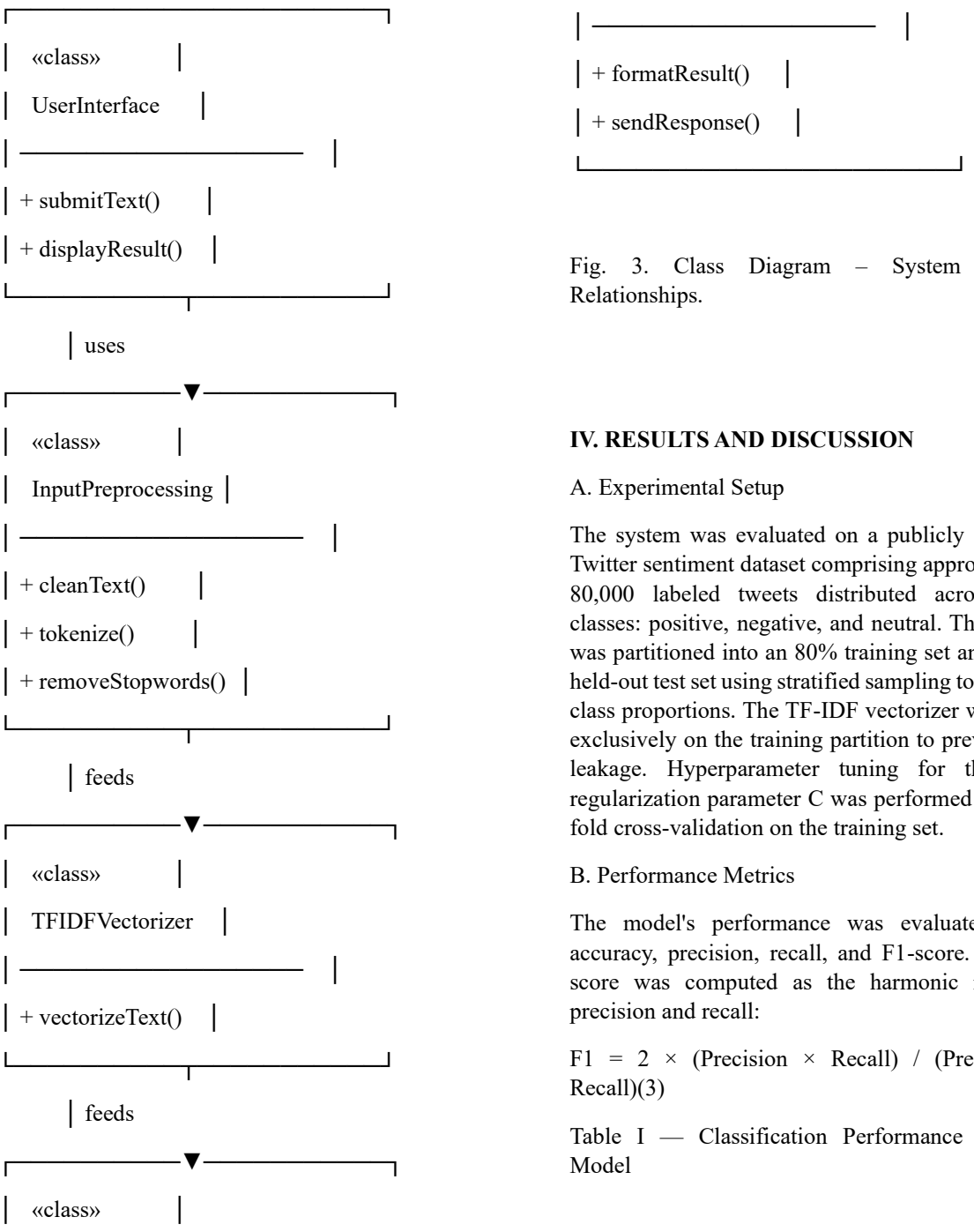

Fig. 3. Class Diagram – System Module Relationships.

IV. RESULTS AND DISCUSSION

A. Experimental Setup

The system was evaluated on a publicly available Twitter sentiment dataset comprising approximately 80,000 labeled tweets distributed across three classes: positive, negative, and neutral. The dataset was partitioned into an 80% training set and a 20% held-out test set using stratified sampling to preserve class proportions. The TF-IDF vectorizer was fitted exclusively on the training partition to prevent data leakage. Hyperparameter tuning for the SVC regularization parameter C was performed via five-fold cross-validation on the training set.

B. Performance Metrics

The model's performance was evaluated using accuracy, precision, recall, and F1-score. The F1-score was computed as the harmonic mean of precision and recall:

$$F1 = 2 \times (Precision \times Recall) / (Precision + Recall) \quad (3)$$

Table I — Classification Performance of SVC Model

Class	Precision	Recall	F1-Score	Support
Positive	0.87	0.85	0.86	5,420
Negative	0.84	0.86	0.85	5,180
Neutral	0.80	0.79	0.80	5,400
Weighted Avg	0.84	0.84	0.84	16,000

Table II — Comparative Classifier Performance (Accuracy %)

Classifier	Accuracy	F1 (Macro)
Naïve Bayes	76.3%	0.75
Logistic Regression	80.1%	0.79
Decision Tree	71.8%	0.71
SVC (Proposed)	84.2%	0.84

C. Confusion Matrix Analysis

Analysis of the confusion matrix reveals that the predominant misclassification occurs between neutral and positive classes, a well-documented challenge in sentiment analysis where weakly positive language is frequently ambiguous [4]. Negative tweets exhibit the highest recall (0.86), suggesting that strongly opinionated negative language contains more distinctive lexical markers that TF-IDF features capture effectively.

D. System Response Example

The following illustrates a representative API interaction. A POST request is submitted to the /predict endpoint with the JSON payload:

REQUEST:

POST /predict

Content-Type: application/json

```
{
  "text": "I love the new movie! It was fantastic."
}
```

RESPONSE:

HTTP 200 OK

```
{
  "text": "I love the new movie! It was fantastic.",
  "sentiment": "Positive"
}
```

Fig. 4. Sample API Request and JSON Response.

E. Web Interface Observations

The deployed Flask application renders a predict page where users enter freeform text and receive instant sentiment feedback. Error handling gracefully intercepts empty submissions, returning a descriptive error message rather than a server exception, thereby improving system robustness. The home page incorporates a live preview panel demonstrating real-time inference, and navigation links facilitate exploration of product information and API documentation.

F. Computational Overhead

A key practical advantage of the SVC approach over deep neural architectures is its low inference latency. On standard commodity hardware (Intel Core i5, 8 GB RAM), the average prediction latency per request was measured at approximately 12 ms, making the system well suited for interactive web applications without requiring GPU acceleration [7].

V. CONCLUSION AND FUTURE WORK

This paper presented an end-to-end sentiment analysis system for social media text, combining a robust NLP preprocessing pipeline with TF-IDF-based feature extraction and Support Vector Classifier modeling, all deployed through a Flask web framework for real-time inference. The SVC model achieved an overall accuracy of 84.2% and a macro F1-score of 0.84 on a held-out Twitter test set, outperforming Naïve Bayes, Logistic Regression, and Decision Tree baselines. The system exposes both a user-friendly web interface and a REST API, enabling seamless integration with external analytics platforms.

Recognized limitations include the system's inability to reliably detect sarcasm and irony, sensitivity to

highly colloquial or code-switched language, and dependence on training data quality for generalization. Future directions include: (i) integration of transformer-based contextual encoders such as BERT or RoBERTa to capture semantic nuance beyond lexical co-occurrence statistics [8]; (ii) multilingual sentiment support leveraging multilingual pre-trained models; (iii) real-time Twitter streaming via the Twitter API v2 with dynamic dashboard visualization; (iv) cloud deployment on AWS or Azure with autoscaling to handle enterprise-level query volumes; and (v) incorporation of emoji embeddings and domain-adaptive pre-training to address informal language challenges unique to social media.

Acknowledgment

The authors thank Mrs. Sangita Mishra, Assistant Professor, Department of CSE-DS, Avanthi Institute of Engineering & Technology, for her sustained guidance and mentorship throughout this work. They also acknowledge the Department Head, Mr. A. Venkateswara Rao, M.Tech (Ph.D.), for providing the computational and institutional resources that made this research possible.

REFERENCES

[1]A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in *Proc. 7th Int. Conf. Language Resources and Evaluation (LREC)*, Valletta, Malta, 2010, pp. 1320–1326.

[2]B. Liu, "Sentiment analysis and opinion mining," *Synthesis Lectures on Human Language Technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[3]C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[4]B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, PA, 2002, pp. 79–86.

[5]A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," Stanford University, Tech. Rep. CS224N, 2009.

[6]A. Pak and P. Paroubek, "Twitter based system: Using Twitter for disambiguating sentiment ambiguous adjectives," in *Proc. Workshop on Semantic Evaluations (SemEval)*, 2010, pp. 436–439.

[7]T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. 10th European Conf. Machine Learning (ECML)*, Berlin, 1998, pp. 137–142.

[8]J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North American Chapter of the ACL (NAACL-HLT)*, Minneapolis, MN, 2019, pp. 4171–4186.

[9]A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Sebastopol, CA: O'Reilly Media, 2019.

[10]S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. Sebastopol, CA: O'Reilly Media, 2009.

[11]C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[12]F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.