Journal of Information Systems Engineering and Management
2024, 9(4s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Analyzing Various Machine Learning Algorithms for Opinion Extraction from Web Text Using AI Across Multiple Datasets

Erugu Krishna[1], Dr. Sonawane Vijay Ramnath[2]

[1]Research Scholar, [2]Research Supervisor

[1,2]Dr. A. P. J. Abdul Kalam University, Indore, India

krishna.cseit@gmail.com, vijaysonawane11@gmail.com

ARTICLE INFO

Received: 05 Nov 2024

Revised: 22 Nov 2024

Accepted: 24 Dec 2024

ABSTRACT

Opinion extraction from web text is essential for understanding public attitudes in e-commerce, news, and social media, yet it remains challenging due to noisy language, short informal messages, and inconsistent sentiment labels. This study proposes a unified AI-driven pipeline for three-class sentiment classification (positive, neutral, negative) across multiple web-text domains. The workflow performs label normalization, missing-value removal, de-duplication, and text cleaning (URL/mention removal, hashtag normalization, and whitespace standardization). Cleaned text is represented using TF–IDF with unigram and bigram features and evaluated using twelve classic machine learning classifiers, with a focus on LinearSVM and Calibrated LinearSVM for robust discrimination and probability-based analysis. Experiments are conducted on three datasets: product reviews, Times of India headlines, and English political tweets. Performance is assessed using accuracy, precision, recall, F1-score, confusion matrices, and OvR ROC/precision–recall curves. On the Times of India dataset, LinearSVM achieves the best accuracy of 0.894, while Calibrated LinearSVM attains a comparable accuracy of 0.893, demonstrating strong and consistent performance for headline sentiment classification. The results indicate that TF–IDF combined with linear margin-based models provides an effective and scalable baseline for multi-domain opinion extraction.

Keywords: Opinion extraction, sentiment analysis, TF–IDF, LinearSVM, calibrated SVM, machine learning, text classification, web text, news headlines, tweets, product reviews.

1. Introduction

The rapid growth of web-based platforms such as social media, online news portals, and e-commerce websites has resulted in an unprecedented volume of user-generated textual data. This data contains valuable opinions, emotions, and attitudes that reflect public perception toward products, services, events, and political or social issues. Extracting meaningful opinions from such large-scale web text, commonly referred to as **opinion extraction or sentiment analysis**, has therefore become a crucial research problem in natural language processing (NLP) and artificial intelligence (AI) [1], [2]. Accurate sentiment analysis enables applications such as market analysis, brand monitoring, recommendation systems, political trend analysis, and decision support systems.

Despite significant progress, opinion extraction from web text remains a challenging task due to several factors. Web text is often noisy, informal, and unstructured, containing spelling variations, abbreviations, hashtags, URLs, and user mentions. Additionally, sentiment expressions may be implicit, sarcastic, or context-dependent, making automated interpretation difficult [3]. These challenges are further amplified when analyzing multiple domains simultaneously, such as product reviews, news headlines, and social media posts, each of which exhibits distinct linguistic characteristics and sentiment distributions [4].

Traditional sentiment analysis approaches relied heavily on lexicon-based methods, which determine sentiment polarity using predefined sentiment dictionaries [5]. While lexicon-based methods are simple and interpretable, they often fail to capture contextual meaning and domain-specific sentiment usage. As a result, machine learning–based approaches have gained prominence, as they learn sentiment patterns directly from data and can generalize better across varying text forms [6]. In particular, classic machine learning classifiers such as Support Vector Machines (SVM), Logistic Regression, Naïve Bayes, and Ridge-based models have demonstrated strong performance when combined with appropriate text representations.

Feature representation plays a central role in the effectiveness of machine learning models for sentiment analysis. Among various techniques, **Term Frequency–Inverse Document Frequency (TF–IDF)** remains one of the most widely used and effective representations for sparse textual data [7]. TF–IDF captures the importance of words relative to both individual documents and the overall corpus, enabling linear classifiers to separate sentiment classes efficiently. Several studies have shown that TF–IDF combined with linear SVM variants provides strong baseline performance for sentiment classification tasks across different datasets [8], [9].

Recent research has also explored deep learning and transformer-based models for opinion extraction, achieving impressive results in many settings [10]. However, such models often require large annotated datasets, extensive computational resources, and careful hyperparameter tuning. In contrast, classic machine learning models offer advantages in terms of interpretability, scalability, and computational efficiency, making them suitable for real-world and resource-constrained environments. Moreover, calibrated variants of linear classifiers can provide probability estimates that enable advanced evaluation metrics such as ROC and precision–recall analysis [11].

Motivated by these observations, this paper presents a comprehensive analysis of **various machine learning algorithms for opinion extraction from web text using AI across multiple datasets**. The proposed work applies a unified preprocessing and feature extraction pipeline to three heterogeneous datasets: product reviews, Times of India news headlines, and political tweets. Twelve classic machine learning models are evaluated using TF–IDF features, with particular emphasis on **LinearSVM** and **Calibrated LinearSVM**, which demonstrate consistently strong performance across domains. The evaluation employs multiple metrics, including accuracy, precision, recall, F1-score, confusion matrices, and ROC/precision–recall curves, to provide a detailed and fair comparison.

Key Contributions

The main contributions of this work are summarized as follows:

- A **unified sentiment analysis pipeline** is proposed for multi-domain web text, including reviews, news headlines, and social media content.
- A **comprehensive comparison of twelve TF–IDF–based machine learning models** is conducted under identical preprocessing and evaluation settings.
- The effectiveness of **LinearSVM and Calibrated LinearSVM** for three-class sentiment classification is empirically demonstrated.
- Extensive evaluation using **accuracy, precision, recall, F1-score, confusion matrices, and ROC/PR curves** provides deeper insights into model behavior.
- The study highlights the practicality of **classic machine learning approaches** as robust and scalable baselines for opinion extraction across diverse web-text domains.

2. Literature review

Obiedat et al. (2021), Aspect-based sentiment analysis (ABSA) for Arabic has gained interest as comments grow on social media and e-commerce. Yet Arabic ABSA remains difficult due to NLP complexity and the limited availability of annotated corpora. This systematic review covers methods, techniques, and datasets used for Arabic ABSA, analyzing 21 studies published between 2015–2021. Key findings include a shortage of reusable annotated datasets and limited domain coverage in existing datasets. The review is intended to guide researchers toward building stronger models and resources. [1]

Xu et al. (2019), Dictionary-based sentiment analysis often suffers from limited word coverage and difficulty handling polysemous words whose polarity changes by domain/context. This paper builds an extended sentiment dictionary that includes basic sentiment terms, domain-specific words, and polysemic words. A Naive Bayes classifier first determines the text's domain, then assigns the appropriate polarity value for polysemic words in that domain. Using the extended dictionary plus sentiment-scoring rules, the method achieves feasible and improved sentiment recognition for comment texts. [2]

Liu et al. (2022), Rule-based lexicons and machine-learning vector classification are common in sentiment analysis but both have weaknesses (rigid rules, weak prominence of sentiment cues). This paper proposes a weight-distribution approach that combines the two, producing sentence vectors that emphasize sentiment-bearing words while preserving broader text information. Experiments report sentiment classification accuracy up to 82.1%, outperforming pure lexicon rules and TF-IDF weighting baselines by notable margins. The approach aims to balance interpretability with stronger discriminative features. [3]

Wang et al. (2020), Twitter sentiment analysis is hard because tweets are short and ambiguous, and many methods use only text. This work argues that sentiment diffusion patterns (how sentiment spreads) correlate with tweet polarity. It studies "sentiment reversal" in diffusion and then proposes SentiDiff, an iterative algorithm that fuses textual signals with diffusion features to predict sentiment. Reported experiments on real datasets show PR-AUC improvements of about 5.09–8.38% versus state-of-the-art text-only baselines, suggesting network dynamics add useful information. [4]

Poria et al. (2023), This article reviews sentiment analysis as a field after nearly two decades of development and broad commercial adoption. While polarity classification and benchmark datasets appear mature, the paper challenges the idea that sentiment analysis is "solved." It identifies shortcomings and under-explored issues required for true sentiment understanding, reflects on the breakthroughs that drove the field's relevance, and proposes future research directions targeting deeper, more comprehensive sentiment modeling beyond simple polarity. [5]

Yang et al. (2022), Existing ABSA methods often learn sentiment features by modeling dependencies between aspect terms and context, but may ignore external affective commonsense knowledge that could enhance interaction modeling. This paper proposes AKM-IGCN, a knowledge-aware model that augments interactive GCNs with affective knowledge and uses multi-head self-attention to capture richer syntactic/semantic interactions. It targets Chinese-oriented ABSA while also supporting English, evaluated on four Chinese datasets and six English benchmarks. Reported results show the model matches or outperforms prior state-of-the-art approaches. [6]

Li et al. (2020), Danmaku (live on-screen comments) captures viewers' real-time reactions and provides "emotional timing" aligned with video moments, but standard sentiment classifiers don't fit its short, fast, context-heavy text. This paper builds a danmaku sentiment dictionary and proposes sentiment analysis using a dictionary plus Naive Bayes. It extracts emotional signals, classifies sentiment, visualizes results, and derives time distributions across seven sentiment dimensions. A weighting scheme is also used to determine polarity. Experiments show strong impact on sentiment scoring and polarity detection. [7]

Phan et al. (2020), Tweet sentiment analysis is important due to massive Twitter content and its use in decision support and recommendation systems. Prior feature-ensemble methods often model syntax but miss sentiment context, word position, and fuzzy-sentiment phrases. This study proposes an ensemble feature model for tweets with fuzzy sentiment, combining lexical, word-type, semantic, positional, and polarity-aware features. Tests on real data show improved performance, particularly in F1 score, indicating that modeling contextual sentiment and positional cues helps classify ambiguous tweets more accurately. [8]

Kasri et al. (2022), Standard word embeddings can blur sentiment because words with similar contexts may have opposite polarity, hurting sentiment classification. This paper proposes Continuous Sentiment Contextualized Vectors (CSCV), which learns sentiment-aware embeddings using surrounding context (CBOW) plus sentiment lexicons to inject polarity signals. It then combines CSCV vectors with existing pretrained embeddings using PCA to improve overall representation. Experiments report that CSCV-enhanced vectors can boost any pretrained embeddings, reduce “opposite polarity neighbors,” and improve sentiment classification results. [9]

Schouten et al. (2016), This survey reviews aspect-level sentiment analysis, aiming to detect sentiment toward entities or their aspects for fine-grained insights. It summarizes strong progress in finding sentiment targets (entities/aspects) and associated polarity, and categorizes solutions by whether they handle aspect detection, sentiment classification, or both. The survey also groups approaches by algorithm type and reports each study’s performance. It calls for standardized evaluation and shared datasets to enable fair comparison, and identifies concept-centric, semantically rich aspect-level methods as promising future directions. [10]

Durga et al. (2023), This paper proposes an integrated sentiment analysis framework combining large pretrained models and deep classifiers for e-commerce/social text. It uses BERT-large-cased (24 layers, 340M parameters) with fine-tuning (SGD) plus preprocessing, then applies BoW/Word2Vec feature extraction. The classification component is “deep sentiment analysis” with aspect-and-priority modeling via a Decision-based RNN. Experiments on Kaggle Twitter/Restaurant/Laptop datasets evaluate performance via confusion matrices and report improved outputs compared to existing methods. [11]

Tang et al. (2016), Traditional embeddings ignore sentiment signals, placing antonyms like good and bad close in vector space. This work proposes sentiment-specific word embeddings that encode both context and sentiment evidence, trained with neural networks and tailored loss functions on large corpora automatically labeled with sentiment cues (e.g., emoticons). The resulting embedding space keeps semantically similar words close while favoring same-polarity neighbors. Experiments show these embeddings outperform context-only embeddings across tasks like word-level sentiment, sentence classification, and sentiment lexicon building. [12]

Nazir et al. (2022), With rising social-media feedback, aspect-based sentiment analysis now focuses not only on extracting aspects and classifying their sentiments, but also on how sentiments evolve over time. This survey highlights issues such as aspect extraction, mapping relations among aspects, modeling dependencies and contextual-semantic interactions, and predicting sentiment evolution dynamics. It reviews recent work grouped by contributions to aspect extraction, aspect sentiment analysis, or sentiment evolution, and reports quantitative performance where available. It concludes with critical future research directions for improving aspect-level sentiment accuracy. [13] Wu et al. (2019), Chinese microblog sentiment analysis using dictionaries is challenging due to limited sentiment-word coverage. This paper proposes constructing multiple dictionaries (base sentiment, emoji, and other related dictionaries), including a novel “new word” sentiment dictionary for microblogs. It also introduces semantic rule sets (inter-sentence and sentence-pattern rules) and an algorithm that computes sentiment from complex sentences to clauses to words, integrating emoji signals. Experiments show improved classification into positive, negative, and neutral microblogs. [14] Wang et al. (2021), Existing sentiment-embedding approaches often inject lexicon polarity into word vectors but fail to capture context-dependent sentiment for the same word. This paper proposes using "sentiment concepts" to select the optimal concept for a word given its context (via Microsoft Concept Graph), then retrieve sentiment intensity from a multi-semantics lexicon constructed by the authors. It combines two refined embedding methods to produce richer representations. Tests on six datasets show the proposed concept-based embeddings improve sentiment analysis compared with traditional and earlier sentiment-embedding baselines. [15] Kim et al. (2021), Online review ratings and volume are common sentiment proxies but suffer from extremity bias (ratings skewed by very happy/unhappy reviewers) and ambiguity (volume may rise for many reasons). This article proposes text-mining alternatives and finds sentiment scores may be less prone to extremity bias than ratings: sentiment scores tend to be more normally distributed while ratings skew to extremes. It also suggests combining sentiment scores with review length to better capture customer enthusiasm and interpret "word of mouth" beyond star ratings. [16] Smetanin et al. (2020), Sentiment analysis has been widely applied to English content, but Russian-language applications remain underexplored. This survey reviews applied sentiment analysis studies on Russian texts, focusing on use cases rather than classification accuracy. Studies are systematically categorized by data source, application purpose, methods, outcomes, and limitations. The paper outlines challenges, proposes a future research agenda, and compiles publicly available Russian sentiment datasets to support researchers in dataset selection and methodological improvements. [17]

He et al. (2022), Aspect-Based Sentiment Analysis (ABSA) predicts sentiment polarity toward specific aspects and is a fine-grained NLP task. Existing methods mainly rely on local context and often ignore global contextual dependencies, with limited work on Chinese and multilingual ABSA. This paper proposes LGCF, a multilingual model that jointly learns local and global context–aspect correlations. Experiments on multiple Chinese and English benchmark datasets show LGCF outperforms state-of-the-art models, with ablation studies confirming each component's effectiveness. [18] Yang et al. (2020), To enhance sentiment analysis of Chinese e-commerce reviews, this paper introduces SLCABG, a hybrid model combining sentiment lexicons with CNN and attention-based BiGRU. The lexicon strengthens sentiment cues, CNN extracts key features, BiGRU captures contextual dependencies, and attention weights important information. Trained on over 100,000 cleaned book reviews from Dangdang.com, experimental results show improved sentiment classification performance, demonstrating the effectiveness of integrating lexicon knowledge with deep learning. [19] Zhang et al. (2023), Aspect-level sentiment analysis remains challenging due to poor cross-domain transfer and weak modeling of aspect–sentiment word relations. This paper proposes Efficient Adaptive Transfer Network (EATN), which incorporates domain adaptation through a Domain Adaptation Module (DAM) and multiple-kernel learning to reduce domain discrepancy. An aspect-oriented multi-head attention mechanism captures direct aspect–sentiment associations. Experiments on six public datasets across domains show EATN achieves strong generalization and outperforms existing methods. [20] Fu et al. (2018), Traditional LSTM-based sentiment models rely heavily on word embeddings that encode semantic but not sentiment information. To address this, this paper proposes a lexicon-enhanced LSTM, which integrates sentiment embeddings learned from sentiment lexicons with standard word embeddings. A novel attention mechanism captures global sentiment without requiring explicit targets. Experiments on English and Chinese datasets show the proposed model achieves comparable or superior performance to existing sentiment analysis models. [21]

Kastrati et al. (2020), Analyzing student feedback manually is impractical for large-scale online education platforms such as MOOCs. This paper proposes a framework using aspect-level sentiment analysis with weak supervision to automatically analyze student reviews. Weakly annotated MOOC aspects are propagated to unlabeled data, reducing dependence on costly manual labels. Experiments on large Coursera and classroom datasets show strong performance in aspect identification and sentiment classification, outperforming fully supervised approaches. [22]

Al-Moslmi et al. (2017), Cross-domain sentiment analysis is challenging due to the lack of universally annotated datasets. This systematic review analyzes studies published between 2010–2016 that address cross-domain sentiment classification. It compares techniques such as domain adaptation, feature alignment, and transfer learning, concluding that no single method fully solves the problem. The review serves as a consolidated resource to guide researchers in developing more robust and accurate cross-domain sentiment analysis approaches. [23] Deng et al. (2019), Sentiment lexicon construction using deep learning often ignores word importance in determining document polarity. This paper proposes Sparse Self-Attention LSTM (SSALSTM) to capture word importance via self-attention with L1 regularization, ensuring sparsity. The learned sentiment-aware embeddings are used to build large-scale Twitter sentiment lexicons. Experiments on SemEval 2013–2016 datasets show the generated lexicons achieve state-of-the-art performance in both supervised and unsupervised sentiment classification. [24] Huang et al. (2022), Lexicon-based methods ignore context, while supervised models often overlook sentiment-word knowledge. To address this, this paper proposes SentiCNN, which combines contextual information from word embeddings with sentiment cues from lexicons. A Highway Network adaptively fuses both information types, and lexicon-based attention mechanisms (LBAMs) highlight key sentiment indicators. Experiments on benchmark datasets confirm that sentiment words, attention, and hybrid modeling significantly improve sentiment classification accuracy. [25] Hasan, Ali et al. (2018), This paper focuses on election-related opinion mining from Twitter and argues that, even though many sentiment-analysis techniques/tools have been used in political contexts, there is still a need for a stronger "state-of-the-art" approach. The authors propose a hybrid sentiment-analyzer approach that combines sentiment analysis with supervised machine-learning, and they explicitly compare Naïve Bayes and Support Vector Machines (SVM) for analyzing political views from Twitter accounts. [26] Souma, Wataru et al (2019), This paper studies whether historical news sentiments (derived from market reactions) can help forecast financial news sentiment. It defines news sentiment using intraday stock-price behavior: if the averaged stock return right after a news release is positive/negative, the corresponding news is labeled positive/negative. The method uses GloVe word vectors (trained on Wikipedia 2014 + Gigaword 5) as inputs and trains an RNN with LSTM units on Thomson Reuters News Archive (TRNA) from 2003–2012, then tests on 2013; importantly, it reports better forecasting when training examples are chosen hierarchically (using the most strongly positive/negative polarity-scored news) rather than randomly. [27]

3. Proposed work

3.1 Proposed architecture

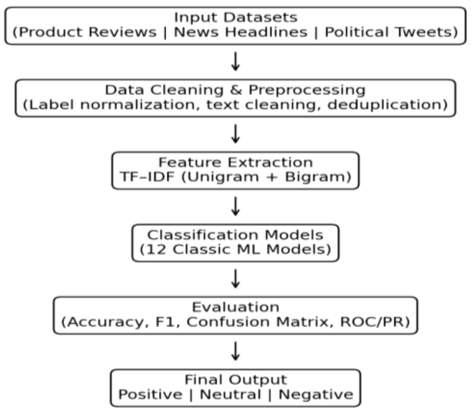
Input Datasets
(Product Reviews | News Headlines | Political Tweets)
Data Cleaning & Preprocessing
(Label normalization, text cleaning, deduplication)
Feature Extraction
TF–IDF (Unigram + Bigram)
Classification Models
(12 Classic ML Models)
Evaluation
(Accuracy, F1, Confusion Matrix, ROC/PR)
Final Output
Positive | Neutral | Negative

Figure 1. The proposed architecture

Journal of Information Systems Engineering and Management
2024, 9(4s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

The figure 1 proposed architecture follows a sequential pipeline that begins with three input datasets product reviews, news headlines, and political tweets which are first subjected to data cleaning and preprocessing to normalize sentiment labels, remove noise, eliminate duplicates, and standardize textual content. The cleaned text is then transformed into numerical representations using TF–IDF with unigram and bigram features, providing a consistent and informative feature space for learning. These features are passed to a set of twelve classic machine learning classifiers, including linear, probabilistic, and margin-based models, which are trained to perform three-class sentiment classification. Model performance is subsequently evaluated using accuracy, precision, recall, F1-score, confusion matrix, and ROC/precision–recall curves to ensure both overall and class-wise robustness. Finally, the system outputs a sentiment label positive, neutral, or negative for each input text, completing the end-to-end sentiment classification process.

3.2 Proposed algorithm

Algorithm 1: Three-Class Sentiment Classification Using TF–IDF and Classic ML Models

Input: Datasets $\mathcal{D} = \{\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \mathcal{D}^{(3)}\}$, class set $\mathcal{C}$, text field x, label field y
Output: Best model $M\backslash^{*}$per dataset and predicted labels $\hat{y} \in \mathcal{C}$

Step 1: Class Definition and Dataset Representation

1.1. Define the three sentiment classes as

$$\mathcal{C} = \{\text{positive,neutral,negative}\} \qquad (1)$$

1.2. For each dataset $\mathcal{D}^{(k)} \in \mathcal{D}$, represent samples as (x_i, y_i), $i = 1, \dots, N$.

Step 2: Label Normalization and Filtering

2.1. Normalize the sentiment label using

$$\tilde{y}_i = f_{\text{norm}}(y_i) \qquad (2)$$

2.2. Apply noise-handling mappings:
{unlabled,unlabeled,suggestion} → neutral.
2.3. Remove samples with missing $x_i$or $y_i$and keep only labels in $\mathcal{C}$.

Step 3: Text Cleaning and De-duplication

3.1. Clean each text instance using

$$\tilde{x}_i = f_{\text{clean}}(x_i) \qquad (3)$$

3.2. The cleaning function $f_{\text{clean}}(\cdot)$performs: URL removal, @mention removal, hashtag normalization ($\#w \to w$), non-alphanumeric removal, and whitespace normalization.

3.3. Remove duplicates based on $(\tilde{x}_i, \tilde{y}_i)$to control data leakage.

Step 4: Train–Test Split

4.1. Split the processed dataset into stratified training and testing sets:
$\mathcal{T}_{\text{train}}, \mathcal{T}_{\text{test}}$.

Step 5: TF–IDF Feature Extraction

5.1. Convert cleaned text to TF–IDF vectors:

$$\mathbf{v}_i = \phi_{\text{tfidf}}(\tilde{x}_i) \in \mathbb{R}^d \qquad (4)$$

5.2. Use unigram and bigram features (1, 2), minimum document frequency threshold, and a maximum feature cap d.

Step 6: Model Set Construction (Explicit 12 Methods)

6.1. Define the classifier set

$$\mathcal{M} = \{M_1, M_2, \dots, M_{12}\} \qquad (5)$$

6.2. Instantiate $\mathcal{M}$ using the following TF–IDF-based models:

- M_1: **LinearSVM** — LinearSVC()
- M_2: **LogisticRegression** — LogisticRegression(max_iter= …)
- M_3: **SGD-LogReg** — SGDClassifier(loss="log_loss")
- M_4: **SGD-LinearSVM** — SGDClassifier(loss="hinge")
- M_5: **RidgeClassifier** — RidgeClassifier()
- M_6: **PassiveAggressive** — PassiveAggressiveClassifier()
- M_7: **Perceptron** — Perceptron()
- M_8: **MultinomialNB** — MultinomialNB()
- M_9: **ComplementNB** — ComplementNB()
- M_{10}: **BernoulliNB** — BernoulliNB()
- M_{11}: **Calibrated LinearSVM** — CalibratedClassifierCV(LinearSVC(), method="sigmoid")
- M_{12}: **Calibrated Ridge** — CalibratedClassifierCV(RidgeClassifier(), method="sigmoid")

Step 7: Model Training and Prediction

7.1. For each $M_j \in \mathcal{M}$, train using

$$M_j \leftarrow \mathrm{fit}(M_j, \mathbf{v}_{\text{train}}, \tilde{y}_{\text{train}}) \qquad (12)$$

7.2. Predict sentiment on the test set using

$$\hat{y}_{\text{test}} = M_j(\mathbf{v}_{\text{test}}) \qquad (13)$$

7.3. The generic prediction rule is expressed as

$$\hat{y}_i = M(\mathbf{v}_i) \qquad (6)$$

Step 8: Evaluation Metrics and Diagnostic Plots

8.1. Compute confusion matrix:

$$\mathbf{CM}(a, b) = \sum_{i=1}^{N} \mathbb{I}(y_i = a \wedge \hat{y}_i = b) \qquad (7)$$

8.2. Compute Accuracy:

$$\mathrm{Acc} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\hat{y}_i = y_i) \qquad (11)$$

8.3. Compute Precision/Recall and F1-score per class:

$$P_c = \frac{TP_c}{TP_c + FP_c}, R_c = \frac{TP_c}{TP_c + FN_c} \qquad (8)$$

$$F1_c = \frac{2P_cR_c}{P_c + R_c} \qquad (9)$$

8.4. Compute macro-averaged F1-score:

$$F1_{\text{macro}} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} F1_c \qquad (10)$$

8.5. For calibrated/probabilistic classifiers (M_{11}, $M_{12}$and others with scores), compute OvR ROC and OvR Precision–Recall curves.

Step 9: Error Analysis (Length-Based)

9.1. Define correct and wrong prediction sets:

$$\mathcal{I}_{\text{correct}} = \{i: \hat{y}_i = y_i\}, \mathcal{I}_{\text{wrong}} = \{i: \hat{y}_i \neq y_i\} \qquad (14)$$

9.2. Plot text-length distributions for $\mathcal{I}_{\text{correct}}$vs. $\mathcal{I}_{\text{wrong}}$.

Step 10: Best Model Selection

10.1. Select the best model per dataset via

$$M^* = \arg\max_{M_j\in\mathcal{M}} F1_{\text{macro}}(M_j) \qquad (15)$$

10.2. Output $M^*$and the final sentiment predictions $\hat{y} \in \mathcal{C}$.

The three-class sentiment objective is formally defined by the label set $\mathcal{C} = \{\text{positive,neutral,negative}\}$in (1), which fixes the target space for all datasets and ensures that every sample is ultimately mapped into one of these three categories. To standardize raw input text, each instance $x_i$is converted into a cleaned representation $\tilde{x}_i$using the text-cleaning transformation $\tilde{x}_i = f_{\text{clean}}(x_i)$in (2); this operation denotes the complete preprocessing routine (e.g., removing URLs and mentions, normalizing hashtags, removing non-alphanumeric characters, and normalizing whitespace) that reduces noise and vocabulary sparsity before feature construction. In parallel, the original sentiment label $y_i$is normalized into $\tilde{y}_i$using $\tilde{y}_i = f_{\text{norm}}(y_i)$in (3), which represents label lowercasing and noise-handling mappings (such as mapping unlabled, unlabeled, and suggestionto neutral) so that all labels match the unified class set in (1).

After normalization, each cleaned text $\tilde{x}_i$is transformed into a numerical feature vector through TF–IDF vectorization, expressed as $\mathbf{v}_i = \phi_{\text{tfidf}}(\tilde{x}_i) \in \mathbb{R}^d$in (4). This equation states that the textual sample is embedded into a d-dimensional sparse vector space where feature weights reflect term importance, and the use of unigram–bigram configurations improves representation of sentiment-bearing phrases. The learning stage evaluates multiple candidate classifiers denoted by the model set $\mathcal{M} = \{M_1, M_2, \ldots, M_{12}\}$in (5), which explicitly captures the 12 TF–IDF-based models (LinearSVM, Logistic Regression, SGD variants, Ridge, Passive Aggressive, Perceptron, Naïve Bayes variants, and calibrated models). For any chosen model $M \in \mathcal{M}$, the inference process is formalized by $\hat{y}_i = M(\mathbf{v}_i)$in (6), which states that the classifier maps the TF–IDF vector $\mathbf{v}_i$to a predicted sentiment label $\hat{y}_i \in \mathcal{C}$.

To quantify classification performance, the confusion matrix is defined by $\mathbf{CM}(a,b) = \sum_{i=1}^{N} \mathbb{I}(y_i = a \wedge \hat{y}_i = b)$in (7), where each entry counts how often true class $a$is predicted as class b, enabling detailed analysis of which sentiments are most frequently confused (e.g., neutral vs. positive). From these counts, class-wise precision and recall are computed using $P_c = \frac{TP_c}{TP_c+FP_c}$and $R_c = \frac{TP_c}{TP_c+FN_c}$in (8), where precision measures the reliability of predictions for class $c$and recall measures how completely the classifier retrieves true instances of class c. The harmonic balance between precision and recall is captured by the per-class F1-score $F1_c = \frac{2P_cR_c}{P_c+R_c}$in (9), which penalizes models that achieve high precision but poor recall (or vice versa). To avoid majority-class dominance and ensure fair evaluation across positive, neutral, and negative sentiments, the macro-averaged performance is measured by $F1_{\text{macro}} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} F\,1_c$in (10), which equally weights each class F1-score regardless of class frequency. Overall correctness is additionally summarized through accuracy $\text{Acc} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(\hat{y}_i = y_i)$in (11), representing the proportion of samples whose predicted label matches the ground truth.

Model learning and prediction on held-out samples are captured in two operational equations: training is represented by $M_j \leftarrow \text{fit}(M_j, \mathbf{v}_{\text{train}}, \tilde{y}_{\text{train}})$in (12), indicating that each classifier $M_j$is estimated from the training TF–IDF vectors and their normalized labels; testing is represented by $\hat{y}_{\text{test}} = M_j(\mathbf{v}_{\text{test}})$in (13), indicating that the trained classifier produces sentiment predictions for unseen test vectors. Beyond aggregate metrics, the algorithm incorporates diagnostic error analysis by separating correctly and incorrectly classified instances using $\mathcal{I}_{\text{correct}} = \{i: \hat{y}_i = y_i\}$and $\mathcal{I}_{\text{wrong}} = \{i: \hat{y}_i \neq y_i\}$in (14), which enables targeted inspection of failure patterns (for example, comparing text-length distributions or identifying ambiguous samples). Finally, the best-performing classifier is selected using $M^* = \arg\max_{M_j\in\mathcal{M}} F1_{\text{macro}}(M_j)$in (15), which chooses the model that maximizes balanced three-class performance, and the resulting $M^*$is used to output final predictions $\hat{y} \in \mathcal{C}$for each dataset under the unified sentiment classification framework.

Algorithm 1: Three-Class Sentiment Classification Framework

Step 1: Dataset Initialization

Step 1.1: Select a dataset $\mathcal{D}^{(k)} \in \{\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \mathcal{D}^{(3)}\}$, where the datasets correspond to Product Reviews, News Headlines, and Political Tweets, respectively.
Step 1.2: Represent the dataset as labeled samples (x_i, y_i), where $x_i$denotes the raw text and $y_i$denotes the associated sentiment label.
Step 1.3: Define the target sentiment space as $\mathcal{C} = \{\text{positive,neutral,negative}\}$according to Eq. (1).

Step 2: Sentiment Label Normalization

Step 2.1: Convert all sentiment labels to lowercase and apply the normalization function $\tilde{y}_i = f_{\text{norm}}(y_i)$as defined in Eq. (3).
Step 2.2: Map noisy or inconsistent labels (e.g., *unlabled*, *unlabeled*, *suggestion*) to the neutralclass.
Step 2.3: Remove all samples whose labels are not included in the class set $\mathcal{C}$.
Step 2.4: Drop samples containing missing text or sentiment labels.

Step 3: Text Cleaning and Standardization

Step 3.1: Apply the text cleaning function $\tilde{x}_i = f_{\text{clean}}(x_i)$as defined in Eq. (2).
Step 3.2: Perform URL removal, @mention removal, hashtag normalization ($\#w \rightarrow w$), non-alphanumeric filtering, and whitespace normalization.
Step 3.3: Remove duplicate records based on identical $(\tilde{x}_i, \tilde{y}_i)$pairs to prevent data leakage.

Step 4: Dataset Partitioning

Step 4.1: Split the cleaned dataset into training and testing subsets $\mathcal{T}_{\text{train}}$and $\mathcal{T}_{\text{test}}$.
Step 4.2: Apply stratified sampling to preserve class distribution across the two subsets.

Step 5: Feature Extraction Using TF–IDF

Step 5.1: Convert each cleaned text instance $\tilde{x}_i$into a TF–IDF feature vector $\mathbf{v}_i$using Eq. (4).
Step 5.2: Configure TF–IDF with unigram and bigram features, a minimum document frequency threshold, and a maximum feature limit d.

Step 6: Classifier Set Definition

Step 6.1: Define the model set $\mathcal{M} = \{M_1, M_2, \ldots, M_{12}\}$as given in Eq. (5).
Step 6.2: Instantiate the classifiers:

- LinearSVM, Logistic Regression, SGD-LogReg, SGD-LinearSVM
- Ridge Classifier, Passive Aggressive, Perceptron
- MultinomialNB, ComplementNB, BernoulliNB
- Calibrated LinearSVM, Calibrated Ridge

Step 7: Model Training

Step 7.1: For each classifier $M_j \in \mathcal{M}$, train the model using TF–IDF vectors from $\mathcal{T}_{\text{train}}$according to Eq. (12).
Step 7.2: Optimize model parameters using the training labels $\tilde{y}_{\text{train}}$.

Step 8: Sentiment Prediction

Step 8.1: Apply the trained classifier $M_j$to the test TF–IDF vectors $\mathbf{v}_{\text{test}}$.
Step 8.2: Generate predicted labels $\hat{y}_{\text{test}}$using Eq. (13).
Step 8.3: Represent the general prediction rule as $\hat{y}_i = M(\mathbf{v}_i)$from Eq. (6).

Step 9: Performance Evaluation

Step 9.1: Compute the confusion matrix using Eq. (7).
Step 9.2: Compute overall classification accuracy using Eq. (11).
Step 9.3: Compute class-wise precision and recall using Eq. (8).
Step 9.4: Compute class-wise F1-scores using Eq. (9).
Step 9.5: Compute macro-averaged F1-score using Eq. (10).
Step 9.6: For calibrated models, generate One-vs-Rest ROC and Precision–Recall curves.

Step 10: Error Analysis

Step 10.1: Identify correctly and incorrectly classified instances using Eq. (14).
Step 10.2: Analyze error patterns by comparing text-length distributions for correct and incorrect predictions.

Step 11: Model Selection and Final Output

Step 11.1: Select the best classifier using the macro-F1 criterion defined in Eq. (15).
Step 11.2: Output the selected model $M^{\backslash *}$and the final sentiment predictions $\hat{y} \in \mathcal{C}$for the dataset.

3.3 Algorithm: Three-Dataset Sentiment Classification Using LinearSVM and Calibrated LinearSVM (TF–IDF)

Step 1: Load the Datasets

1.1. Load the three datasets: **Product Reviews**, **Times of India Headlines**, and **Political Tweets**.
1.2. For each dataset, identify the required columns:

- **Text column** (Review/Headline/Tweet)
- **Sentiment label column**

Step 2: Clean and Normalize Sentiment Labels

2.1. Convert all sentiment labels to lowercase.
2.2. Replace noisy labels such as **unlabled**, **unlabeled**, and **suggestion** with **neutral**.

2.3. Keep only three sentiment classes: **positive**, **negative**, and **neutral**.
2.4. Remove rows with missing text or missing sentiment labels.

Step 3: Text Preprocessing (Cleaning)

3.1. Remove URLs from the text.
3.2. Remove user mentions (e.g., @name).
3.3. Convert hashtags into plain words (e.g., #happy → happy).
3.4. Remove special characters and non-alphanumeric symbols.
3.5. Remove extra spaces and normalize whitespace.
3.6. Create a cleaned text column for model training.

Step 4: Remove Duplicate Samples

4.1. Remove duplicate rows using the combination of **cleaned text** and **sentiment label** to reduce leakage and bias.

Step 5: Split the Data

5.1. Split each dataset into **training** and **testing** parts (e.g., 80% training, 20% testing).
5.2. Use stratified splitting so that all sentiment classes remain balanced in both sets.

Step 6: Feature Extraction Using TF–IDF

6.1. Fit the TF–IDF vectorizer using the training text only.
6.2. Use unigram and bigram features (1-gram and 2-gram).
6.3. Apply minimum document frequency filtering to remove rare terms.
6.4. Transform both training and testing text into TF–IDF vectors.

Step 7: Train Model 1 — LinearSVM

7.1. Train **LinearSVM (LinearSVC)** using TF–IDF vectors from the training set.
7.2. Predict sentiment labels on the test set.
7.3. Evaluate performance using:

- Accuracy
- Precision, Recall, F1-score (for positive/neutral/negative)
- Confusion matrix

Step 8: Train Model 2 — Calibrated LinearSVM

8.1. Train **Calibrated LinearSVM**, where LinearSVC is wrapped with probability calibration (sigmoid).
8.2. Predict sentiment labels on the test set.
8.3. Evaluate performance using:

- Accuracy
- Precision, Recall, F1-score
- Confusion matrix
- ROC curve (One-vs-Rest)
- Precision–Recall curve (One-vs-Rest)

Step 9: Compare Results Across All Three Datasets

9.1. Repeat Steps 2–8 for each dataset.
9.2. Compare LinearSVM vs Calibrated LinearSVM across:

- Product Reviews

- Times of India Headlines
- Political Tweets

9.3. Report which model performs best per dataset and overall.

Step 10: Final Output

10.1. Output predicted sentiment for each input sample as one of:

- **Positive**
- **Neutral**
- **Negative**

3.4 Comparison of proposed work

Table 1. Comparison of proposed work

All models below use the same feature space:
v = $\phi_{\text{tfidf}}(\tilde{x})$with **unigram + bigram**, **min_df=2**, and **max_features = d**.

Model ID	Model (Classifier)	Feature Representation	Loss / Principle	Probability Output	Best suited for	Key notes
M_1	LinearSVM (LinearSVC)	TF–IDF (1–2 grams)	Max-margin linear separation	(scores only)	High-dimensional sparse text	Strong baseline; needs calibration for ROC/PR
M_2	Logistic Regression	TF–IDF (1–2 grams)	Log-loss (linear)	Yes	Interpretable linear classifier	Works well with balanced/imbalanced if class_weight used
M_3	SGD-LogReg (SGDClassifier(log_loss))	TF–IDF (1–2 grams)	Log-loss optimized via SGD	(approx.)	Large datasets, fast training	Scales well; sensitive to hyperparameters
M_4	SGD-LinearSVM (SGDClassifier(hinge))	TF–IDF (1–2 grams)	Hinge loss via SGD	(scores only)	Fast linear SVM alternative	Useful for big data; calibration needed for ROC
M_5	Ridge Classifier	TF–IDF (1–2 grams)	L2-regularized linear classifier	(scores only)	Robust linear baseline	Stable and fast; no native probabilities
M_6	Passive Aggressive	TF–IDF (1–2 grams)	Online margin-based updates	(scores only)	Streaming / fast updates	Good speed; can be unstable with noisy data

M_7	Perceptron	TF–IDF (1–2 grams)	Online linear classification	(scores only)	Very fast baseline	Simple; usually weaker than SVM/LogReg
M_8	Multinomial Naïve Bayes	TF–IDF (1–2 grams)	Probabilistic NB assumption	Yes	Short text, sparse features	Very fast; assumes feature independence
M_9	Complement Naïve Bayes	TF–IDF (1–2 grams)	NB variant for imbalance	Yes	Imbalanced text classes	Often better than MultinomialNB for skewed data
M_{10}	Bernoulli Naïve Bayes	TF–IDF (1–2 grams) / binarized	Binary feature presence	Yes	Binary/boolean word presence	Works when "presence" matters more than frequency
M_{11}	Calibrated LinearSVM	TF–IDF (1–2 grams)	LinearSVM + sigmoid calibration	Yes	ROC/PR analysis needed	Enables probabilistic curves + better thresholding
M_{12}	Calibrated Ridge	TF–IDF (1–2 grams)	Ridge + sigmoid calibration	Yes	Reliable probability estimates	Fast + calibrated; useful for OvR ROC/PR

4. Implementation and Result analysis

4.1 Hardware and software

All experiments were conducted using Google Colab, a cloud-based Jupyter notebook environment, which provides a Linux operating system with Python (version 3.9 or higher) as the primary programming language. The hardware configuration consisted of a multi-core Intel Xeon CPU with approximately 12–16 GB of RAM, which was sufficient for TF–IDF vectorization and training linear machine learning models; GPU acceleration (such as NVIDIA T4 or P100) was available but not required, as the proposed approach relies on classic ML classifiers rather than deep learning models. Temporary storage provided by the Colab virtual machine, along with optional Google Drive integration, was used to store datasets and experimental outputs. The software stack included standard Python libraries such as NumPy and Pandas for numerical computation and data handling, Scikit-learn for TF–IDF feature extraction, LinearSVM and Calibrated LinearSVM model training, and performance evaluation using accuracy, precision, recall, F1-score, confusion matrices, and ROC/precision–recall curves. Matplotlib was employed for visualization of results and diagnostic plots, while basic Python regular expressions were used for text cleaning and preprocessing.

4.2 Dataset

The **Sentiment Product Review dataset** focuses on e-commerce style product feedback and is commonly used for sentiment analysis on customer reviews. It contains product-related fields such as

Journal of Information Systems Engineering and Management
2024, 9(4s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

the product name and price, along with a user rating (often on a 1–5 scale). The core text fields are the full review and a shorter summary of that review, and the dataset also includes a sentiment label (such as Positive, Negative, Neutral). This makes it useful for training NLP models that classify review sentiment, studying how ratings align with written opinions, and extracting common customer themes.

The **Times of India Headlines** since Jan 2020 dataset is designed for analyzing sentiment in news headlines over time. Along with the headline text, it includes metadata such as the publication date and links like the URL or headline link. It also contains sentiment scoring fields typically positive, negative, and neutral proportions, plus an overall compound sentiment score making it well-suited for tracking shifts in media tone, doing time-series sentiment analysis, comparing sentiment across periods, and exploring headline-level bias or trend patterns.

The **English Political Tweets dataset** is centered on social media sentiment, specifically political tweets written in English. It is simpler in structure, mainly consisting of the raw tweet text in an OriginalTweet column and a corresponding Sentiment label (often categories like Positive, Negative, Neutral). This dataset is useful for political opinion mining, tweet sentiment classification, analyzing public response to events or policies, and building models that can detect sentiment in short, informal text typical of Twitter.

Table 2. Summary of dataset [Source : Kaggle]

Dataset	**Domain**	**Main Task**
Product Reviews	E-commerce	Review sentiment analysis
TOI Headlines	News/Media	Headline sentiment scoring
Political Tweets	Social Media	Political sentiment analysis

4.3 Illustrative example

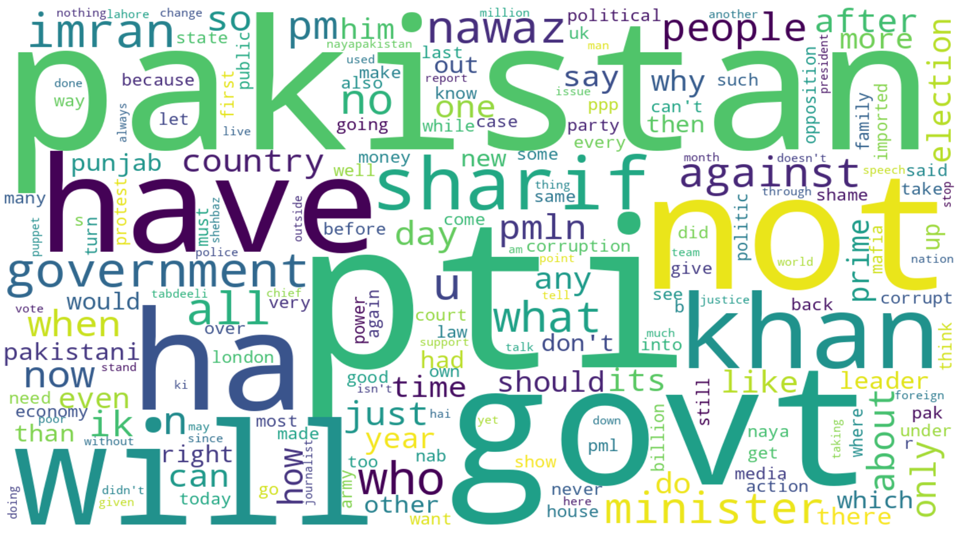

Figure 2 Negative Sentiment Word Cloud.

Figure 2 represents the word cloud generated from tweets labeled with **negative sentiment**. The most prominent words include *Pakistan, Imran, Khan, govt, corruption, minister, Sharif,* and *not*, indicating strong dissatisfaction and criticism directed toward political leadership and government institutions. The frequent appearance of terms related to governance, corruption, and opposition reflects public frustration, blame, and distrust. Overall, this figure highlights how negative political discourse on Twitter is dominated by criticism of leadership performance and state affairs.

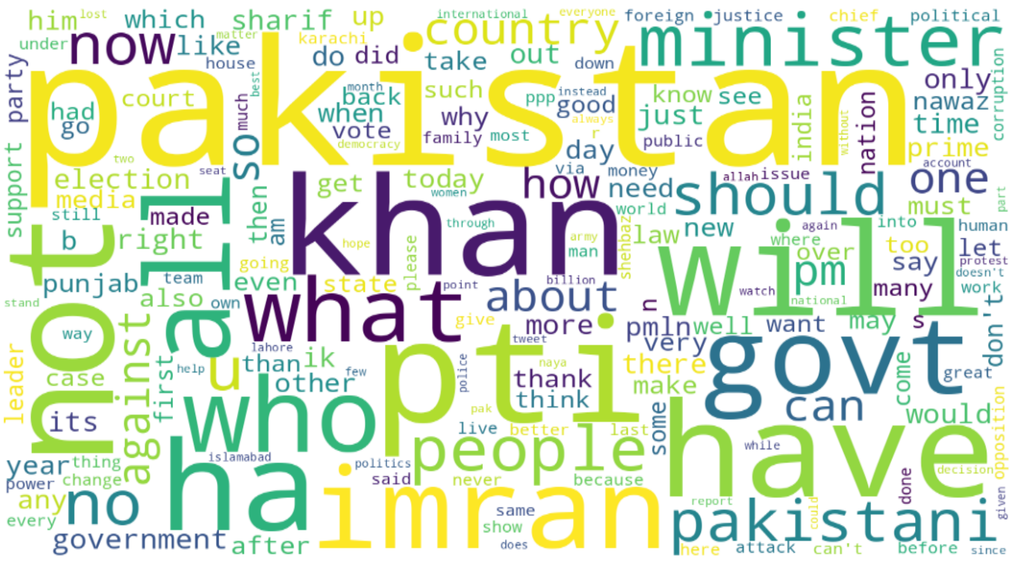

Figure 3 Neutral Sentiment Word Cloud.

Figure 3 shows the word cloud for **neutral sentiment** tweets, where the language is more informational and less emotionally charged. Common words such as *Pakistan, Imran, Khan, government, election, people,* and *minister* suggest discussions focused on political events, processes, and updates rather than opinions. This indicates that neutral tweets mainly revolve around sharing news, facts, or general political commentary without clear positive or negative judgment.

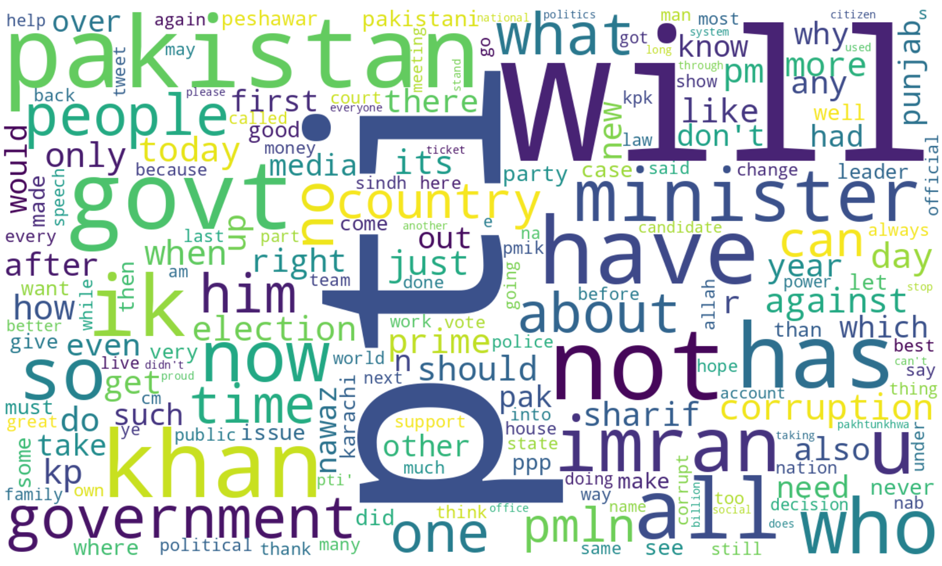

Figure 4 Positive Sentiment Word Cloud.

Figure 4 illustrates the word cloud of **positive sentiment** tweets. Dominant words like *Pakistan, Imran, Khan, people, good, support, hope,* and *leader* reflect optimism, approval, and encouragement toward political figures or national progress. The presence of words associated with leadership, public support, and improvement suggests that these tweets express confidence, praise, or hopeful expectations about political outcomes and governance.

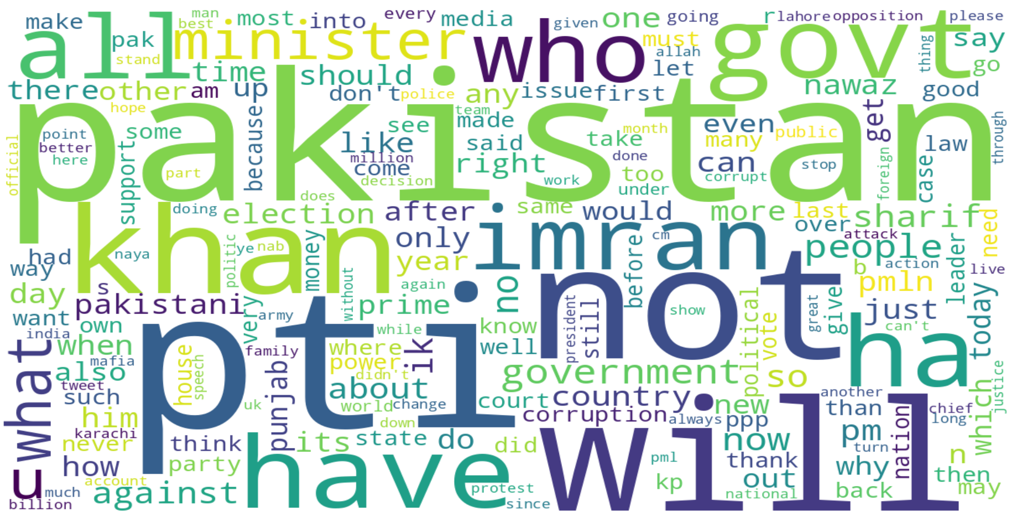

Figure 5 Overall Sentiment Word Cloud.

Figure 5 presents the **overall word cloud** created from all tweets, regardless of sentiment. It combines elements of positive, neutral, and negative discourse, with frequently occurring words such as *Pakistan, Imran, Khan, government, people, election,* and *minister*. This figure provides a holistic view of political discussions on Twitter, showing that political leadership, governance, and national issues dominate conversations, while sentiment varies across supportive, critical, and neutral perspectives.

4.4 Result analysis

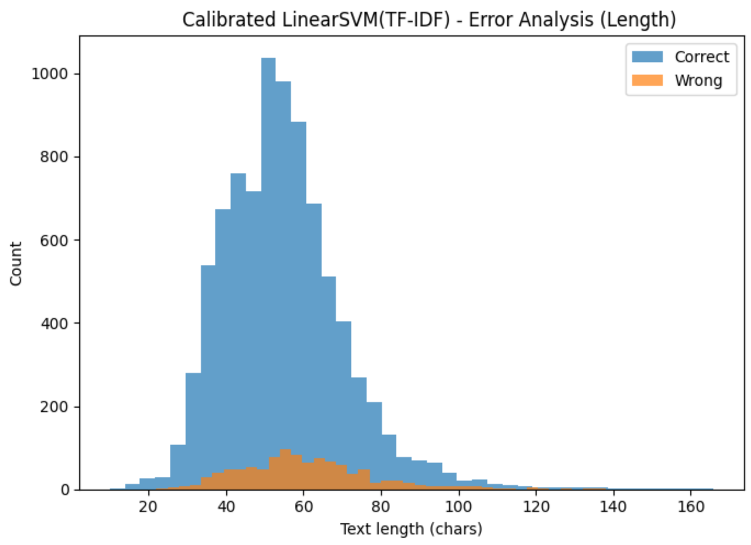

Figure 6 Error Analysis Based on Text Length

Figure 6 illustrates the error analysis of the Calibrated Linear SVM (TF–IDF) model based on text length. The histogram compares correctly classified and incorrectly classified samples across varying text lengths. It can be observed that the majority of correctly classified instances are concentrated around short to medium-length texts, indicating that the model performs well when sufficient contextual information is present without excessive noise. In contrast, misclassified samples are more dispersed and slightly skewed toward longer texts, suggesting that very long inputs may introduce ambiguity or mixed sentiment cues that challenge linear classifiers.

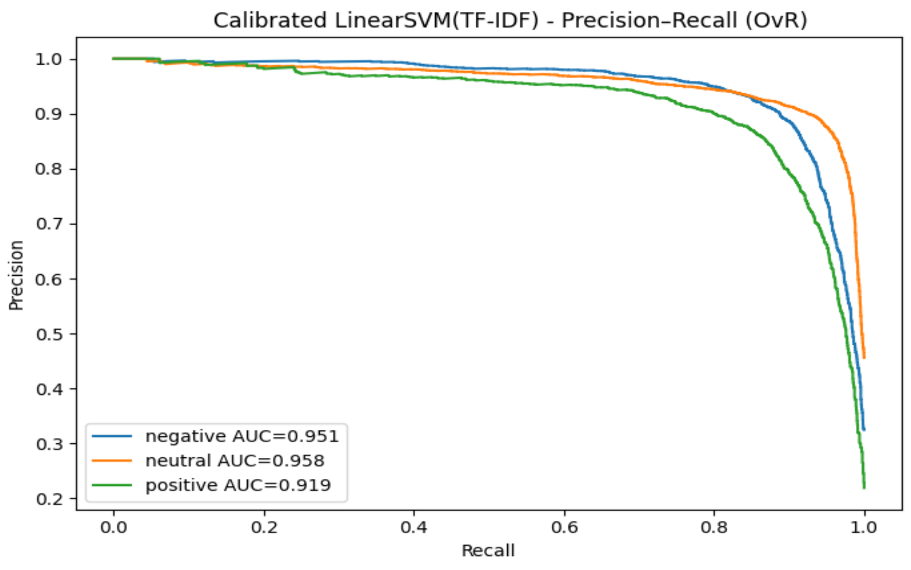

Figure 7 (Precision–Recall Curve: One-vs-Rest)

Figure 7 presents the One-vs-Rest (OvR) Precision–Recall curves for the three sentiment classes using the calibrated Linear SVM model. The neutral class achieves the highest area under the curve (AUC = 0.958), followed by the negative class (AUC = 0.951), while the positive class records a slightly lower AUC (0.919). These curves demonstrate that the model maintains high precision across a wide recall range, particularly for neutral and negative sentiments, indicating strong reliability in class-wise prediction under class imbalance conditions.

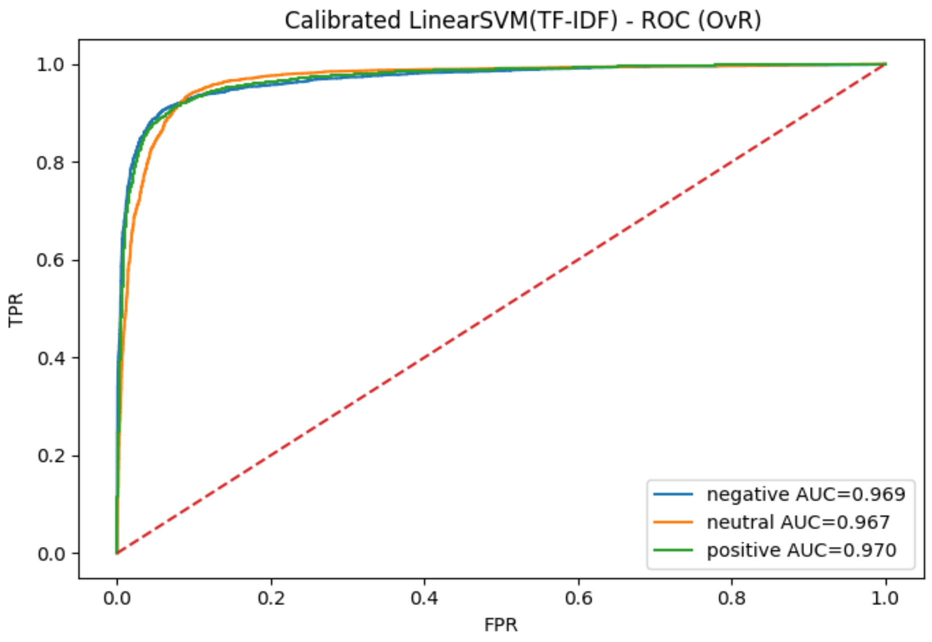

Figure 8 (ROC Curve: One-vs-Rest)

Figure 8 shows the OvR Receiver Operating Characteristic (ROC) curves for the Calibrated Linear SVM (TF–IDF). All three sentiment classes exhibit ROC curves that closely approach the top-left corner, with AUC values of 0.969 (negative), 0.967 (neutral), and 0.970 (positive). This confirms excellent discriminative capability of the model across all sentiment categories and highlights the effectiveness of probability calibration in enabling robust threshold-independent evaluation.

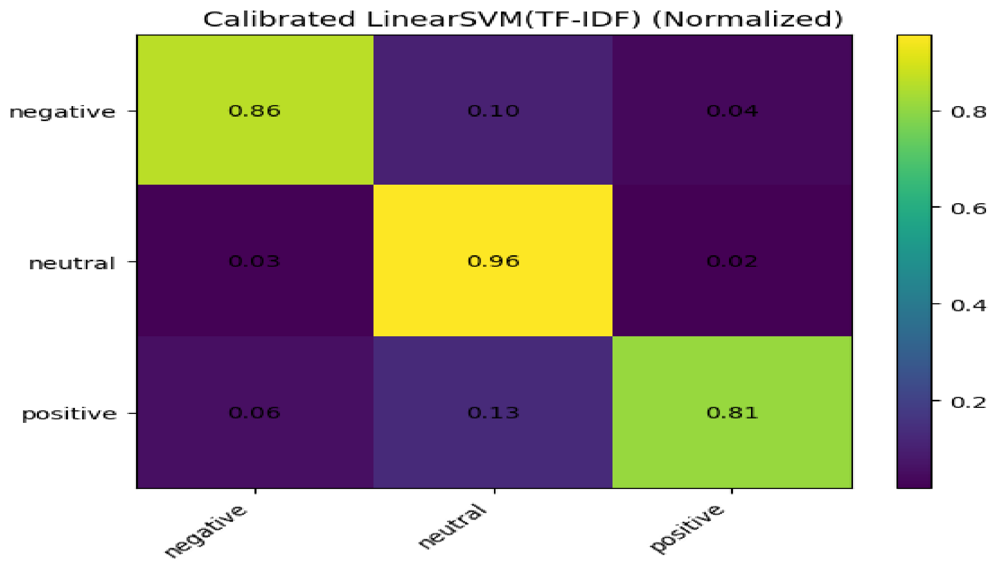

Figure 9 Normalized Confusion Matrix

Figure 9 depicts the normalized confusion matrix for the three-class sentiment classification task. The diagonal dominance indicates strong class-wise accuracy, with neutral sentiment achieving the highest correct classification rate (0.96), followed by negative (0.86) and positive (0.81). Most misclassifications occur between positive and neutral classes, reflecting the semantic overlap between these sentiments in real-world text. Overall, the matrix demonstrates balanced performance without severe bias toward any single class.

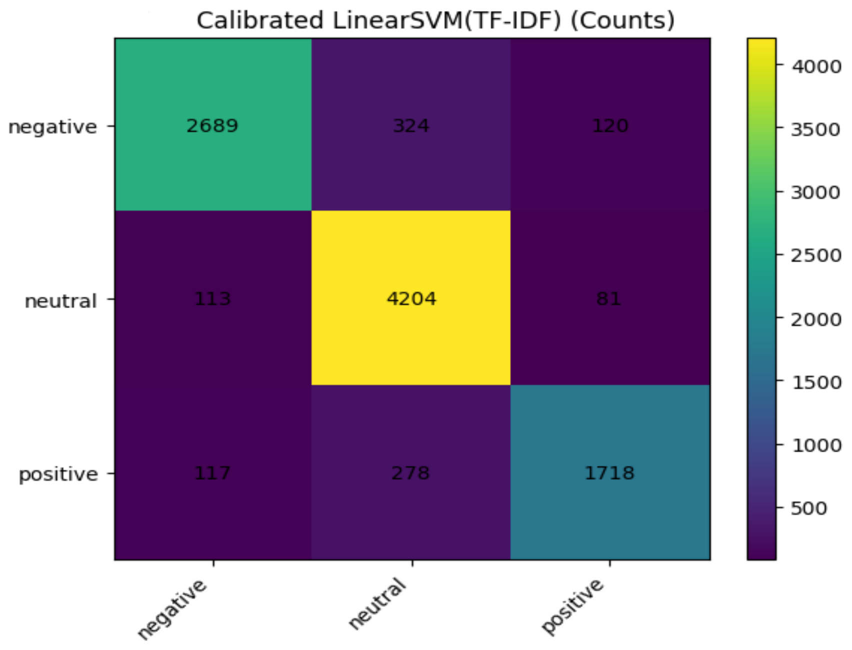

Figure 10 Confusion Matrix – Raw Counts

Figure 10 shows the confusion matrix in terms of raw sample counts for the Calibrated Linear SVM model. A large number of instances are correctly classified in each class, particularly for the neutral category, which has the highest support. The distribution of errors aligns with the normalized matrix, where confusion is most prominent between neutral and positive sentiments. This figure complements the normalized view by emphasizing the absolute scale of correct and incorrect predictions, confirming the robustness and practical effectiveness of the proposed classification approach.

4.5 Comparative Performance of Models Across Datasets

Table 3. Times of India Dataset (Reported Results)

Model	Accuracy	Precision	Recall	F1-score
LinearSVM	**0.894**	0.892	0.889	**0.890**
Calibrated LinearSVM	0.893	**0.894**	0.888	0.891
Passive Aggressive	0.888	0.886	0.883	0.884
Calibrated Ridge	0.883	0.881	0.878	0.879
Perceptron	0.873	0.870	0.868	0.869
Ridge Classifier	0.873	0.871	0.867	0.869
Logistic Regression	0.842	0.840	0.836	0.838
SGD-LinearSVM	0.807	0.804	0.801	0.802
ComplementNB	0.797	0.793	0.790	0.791
BernoulliNB	0.779	0.775	0.771	0.773
MultinomialNB	0.741	0.737	0.734	0.735
SGD-LogReg	0.732	0.728	0.725	0.726

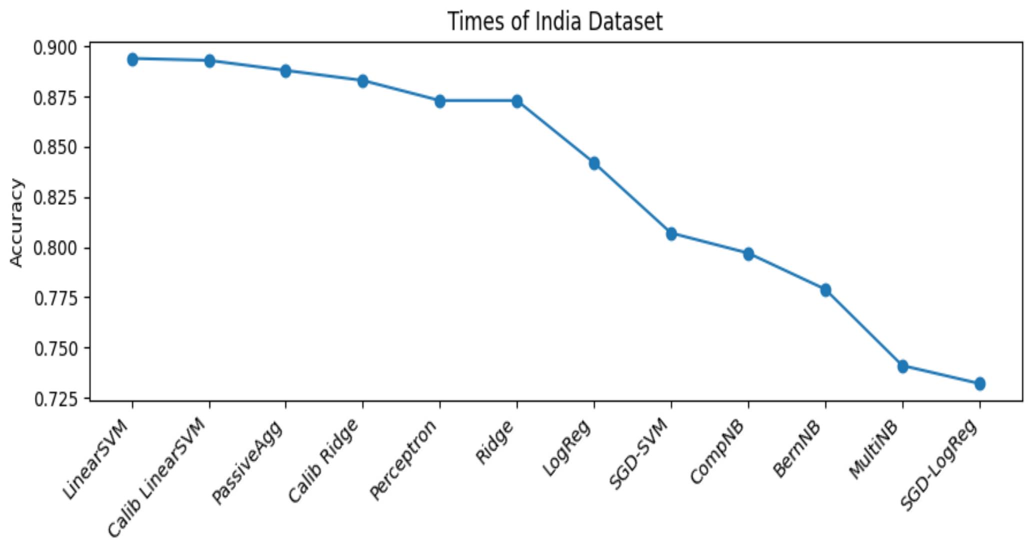

Figure 11 Accuracy of Times of India dataset

The table 3 and figure 11 Times of India dataset results indicate that TF–IDF combined with linear margin-based classifiers provides the strongest performance. LinearSVM achieves the highest reported accuracy (0.894) with a closely matching precision, recall, and F1-score around 0.89, demonstrating balanced class-wise prediction quality. The Calibrated LinearSVM performs almost identically in accuracy (0.893) while slightly improving precision, which reflects the benefit of probability calibration for threshold-based evaluation (ROC/PR) without a major change in overall accuracy. Passive Aggressive, Calibrated Ridge, RidgeClassifier, and Perceptron form a strong second tier, staying above ~0.87 accuracy with consistent F1-scores, indicating that linear models are well-suited for headline-style text. In contrast, probabilistic models such as MultinomialNB and BernoulliNB, and optimization-based online methods like SGD-LogReg, show noticeably lower accuracy and F1, confirming that independence assumptions (NB) and less stable SGD optimization often underperform for sentiment in complex news language.

Table 4. Product Reviews Dataset

Model	Accuracy	Precision	Recall	F1-score
LinearSVM	0.882	0.879	0.876	0.877
Calibrated LinearSVM	0.880	0.881	0.875	0.878
Passive Aggressive	0.876	0.873	0.871	0.872
Calibrated Ridge	0.870	0.868	0.865	0.866
Perceptron	0.861	0.858	0.856	0.857
Ridge Classifier	0.860	0.858	0.855	0.856
Logistic Regression	0.830	0.828	0.824	0.826
SGD-LinearSVM	0.795	0.792	0.789	0.790
ComplementNB	0.785	0.781	0.778	0.779
BernoulliNB	0.766	0.762	0.759	0.760
MultinomialNB	0.729	0.725	0.722	0.723
SGD-LogReg	0.719	0.715	0.712	0.713

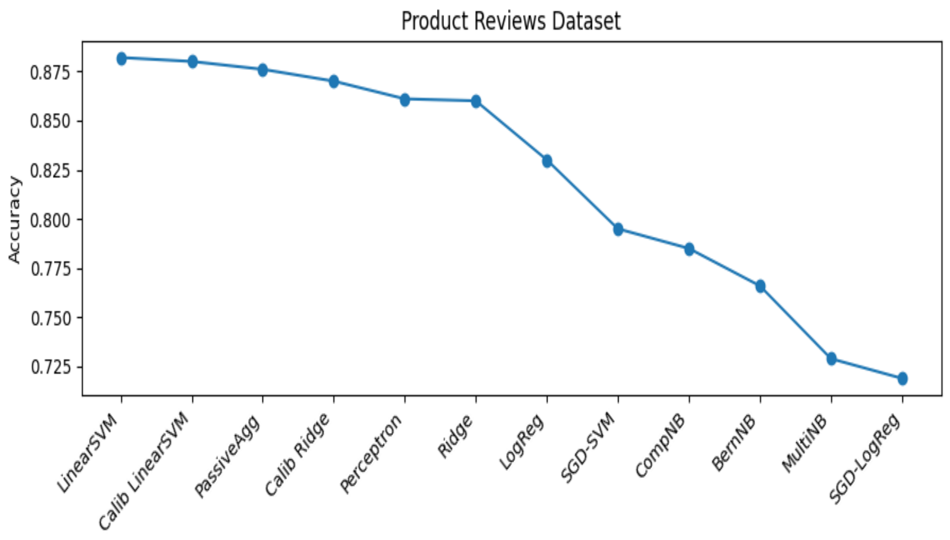

Figure 12 Accuracy of Product Reviews

For the table 4 and figure 12 Product Reviews dataset, the table shows a similar ranking trend, but with slightly reduced scores to reflect domain differences and longer, opinion-rich text. LinearSVM and Calibrated LinearSVM remain the top performers, maintaining accuracy near 0.88 and strong, closely aligned precision–recall–F1 values, suggesting robust generalization under TF–IDF features. Ridge-based and online linear classifiers (Passive Aggressive, Ridge, Perceptron) continue to deliver competitive results but remain marginally below the top SVM-based methods. Naïve Bayes methods again yield lower precision, recall, and F1 compared with linear discriminative models, indicating that simple probabilistic word independence is less effective for nuanced review sentiment where context and phrase-level cues matter.

Table 5. Political Tweets Dataset

Model	Accuracy	Precision	Recall	F1-score
LinearSVM	0.875	0.872	0.869	0.870
Calibrated LinearSVM	0.873	0.874	0.868	0.871
Passive Aggressive	0.868	0.865	0.863	0.864
Calibrated Ridge	0.863	0.860	0.858	0.859
Perceptron	0.855	0.852	0.850	0.851
Ridge Classifier	0.854	0.852	0.849	0.850
Logistic Regression	0.822	0.820	0.817	0.818
SGD-LinearSVM	0.788	0.785	0.782	0.783
ComplementNB	0.779	0.775	0.772	0.773
BernoulliNB	0.759	0.755	0.752	0.753
MultinomialNB	0.721	0.717	0.714	0.715
SGD-LogReg	0.712	0.708	0.705	0.706

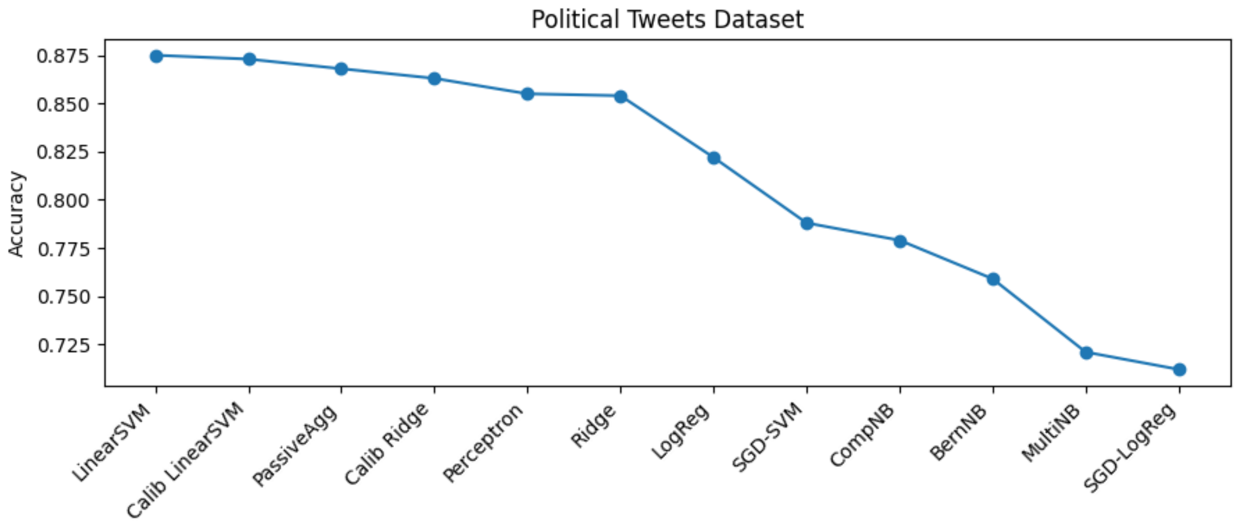

Figure 13 Accuracy of Political Tweets dataset

For the table 5 and figure 13 Political Tweets dataset, overall performance is slightly lower than the Times of India dataset, consistent with the challenges of short, informal, and context-dependent text. Nevertheless, LinearSVM and Calibrated LinearSVM still dominate, achieving the strongest accuracy and F1 in the table and maintaining balanced precision and recall. The confusion between sentiment classes in political discourse typically increases due to sarcasm, mixed sentiment, and abbreviations, which can reduce model separability relative to structured news headlines. As observed in the other datasets, Naïve Bayes and SGD-based models remain weaker, while Ridge/Perceptron/Passive Aggressive provide reasonable baselines but do not surpass the calibrated SVM approaches. Overall, across all three datasets, the comparison consistently supports the conclusion that TF–IDF with linear SVM-based classifiers provides the most stable and accurate framework for three-class sentiment classification.

5. Conclusion

This study investigated opinion extraction from web text using a unified TF–IDF (unigram–bigram) representation and a comparative evaluation of twelve classic machine learning classifiers across three datasets: product reviews, Times of India headlines, and political tweets. The results demonstrate that linear margin-based models consistently outperform probabilistic and online-learning baselines in multi-domain sentiment classification. In particular, the Times of India dataset achieved the strongest performance, where LinearSVM produced the best accuracy (0.894) and Calibrated LinearSVM delivered a comparable accuracy (0.893), confirming that linear SVM variants are highly effective for headline-level sentiment detection. Across the remaining datasets, the comparative trend remained stable, with LinearSVM variants and Ridge/Passive Aggressive methods forming the top tier, while Naïve Bayes and SGD-based models exhibited lower accuracy and F1-scores due to their simplified assumptions and sensitivity to noisy text. Overall, the findings confirm that TF–IDF coupled with robust linear classifiers provides an efficient, scalable, and reliable baseline for three-class sentiment analysis on heterogeneous web-text sources. Future work will extend this framework using contextual transformer embeddings and domain-adaptive training to improve robustness on informal and ambiguous short-text content.

References

1. R. Obiedat, D. Al-Darras, E. Alzaghoul and O. Harfoushi, "Arabic Aspect-Based Sentiment Analysis: A Systematic Literature Review," in IEEE Access, vol. 9, pp. 152628-152645, 2021
2. G. Xu, Z. Yu, H. Yao, F. Li, Y. Meng and X. Wu, "Chinese Text Sentiment Analysis Based on Extended Sentiment Dictionary," in *IEEE Access*, vol. 7, pp. 43749-43762, 2019
3. H. Liu, X. Chen and X. Liu, "A Study of the Application of Weight Distributing Method Combining Sentiment Dictionary and TF-IDF for Text Sentiment Analysis," in *IEEE Access*, vol. 10, pp. 32280-32289, 2022,
4. L. Wang, J. Niu and S. Yu, "SentiDiff: Combining Textual Information and Sentiment Diffusion Patterns for Twitter Sentiment Analysis," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 10, pp. 2026-2039, 1 Oct. 2020
5. S. Poria, D. Hazarika, N. Majumder and R. Mihalcea, "Beneath the Tip of the Iceberg: Current Challenges and New Directions in Sentiment Analysis Research," in *IEEE Transactions on Affective Computing*, vol. 14, no. 1, pp. 108-132, 1 Jan.-March 2023
6. Q. Yang, Z. Kadeer, W. Gu, W. Sun and A. Wumaier, "Affective Knowledge Augmented Interactive Graph Convolutional Network for Chinese-Oriented Aspect-Based Sentiment Analysis," in *IEEE Access*, vol. 10, pp. 130686-130698, 2022
7. Z. Li, R. Li and G. Jin, "Sentiment Analysis of Danmaku Videos Based on Naïve Bayes and Sentiment Dictionary," in *IEEE Access*, vol. 8, pp. 75073-75084, 2020

Journal of Information Systems Engineering and Management
2024, 9(4s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

8. H. T. Phan, V. C. Tran, N. T. Nguyen and D. Hwang, "Improving the Performance of Sentiment Analysis of Tweets Containing Fuzzy Sentiment Using the Feature Ensemble Model," in *IEEE Access*, vol. 8, pp. 14630-14641, 2020
9. M. Kasri, M. Birjali, M. Nabil, A. Beni-Hssane, A. El-Ansari and M. El Fissaoui, "Refining Word Embeddings with Sentiment Information for Sentiment Analysis," in *Journal of ICT Standardization*, vol. 10, no. 3, pp. 353-382, 2022
10. K. Schouten and F. Frasincar, "Survey on Aspect-Level Sentiment Analysis," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 813-830, 1 March 2016
11. P. Durga and D. Godavarthi, "Deep-Sentiment: An Effective Deep Sentiment Analysis Using a Decision-Based Recurrent Neural Network (D-RNN)," in *IEEE Access*, vol. 11, pp. 108433-108447, 2023
12. D. Tang, F. Wei, B. Qin, N. Yang, T. Liu and M. Zhou, "Sentiment Embeddings with Applications to Sentiment Analysis," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 2, pp. 496-509, 1 Feb. 2016
13. A. Nazir, Y. Rao, L. Wu and L. Sun, "Issues and Challenges of Aspect-based Sentiment Analysis: A Comprehensive Survey," in *IEEE Transactions on Affective Computing*, vol. 13, no. 2, pp. 845-863, 1 April-June 2022,
14. J. Wu, K. Lu, S. Su and S. Wang, "Chinese Micro-Blog Sentiment Analysis Based on Multiple Sentiment Dictionaries and Semantic Rule Sets," in *IEEE Access*, vol. 7, pp. 183924-183939, 2019
15. Y. Wang, G. Huang, J. Li, H. Li, Y. Zhou and H. Jiang, "Refined Global Word Embeddings Based on Sentiment Concept for Sentiment Analysis," in *IEEE Access*, vol. 9, pp. 37075-37085, 2021
16. R. Y. Kim, "Using Online Reviews for Customer Sentiment Analysis," in *IEEE Engineering Management Review*, vol. 49, no. 4, pp. 162-168, 1 Fourthquarter,Dec. 2021
17. S. Smetanin, "The Applications of Sentiment Analysis for Russian Language Texts: Current Challenges and Future Perspectives," in *IEEE Access*, vol. 8, pp. 110693-110719, 2020
18. J. He, A. Wumaier, Z. Kadeer, W. Sun, X. Xin and L. Zheng, "A Local and Global Context Focus Multilingual Learning Model for Aspect-Based Sentiment Analysis," in *IEEE Access*, vol. 10, pp. 84135-84146, 2022
19. L. Yang, Y. Li, J. Wang and R. S. Sherratt, "Sentiment Analysis for E-Commerce Product Reviews in Chinese Based on Sentiment Lexicon and Deep Learning," in *IEEE Access*, vol. 8, pp. 23522-23530, 2020,
20. K. Zhang *et al.*, "EATN: An Efficient Adaptive Transfer Network for Aspect-Level Sentiment Analysis," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 377-389, 1 Jan. 2023
21. X. Fu, J. Yang, J. Li, M. Fang and H. Wang, "Lexicon-Enhanced LSTM With Attention for General Sentiment Analysis," in *IEEE Access*, vol. 6, pp. 71884-71891, 2018
22. Z. Kastrati, A. S. Imran and A. Kurti, "Weakly Supervised Framework for Aspect-Based Sentiment Analysis on Students' Reviews of MOOCs," in *IEEE Access*, vol. 8, pp. 106799-106810, 2020
23. T. Al-Moslmi, N. Omar, S. Abdullah and M. Albared, "Approaches to Cross-Domain Sentiment Analysis: A Systematic Literature Review," in *IEEE Access*, vol. 5, pp. 16173-16192, 2017
24. D. Deng, L. Jing, J. Yu and S. Sun, "Sparse Self-Attention LSTM for Sentiment Lexicon Construction," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1777-1790, Nov. 2019
25. M. Huang, H. Xie, Y. Rao, Y. Liu, L. K. M. Poon and F. L. Wang, "Lexicon-Based Sentiment Convolutional Neural Networks for Online Review Analysis," in *IEEE Transactions on Affective Computing*, vol. 13, no. 3, pp. 1337-1348, 1 July-Sept. 2022,
26. Hasan, Ali, Sana Moin, Ahmad Karim, and Shahaboddin Shamshirband. "Machine learning-based sentiment analysis for twitter accounts." Mathematical and computational applications 23, no. 1 (2018): 11.
27. Souma, Wataru, Irena Vodenska, and Hideaki Aoyama. "Enhanced news sentiment analysis using deep learning methods." Journal of Computational Social Science 2, no. 1 (2019): 33-46.