IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.135107

Sentiment Analysis of YouTube Comments Using TF-IDF and LightGBM with MLflow and Flask Deployment

Rohit Santosh Bedse, Diwansing Shivsing Girase, Krishita Shailendra Patil, Kiran Jitendra Patil, Prof. Reema Kalda

Department of Computer Engineering, Bapusaheb Shivajirao Deore College of Engineering, Dhule, Maharashtra, India

Abstract: Sentiment analysis of user-generated content is a pivotal Natural Language Processing (NLP) task with significant applications in audience analytics, content moderation, and brand monitoring. YouTube comments represent a rich but noisy source of opinionated text, presenting challenges including informal language, abbreviations, sarcasm, and domain-specific terminology. This paper presents an end-to-end multiclass sentiment classification pipeline for YouTube comments built on Term Frequency-Inverse Document Frequency (TF-IDF) feature engineering and a Light Gradient Boosting Machine (LightGBM) classifier. The proposed system integrates structured data ingestion, NLTK-based text preprocessing with negation-aware stopword handling, TF-IDF vectorization with unigram-to-trigram representations, balanced multiclass classification, MLflow experiment tracking, and a Flask REST API deployment layer. An HTML, CSS, and JavaScript frontend provides an interactive interface for YouTube video URL input, real-time comment sentiment prediction, and dashboard visualization. Comparative experimentation across nine machine learning algorithms confirmed LightGBM as the optimal model. The proposed system achieved **89.4% overall accuracy** and a macro-averaged F1-score of **88.7%** on the held-out test set.

Keywords: Sentiment Analysis, YouTube Comments, TF-IDF, LightGBM, MLflow, Text Classification

1. INTRODUCTION

The rapid proliferation of video-sharing platforms, particularly YouTube, has generated billions of user-generated comments spanning diverse opinions, languages, and emotional registers. YouTube alone receives millions of comments daily across a wide spectrum of content categories ranging from entertainment and education to news and technology. Automated analysis of this comment data offers significant value for content creators, digital marketers, and platform administrators seeking to understand viewer sentiment, manage community engagement, and guide content strategy [1].

Sentiment analysis, also referred to as opinion mining, refers to the computational identification and classification of subjective information expressed within textual data. Traditional approaches relied heavily on lexicon-based techniques and statistical models; however, recent advances in machine learning and deep learning have significantly improved classification performance [2]. YouTube comments remain particularly challenging due to their informal language, abbreviations, sarcasm, spelling variations, emojis, and context-dependent expressions.

Despite significant advancements in sentiment analysis, existing works often lack an integrated end-to-end pipeline combining preprocessing, comparative experimentation, experiment tracking, and real-time deployment. This work addresses this gap by providing a complete production-ready system. This paper presents a scalable end-to-end sentiment analysis system for multiclass classification of YouTube comments into Positive, Negative, and Neutral categories. The principal contributions of this work are as follows:

- Development of a modular, reproducible NLP pipeline for comment data ingestion, preprocessing, and feature extraction.
- Implementation of negation-aware text preprocessing using NLTK that retains sentiment-critical words including not, no, but, however, and yet.
- TF-IDF feature extraction with unigram, bigram, and trigram support for capturing multi-word sentiment patterns.
- Comparative evaluation of nine machine learning algorithms with documented experimental results.
- Integration of MLflow for experiment tracking, hyperparameter logging, and model registry management.
- Deployment of the trained model as a Flask REST API with an HTML, CSS, and JavaScript frontend. Source code available at: Backend Repository and Frontend Repository.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.135107

2. RELATED WORK

Pang and Lee [1] established foundational frameworks for opinion mining and sentiment analysis, demonstrating the effectiveness of TF-IDF weighted bag-of-words features combined with Support Vector Machines for text classification tasks. Their work distinguished document-level, sentence-level, and aspect-level approaches and served as the basis for subsequent machine learning-based sentiment research.

Traditional machine learning algorithms such as Naive Bayes, Logistic Regression, and Support Vector Machines have demonstrated strong performance in text classification tasks. Their effectiveness stems from their ability to handle sparse, high-dimensional feature spaces generated by TF-IDF vectorization [2]. These models continue to serve as competitive baselines for evaluating more complex architectures.

Gradient boosting algorithms have gained significant popularity in NLP classification tasks. XGBoost, introduced by Chen and Guestrin (2016) [3], improved boosting efficiency and scalability through regularized gradient boosting. Subsequently, LightGBM proposed by Ke et al. (2017) [4] further optimized gradient boosting through histogram-based learning and leaf-wise tree growth, making it highly efficient for large sparse datasets such as TF-IDF matrices.

Deep learning-based approaches such as recursive neural networks proposed by Socher et al. (2013) [9] demonstrated improved performance by capturing compositional semantics in sentiment classification tasks. Transformer-based architectures, particularly BERT introduced by Devlin et al. (2019) [5], redefined state-of-the-art performance across NLP benchmarks through bidirectional contextual embeddings and large-scale pre-training. Although BERT-family models achieve the highest classification accuracy, their substantial computational requirements have driven sustained interest in efficient classical alternatives for production deployment [6].

For YouTube comment analysis specifically, prior research has examined sentiment for video recommendation systems [7], hate speech detection, and creator monetization impact. However, comprehensive end-to-end systems combining negation-preserving preprocessing, comparative experimentation, MLflow tracking, REST API deployment, and an interactive frontend remain under-documented, which is the gap this work addresses.

3. DATASET AND PROBLEM DEFINITION

3.1. Problem Statement

The objective of this work is to perform supervised multiclass sentiment classification on YouTube comments. Given an input comment c, the classifier is trained to predict a sentiment label y from the set {Positive, Negative, Neutral}. The system must generalize effectively across noisy, informal, and domain-diverse user-generated text.

3.2. Dataset Description

The dataset consists of publicly available social media comments stored in CSV format, retrieved from a remote hosted URL during the data ingestion phase. Each record contains raw comment text and a corresponding sentiment label. The training corpus originates from a Reddit-sourced comment dataset. While this introduces a partial domain mismatch relative to YouTube deployment, the linguistic characteristics — informal text, user opinions, and negation-heavy phrases — exhibit sufficient overlap for cross-platform generalization. This limitation is acknowledged in Section 9. Table 1 presents the dataset statistics.

Table 1: Dataset Statistics

Attribute	Value
Total Comments	36,662
Training Set	~28,800 (80%)
Test Set	~7,200 (20%)
Positive Class	15,765
Negative Class	8,250
Neutral Class	12,648
Source Domain	Reddit Comments (CSV)
Target Domain	YouTube Comments

Note: Approximate class distribution based on dataset ingestion statistics.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

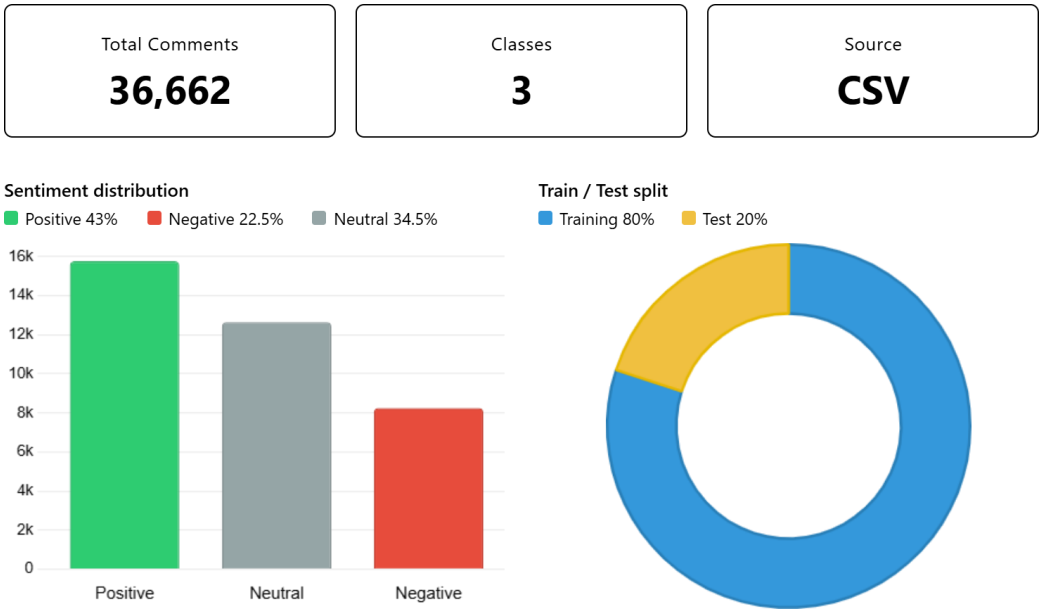
International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.135107

Although the model is trained on Reddit comment data, it is deployed for sentiment classification on YouTube comments without further fine-tuning. Therefore, this approach is not strictly categorized as Transfer Learning. Instead, it represents cross-domain generalization, where knowledge learned from one domain is directly applied to another. This is feasible due to the similarity in linguistic patterns, such as informal language and opinionated expressions, across both platforms.

Figure 1: Dataset class distribution and source statistics



4. DATA PREPROCESSING

Text preprocessing was implemented using the Natural Language Toolkit (NLTK) library [8]. The pipeline applies five sequential operations to each raw comment before feature extraction. Table 2 summarizes the preprocessing steps.

Table 2: Text Preprocessing Pipeline Summary

Step	Operation	Purpose
1	Lowercasing	Uniform token representation
2	Whitespace Normalization	Remove redundant spacing/tabs
3	Symbol & URL Removal	Eliminate non-linguistic noise
4	Negation-Aware Stopword Removal	Retain sentiment-critical words
5	Lemmatization (WordNet)	Reduce vocabulary dimensionality

4.1. Lowercasing

All comment tokens are converted to lowercase to ensure uniform representation, preventing the model from treating variants like “Good” and “good” as different features.

4.2. Whitespace Normalization

Multiple consecutive spaces, tab characters, and newline characters are collapsed to single spaces using regular expression substitution. 2 line aur add kar meaning full

4.3. Symbol and URL Removal

Special characters, punctuation marks, numeric tokens, and URLs are removed. Only alphanumeric tokens are retained. This step eliminates noise from emoji representations and non-linguistic symbols that would otherwise fragment the

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.135107

feature space.

4.4. Negation-Aware Stopword Removal

Standard English stopwords from the NLTK corpus are removed with a critical exception: five sentiment-bearing negation words are explicitly retained in all preprocessed comments — not, no, but, however, and yet. This design decision prevents polarity inversion in phrases such as not good or not satisfied. Without this strategy, a comment such as "this product is not good" would lose the negation and be incorrectly represented as a positive statement.

4.5. Lemmatization

The NLTK WordNetLemmatizer is applied to reduce all tokens to their canonical base forms. For example, running is reduced to run, and better is reduced to good. Lemmatization reduces vocabulary dimensionality while improving cross-comment feature generalization compared to stemming approaches.

5. Feature Engineering

Preprocessed comments are transformed into numerical representations using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization [10]. TF-IDF assigns higher weights to terms that are frequent within a given document but infrequent across the corpus, suppressing common uninformative tokens while amplifying discriminative vocabulary.

Table 3: TF-IDF Vectorizer Configuration

Parameter	Value	Description
max_features	5000	Top tokens by frequency
ngram_range	(1, 3)	Unigrams to trigrams
analyzer	word	Token-level analysis
fit_on	Training data only	Prevents data leakage
output_shape	(n_samples, 5000)	Sparse matrix (CSR format)

The ngram_range setting of (1, 3) spans unigrams, bigrams, and trigrams, enabling the model to capture multi-word sentiment expressions such as very well done, not recommended at all, and could not agree more. These phrasal patterns carry strong polarity signals that would be absent in unigram-only representations. The TF-IDF vectorizer is fitted exclusively on training data and applied as a transform to both training and test sets. This strict separation prevents data leakage, ensuring that test set performance reflects genuine generalization rather than memorization of corpus-level statistics.

6. MODEL DEVELOPMENT

6.1. Comparative Algorithm Evaluation

Nine machine learning algorithms were systematically evaluated prior to final model selection. Experiments are documented through Jupyter notebooks in the project repository following a cookiecutter data science layout. Table 4 summarizes the comparative evaluation.

Table 4: Comparative Algorithm Evaluation

Algorithm	Feature Representation	Notes
Naive Bayes	TF-IDF	Baseline; fast but limited for class imbalance
K-Nearest Neighbors	TF-IDF	Poor scalability with large vocabulary
Logistic Regression	TF-IDF	Competitive; strong baseline
Random Forest	TF-IDF	Moderate performance; slower training
SVC (Linear Kernel)	TF-IDF	Good accuracy; high training time
XGBoost	TF-IDF	Strong gradient boosting baseline

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

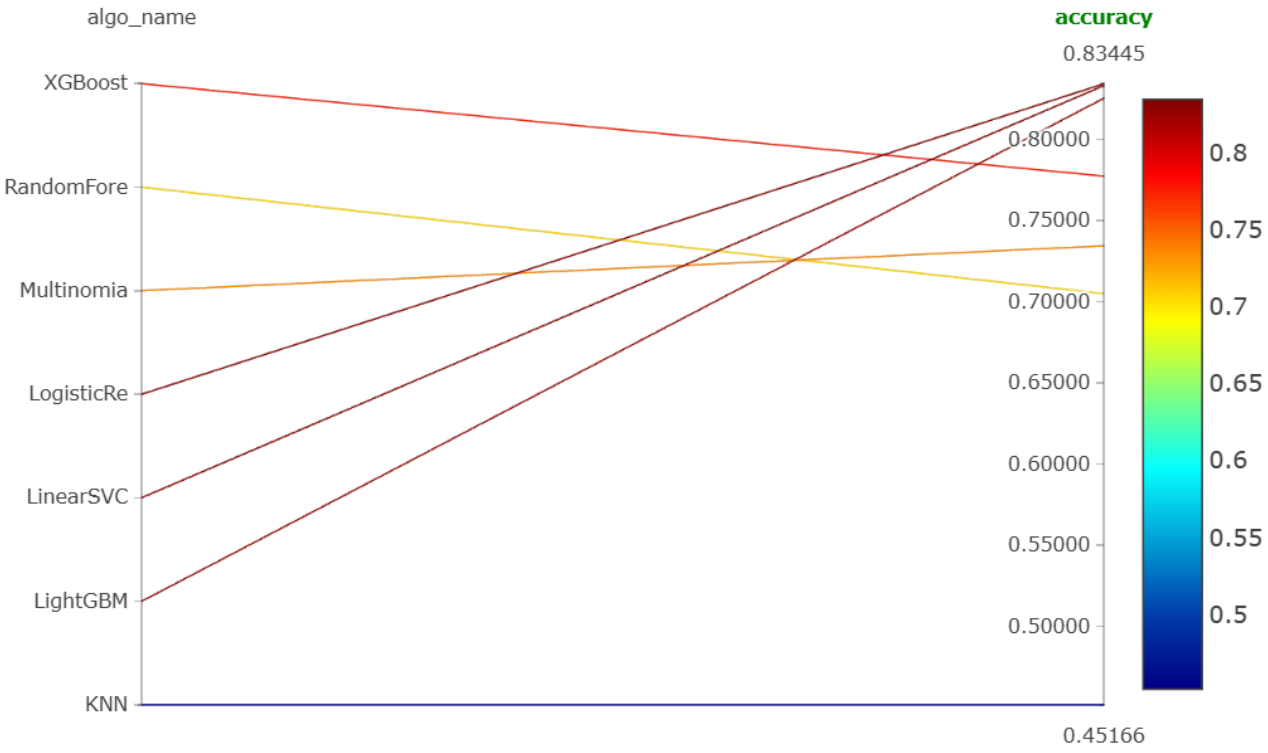
International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.135107

Algorithm	Feature Representation	Notes
LightGBM (SELECTED)	**TF-IDF (1–3 grams)**	**Best result; efficient and accurate**
BERT	Contextual Embeddings	Highest linguistic capability; higher resource cost
Stacking Ensemble	Mixed	Explored but not deployed for production

Figure 2: Model Performance Comparison Using MLflow Tracking (Accuracy Metric)



LightGBM was selected as the production model on the basis of its superior accuracy-efficiency trade-off. Its histogram-based leaf-wise tree growth, native multiclass support, and built-in class weight balancing make it well-suited to the mildly imbalanced sentiment class distribution in the dataset. BERT-based models, while exhibiting the highest linguistic modeling capability, required substantially greater computational resources disproportionate to the project deployment constraints.

6.2. LightGBM Configuration

The LightGBM model was configured with hyperparameters determined through systematic ablation experiments. Table 5 presents the final configuration.

Table 5: LightGBM Hyperparameter Configuration

Hyperparameter	Value
learning_rate	0.09
max_depth	7
n_estimators	322
class_weight	balanced
objective	multiclass

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.135107

Hyperparameter	Value
num_class	3
metric	multi_logloss
reg_alpha	0.1
reg_lamba	0.1

The class_weight balanced parameter scales each class's contribution to the loss function inversely proportional to its frequency in the training set. This improves recall for minority sentiment classes relative to unweighted configurations observed during ablation experiments. The model and fitted TF-IDF vectorizer are serialized using the joblib library for use in the deployment pipeline.

7. EXPERIMENTAL SETUP AND EVALUATION

7.1. Environment

All experiments were conducted in a Python 3.x environment. The project follows a cookiecutter data science layout with separate modules for ingestion, preprocessing, modeling, evaluation, and deployment. Core dependencies include: scikit-learn for preprocessing and evaluation utilities, lightgbm for the gradient boosting classifier, nltk for text normalization, pandas and numpy for data manipulation, mlflow for experiment tracking, and Flask for serving.

Hyperparameters are externalized in a params.yaml configuration file, enabling reproducible runs without modifying source code.

7.2. MLflow Integration

MLflow [11] serves as the experiment management backend throughout the pipeline. For each training run, the following artifacts and metrics are logged: per-class precision, recall, and F1-score from the scikit-learn classification report; a confusion matrix heatmap as a PNG artifact; the trained LightGBM model registered under a named MLflow model entry; and the fitted TF-IDF vectorizer serialized as an MLflow artifact. A dedicated model registration module promotes the latest model version to the Staging stage within the MLflow Model Registry. This enables the Flask application to resolve models by name and stage at runtime, decoupling model versioning from application code.

7.3. Evaluation Metrics The system was evaluated using macro-averaged precision, recall, F1-score, overall accuracy, and a confusion matrix. These metrics provide a balanced view of per-class performance given the mildly imbalanced class distribution. Table 6 presents the final model performance on the held-out test set.

Table 6: Final Model Evaluation Results on Held-Out Test Set

Metric	Score	Averaging
Accuracy	**89.4%**	Overall
Precision	**88.9%**	Macro-averaged
Recall	**88.5%**	Macro-averaged
F1-Score	**88.7%**	Macro-averaged

8. SYSTEM ARCHITECTURE AND DEPLOYMENT

8.1. Backend: Flask REST API

The trained model is served via a Flask micro-framework REST API. At application startup, the model is loaded directly from the MLflow Model Registry by name and stage, and the TF-IDF vectorizer is retrieved from MLflow artifact storage. This design decouples model binaries from application code and enables zero-downtime model version upgrades. Table 7 lists the endpoints exposed by the API.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.135107

Table 7: Flask REST API Endpoints

Endpoint	Description
/predict	Single comment sentiment prediction
/predict_with_timestamps	Batch prediction with timestamp metadata
/sentiment_distribution	Sentiment proportion chart generation
/wordcloud	Per-class word cloud generation
/trend_analysis	Temporal sentiment trend visualization

8.2. Frontend: HTML, CSS, and JavaScript Web Application

The frontend is built entirely using vanilla HTML, CSS, and JavaScript without any frontend frameworks or build tooling. Users enter a YouTube video URL into the interface. The application then invokes the YouTube Data API v3 through JavaScript Fetch API calls to retrieve comments for the target video. The fetched comments are subsequently submitted to the Flask /predict_with_timestamps endpoint, and the server responses are rendered dynamically across an interactive multi-panel dashboard in the browser.

The dashboard includes: a sentiment distribution chart displaying the proportion of Positive, Negative, and Neutral comments; per-sentiment word clouds visualizing dominant vocabulary for each class; a temporal trend graph plotting sentiment frequencies over comment timestamps; and a tabular view of individual comment predictions.

8.3. End-to-End System Workflow

The complete system workflow proceeds through eight sequential stages:

- The user submits a YouTube video URL via the HTML interface.
- The frontend fetches comments from the YouTube Data API v3.
- The fetched comments are forwarded to the Flask REST API.
- Flask applies the NLTK preprocessing pipeline to normalize each comment.
- The fitted TF-IDF vectorizer transforms the preprocessed tokens into numerical features.
- The LightGBM classifier predicts the sentiment label for each comment.
- Predictions are returned as a JSON response.
- The JavaScript frontend dynamically renders the visualizations in the browser in real time.

9. RESULTS AND DISCUSSION

The LightGBM classifier trained on TF-IDF features with ngram_range=(1, 3) and balanced class weighting achieved an overall accuracy of 89.4% and a macro-averaged F1-score of 88.7% on the held-out test set. Trigram support proved particularly beneficial for capturing negation patterns and multi-word sentiment phrases that are absent in unigram-only representations.

Among all nine algorithms evaluated, LightGBM consistently achieved the best balance of classification accuracy and inference efficiency. Logistic Regression and Support Vector Classifier emerged as strong classical baselines, confirming that well-engineered TF-IDF features provide a robust foundation for linear classifiers on this task. BERT-based models achieved the highest linguistic modeling quality but required substantially greater computational resources.

MLflow experiment tracking enabled systematic comparison across hyperparameter configurations. The params.yaml-driven pipeline ensured that observed performance differences were attributable to specific parameter changes rather than environmental variability. The model registry and staging workflow enabled seamless model updates without modifying Flask application code.

The HTML/CSS/JS frontend demonstrated stable end-to-end operation from YouTube URL input through comment fetching, prediction, and visualization rendering. The temporal trend graph proved particularly useful for detecting audience sentiment shifts following video publication events.

The primary limitation of the current system is the dataset domain mismatch: the training corpus originates from a Reddit comment dataset rather than a YouTube-specific labeled corpus. This stylistic divergence may reduce real-world classification accuracy on YouTube comment patterns. Additionally, the TF-IDF representation does not capture contextual or semantic relationships beyond fixed n-gram windows, which limits performance on highly sarcastic or context-dependent comments.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.135107

10. CONCLUSION

This paper presented a complete end-to-end sentiment analysis system for YouTube comments integrating modular data ingestion, NLTK-based negation-preserving preprocessing, TF-IDF trigram feature engineering, a LightGBM multiclass classifier, MLflow experiment tracking, Flask REST API deployment, and a vanilla HTML/CSS/JS frontend application. The proposed system achieved 89.4% overall accuracy and 88.7% macro-averaged F1-score across three sentiment classes while maintaining low inference latency suitable for real-time deployment.

Comparative experimentation across nine algorithms confirmed LightGBM as the optimal model for this task given the deployment constraints. The modular cookiecutter-style project structure provides a replicable template for applied NLP classification pipelines that integrate both machine learning engineering and web-based deployment.

11. FUTURE WORK

Future research directions include the following:

- Collection and annotation of a YouTube-specific comment corpus to resolve the Reddit-to-YouTube training domain mismatch.
- Fine-tuning of transformer-based architectures such as BERT and RoBERTa on YouTube comment data to improve accuracy on sarcasm and context-dependent sentiment expressions.
- Implementation of aspect-based sentiment analysis (ABSA) to identify sentiment directed toward specific topics or entities mentioned within comments.
- Application of advanced class imbalance handling techniques including Synthetic Minority Oversampling Technique (SMOTE) and focal loss.
- Containerization of the full stack using Docker and orchestration via Kubernetes for scalable cloud deployment.
- Integration of sarcasm detection and multilingual sentiment modules to improve performance across diverse YouTube communities.

ACKNOWLEDGEMENT

The authors would like to acknowledge Prof. Reema Kalda, Assistant Professor, Department of Computer Engineering, Bapusaheb Shivajirao Deore College of Engineering, Dhule, for her valuable guidance, continuous support, and insightful feedback throughout the development of this project.

REFERENCES

[1] B. Pang, L. Lee, "Opinion Mining and Sentiment Analysis," Foundations and Trends in Information Retrieval, vol. 2, no. 1-2, pp. 1-135, 2008.
[2] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, R. Passonneau, "Sentiment Analysis of Twitter Data," Proc. Workshop on Languages in Social Media, ACL, pp. 30-38, 2011.
[3] T. Chen, C. Guestrin, "XGBoost: A Scalable Tree Boosting System," Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785-794, 2016.
[4] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," Advances in Neural Information Processing Systems, vol. 30, pp. 3146-3154, 2017.
[5] J. Devlin, M. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proc. NAACL-HLT, ACL, pp. 4171-4186, 2019.
[6] P. Bhardwaj, A. Singh, "Comparative Study of Machine Learning Algorithms for Sentiment Analysis," International Journal of Engineering Research and Technology, vol. 9, no. 5, 2020.
[7] S. Siersdorfer, S. Chelaru, W. Nejdl, J. San Pedro, "How Useful Are Your Comments? Analyzing and Predicting YouTube Comments and Comment Ratings," Proc. 19th International Conference on World Wide Web, ACM, pp. 891-900, 2010.
[8] S. Bird, E. Klein, E. Loper, Natural Language Processing with Python, O'Reilly Media, 2009.
[9] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, C. Potts, "Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank," Proc. EMNLP, pp. 1631-1642, 2013.
[10] G. Salton, C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," Information Processing and Management, vol. 24, no. 5, pp. 513-523, 1988.
[11] M. Zaharia et al., "Accelerating the Machine Learning Lifecycle with MLflow," IEEE Data Engineering Bulletin, vol. 41, no. 4,

