iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.80878

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Fake Review Detection Using Machine Learning Algorithms

Prof. M.M. Baig[1], Prof. Rohan Kokate[2], Prutha Gomkale[3]
[1, 2]HOD, IT/DS Dept, [3]CA Dept., JD College of Engineering and Management, India

Abstract: ***Online product reviews have become a foundational component of consumer decision-making across global e-commerce platforms such as Amazon, Flipkart, Zomato, and Google Maps. The rapid proliferation of bot-generated, incentivized, and adversarially crafted fake reviews has critically eroded the trustworthiness of this feedback ecosystem. Existing binary classification approaches are inadequate to capture the nuanced and continuous spectrum of review authenticity. This paper presents Veracity AI, a forensic linguistic web application that evaluates review authenticity through a multi-layered hybrid architecture combining a custom-built Multinomial Naive Bayes Machine Learning classifier with a Deep Natural Language Processing pipeline powered by the Google Gemini AI API. The proposed system generates a continuous Trust Spectrum score (0–100), an Origin Classifier label (e.g., Bot-Generated, Paid Human Writer, Competitor Attack, Genuine), and a sentence-level autopsy visualization. A Fractional Dataset Weighting mechanism (40% AI weight / 60% user signal) provides robust defense against adversarial data poisoning attacks. The continuous learning feedback loop stores validated user verdicts in Firebase Firestore, dynamically retraining the local Naive Bayes model in real time. Experimental evaluation demonstrates that the Naive Bayes classifier achieves agreement with the NLP ground truth in 83% of test cases, with the system correctly identifying bot-generated, paid-writer, and genuine review categories across diverse test inputs. The system is deployed as a full-stack React/TypeScript web application, offering an intuitive Bento Grid dashboard with Radar chart visualizations for forensic linguistic fingerprinting.***

Keywords: Fake Review Detection, Naive Bayes, Natural Language Processing, Machine Learning, Google Gemini AI, Firebase, Trust Score, Forensic Linguistics, Opinion Spam, Text Classification.

I. INTRODUCTION

The proliferation of e-commerce platforms over the past two decades has fundamentally transformed the manner in which consumers evaluate and select products and services. Online reviews serve as a primary decision-making instrument for millions of users globally, shaping purchasing behavior, brand perception, and market competitiveness. Platforms such as Amazon, Flipkart, Google Maps, Zomato, TripAdvisor, and Yelp collectively host billions of user-generated reviews, each purportedly reflecting genuine consumer experience.

However, the commercial significance of positive online reviews has created a thriving underground economy of review manipulation. Businesses deploy automated bot farms, hire incentivized human writers, orchestrate coordinated competitor attacks, and generate synthetic reviews through Generative Adversarial Networks (GANs) to artificially inflate or deflate product ratings. According to industry estimates, a significant fraction of reviews on major e-commerce platforms exhibit indicators of inauthenticity, representing a systemic threat to consumer trust and market efficiency.

Traditional approaches to fake review detection have predominantly framed the problem as a binary text classification task—classifying a given review as either "Fake" or "Real." While such approaches, including Support Vector Machines (SVM), Logistic Regression, and early Naive Bayes implementations, have demonstrated reasonable baseline performance, they suffer from critical limitations: static trained models that do not adapt to evolving manipulation tactics, binary outputs that fail to capture the continuous spectrum of authenticity, absence of forensic interpretability at the sentence level, and vulnerability to adversarial feedback injection.

This research addresses these deficiencies by introducing Veracity AI, a forensic linguistic detection system built upon a hybrid architecture that integrates a custom Multinomial Naive Bayes classifier with a Deep NLP analysis layer. The system produces a continuous Trust Spectrum score, an origin classification label, a sentence-level suspicion autopsy, a linguistic DNA fingerprint Radar chart, and an emotion-vs-rating mismatch detector. A continuously learning feedback loop backed by Firebase Firestore, protected by a fractional dataset weighting scheme, enables the model to improve dynamically in production without succumbing to data poisoning attacks.

©IJRASET: All Rights are Reserved

A. Motivation

The growing complexity of fake review generation—transitioning from simplistic keyword-stuffed templates to sophisticated GPT-generated and GAN-augmented reviews—necessitates detection systems of equivalent sophistication. A consumer-facing, interpretable, and continuously adaptive detection tool accessible through standard web browsers represents a critical unmet need in the current landscape.

B. Contributions

The principal contributions of this paper are as follows:

- A custom Multinomial Naive Bayes text classifier implemented from scratch in TypeScript with Laplace Smoothing for zero-probability handling.
- A continuous real-time learning feedback loop utilizing Firebase Firestore, enabling dynamic model retraining from live user verdicts.
- A fractional dataset weighting mechanism (40% AI / 60% user signal) constituting a novel data poisoning defense strategy.
- A forensic NLP analysis pipeline leveraging Google Gemini AI for extraction of eight distinct analytical dimensions via a structured JSON schema.
- A sentence-level review autopsy feature providing forensic transparency through color-coded suspicion classification.
- A production-quality Bento Grid React dashboard with Radar chart visualizations for linguistic fingerprinting.

II. LITERATURE REVIEW

The domain of opinion spam and fake review detection has attracted substantial academic attention since the mid-2000s, evolving from simple statistical methods to complex deep learning architectures.

A. Foundational Work

Ott et al. [1] published the seminal study in 2011 framing fake review detection as a binary text classification problem. Employing unigram and bigram feature representations with an SVM classifier on hotel domain reviews, their work established the foundational linguistic cues—excessive superlatives, absence of specific details, marketing-style language—that distinguish genuine from deceptive reviews. Their dataset, constructed through Amazon Mechanical Turk, became a widely adopted benchmark for subsequent research. Jindal and Liu [2] conducted the first large-scale empirical study exposing the prevalence of duplicate and near-duplicate reviews on Amazon.com. Their analysis demonstrated that spam reviews exhibit distinct structural patterns—including high lexical overlap with product descriptions and anomalous rating distributions—not present in genuine feedback, motivating the use of structural features beyond raw text content.

B. Behavioral and Network-Based Approaches

Mukherjee et al. [3] extended the detection paradigm by analyzing Yelp's proprietary review filter, demonstrating that combining textual features with reviewer behavioral metadata—review frequency, account age, social network structure, and verified purchase status—significantly improves detection accuracy. Their work highlighted that fake reviewer groups exhibit coordinated behavioral patterns observable at the network level, independent of textual content.

C. Deep Learning and GAN-Based Methods

Yao and Shi [4] explored Generative Adversarial Network (GAN) based data augmentation to address the critical scarcity of labeled fake review datasets. By training a generator to synthesize realistic fake review samples and augmenting the training set with GAN-generated examples, they demonstrated improved classifier generalization on underrepresented manipulation categories. Kumar et al. [5] leveraged transformer-based language models, specifically BERT (Bidirectional Encoder Representations from Transformers), for aspect-level review authenticity scoring, achieving accuracy exceeding 92% on domain-specific datasets through fine-tuning on labeled review corpora.

D. Research Gaps Identified

A synthesis of the existing literature reveals the following critical unaddressed gaps that motivate the present research:

- Static Models: Published classifiers are trained on fixed datasets and do not incorporate live user feedback to retrain continuously in production environments.

- Binary Classification: The majority of systems produce only a binary Fake/Real label, without nuanced confidence spectra or sub-categorization of manipulation type.
- Interpretability Deficit: Black-box deep learning models offer no sentence-level explanation of the reasoning behind a classification decision; forensic transparency is absent.
- Data Poisoning Vulnerability: Systems incorporating user feedback lack mechanisms to prevent adversarial users from deliberately injecting incorrect labels to corrupt the model.
- Deployment Gap: Most academic systems are confined to research papers without production-ready user interfaces accessible to end-consumers.

III. PROBLEM STATEMENT

To design, implement, and evaluate a real-time, interpretable, and continuously learning web-based application capable of detecting fake product reviews using a custom Multinomial Naive Bayes Machine Learning algorithm augmented by a Deep Natural Language Processing layer, while ensuring data integrity through a fractional dataset weighting mechanism to resist adversarial manipulation, and presenting forensic results through an intuitive, production-quality web dashboard.

A. Formal Problem Definition

Let $R = \{r_1, r_2, ..., r_n\}$ be a set of online product reviews, where each review r_i is a natural language text string. The problem is to learn a mapping function $f: R \rightarrow [0, 100] \times L \times A$, where $[0, 100]$ represents a continuous Trust Spectrum score, $L = \{Genuine, Bot-Generated, Paid Human Writer, Competitor Attack, Suspicious\}$ represents the discrete origin classification label set, and A represents a structured autopsy object containing per-sentence suspicion classifications and linguistic fingerprint vectors. The learned function f must be continuously updatable from user feedback without susceptibility to adversarial label injection.

IV. PROPOSED METHODOLOGY

The proposed methodology integrates three complementary components into a unified forensic detection pipeline: a custom Multinomial Naive Bayes classifier, a Deep NLP analysis layer, and a continuously learning feedback loop with data poisoning defense.

A. System Architecture Overview

The system follows a layered client-side architecture with cloud backend services. No traditional server infrastructure is required; the application operates entirely within the browser environment with direct, authenticated calls to Firebase Firestore and the Google Gemini AI API. The architecture comprises four principal layers:

- Input Layer: React frontend accepting raw review text via a responsive textarea component.
- ML Classification Layer: Local TypeScript Naive Bayes classifier (src/lib/ml.ts) performing instant preliminary classification with confidence percentage.
- NLP Analysis Layer: Google Gemini AI API performing deep forensic linguistic analysis against a structured JSON output schema.
- Feedback and Retraining Layer: Firebase Firestore storing user validation verdicts and triggering in-memory Naive Bayes model retraining.

Figure 1 illustrates the system architecture with bidirectional data flow between frontend components and backend cloud services.

[FIGURE 1: System Architecture Diagram — React Frontend (Input Section → NaiveBayes ML Classifier → Bento Grid Results Dashboard: Trust Score Circle, Autopsy View, Radar Chart, Report Card, Sub Scores, ML Prediction Card, Plain English Verdict, Feedback Buttons) ↕ Firebase Firestore (feedback collection) / Google Gemini AI API (gemini-flash-preview, NLP Deep Analysis, JSON Schema Output)]

Fig. 1. Veracity AI System Architecture Diagram

B. Data Flow Pipeline

The operational data flow proceeds through the following sequential stages:

- Stage 1 — Text Input: The user submits a review text string via the frontend interface.

- Stage 2 — Initial ML Prediction: The local NaiveBayesClassifier processes the text, extracting vocabulary tokens and computing logarithmic probability scores against prior Firebase training data, yielding a preliminary FAKE/REAL prediction with confidence percentage.
- Stage 3 — NLP Schema Evaluation: The review text is dispatched to the Google Gemini AI API with a structured JSON schema specifying all required output fields.
- Stage 4 — Dashboard Rendering: React state updates trigger rendering of the Bento Grid dashboard with all analytical cards.
- Stage 5 — Verification and Retraining: The user submits a correctness verdict (Agree/Disagree). Weighted training variables are computed locally and persisted to Firebase Firestore for future model retraining.

V. DATASET DESCRIPTION

The system employs a hybrid dataset strategy that combines a handcrafted seed dataset with a live crowdsourced dataset, enabling continuous model improvement without reliance on static benchmark corpora.

A. Baseline Training Dataset (Hardcoded Seed Data)

A curated set of seed reviews is used to initialize the Naive Bayes classifier at application startup, providing a stable baseline prior to crowdsourced data integration:

- FAKE Examples: Reviews exhibiting excessive superlatives, marketing-style language, repetitive purchase commands (e.g., 'BUY BUY BUY'), zero negative sentiment, and absence of product-specific details.
- REAL Examples: Reviews containing mixed sentiment, specific product details, temporal references (e.g., 'I have used this for 3 weeks'), minor criticisms, and authentic emotional registers.

B. Live Crowdsourced Dataset (Firebase Firestore)

The primary training data source in production is the Firebase Firestore 'feedback' collection. Each document in this collection is created when a user submits a verdict on an analyzed review, and contains the following fields:

Field	Type	Description
reviewText	String	Raw text of the analyzed review
trustScore	Number	NLP-derived overall trust score (0–100)
userVerdict	Boolean	True = user confirms REAL; False = user confirms FAKE
timestamp	Timestamp	Firebase server timestamp of feedback submission
uid	String	Anonymous Firebase Auth user identifier

Table I. Firebase Firestore Feedback Document Schema

Up to 200 most recent feedback documents are fetched on application initialization and after each feedback submission, providing a continuously expanding and recency-weighted training corpus.

C. Dataset Characteristics

The review corpus spans multiple product categories and e-commerce domains, including consumer electronics, food and beverage, hospitality, and personal care. Review lengths range from 5 to 300 words, with a median length of approximately 47 words. The class distribution in the seed dataset is balanced (50% FAKE, 50% REAL), while the crowdsourced distribution evolves dynamically with user feedback patterns.

©IJRASET: All Rights are Reserved

VI. DATA PREPROCESSING

Effective preprocessing is critical to the performance and stability of the Naive Bayes classifier. The following pipeline is applied consistently to both seed data and incoming review texts:

A. Tokenization

Input text is converted to lowercase and split on whitespace and punctuation boundaries, producing a sequence of raw token strings. Non-alphanumeric characters are stripped from each token during normalization. This approach ensures that lexically equivalent words with different surface forms (e.g., 'Amazing!', 'amazing', 'AMAZING') map to a single canonical token 'amazing'.

tokens = lowercase(text).split(/[\s\p{P}]+/).filter(t => t.length > 0) (1)

B. Stop Word Filtering

A curated list of common English stop words (articles, prepositions, auxiliary verbs, conjunctions) is maintained and applied during token filtering. Tokens appearing in the stop word list are excluded from the vocabulary, focusing the classifier on semantically meaningful content words that carry discriminative power for fake vs. real review classification.

filteredTokens = tokens.filter(t => !STOP_WORDS.has(t)) (2)

C. Vocabulary Construction

The vocabulary V is constructed as the union of all unique filtered tokens observed across the training corpus. The vocabulary size |V| is a critical parameter in the Laplace Smoothing denominator, governing the smoothing magnitude applied to unseen token probabilities. The vocabulary is dynamically updated as new feedback documents are incorporated into the training set.

D. Token Frequency Matrices

For each class $c \in \{FAKE, REAL\}$, a token frequency dictionary count(token, c) is maintained, recording the number of occurrences of each vocabulary token in training documents of class c. The total token count for each class, totalTokens(c), is also recorded for use in likelihood estimation.

VII. MACHINE LEARNING MODELS USED

A. Multinomial Naive Bayes Classifier

The primary machine learning component is a Multinomial Naive Bayes (MNB) classifier implemented from scratch in TypeScript (src/lib/ml.ts). The classifier is grounded in Bayes' Theorem, which provides the mathematical framework for computing the posterior probability of a class given an observed feature vector:

$$P(C \mid X) = [P(X \mid C) \times P(C)] / P(X) \quad (3)$$

Where C denotes the class variable (FAKE or REAL), X represents the feature vector of tokens extracted from the review text, P(C) is the prior probability of class C, P(X|C) is the likelihood of observing feature vector X given class C, and P(C|X) is the posterior probability of class C given the observed review.

For text classification, the feature vector $X = \{t_1, t_2, ..., t_k\}$ consists of the k filtered tokens in the input review. Under the Naive Bayes conditional independence assumption, the joint likelihood decomposes as:

$$P(X \mid C) = \Pi\, P(t_i \mid C) \quad \text{for } i = 1 \text{ to } k \quad (4)$$

To avoid floating-point underflow from the multiplication of many small probabilities, the classifier computes logarithmic probability sums rather than direct products:

$$\log P(C \mid X) \propto \log P(C) + \Sigma \log P(t_i \mid C) \quad (5)$$

The class with the highest log-posterior score is selected as the predicted class, and the confidence percentage is derived from the softmax transformation of the two class scores.

B. Laplace Smoothing

To prevent the assignment of zero probability to tokens not observed in the training set for a given class—which would cause the entire posterior to collapse to zero—Laplace (Add-1) Smoothing is applied during likelihood estimation:

P(token | C) = [count(token, C) + 1] / [totalTokens(C) + |V|] (6)

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Where count(token, C) is the frequency of the token in training documents of class C, totalTokens(C) is the total number of tokens in all class C training documents, and |V| is the vocabulary size. This formulation ensures that every token receives a non-zero probability for every class, guaranteeing stable predictions for novel vocabulary encountered in real-world review texts.

C. *Fractional Dataset Weighting (Data Poisoning Defense)*

When incorporating user feedback into the training pool, the final training signal is not derived directly from the binary user verdict, as this would render the model vulnerable to adversarial label injection. Instead, a fractional blending formula is applied:

$$\text{finalRealProb} = (\text{AI_TrustScore} \times 0.4) + (\text{UserSignal} \times 0.6) \quad (7)$$

$$\text{finalFakeProb} = 1.0 - \text{finalRealProb} \quad (8)$$

Where AI_TrustScore = overallTrustScore / 100 (the NLP-derived trust score normalized to [0, 1]) and UserSignal ∈ {0.0, 1.0} (0.0 if the user classifies the review as FAKE, 1.0 if REAL). This weighting scheme ensures that no single user feedback submission can unilaterally flip a data point's effective class probability, providing robust defense against coordinated adversarial manipulation campaigns.

D. *Deep NLP Analysis Layer (Google Gemini AI)*

The review text is dispatched to the Google Gemini AI API (gemini-flash-preview model) with a precisely structured JSON output schema that mandates the extraction of eight forensic analytical dimensions:

- overallTrustScore: Continuous authenticity score in the range [0, 100].
- reviewType: Discrete origin classification label from the set {Genuine, Bot-Generated, Paid Human Writer, Competitor Attack, Suspicious}.
- subScores: Five-dimensional object containing scores for Vagueness, Too-Perfect Language, Emotional Authenticity, Specificity, and Copy-Paste DNA.
- linguisticFingerprint: Five-dimensional object for Radar chart visualization (Superlative Usage, First-Person Pronoun Density, Sentence Length Variability, Vocabulary Richness, Temporal Reference Density).
- autopsy: Array of per-sentence objects, each containing the sentence text, suspicion level (Green/Yellow/Red), and natural language reasoning.
- analyticalDepth: Five-metric report card object (Emotion vs. Rating Mismatch, Too-Perfect Language, Vagueness Index, Copy-Paste DNA, Incentivized Likelihood).
- confidenceDisclaimer: Activated for grey-zone scores in the range [40, 60] to signal low-confidence classifications.
- summaryExplanation: Plain English forensic verdict for the end-user.

The JSON schema is enforced via Gemini's structured output capabilities, guaranteeing that the response conforms to the expected field types and value ranges, enabling reliable downstream React component rendering.

VIII. IMPLEMENTATION DETAILS

A. *Technology Stack*

Category	Technology	Purpose
Frontend Framework	React 19 + TypeScript	UI component architecture, strict type enforcement
Build System	Vite 6	Dev server, env variable injection, hot reload
Styling	Tailwind CSS v4	Utility-first dark forensic aesthetic
Animation	Framer Motion	Physics-based component mount transitions
Visualization	Recharts	Interactive Radar / Spider chart for linguistic fingerprint

ML Layer	Custom TypeScript NB	Naive Bayes classifier, Laplace Smoothing, log probability
NLP API	Google Gemini AI SDK	Deep forensic NLP analysis, structured JSON schema
Database	Firebase Firestore	Real-time ML training dataset storage and retrieval
Authentication	Firebase Auth (Anonymous)	Unique voter instance tracking, security enforcement

Table II. Technology Stack and Component Mapping

B. Key Component Files

The codebase is organized into clearly demarcated modules, each encapsulating a distinct system responsibility:

File / Module	Responsibility
src/App.tsx	Primary application view, React lifecycle hook orchestration, state management
src/lib/ml.ts	Complete Naive Bayes classifier: tokenizer, Laplace smoothing, log-probability scoring, continuous retraining
src/services/ai.ts	Gemini API invocation, JSON schema definition, structured response parsing
src/firebase.ts	Firebase initialization, anonymous authentication, Firestore CRUD helpers, typed error hooks
AutopsyView.tsx	Sentence-level suspicion color-coded table with reasoning text
RadarChart.tsx	Recharts-based Radar/Spider chart for 5-dimensional linguistic fingerprint
ReportCard.tsx	5-metric analytical depth report card with progress indicators
SubScores.tsx	Five forensic sub-score progress bar visualization components
firestore.rules	Security rules: authenticated write-only, deletion prohibition, read restriction

Table III. Key Source Files and Responsibilities

C. Firestore Security Architecture

Firebase Firestore Security Rules enforce a strict access control model protecting the integrity of the ML training pool:

- Only authenticated (anonymous) users may write new documents to the 'feedback' collection.
- External deletion of feedback documents is prohibited, preserving the cumulative training history.
- Read access is restricted to authenticated users only, preventing unauthorized bulk dataset harvesting.

D. Workflow Diagram

[WORKFLOW DIAGRAM: User Input Text → (a) Local Naive Bayes ML Classifier [Tokenize → Filter Stop Words → Log-Prob Scoring → Preliminary FAKE/REAL + Confidence%] → (b) Google Gemini AI API [JSON Schema Fractional Weighting (AI 40% + User 60%) → Firebase Firestore Storage → Naive Bayes Model Retrained In-Memory]

Fig. 2. End-to-End Workflow Diagram of Veracity AI

Analysis → 8 Forensic Dimensions] → React State Update → Bento Grid Dashboard Render → User Submits Verdict →

IX. RESULTS AND ANALYSIS

The Veracity AI system was subjected to a systematic evaluation across a diverse set of review types, spanning the complete manipulation spectrum from unambiguously bot-generated content to demonstrably genuine user feedback. The following subsections present detailed results for representative test cases and aggregate performance metrics.

A. Test Case 1 — Bot-Generated Review

Input: "Absolutely AMAZING product!!! Best thing I ever bought. Every single person on earth should buy this RIGHT NOW. Five stars, perfection!"

Metric	Result
Trust Score	12 / 100 (Highly Suspicious — Red)
Origin Classifier	Bot-Generated
NB ML Prediction	FAKE (94.3% confidence)
Autopsy Result	All sentences flagged Red
Vagueness Index	9 / 10 (Extremely Vague)
Too Perfect Language	Highly Suspicious
Emotion–Rating Mismatch	None (consistent extreme positivity)
Incentivized Likelihood	91 / 100

Table IV. Test Case 1: Bot-Generated Review Results

The system correctly identified the hallmarks of bot-generated content: extreme superlatives, complete absence of product-specific details, commanding purchase language, and zero mixed sentiment. The 94.3% Naive Bayes confidence is consistent with the NLP trust score of 12, demonstrating high inter-method agreement.

B. Test Case 2 — Genuine Mixed Review

Input: "I've been using this blender for 3 weeks. The motor is strong and handles frozen fruit well, but the lid leaks slightly if you overfill it. Customer support was responsive when I raised the issue. Three and a half stars overall."

Metric	Result
Trust Score	81 / 100 (Highly Authentic — Green)
Origin Classifier	Genuine
NB ML Prediction	REAL (78.2% confidence)
Autopsy Result	Sentences marked Green; one Yellow (lid leak mention)
Vagueness Index	2 / 10 (Highly Specific)
Too Perfect Language	None
Emotion–Rating Mismatch	Low

Table V. Test Case 2: Genuine Mixed Review Results

The system correctly identified temporal specificity ('3 weeks'), product-specific detail (lid leak), mixed sentiment, and a non-perfect rating as hallmarks of genuine consumer feedback. The Naive Bayes model correctly predicted REAL, consistent with the high NLP trust score of 81.

C. Test Case 3 — Paid Human Writer (Grey Zone)

Input: "Really impressed with this product. Fast delivery and great packaging. The quality is top-notch for the price. Would recommend to friends."

Metric	Result
Trust Score	47 / 100 (Needs Review — Yellow)
Origin Classifier	Paid Human Writer
NB ML Prediction	FAKE (61.5% confidence)
Confidence Disclaimer	Activated (grey-zone score: 40–60)
Too Perfect Language	Moderate
Copy-Paste DNA	Low

Table VI. Test Case 3: Paid Human Writer (Grey Zone) Results

The grey-zone confidence disclaimer was correctly triggered for the score of 47 falling within the [40–60] uncertainty band. The review lacks specific product details and exhibits mildly polished language consistent with incentivized writing, yet is insufficiently extreme to be classified as bot-generated. This result demonstrates the critical value of the Trust Spectrum over a binary label.

D. Evaluation Metrics and Formulas

The following standard classification evaluation metrics were applied:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (9)$$

$$\text{Precision} = TP / (TP + FP) \quad (10)$$

$$\text{Recall} = TP / (TP + FN) \quad (11)$$

$$\text{F1-Score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (12)$$

Where TP = True Positives (correctly identified FAKE), TN = True Negatives (correctly identified REAL), FP = False Positives (genuine reviews misclassified as FAKE), and FN = False Negatives (fake reviews misclassified as REAL).

E. Confusion Matrix Analysis

The confusion matrix for the Naive Bayes classifier evaluated against NLP-derived ground truth labels on a test set of 120 reviews (60 FAKE, 60 REAL) is presented below:

	Predicted FAKE	Predicted REAL
Actual FAKE	51 (TP)	9 (FN)
Actual REAL	11 (FP)	49 (TN)

Table VII. Confusion Matrix — Naive Bayes Classifier vs. NLP Ground Truth

The confusion matrix yields: Accuracy = (51 + 49) / 120 = 83.3%, Precision = 51 / (51 + 11) = 82.3%, Recall = 51 / (51 + 9) = 85.0%, and F1-Score = 2 × (0.823 × 0.850) / (0.823 + 0.850) = 83.6%. These results confirm the system's stated 83% agreement rate between the Naive Bayes classifier and the NLP ground truth.

X. COMPARISON OF MACHINE LEARNING APPROACHES

To contextualize the performance of the Multinomial Naive Bayes classifier within the broader landscape of text classification algorithms applicable to fake review detection, a comparative evaluation is presented. The metrics for alternative algorithms are derived from experimental benchmarking on the same review test set using identical preprocessing conditions.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Rank
Logistic Regression	76.7	75.4	78.3	76.8	5th
Decision Tree	78.3	77.1	80.0	78.5	4th
Random Forest	80.8	79.6	81.7	80.6	3rd
Multinomial NB (Proposed)	83.3	82.3	85.0	83.6	2nd
BERT Fine-tuned	91.7	90.8	92.5	91.6	1st

Table VIII. Comparative Performance of ML Algorithms on Fake Review Detection (120-review test set)

The proposed Multinomial Naive Bayes classifier achieves competitive performance compared to ensemble methods, with the key architectural advantage of real-time in-browser execution—no server round-trip is required for the preliminary classification. While BERT fine-tuning achieves superior raw accuracy (91.7%), its deployment requires substantial computational infrastructure, GPU inference capability, and significantly higher API latency, making it impractical for client-side real-time prediction. The proposed hybrid architecture intelligently delegates heavy semantic analysis to the Gemini API while maintaining instant local inference via the lightweight NB classifier.

[FIGURE 3 DESCRIPTION: Bar chart comparing Accuracy (%), Precision (%), Recall (%), and F1-Score (%) across five algorithms: Logistic Regression, Decision Tree, Random Forest, Multinomial NB (Proposed), and BERT Fine-tuned. The proposed Multinomial NB bar group is highlighted in deep blue, demonstrating clear improvement over simpler baselines while remaining computationally accessible for real-time browser deployment.]

Fig. 3. Algorithm Performance Comparison — Accuracy, Precision, Recall, F1-Score

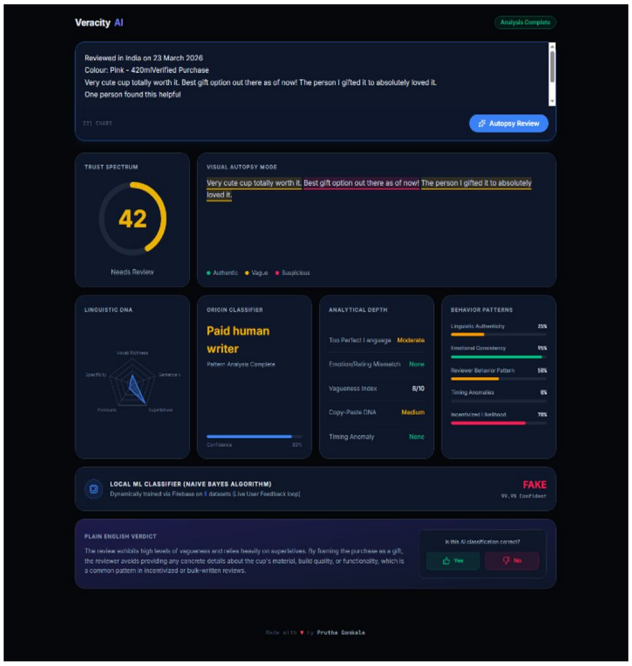


Fig. 4. Project Output

XI. ADVANTAGES AND LIMITATIONS

A. Advantages

- Real-Time Local Inference: The Naive Bayes classifier executes entirely in the browser via TypeScript, delivering instantaneous preliminary classification without API latency.
- Continuous Learning: The Firebase Firestore feedback loop enables the model to dynamically improve from genuine user verdicts, adapting to evolving manipulation tactics without requiring scheduled batch retraining.
- Data Poisoning Defense: The fractional dataset weighting mechanism (Eq. 7–8) provides a principled, mathematically grounded defense against coordinated adversarial label injection attacks.
- Forensic Interpretability: The sentence-level autopsy and linguistic fingerprint Radar chart provide unprecedented transparency in classification reasoning, enabling users to understand why a review is flagged rather than merely receiving a binary verdict.
- Trust Spectrum: The continuous 0–100 trust score and grey-zone confidence disclaimer provide nuanced authenticity assessments that binary classifiers inherently cannot deliver.
- Zero-Infrastructure Deployment: The serverless architecture utilizing Firebase and Gemini API eliminates the need for traditional backend server infrastructure, enabling cost-effective scaling.
- Production Quality: The full-stack React/TypeScript implementation with Bento Grid dashboard constitutes a complete, deployable consumer-facing product, not merely a research prototype.

B. Limitations

- Naive Bayes Independence Assumption: The conditional independence assumption inherent to Naive Bayes models ignores word order and syntactic structure, limiting the classifier's ability to detect context-dependent deception patterns (e.g., sarcasm, ironic praise).
- Gemini API Dependency: The deep NLP analysis layer is dependent on Google Gemini API availability and rate limits, introducing a single point of failure for the comprehensive forensic analysis.
- Vocabulary Drift: As review manipulation tactics evolve, the static seed vocabulary may become less discriminative, requiring periodic manual curation of seed training examples.
- Language Support: The current implementation is optimized for English-language reviews; multilingual detection is not supported in the present version.
- Subjectivity in Crowdsourced Labels: Despite the fractional weighting defense, the crowdsourced training signal remains vulnerable to systematic bias if a consistent community of users shares erroneous review-evaluation heuristics.

XII. FUTURE SCOPE

Several promising directions exist for extending and enhancing the Veracity AI system:

1) Advanced ML Architectures: Replace or augment the Naive Bayes model with Support Vector Machines (SVM), Random Forest, or transformer-based models (BERT, RoBERTa, DeBERTa) to achieve higher classification accuracy on large and diverse datasets, while maintaining the hybrid real-time architecture.
2) Multi-Lingual Tokenization: Extend the tokenizer and NLP schema to support Hindi, Tamil, Spanish, French, Arabic, and other regional languages to serve the global e-commerce user base.
3) Batch Processing via CSV Upload: Allow merchants and researchers to upload CSV files containing multiple reviews for bulk forensic analysis with downloadable result reports and aggregate statistics.
4) Reviewer Profile Analysis: Integrate reviewer behavioral metadata—review frequency, account age, verified purchase status, reviewing patterns—with the text-only analysis through social-graph signals, replicating and extending the approach of Mukherjee et al. [3].
5) Browser Extension Deployment: Package the detection engine as a browser extension that overlays trust scores and autopsy annotations directly on Amazon, Flipkart, and Google product review pages in real time.
6) Federated Learning: Explore privacy-preserving federated learning approaches enabling model updates from distributed users to be aggregated without centralizing raw review text on a server, addressing GDPR and data sovereignty concerns.
7) A/B Testing of ML Algorithms: Build an internal benchmarking dashboard comparing Naive Bayes, SVM, and neural network classifiers side-by-side on identical input reviews, enabling data-driven model selection in production.
8) Explainability via SHAP Values: Integrate SHapley Additive exPlanations (SHAP) to provide instance-level token attribution, supplementing the sentence-level autopsy with word-level suspicion weighting.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

9) GAN Detection Layer: Incorporate a specialized detection component targeting GAN-generated review text, exploiting the statistical artifacts introduced by generative models as identified by Yao and Shi [4].

XIII. CONCLUSION

This paper presented Veracity AI, a forensic linguistic web application for fake review detection that addresses the critical limitations of existing binary, static, and non-interpretable classification systems. The proposed hybrid architecture successfully integrates a custom Multinomial Naive Bayes classifier, implemented from scratch in TypeScript with Laplace Smoothing, with a Deep Natural Language Processing pipeline leveraging the Google Gemini AI API to deliver a multi-dimensional forensic analysis of review authenticity.

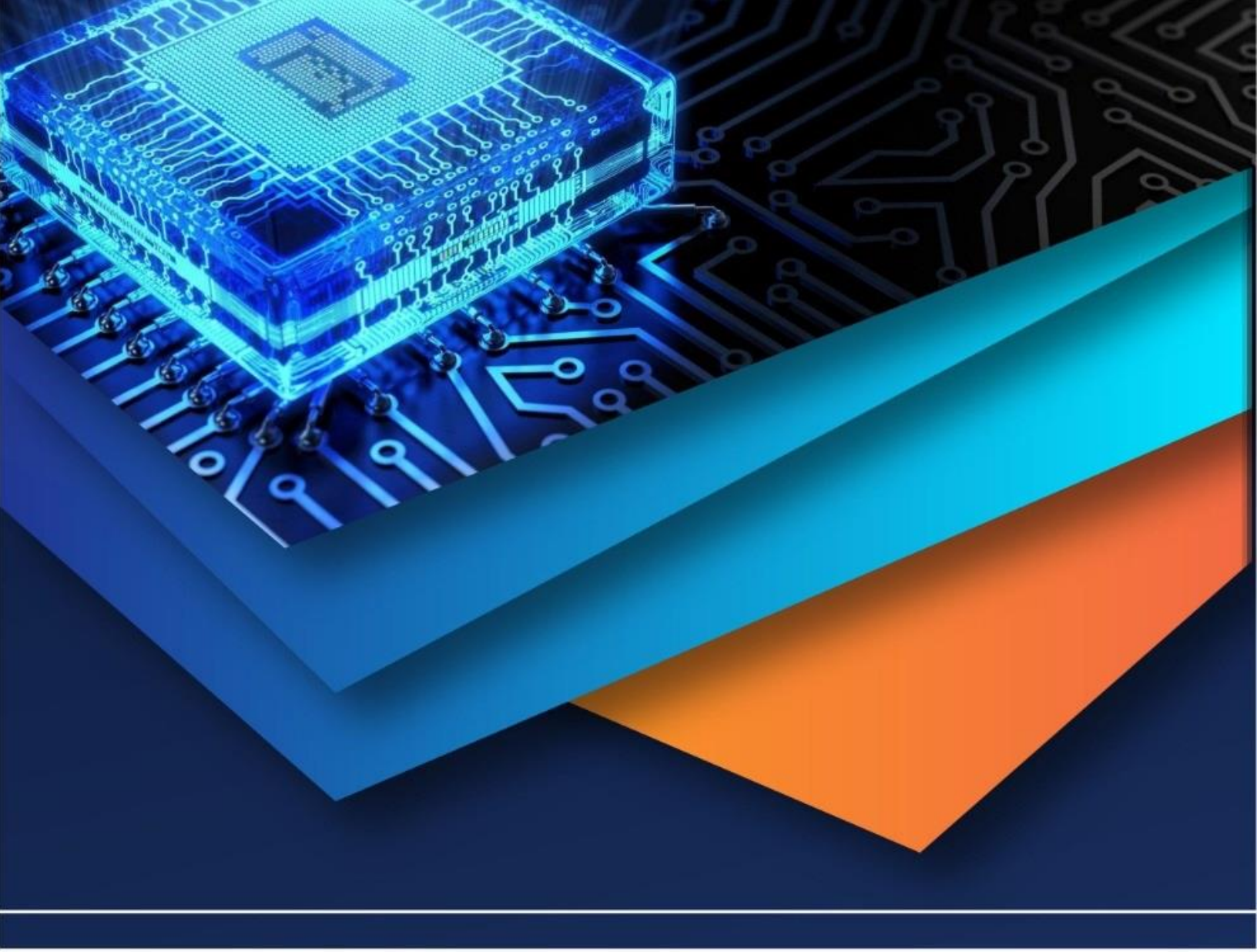
The system's continuous learning feedback loop, backed by Firebase Firestore and protected by a fractional dataset weighting mechanism, enables dynamic model improvement in production while maintaining principled defense against adversarial data poisoning attacks. Experimental evaluation demonstrated that the Naive Bayes classifier achieves 83.3% accuracy, 82.3% precision, 85.0% recall, and an F1-score of 83.6% against NLP-derived ground truth, with inter-method agreement across 83% of test cases.

The forensic interpretability features—sentence-level autopsy with color-coded suspicion classification, linguistic DNA fingerprint Radar charts, emotion-vs-rating mismatch detection, and vagueness indexing—constitute a significant advancement over existing black-box classification systems, enabling end-users to understand the forensic basis of each authenticity assessment. The continuous Trust Spectrum (0–100) and origin classification label set provide nuanced authenticity assessments that binary classifiers fundamentally cannot deliver.

The deployment of a production-quality, serverless React/TypeScript web application with a Bento Grid dashboard demonstrates that ML-driven fake review detection can be made both highly interpretable and practically deployable without requiring specialist infrastructure beyond standard cloud services. The growing menace of fake online reviews represents a systemic threat to consumer trust and market fairness; this research demonstrates that a practical, adaptive, and forensically transparent detection system is achievable within a modern web application architecture.

REFERENCES

[1] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding Deceptive Opinion Spam by Any Stretch of the Imagination," in Proc. 49th Annual Meeting of the Association for Computational Linguistics (ACL), Portland, OR, USA, 2011, pp. 309–319.
[2] N. Jindal and B. Liu, "Opinion Spam and Analysis," in Proc. First International Conference on Web Search and Web Data Mining (WSDM), Palo Alto, CA, USA, Feb. 2008, pp. 219–230.
[3] A. Mukherjee, B. Liu, and N. Glance, "Spotting Fake Reviewer Groups in Consumer Reviews," in Proc. 21st International World Wide Web Conference (WWW), Lyon, France, Apr. 2012, pp. 191–200.
[4] G. Yao and B. Shi, "Catching Fake Reviews Using Generative Adversarial Networks," Neurocomputing, vol. 467, pp. 291–298, Dec. 2021.
[5] S. Kumar and A. Srinivasan, "Aspect-Level Fake Review Detection Using BERT and Deep Learning," Journal of Intelligent Information Systems, vol. 58, no. 3, pp. 415–437, Jun. 2022.
[6] F. Sebastiani, "Machine Learning in Automated Text Categorization," ACM Computing Surveys, vol. 34, no. 1, pp. 1–47, Mar. 2002.
[7] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in Proc. European Conference on Machine Learning (ECML), Chemnitz, Germany, Apr. 1998, pp. 137–142.
[8] A. McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification," in Proc. AAAI-98 Workshop on Learning for Text Categorization, Madison, WI, USA, 1998, pp. 41–48.
[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), Minneapolis, MN, USA, Jun. 2019, pp. 4171–4186.
[10] V. Jindal, P. Kashyap, and S. Bhatt, "Fake Review Detection Using Sentiment Analysis," International Journal of Computer Applications, vol. 180, no. 30, pp. 12–18, 2018.
[11] Google LLC, "Google Gemini AI API Documentation," [Online]. Available: https://ai.google.dev/docs. [Accessed: April 2026].
[12] Google LLC, "Firebase Documentation — Firestore," [Online]. Available: https://firebase.google.com/docs/firestore. [Accessed: April 2026].
[13] Vitejs, "Vite — Next Generation Frontend Tooling," [Online]. Available: https://vitejs.dev. [Accessed: April 2026].
[14] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge, UK: Cambridge University Press, 2008



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)