CAUCHY – Jurnal Matematika Murni dan Aplikasi
Volume 10 (2) (2025), Pages 821-834
p-ISSN: 2086-0382; e-ISSN: 2477-3344

Reliable and Efficient Sentiment Analysis on IMDb with Logistic Regression

Diah Mariatul Ulya, Juhari, Rossima Eva Yuliana, and Mohammad Jamhuri*

Department of Mathematics, Faculty of Science and Technology,
Universitas Islam Negeri Maulana Malik Ibrahim, Malang, Indonesia

Abstract

Understanding public opinion at scale is essential for modern media analytics. We present a reproducible, leakage-safe evaluation of logistic regression (LR) for *binary* sentiment classification on the IMDb Large Movie Review dataset and compare it with five widely used baselines: multinomial Naive Bayes, linear support vector machine (SVM), decision tree, k-nearest neighbors, and random forest. Using a standardized text pipeline (HTML stripping, stopword removal, WordNet lemmatization) with TF–IDF unigrams–bigrams and *nested*, stratified cross-validation, we assess threshold-dependent and threshold-independent performance, probability calibration, and computational efficiency. LR attains the best overall balance of quality and speed, achieving 88.98% accuracy and 89.13% F1, with strong ranking performance (OOF ROC–AUC $\approx$ 0.9568; PR–AUC $\approx$ 0.9554) and well-behaved calibration (Brier $\approx$ 0.0858). Training completes in seconds per fold and CPU inference reaches $\sim 2.46\times10^6$ samples/s. While a calibrated linear SVM yields slightly higher precision, LR delivers higher F1 at markedly lower compute. These results establish LR as a robust, transparent baseline that remains competitive with more complex neural and ensemble approaches, offering a favorable performance–efficiency trade-off for practical deployment and reproducible research on IMDb sentiment classification.

Keywords: classification; IMDb; logistic regression; sentiment analysis; text mining.

1 Introduction

Online platforms concentrate public opinion about films at unprecedented scale. On the Internet Movie Database (IMDb), millions of user ratings and free-form reviews shape audience decisions and provide feedback to studios [1]. The volume, informality, and linguistic variability of these texts make manual synthesis impractical, motivating automated sentiment analysis to classify opinions reliably and at scale [2].

A standard benchmark for this task is the IMDb Large Movie Review Dataset, comprising 50,000 labeled reviews (balanced 25,000 positive / 25,000 negative) [3]. Beyond convenience, the dataset's heterogeneity and review-length dispersion stress-test both predictive performance and computational efficiency, two attributes that matter in practical deployments.

Among classical machine-learning approaches, logistic regression (LR) remains a strong baseline for *binary* text classification: it trains quickly on high-dimensional sparse features, yields

*Corresponding author. E-mail: m.jamhuri@live.com

Submitted: June 10, 2025 Reviewed: July 8, 2025 Accepted: August 20, 2025
DOI: https://doi.org/10.18860/cauchy.v10i2.33809

calibrated probabilities, and affords interpretability through feature weights. In text settings, LR is typically paired with bag-of-n-gram term frequency–inverse document frequency (TF–IDF) representations and competes closely with linear margin-based methods such as support vector machines (SVMs) on sparse, high-dimensional inputs [4], [5].

Prior studies on IMDb span both classical and deep architectures. Long short-term memory (LSTM) networks routinely exceed 80% accuracy under varied preprocessing and tuning [6]; classical methods remain competitive, with LR+RF voting ensembles approaching ∼89% accuracy [7]. However, cross-paper comparisons are often confounded by inconsistent pipelines (tokenization, stopword handling, lemmatization), featureization choices (e.g., n-gram range, vocabulary caps), evaluation protocols (single split vs. K-fold; nested vs. non-nested), decision thresholds, and occasional *train–test leakage* when vectorizers are fit on the full corpus prior to cross-validation [8], [9]. Probability-aware reporting (ROC–AUC, PR–AUC, calibration) is also less common, obscuring reliability beyond point metrics.

Here we provide a focused, *reproducible* assessment of LR on IMDb using a standardized, leakage-safe text pipeline and a nested, stratified cross-validation protocol that cleanly separates model selection from generalization estimation. We compare LR against five widely used baselines—multinomial Naive Bayes, a *calibrated* linear SVM, decision tree, k-nearest neighbors, and random forest—under matched outer folds. To ensure tractable and fair evaluation for non-linear/instance-based methods, we include a pragmatic TF–IDF → TruncatedSVD → standardization projection. Evaluation covers threshold-dependent metrics (accuracy, precision, recall, F1), threshold-independent metrics (ROC–AUC, PR–AUC), probability calibration (Brier; reliability diagrams), and computational efficiency; statistical comparisons use paired Wilcoxon tests with Holm correction.

Under this protocol, LR delivers the best overall balance of predictive quality and efficiency among the evaluated methods, achieving 88.98% accuracy and 89.13% F1, with strong threshold-independent performance (OOF ROC–AUC ≈ 0.9568, PR–AUC ≈ 0.9554) and markedly shorter training and inference times. While a calibrated linear SVM attains slightly higher precision, LR's recall and speed yield the highest F1 and a favorable performance–efficiency trade-off for practical deployment.

The remainder of this article is organized as follows. Section 2 details the dataset, text-processing pipeline, model design, and the nested cross-validation protocol, including calibration and statistical testing. Section 3 reports quantitative results, threshold-independent diagnostics, probability calibration, computational efficiency, statistical comparisons, and qualitative error analysis. Section 4 discusses implications for practice and outlines avenues for lightweight improvements and future work.

2 Methods

We compare logistic regression (LR) against five widely used classifiers—multinomial Naive Bayes (MNB), calibrated linear SVM, decision tree (DT), k-nearest neighbors (KNN), and random forest (RF)—for binary sentiment classification on IMDb. All text processing and learning are encapsulated in `scikit-learn` `Pipeline`s, and model selection is separated from generalization estimation via nested, stratified cross-validation (CV). Evaluation spans threshold-dependent metrics (accuracy, precision, recall, F1), threshold-independent metrics (ROC–AUC, PR–AUC), probability calibration (Brier; reliability diagrams), and computational efficiency (training wall time; inference throughput). All experiments are CPU-only.

2.1 Dataset

We use the IMDb Large Movie Review Dataset, a balanced corpus of 50,000 user-written English reviews paired with binary sentiment labels (1=positive, 0=negative) [10]. The dataset comprises

25,000 positive and 25,000 negative reviews and is widely adopted as a benchmark for sentiment analysis research. We follow the standard usage of the full labeled corpus for cross-validated evaluation; Fig. 1 confirms the intended label balance.

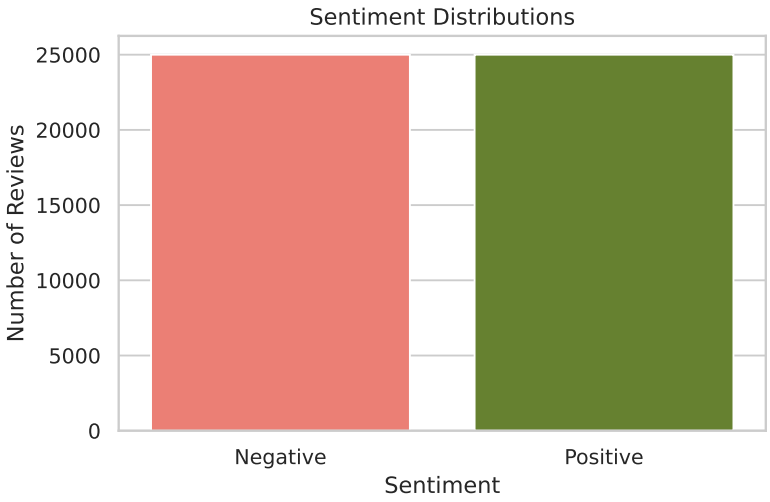


Figure 1: IMDb sentiment label distribution (balanced).

Data integrity and access. The data are publicly available and mirrored on common research repositories[1][2]. Reviews are free-form and may contain punctuation, markup fragments, and idiosyncratic orthography typical of user content. We do not alter labels, and we exclude no examples beyond the automated cleaning steps described below. With the corpus fixed, we next describe the end-to-end text pipeline applied uniformly across models and folds.

2.2 Text preprocessing and featurization

All text processing resides inside a single `Pipeline` to prevent train–test leakage (the vectorizer and any reducers are fit only on training partitions within the relevant CV loop). The pipeline is tailored to informal, user-generated text [11], [12] and proceeds in two stages: cleaning and vectorization.

Cleaning. Before vectorization, we standardize raw text to reduce noise while preserving polarity cues. Concretely, we apply:

(i) **HTML stripping:** remove inline tags and markup artifacts via a lightweight regular-expression pass;
(ii) **case folding:** lowercase all characters to avoid treating "Good" and "good" as distinct tokens [13];
(iii) **non-alphabetic filtering:** drop characters outside `[a-z]` and whitespace to reduce punctuation/digit noise;
(iv) **tokenization:** split on whitespace to obtain word-level tokens;
(v) **stopword removal:** remove common function words using the NLTK English list to emphasize sentiment-bearing terms;
(vi) **lemmatization:** map tokens to dictionary forms using WordNet (e.g., *running*→*run*) to mitigate sparsity [14].

We deliberately retain negation terms (e.g., *not*) so that bigram features downstream can capture scoped polarity (e.g., *not good*).

[1] https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
[2] https://github.com/Karthik-Bhaskar/Sentiment-Analysis

Vectorization (TF–IDF). After cleaning, texts are converted into numeric vectors using term frequency–inverse document frequency (TF–IDF), a strong baseline for linear and margin-based classifiers on high-dimensional text [15]–[17]. The weighting is

$$\mathrm{TF}(t,d) = \frac{f_{t,d}}{\sum_k f_{k,d}}, \quad \mathrm{IDF}(t) = \log\!\left(\frac{N}{df_t}\right), \quad \text{TF-IDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t), \tag{1}$$

where $f_{t,d}$ is the frequency of term t in document d, N is the number of documents, and df_t is the document frequency of t. We instantiate *TfidfVectorizer* with the configuration in Table 1. Because stopwords are removed upstream, we set `stop_words=None` to avoid double filtering. We include bigrams to encode short-range context (e.g., *not good*), use sublinear TF scaling, apply ℓ_2 normalization, and cap the vocabulary at 10,000 features to control dimensionality.

Table 1: Configuration of *TfidfVectorizer*.

Parameter	Value / Description
`lowercase`	`True` (case folding)
`stop_words`	`None` (stopwords removed upstream)
`ngram_range`	`(1, 2)` (unigrams + bigrams)
`min_df`	`2` (ignore terms seen < 2 documents)
`max_df`	`0.9` (ignore overly frequent terms)
`max_features`	`10000` (vocabulary cap)
`sublinear_tf`	`True` (log-scaling of term frequency)
`use_idf`	`True`
`smooth_idf`	`True`
`norm`	`'l2'`

Dense projection for non-linear/instance-based baselines. Certain classifiers perform poorly or incur prohibitive costs on extremely high-dimensional sparse inputs. To provide a fair and tractable setting, we prepend a linear dimensionality-reduction stage to *only* DT, KNN, and RF: TF–IDF vectors are projected via truncated singular value decomposition (TruncatedSVD) to 300 dense components, followed by standardization (zero mean, unit variance). Linear LR and linear SVM operate directly on the original sparse TF–IDF. This yields *TF–IDF* $\rightarrow$ *SVD* $\rightarrow$ *StandardScaler* $\rightarrow$ *classifier* for DT/KNN/RF and *TF–IDF* $\rightarrow$ *classifier* for LR/SVM. With featurization established, we next specify the model families.

2.3 Models

We evaluate one primary classifier and five baselines under matched outer folds and shared featurization. This ordering mirrors the subsequent analysis and ablations.

Primary model: Logistic Regression (LR). LR estimates $P(y{=}1 \mid \mathbf{x})$ using the sigmoid link on a linear score,

$$P(y{=}1 \mid \mathbf{x}) = \sigma(\beta_0 + \beta^\top \mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta^\top \mathbf{x})}}, \tag{2}$$

with coefficients β obtained by maximizing the ℓ_2-regularized log-likelihood (equivalently, minimizing penalized logistic loss):

$$\mathcal{L}(\beta) = -\sum_{i=1}^{m} \big[\, y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i) \,\big] + \frac{\lambda}{2} \|\beta\|_2^2, \tag{3}$$

where $\hat{p}_i = P(y_i{=}1 \mid \mathbf{x}_i)$ and λ controls ℓ_2 regularization (`scikit-learn` convention $C{=}1/\lambda$) [18]. We use solver `lbfgs` with `max_iter`$=1000$ and a default decision threshold of 0.5 unless otherwise specified [19]. We consider `class_weight` $\in$ {`None`, `'balanced'`} during tuning.

Baselines. For comparison breadth, we include five widely used alternatives spanning linear, probabilistic, instance-based, and ensemble methods:

(a) **Linear SVM (calibrated):** `LinearSVC` with hinge loss on sparse TF–IDF; probabilities are obtained via `CalibratedClassifierCV` (Platt's sigmoid) *within* the inner loop, and classification uses a 0.5 threshold on the calibrated probability [20].

(b) **Multinomial Naive Bayes (MNB):** an efficient probabilistic baseline with additive smoothing parameter α over TF–IDF [7].

(c) **Decision Tree (DT):** interpretable axis-aligned partitions with depth/leaf regularization; trained on SVD-reduced, standardized features [21].

(d) **k-Nearest Neighbors (KNN):** Euclidean metric on standardized SVD projections; we tune k and weighting (`uniform` vs. `distance`).

(e) **Random Forest (RF):** bagged decision-tree ensemble trained on standardized SVD projections with tuned tree count and feature subsampling.

Having defined model families, we now describe training, validation, and tuning.

2.4 Training, validation, and tuning

We adopt nested, stratified CV to remove selection bias and obtain an unbiased estimate of generalization. All text processing and learning are wrapped in `Pipeline`s so that vectorizers, reducers, calibrators, and classifiers are fit *only* on the relevant training folds.

Outer evaluation loop. We use K_{out}=5 stratified folds. For each outer fold $i \in \{1, \ldots, 5\}$, the held-out split D_i^{test} serves exclusively for evaluation; the complement $D_{\setminus i}$ is reserved for model selection.

Inner model-selection loop. On $D_{\setminus i}$ we run an inner stratified CV with K_{in}=5 folds to tune hyperparameters (and, where relevant, the calibration model). The entire workflow is expressed as `Pipeline`: TF–IDF $\rightarrow$ (optional SVD $\rightarrow$ StandardScaler) $\rightarrow$ classifier (and calibrator). Vectorizers/reducers are fit strictly within each inner training partition. The primary selection metric is F1 (positive class = 1); accuracy serves as a secondary tie-breaker. The selected configuration is refit on full $D_{\setminus i}$ before a single evaluation on D_i^{test}.

Hyperparameter search spaces. All models share the TF–IDF configuration in Table 1. We search the following grids:

- `Logistic Regression (LR)`: $C \in \{0.25, 0.5, 1, 2, 4\}$; `penalty = 'l2'`; `solver = 'lbfgs'`; `max_iter = 1000`; `class_weight` $\in \{$`None`, `'balanced'`$\}$.
- `Linear SVM (calibrated)`: `LinearSVC` with $C \in \{0.25, 0.5, 1, 2, 4\}$; `class_weight` $\in \{$`None`, `'balanced'`$\}$; probabilities via inner-loop `CalibratedClassifierCV` (sigmoid).
- `Multinomial Naive Bayes (MNB)`: $\alpha \in \{0.1, 0.5, 1.0, 2.0\}$.
- `Decision Tree (DT)`: `max_depth` $\in \{$`None`$, 20, 40\}$; `min_samples_leaf` $\in \{1, 5, 10\}$; trained on SVD-reduced, standardized features.
- `KNN`: $k \in \{5, 15, 31\}$; `weights` $\in \{$`uniform`, `distance`$\}$; metric $= \ell_2$ on standardized SVD projections.
- `Random Forest (RF)`: `n_estimators` $\in \{200, 400\}$; `max_features` $\in \{$`sqrt`, `log2`$\}$; `min_samples_leaf` $\in \{1, 5\}$; trained on SVD-reduced, standardized features.

With model selection defined, we next describe evaluation criteria and aggregation.

2.5 Evaluation metrics and aggregation

On each outer test split we compute accuracy, precision, recall, and F1 using a fixed decision threshold of 0.5 on predicted probabilities (or calibrated probabilities). For methods

that yield decision scores without native probabilities, we obtain calibrated probabilities via `CalibratedClassifierCV` in the inner loop [22], [23]. Threshold-independent metrics include ROC–AUC and PR–AUC (average precision). We assess probability calibration using the Brier score and reliability diagrams. For each model we report mean±standard deviation over the K_{out} outer folds and provide 95% confidence intervals using the t-distribution. These computations correspond to the "record metrics" step in Algorithm 1 and set the stage for runtime measurement.

Algorithm 1 Nested CV for fair model selection and evaluation

1: Split D into K_{out} stratified folds $\{D_i^{\text{test}}\}$
2: **for** $i = 1$ to K_{out} **do**
3: $D_{\setminus i} \leftarrow D \setminus D_i^{\text{test}}$
4: Inner CV (K_{in}) on $D_{\setminus i}$ to select hyperparameters (and perform calibration where applicable)
5: Refit the selected pipeline on full $D_{\setminus i}$
6: Evaluate once on D_i^{test}; record metrics (accuracy, precision, recall, F1, ROC–AUC, PR–AUC, Brier) and timings (train time, throughput)
7: **end for**
8: Aggregate metrics over outer folds; compute mean±std, 95% CIs, and run Wilcoxon tests (LR vs. baselines) with Holm correction

Calibration details and operating point. All reported classification metrics use a threshold of 0.5. Probability quality is summarized by the Brier score; reliability diagrams are computed from out-of-sample (OOF) predictions by concatenating outer-fold test probabilities. We additionally use OOF probabilities to produce dataset-wide ROC and PR curves without leakage. Having specified metrics, we turn to computational efficiency.

Computational efficiency. We measure (i) end-to-end training wall time per outer fold (including vectorization/reduction and model fitting within the trained pipeline) and (ii) inference throughput in samples/second on the corresponding outer test split using the refit model. We exclude data-loading time. All runs are CPU-only. Environment details are summarized in Table 8. Finally, we describe statistical testing.

Statistical analysis. To test whether LR outperforms each baseline, we perform paired Wilcoxon signed-rank tests on outer-fold F1 scores with a one-sided alternative (LR > baseline), and control family-wise error rate using Holm's step-down procedure. This nonparametric test is robust to modest deviations from normality and accommodates the small-sample regime induced by K_{out}=5. We conclude with reproducibility safeguards.

2.6 Reproducibility and safeguards against leakage

All stochastic components use `random_state`= 42. Identical outer folds are used across all models. Implementations rely on `scikit-learn`. Expressing the entire workflow in `Pipeline` ensures that TF–IDF fitting, SVD, standardization, calibration, and classifier training occur strictly within the appropriate training partitions, eliminating train–test leakage. Code artifacts (single executable script), saved fold indices, and exported tables/figures enable exact reproduction of all reported numbers[3].

[3] https://www.kaggle.com/code/edumath/paper-cauchy-33809

3 Results and Discussion

We report outcomes on the IMDb benchmark under a unified, leakage-safe pipeline. Results are presented in four parts: (i) topline performance and stability for logistic regression (LR), (ii) head-to-head comparison with five baselines, (iii) threshold-independent diagnostics and calibration, and (iv) efficiency. We then discuss implications, positioning against prior work, and error patterns.

3.1 Results

Topline performance and stability. Using stratified 5-fold cross-validation with identical outer folds across models, LR achieves mean±std accuracy $\mathbf{0.8898 \pm 0.0023}$, precision 0.8792 ± 0.0041, recall 0.9037 ± 0.0029, and F1 $\mathbf{0.8913 \pm 0.0022}$ (Tables 2 and 3). The narrow dispersion indicates stability to resampling.

Table 2: Per-fold performance of logistic regression (stratified 5-fold CV).

fold	accuracy	precision	recall	f1	roc_auc	pr_auc	brier	train_time_s
1	0.8909	0.8805	0.9047	0.8923	0.9576	0.9564	0.0856	2.9911
2	0.8927	0.8826	0.9060	0.8942	0.9575	0.9542	0.0866	2.6329
3	0.8892	0.8775	0.9030	0.8900	0.9569	0.9558	0.0854	2.4205
4	0.8868	0.8759	0.9003	0.8880	0.9555	0.9549	0.0859	3.1002
5	0.8896	0.8796	0.9045	0.8911	0.9565	0.9556	0.0861	2.2204

Table 3: Mean and standard deviation across five folds for logistic regression.

	accuracy	precision	recall	f1	roc_auc	pr_auc	brier	train_time_s
mean	0.8898	0.8792	0.9037	0.8913	0.9568	0.9554	0.0858	2.6730
std	0.0023	0.0041	0.0029	0.0022	0.0008	0.0017	0.0007	0.5610

Per-class behavior. The confusion matrix (Fig. 2) shows that LR correctly classifies the large majority of both classes, with relatively few false positives/negatives, consistent with the precision and recall above.

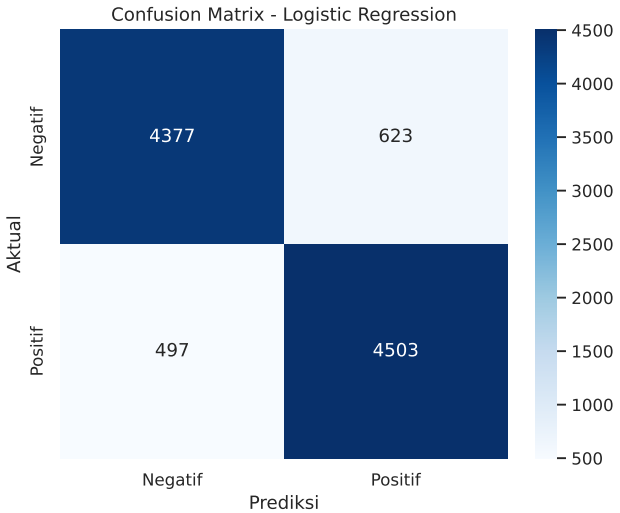


Figure 2: Confusion matrix for logistic regression on a representative validation fold.

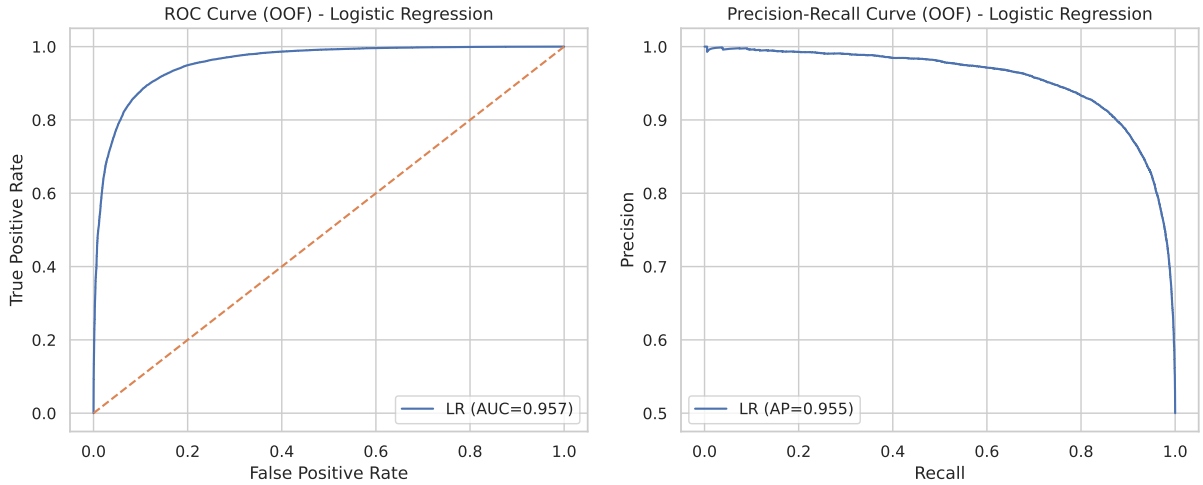
Head-to-head with baselines. Table 4 summarizes LR against multinomial Naive Bayes (MNB), linear SVM, decision tree (DT), k-nearest neighbors (KNN), and random forest (RF), all under the same folds and TF–IDF features. The time column reports mean *training wall time per fold*.

Table 4: Performance comparison across classifiers (stratified 5-fold averages). Time = mean training wall time per fold.

Classifier	Accuracy	Precision	Recall	F1-score	Time (s)
Naive Bayes (MNB)	0.8570	0.8550	0.8598	0.8574	0.03
SVM (linear)	0.8874	0.8801	0.8971	0.8885	1007.11
Decision Tree	0.7246	0.7275	0.7183	0.7228	46.46
KNN	0.7589	0.7261	0.8316	0.7752	852.16
Random Forest	0.8501	0.8544	0.8440	0.8492	90.72
Logistic Regression	**0.8898**	**0.8792**	**0.9037**	**0.8913**	**1.85**

LR yields the highest accuracy (88.98%) and F1 (89.13%), reflecting the best precision–recall balance. Although SVM attains slightly higher precision (88.01% vs. 87.92% for LR), LR's higher recall (90.37%) gives it the top F1. MNB is extremely fast but trails in accuracy/F1. KNN shows recall>precision behavior typical of distance-based methods on low-rank projections. DT underperforms overall (axis-aligned splits struggle on the projected space), and RF improves upon DT but remains below LR while incurring higher compute.

Threshold-independent diagnostics. Using out-of-fold (OOF) probabilities, the ROC–AUC is **0.9568** and the PR–AUC (average precision) is **0.9554** (Fig. 3). These ranking metrics align with the strong F1 and indicate excellent separability prior to thresholding.



(a) ROC curve (OOF); AUC = 0.9568. **(b)** Precision–recall curve (OOF); AP = 0.9554.

Figure 3: Threshold-independent diagnostics for logistic regression on OOF probabilities.

Probability calibration. Out-of-fold probabilities are well calibrated overall (Brier **0.0858** ± **0.0007**). In the reliability diagram (Fig. 4), the LR curve tracks the $y{=}x$ identity closely across most of the support, with three small but systematic departures: (i) at the extreme left ($p \lesssim 0.1$) the curve lies slightly *below* the diagonal, indicating mild *overconfidence*; (ii) throughout the mid–high band ($0.6 \lesssim p \lesssim 0.9$) it sits a little *above* the diagonal, reflecting modest *underconfidence*; and (iii) in the right tail ($p \gtrsim 0.9$) it re-aligns with the diagonal. These deviations are minor and the resulting probabilities remain adequate for downstream applications requiring calibrated confidence estimates.

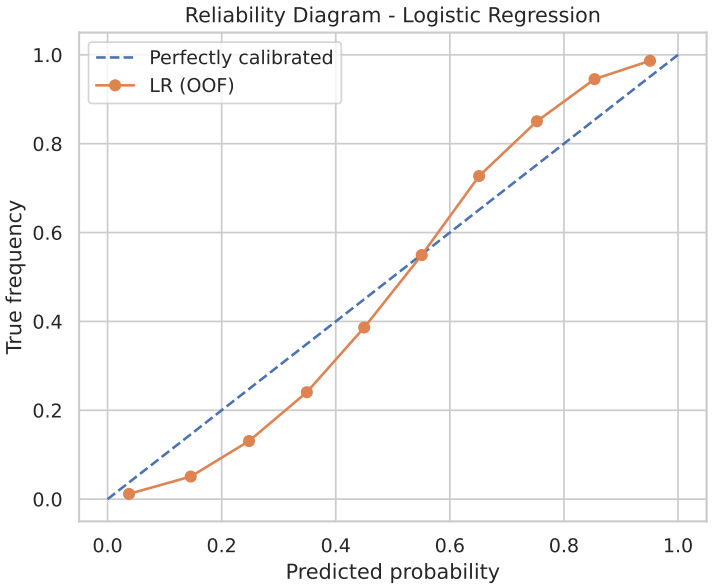


Figure 4: Reliability diagram (OOF) for logistic regression; lower Brier is better.

Complementing these quality assessments, we next evaluate LR's computational efficiency to highlight its suitability for frequent retraining and real-time deployment.

Computational efficiency. Beyond the training times in Table 4, we measure end-to-end training and inference per outer fold. LR's mean training time is **2.673±0.561 s**; mean inference time on the held-out split is **0.0041±0.0005 s**. This implies CPU-only throughput of approximately $2.46\times10^6 \pm 0.26\times10^6$ **samples/s** on our setup (hardware/software details in Table 8), supporting frequent retraining and low-latency deployment.

3.2 Discussion

Internal comparison. Across identical folds, LR offers the most balanced trade-off among accuracy, precision, and recall. Its F1 advantage over SVM stems from higher recall, desirable when false negatives are costlier. MNB remains attractive for speed but sacrifices several points of F1/accuracy. KNN's recall>precision pattern reflects distance concentration effects even after low-rank projection. DT is prone to suboptimal axis-aligned partitions on the projected space; RF mitigates variance but still lags LR while costing more compute. Low standard deviations across folds indicate that these patterns are not split-specific.

External comparison. We situate LR against representative IMDb studies using heterogeneous pipelines (SVM+TF–IDF, LSTM variants, LR+RF voting). Our single-model LR is competitive with the best reported numbers while being simpler and cheaper to train. Cross-paper comparability is imperfect—preprocessing, featurization, splits, and tuning differ—so we avoid definitive claims; nonetheless, the observed gap to stronger ensembles is marginal (Table 5).

Qualitative error analysis. Inspection of anonymized misclassified snippets (Table 6, Table 7) reveals four recurrent sources: (i) ambivalent narratives mixing praise and criticism; (ii) scoped or layered negation ("not entirely bad", "hardly great"); (iii) sarcasm/irony; and (iv) domain-specific references requiring background knowledge. Such phenomena are challenging for linear models on bag-of-n-grams. The high ROC–AUC (≈ 0.9568), PR–AUC (≈ 0.9554), and reasonable calibration (Brier ≈ 0.0858) suggest many near-boundary errors are intrinsically ambiguous.

Table 5: External comparison with representative prior work on IMDb.

Study (Year)	Method	Acc.	Prec.	Rec.	F1-score / Time
Ali et al [24]	MLP	0.86	–	–	–
Ramadhan & Ramadhan [25]	SVM + TF–IDF	0.79	0.75	0.87	–
Kalla et al [6]	LSTM model	0.86	–	–	–
Kaushik & Parmar [7]	Voting (LR + RF)	0.89	0.88	0.90	0.89
Our study	**Logistic Regression**	**0.88**	**0.87**	**0.90**	**0.89 / 1.85 s**

Table 6: Representative false positives (anonymized excerpts).

#	proba	true/pred	excerpt
1	0.943	0/1	. . . the film has a few genuinely touching moments, but the overall pacing drags and the humor falls flat. I wanted to like it more than I did, and maybe on a second viewing I'd catch more of what fans rave about, yet the first half felt like setup for a payoff that never truly lands. . .
2	0.901	0/1	. . . I appreciate the ambition and the cinematography, however the script meanders, the characters lack depth, and the final twist feels unearned. Some performances are solid, but the story never coalesces into something satisfying. . .
3	0.876	0/1	. . . there are flashes of brilliance, but also a lot of filler. The soundtrack is overbearing and scenes linger too long. It's not awful, just not the masterpiece some claim—it's middling with occasional sparks rather than consistently engaging. . .

Table 7: Representative false negatives (anonymized excerpts).

#	proba	true/pred	excerpt
1	0.118	1/0	. . . it's not perfect by any means, but I had a good time. The lead brings charm, the jokes land more often than not, and I walked out with a smile. For a Friday night watch, you could do far worse. . .
2	0.227	1/0	. . . after a slow start, the film hits its stride and becomes surprisingly heartfelt. The chemistry between the two leads works, and the ending, while predictable, is warmly executed. . .
3	0.194	1/0	. . . despite a few clunky lines and budget constraints, the creativity shines through. It's a scrappy, spirited effort that rewards viewers who can look past the rough edges and enjoy the ride. . .

Practical implications and limitations. LR combines interpretability (via coefficients), fast training (seconds), and very high CPU throughput ($\sim 2.46\times10^6$ samples/s), making it a strong default where rapid iteration and predictable latency are priorities. Limitations include figurative language, long-range discourse, and subtle pragmatics. Under class imbalance or drift, threshold tuning, class weighting, and periodic recalibration are advisable. Given heterogeneous setups in prior work, we avoid definitive superiority claims.

Concluding remark. Across threshold-dependent, threshold-independent, and calibration-oriented metrics—and under strict, stratified validation—LR offers the best overall balance among baselines while being markedly more efficient. Full environment details are provided in Table 8 to support reproducibility.

Table 8: Hardware and software environment used in all experiments.

Component	Specification
OS / Platform	Linux-6.6.56+ (x86_64) with glibc 2.35
CPU Architecture	x86_64
Logical CPU Cores	4
RAM (GB)	32
Python	3.11.11
pandas	2.2.3
scikit-learn	1.2.2

4 Conclusion

We conducted a like-for-like evaluation of logistic regression (LR) for binary sentiment classification on the IMDb benchmark under a standardized, leakage-safe text pipeline. Using stratified 5-fold cross-validation with a common TF–IDF featurization across models, LR achieved the best overall balance of predictive quality and efficiency among five widely used baselines (multinomial Naive Bayes, linear SVM, decision tree, k-nearest neighbors, and random forest): accuracy **88.98%**, F1 **89.13%**, strong threshold-independent performance (OOF ROC–AUC $\approx$ **0.9568**; PR–AUC $\approx$ **0.9554**), and well-behaved calibration (Brier $\approx$ **0.0858**). Training completes in a few seconds per fold, and CPU inference throughput is on the order of $\sim 2.46 \times 10^6$ samples/s, underscoring suitability for low-latency deployment.

Relative to more complex approaches reported in prior work (e.g., LSTM variants and LR+RF voting ensembles), this single-model LR configuration remains competitive while preserving transparency, ease of maintenance, and reproducibility. Error analysis indicates that residual mistakes cluster around ambivalent narratives, scoped negation, sarcasm, and domain-specific references—phenomena that challenge linear models over bag-of-n-grams.

Practically, LR is a strong default for movie-review sentiment classification when rapid iteration, interpretability, and predictable latency are priorities. When marginal gains are needed beyond LR, lightweight extensions—explicit negation handling, modest n-gram expansion, calibrated decision thresholds under application-specific costs, and periodic probability recalibration under distribution shift—offer targeted improvements without the complexity or compute demands of deep sequence models. Future work includes assessing cross-platform domain adaptation, robustness under class imbalance and temporal drift, and incorporating light contextual features that retain LR's speed while addressing the identified error modes.

CRediT Author Contributions

Diah Mariatul Ulya: Conceptualization; Methodology; Investigation; Formal analysis; Writing – Original Draft. **Juhari:** Data Curation; Software; Validation; Visualization. **Rossima Eva Yuliana:** Formal analysis; Visualization; Writing – Review & Editing. **Mohammad Jamhuri:** Supervision; Project administration; Resources; Writing – Review & Editing.

Declaration of Generative AI and AI-assisted Technologies

The authors used generative AI tools (e.g., ChatGPT) to assist with language polishing, LaTeX boilerplate generation (tables/figures), and consistency checks. All generated content was reviewed, verified, and edited by the authors, who take full responsibility for the manuscript's integrity and accuracy.

Declaration of Competing Interest

The authors declare no competing interests.

Funding and Acknowledgments

This research received *no external funding.* The authors thank the Faculty of Science and Technology, UIN Maulana Malik Ibrahim Malang, for general support and infrastructure.

Data Availability

The experiments in this study use the standard, publicly available IMDb Large Movie Review Dataset (50,000 labeled reviews; commonly mirrored on research repositories). Preprocessing scripts, configuration files, and instructions to reproduce all reported results (including figures and tables) are provided in the authors' code package and will be made available upon request or publication.

References

[1] S. Banerjee and A. Y. Chua, "Tracing the growth of imdb reviewers in terms of rating, readability and usefulness," in *2018 4th International Conference on Information Management (ICIM)*, IEEE, 2018, pp. 57–61. DOI: 10.1109/INFOMAN.2018.8392809.

[2] S. Ounacer, D. Mhamdi, S. Ardchir, A. Daif, and M. Azzouazi, "Customer sentiment analysis in hotel reviews through natural language processing techniques," *Int. J. Adv. Comput. Sci. Appl*, vol. 14, no. 1, pp. 1–11, 2023. DOI: 10.14569/IJACSA.2023.0140162.

[3] P. Atandoh, F. Zhang, M. A. Al-Antari, D. Addo, and Y. H. Gu, "Scalable deep learning framework for sentiment analysis prediction for online movie reviews," *Heliyon*, vol. 10, no. 10, 2024. DOI: 10.1016/j.heliyon.2024.e30756.

[4] M. Muhathir, "Compares the effectiveness of the bagging method in classifying spices using the histogram of oriented gradient feature extraction technique," *Jurnal Teknik Informatika CIT Medicom*, vol. 15, no. 1, pp. 48–57, 2023. DOI: 10.35335/cit.Vol15.2023.386.pp48-57.

[5] L. Holla and K. Kavitha, "An improved fake news detection model using hybrid time frequency-inverse document frequency for feature extraction and adaboost ensemble model as a classifier," *Journal of Advances in Information Technology*, vol. 15, no. 2, pp. 202–211, 2024. DOI: 10.12720/jait.15.2.202-211.

[6] D. Kalla, N. Smith, and F. Samaah, "Deep learning-based sentiment analysis: Enhancing imdb review classification with lstm models," *Available at SSRN 5103558*, 2025.

[7] K. Kaushik and M. Parmar, "Imdb movie data classification using voting classifier for sentiment analysis," *Int. J. Comput. Sci. Eng.*, vol. 10, no. 1, pp. 18–23, 2022. DOI: 10.26438/ijcse/v10i1.1823.

[8] A. Belz, S. Agarwal, A. Shimorina, and E. Reiter, "A systematic review of reproducibility research in natural language processing," *arXiv preprint arXiv:2103.07929*, 2021. DOI: 10.48550/arXiv.2103.07929.

[9] Y. Xue, X. Cao, X. Yang, Y. Wang, R. Wang, and J. Li, "We need to talk about reproducibility in nlp model comparison," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 9424–9434. DOI: 10.18653/v1/2023.emnlp-main.586.

[10] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150. Available online.

[11] J. Sun and Y. Xia, "Pretreating and normalizing metabolomics data for statistical analysis," *Genes & Diseases*, vol. 11, no. 3, p. 100 979, 2024. DOI: 10.1016/j.gendis.2023.04.018.

[12] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 91–99, 2022. DOI: 10.1016/j.gltp.2022.04.020.

[13] K. Passi and S. Kalakala, "A rule-based sentiment analysis of whatsapp reviews in telugu language," in *IOT with Smart Systems*, Springer, 2023, pp. 167–180. DOI: 10.1007/978-981-19-3575-6_19.

[14] Z. Abidin, A. Junaidi, *et al.*, "Text stemming and lemmatization of regional languages in indonesia: A systematic literature review," *Journal of Information Systems Engineering and Business Intelligence*, vol. 10, no. 2, pp. 217–231, 2024. DOI: 10.20473/jisebi.10.2.217-231.

[15] T. Verdonck, B. Baesens, M. Óskarsdóttir, and S. vanden Broucke, "Special issue on feature engineering editorial," *Machine learning*, vol. 113, no. 7, pp. 3917–3928, 2024. DOI: 10.1007/s10994-021-06042-2.

[16] D. Gibert, J. Planes, C. Mateu, and Q. Le, "Fusing feature engineering and deep learning: A case study for malware classification," *Expert Systems with Applications*, vol. 207, p. 117 957, 2022. DOI: 10.1016/j.eswa.2022.117957.

[17] R. Pan, M. Bagherzadeh, T. A. Ghaleb, and L. Briand, "Test case selection and prioritization using machine learning: A systematic literature review," *Empirical Software Engineering*, vol. 27, no. 2, p. 29, 2022. DOI: 10.1007/s10664-021-10066-6.

[18] M. Jamhuri, I. Mukhlash, and M. I. Irawan, "Performance improvement of logistic regression for binary classification by gauss-newton method," in *Proceedings of the 2022 5th International Conference on Mathematics and Statistics*, 2022, pp. 12–16. DOI: 10.1145/3545839.3545842.

[19] J. Ribeiro, R. Lima, T. Eckhardt, and S. Paiva, "Robotic process automation and artificial intelligence in industry 4.0–a literature review," *Procedia Computer Science*, vol. 181, pp. 51–58, 2021. DOI: 10.1016/j.procs.2021.01.104.

[20] N. Cesario, D. Lewis, C. Rosales, F. Antolini, R. Stojanovic, and L. Vandenberg, "Ransomware detection using opcode sequences and machine learning: A novel approach with t-sne and support vector machines," *Authorea Preprints*, 2024. DOI: 10.36227/techrxiv.172963142.20817264/v1.

[21] J.-Y. Ong, L.-Y. Ong, and M.-C. Leow, "Addressing overfitting in comparative study for deep learning based classification," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 23, no. 3, pp. 673–681, 2025. DOI: 10.12928/telkomnika.v23i3.26451.

[22] D. A. Neu, J. Lahann, and P. Fettke, "A systematic literature review on state-of-the-art deep learning methods for process prediction," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 801–827, 2022. DOI: 10.1007/s10462-021-09960-8.

[23] S. N. Nobel, S. M. R. Swapno, M. R. Islam, M. Safran, S. Alfarhood, and M. Mridha, "A machine learning approach for vocal fold segmentation and disorder classification based on ensemble method," *Scientific reports*, vol. 14, no. 1, p. 14 435, 2024. DOI: 10.1038/s41598-024-64987-5.

[24] N. M. Ali, M. M. Abd El Hamid, and A. Youssif, "Sentiment analysis for movies reviews dataset using deep learning models," *International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol*, vol. 9, 2019. DOI: 10.5121/ijdkp.2019.9302.

[25] N. G. Ramadhan and T. I. Ramadhan, "Analysis sentiment based on imdb aspects from movie reviews using svm," *Sinkron: jurnal dan penelitian teknik informatika*, vol. 6, no. 1, pp. 39–45, 2021. DOI: 10.33395/sinkron.v7i1.11204.