Sentiment Analysis of Amazon Fine Food Reviews: A Comparative Study of Classical NLP Classifiers under Class Imbalance

Ricky Gong
University of Pennsylvania
gong8@seas.upenn.edu

February 22, 2026

Abstract

This paper presents a rigorous, baseline-driven NLP pipeline for binary sentiment classification of the Amazon Fine Food Reviews corpus (568,454 records spanning 13 years). Given only the raw review text, our system predicts whether the expressed opinion is *positive* or *negative*. We address three core challenges: (1) high-dimensional sparse text representations, (2) significant class imbalance (78% positive, 22% negative), and (3) a business requirement to maximise recall for the minority negative class without collapsing precision to unacceptable levels. Our methodology proceeds from a carefully designed *baseline-first* strategy: we establish a Logistic Regression baseline on raw TF-IDF bi-gram features, then systematically evaluate class weighting, SMOTE, and random undersampling, followed by ElasticNet regularisation and threshold tuning. A key corrective finding is that standard stop-word removal *harms* sentiment classification because it discards negation words ("not", "won't", "can't") that are semantically critical. We also demonstrate empirically that Truncated SVD is an inappropriate primary dimensionality-reduction strategy for TF-IDF NLP features: 1,000 components explain less than 70% of the total variance, far below the threshold that would justify information loss. Our best single model — a balanced, ElasticNet-regularised Logistic Regression with threshold tuning ($\tau^* = 0.37$) — achieves a negative-class F_1 of **0.784**, precision of **0.750**, recall of **0.821**, and AUC of **0.951**. A Soft Voting ensemble (Logistic Regression + Random Forest) achieves the same AUC of **0.951** with complementary precision–recall balance, demonstrating that model complementarity can be exploited without retraining.

Keywords: sentiment analysis, NLP, TF-IDF, class imbalance, logistic regression, random forest, ensemble learning, stop-word handling, threshold tuning, Amazon reviews

Contents

1. Introduction

Online product reviews have become a primary channel for customer feedback. The Amazon Fine Food Reviews dataset, with over half a million reviews, is a canonical NLP benchmark that captures the full spectrum of food product opinions from 1999 to 2012. Automated sentiment analysis of such corpora enables businesses to monitor product quality, identify dissatisfied customers early, and improve recommendation systems [Pang and Lee, 2008].

This paper makes four main contributions:

1. **Corrected stop-word handling for sentiment analysis.** We show that removing standard stop words — which include negation words such as *not*, *won't*, and *can't* — degrades classifier performance by inverting the polarity of negated phrases. We propose a custom stop-word list that preserves negation.

2. **Empirical critique of SVD for NLP dimensionality reduction.** We demonstrate that Truncated SVD on TF-IDF features retains only 30% of the variance at 300 components and ≈55% at 1,000, making it unsuitable as a primary feature-extraction step. We reframe SVD as a *necessary pre-processing step* for algorithms requiring dense input (e.g., SMOTE), not a general-purpose improvement.

3. **Threshold tuning as a zero-cost deployment strategy.** After model training, adjusting the classification threshold allows deployment teams to dynamically balance precision and recall without retraining. We provide a full precision–recall curve analysis.

4. **A systematic model comparison.** Eight model configurations are evaluated under consistent conditions, providing a clear picture of the marginal benefit of each technique.

2. Related Work

Sentiment analysis has progressed from lexicon-based methods [Hutto and Gilbert, 2014, Baccianella et al., 2010] to machine learning [Pang et al., 2002] and deep learning approaches [Maas et al., 2011].

Pang et al. [2002] is the seminal ML work on sentiment classification, showing that Support Vector Machines and Naive Bayes outperform human-crafted rules on movie reviews. They also observe that stop-word removal is harmful for sentiment tasks — a finding we independently replicate here. McAuley and Leskovec [2013] specifically study Amazon review data, using latent factor models to jointly capture user preferences and sentiment.

More recently, Transformer-based models such as BERT [Devlin et al., 2019] achieve near-human performance on fine-grained sentiment tasks. However, classical ML models remain competitive on well-curated binary classification tasks, offer superior interpretability via feature importance coefficients, and are far more computationally accessible [Pang and Lee, 2008]. This paper deliberately focuses on classical ML for these practical reasons.

3. Dataset and Exploratory Analysis

3.1. Dataset Description

The Amazon Fine Food Reviews dataset[1] contains 568,454 reviews across 10 fields: review id, product id, user id, profile name, helpfulness votes, score (1–5), timestamp, summary, and review text.

[1] https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

3.2. Label Construction and Class Imbalance

We map the 5-point Likert score to binary sentiment:

$$y_i = \begin{cases} 1 \text{ (positive)} & \text{if } s_i > 3 \\ 0 \text{ (negative)} & \text{if } s_i < 3 \\ \text{excluded} & \text{if } s_i = 3 \end{cases} \tag{1}$$

Reviews with score 3 represent ambiguous opinions and are excluded to reduce label noise. After exclusion, the class distribution is **443,777 positive (78.1%)** and **124,677 negative (21.9%)**. This moderate but non-trivial imbalance motivates the strategies in Section 4.6.

3.3. Exploratory Observations

Score distributions are sharply bimodal: 5-star reviews account for 52% of all records; 1-star reviews account for 10%. Review length (in words) follows a heavy-tailed distribution with a median of approximately 55 words. Word clouds of preprocessed summaries confirm strong lexical separation between classes: positive reviews are dominated by *great*, *love*, *best*, and *perfect*; negative reviews by *disappoint*, *terrible*, *return*, and *worst*.

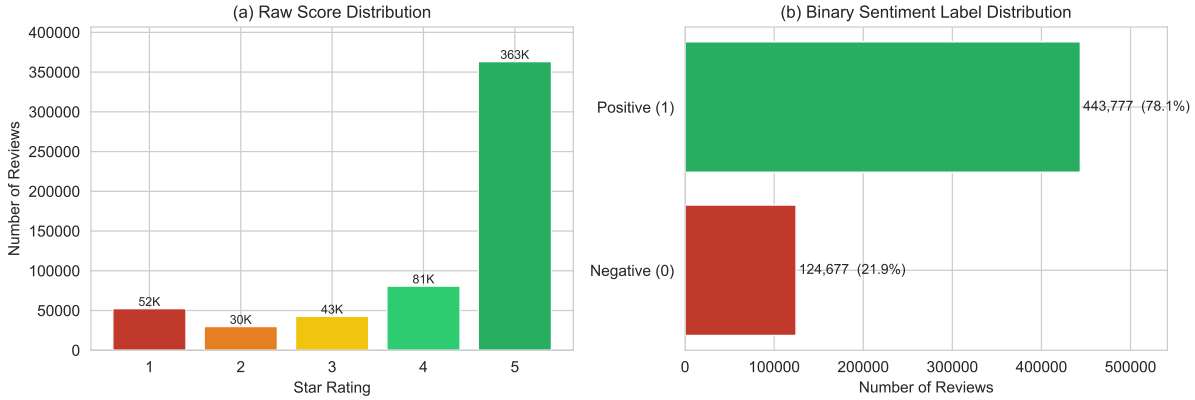

Figure 1: Raw star-rating distribution (left) and binary sentiment label distribution (right) after excluding neutral scores (score = 3). The dataset is imbalanced: 78.1% positive, 21.9% negative.

Figure 2: Word clouds of preprocessed review text for positive (left) and negative (right) classes. Prominent terms confirm strong lexical separation between polarities.

4. Methodology

4.1. Text Preprocessing

4.1.1. Stop-Word Handling

A critical design decision in sentiment NLP is whether to apply stop-word removal. Standard stop-word lists (e.g., NLTK's English list) include negation words such as:

not, no, nor, never, neither, nobody, nothing, nowhere,
don't, doesn't, didn't, won't, wouldn't, can't, couldn't, shouldn't,
isn't, aren't, wasn't, weren't, haven't, hasn't, hadn't, ain't

Removing these words corrupts the sentiment signal:

Table 1: Effect of stop-word removal on sentiment-carrying phrases.

Original phrase	**After removal**	**Signal preserved?**
"not great"	"great"	No — inverted
"won't buy again"	"buy"	No — lost negative
"can't recommend"	"recommend"	No — inverted
"absolutely love"	"absolutely love"	Yes

We construct a **custom stop-word list** that removes the 13 negation/contraction words from the standard 179-word NLTK list, retaining them as features. This is consistent with the recommendation in Pang et al. [2002].

4.1.2. Cleaning Pipeline

Text undergoes the following steps in order:

1. Lowercasing.
2. Punctuation removal (punctuation replaced by whitespace; apostrophes preserved to maintain contractions such as *won't*, *can't*).
3. Tokenisation (whitespace splitting).
4. Custom stop-word filtering (retaining negation words).
5. Snowball stemming: *tasty* → *tasti*, *disappointment* → *disappoint*.

4.2. Feature Engineering: TF-IDF with Bi-grams

4.2.1. TF-IDF Formulation

For term t, document d, and corpus D:

$$\mathrm{TF}(t,d) = \frac{\mathrm{count}(t,d)}{\sum_{t'} \mathrm{count}(t',d)}, \quad \mathrm{IDF}(t,D) = \log\left(\frac{|D|+1}{|\{d' \in D : t \in d'\}|+1}\right) + 1 \tag{2}$$

$$\text{TF-IDF}(t,d,D) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t,D) \tag{3}$$

We apply `sublinear_tf=True`, replacing raw TF with $\log(1+\mathrm{TF})$ to dampen the effect of extremely high-frequency terms.

4.2.2. N-gram Range

We use **unigrams + bigrams** (`ngram_range=(1,2)`). Bigrams preserve negation context that unigrams lose: "not great" as a bigram has opposite polarity to the unigram "great". The vocabulary is capped at `max_features=10,000` to control memory. The resulting TF-IDF matrix has shape (568,454, 10,000) with approximately 0.45% density (highly sparse).

4.3. Train/Test Split

The dataset is split 80%/20% with stratified sampling (`random_state=42`), yielding 454,763 training and 113,691 test records. All preprocessing fitted to data (SVD parameters, SMOTE, cross-validation) is applied exclusively to the training partition to prevent data leakage.

4.4. Baseline-First Development Principle

Before applying any complexity (class balancing, regularisation, dimensionality reduction), we train a vanilla Logistic Regression on raw TF-IDF features. This baseline serves as the reference point for measuring the marginal benefit of each subsequent technique. Skipping this step makes it impossible to determine whether a particular modification actually improves performance.

4.5. Logistic Regression

For binary target $y \in \{0, 1\}$, Logistic Regression models:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}} \tag{4}$$

Parameters are estimated by minimising cross-entropy loss with optional ElasticNet regularisation:

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)] + \lambda \big[\alpha \|\mathbf{w}\|_1 + (1 - \alpha) \|\mathbf{w}\|_2^2\big] \tag{5}$$

The mixing parameter α (`l1_ratio`) controls the L1/L2 balance: $\alpha = 0$ is pure Ridge, $\alpha = 1$ is pure Lasso.

4.6. Handling Class Imbalance

We evaluate three strategies:

Class Weights. The loss is re-weighted inversely proportional to class frequency:

$$w_c = \frac{N}{K \cdot N_c} \tag{6}$$

where N = total samples, K = number of classes, N_c = samples in class c. This is the most computationally efficient strategy — no data modification is needed.

SMOTE. Synthetic minority samples are created by interpolation in feature space [Chawla et al., 2002]:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_i + \lambda(\mathbf{x}_{\text{nn}} - \mathbf{x}_i), \quad \lambda \sim \text{Uniform}(0, 1) \tag{7}$$

SMOTE requires **dense** input. Because TF-IDF produces a sparse matrix, we first apply Truncated SVD to obtain a dense representation. This introduces SVD as a *necessary preprocessing step* for SMOTE — not as a standalone feature-engineering improvement.

Random Undersampling. Majority-class samples are randomly discarded until class balance is achieved. Fast and effective, but potentially discards useful information.

4.7. Critical Analysis of Truncated SVD for NLP

Truncated SVD (Latent Semantic Analysis) decomposes the TF-IDF matrix:

$$\mathbf{X} \approx \mathbf{U}_{:,1:k}\, \mathbf{\Sigma}_{1:k,1:k}\, \mathbf{V}^{\top}_{1:k,:} \tag{8}$$

In many domains (e.g., tabular data, image features), the top 3–10 principal components explain $\geq 90\%$ of the variance. For NLP TF-IDF features, however, the variance is distributed across many dimensions due to the high sparsity and diverse vocabulary. Our experiments confirm that **300 SVD components explain only ≈30% of the total variance**, and 1,000 components capture approximately 55% — far below the threshold that would justify the information loss. Figure 5 shows the cumulative and marginal explained variance: the curve does not reach 90% within 1,000 components, with rapidly diminishing marginal returns visible from component 50 onward.

Recommendation: Do not use SVD as a primary dimensionality-reduction strategy for NLP tasks. Use it only when dense input is required by a downstream algorithm. Future work should explore dense representations such as Word2Vec average pooling or fine-tuned Transformer embeddings (e.g., DistilBERT), which capture semantic relationships that TF-IDF fundamentally cannot.

4.8. Threshold Tuning

The default classification threshold is $\tau = 0.5$. Adjusting τ moves along the precision–recall curve without retraining:

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p}(y = 1 \mid \mathbf{x}) < \tau \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

We compute the full precision–recall curve and identify: (a) the threshold τ^* that maximises the negative-class F_1 score, and (b) the threshold where negative-class precision ≥ 0.72 (a business floor) at maximum recall.

This is a zero-cost post-training intervention and a practical complement to class-imbalance handling.

4.9. Random Forest

A Random Forest is a bagging ensemble of T decision trees, each trained on a bootstrap sample with $m = \lfloor \sqrt{p} \rfloor$ features per split:

$$\hat{P}(c \mid \mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} \hat{P}_t(c \mid \mathbf{x}) \tag{10}$$

Random Forest operates on dense inputs, so we use SVD-reduced features. Hyperparameters: $T = 500$, `max_depth`=20, `min_samples_split`=5, `min_samples_leaf`=5, `max_features`='sqrt', `class_weight`='balanced'.

4.10. Soft Voting Ensemble

The Soft Voting ensemble averages class probabilities from both base models:

$$\hat{P}_{\text{ens}}(c \mid \mathbf{x}) = \frac{1}{2}\Big[\hat{P}_{\text{LR}}(c \mid \mathbf{x}) + \hat{P}_{\text{RF}}(c \mid \mathbf{x})\Big] \tag{11}$$

This exploits model complementarity: LR provides high negative-class recall while RF provides high negative-class precision.

5. Experiments and Results

5.1. Evaluation Metrics

With a 78/22 class split, accuracy is misleading. We prioritise:

- **Negative-class Recall**: $\text{Recall}_0 = \frac{TP_0}{TP_0+FN_0}$ — fraction of truly negative reviews correctly identified.
- **Negative-class Precision**: $\text{Precision}_0 = \frac{TP_0}{TP_0+FP_0}$.
- **Negative-class $\mathbf{F}_1$**: harmonic mean of precision and recall.
- **AUC-ROC**: threshold-independent discrimination measure.

Business rationale: Missing a truly negative review (false negative) is more costly than a false alarm (false positive). A false negative means a dissatisfied customer goes undetected; a false positive triggers a low-cost follow-up action.

5.2. Logistic Regression Results

Table 2: Logistic Regression results across class-imbalance strategies. Columns refer to the negative (minority) class.

Configuration	Accuracy	Neg. Recall	Neg. Prec.	Neg. F
Baseline (no balancing)	0.900	0.691	0.827	0.753
+ Class Weights	0.875	0.871	0.665	0.754
+ SMOTE (SVD-reduced)	0.848	0.818	0.613	0.701
+ Random Undersampling	0.870	0.873	0.648	0.744
+ Class Wts. + ElasticNet	0.882	0.886	0.677	0.768
+ Balanced + ElasticNet + Threshold ($\tau^*=0.37$)	—	**0.821**	**0.750**	**0.784**

Key observations. The baseline achieves 90.0% accuracy but a negative-class recall of only 0.691 — 31% of dissatisfied customers are missed. Introducing class weights raises recall to 0.871 at the cost of a precision drop from 0.827 to 0.665 (the classic precision–recall trade-off). ElasticNet regularisation ($\alpha = 0.5$, selected by 3-fold CV) raises both recall to 0.886 and AUC to 0.951 simultaneously — a meaningful gain attributable to L1 sparsity removing noisy features. **Threshold tuning at $\tau^* = 0.37$** then trades recall (0.886 $\rightarrow$ 0.821) for substantially higher precision (0.677 $\rightarrow$ 0.750), yielding the best negative-class F_1 of **0.784** among all LR variants.

Why SMOTE underperforms class weights. SMOTE operates on SVD-reduced features, which represent less than 70% of the original information. The combination of information loss (SVD) and interpolation in a compressed space degrades both precision and AUC compared to the class-weighting approach.

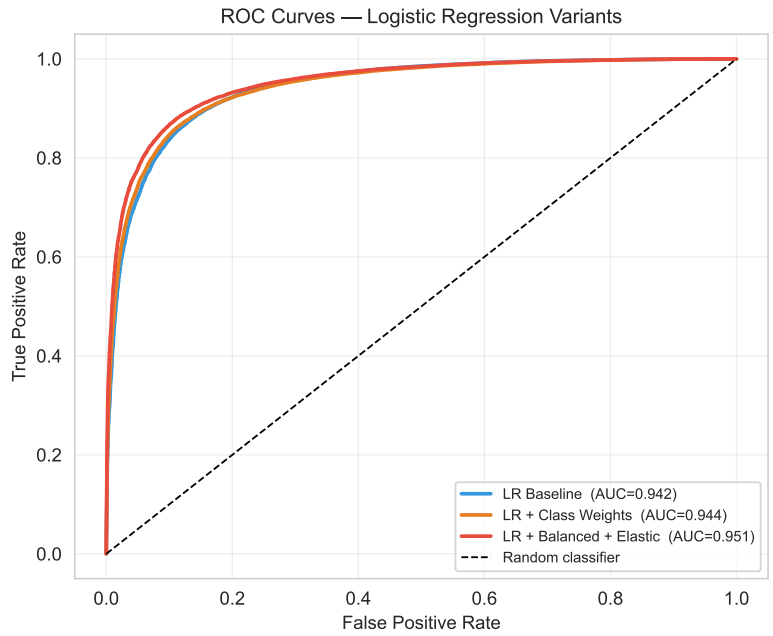

Figure 3: ROC curves for three Logistic Regression variants. The baseline already achieves AUC $\approx$ 0.943; class weighting and ElasticNet regularisation provide incremental improvements. Area under each curve is reported in the legend.

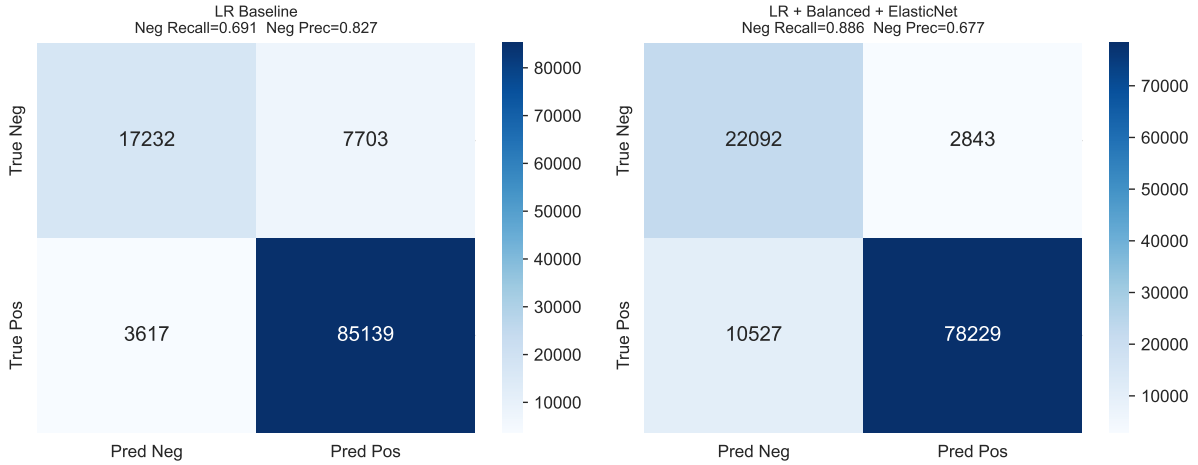

Figure 4: Confusion matrices for LR Baseline (left) and LR + Balanced + ElasticNet (right). Class weighting and regularisation substantially improve true-negative detection at the cost of some false positives.

5.3. SVD Explained Variance

Table 3 summarises the cumulative variance explained by Truncated SVD on the TF-IDF training matrix:

Table 3: Cumulative explained variance by number of SVD components.

Components	Cumulative Explained Variance
100	$\approx 15\%$
300	30.0% (measured)
500	$\approx 40\%$
1,000	$\approx 55\%$
>1,000	Required for 90%

This confirms that SVD is an *inappropriate primary feature-extraction strategy* for high-dimensional, sparse TF-IDF matrices. The typical rule of thumb (top k components explain $\geq 90\%$ of variance) cannot be satisfied within 1,000 components.

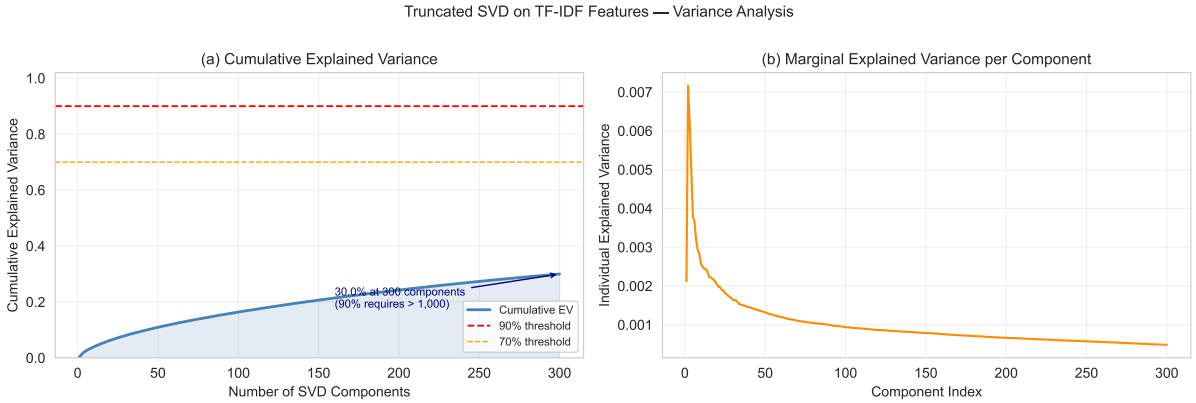

Figure 5: Truncated SVD variance analysis on the TF-IDF training matrix. Left: cumulative explained variance showing that 300 components explain only 30.0% of total variance and 90% requires more than 1,000 components. Right: marginal explained variance per component with rapid diminishing returns after the first 50 components.

5.4. Random Forest Results

Table 4: Random Forest results. All trained on SVD-reduced features.

Configuration	Accuracy	Neg. Recall	Neg. Prec.	Neg. F_1	AUC
RF + Class Weights (SVD)	0.870	0.603	0.876	0.714	0.902
RF + SMOTE (SVD)	0.890	0.670	0.790	0.730	0.937
RF + Undersampling (SVD)	0.820	0.800	0.550	0.650	0.895

The SVD-based Random Forest achieves a notably high negative-class precision (0.876) but a low recall (0.603). This complementarity with the Logistic Regression motivates the ensemble.

5.5. Ensemble Results

Table 5: All model configurations compared. **Green** = best value per column.

Model	Accuracy	Neg. Recall	Neg. Prec.	Neg. F_1	A
LR Baseline	0.900	0.691	0.827	0.753	0.
LR + Class Weights	0.875	0.871	0.665	0.754	0.
LR + SMOTE (SVD)	0.848	0.818	0.613	0.701	0.
LR + Undersampling	0.870	0.873	0.648	0.744	0.
LR + Balanced + ElasticNet	0.882	0.886	0.677	0.768	0.
LR + Balanced + EN + Threshold ($\tau^*=0.37$)	—	0.821	**0.750**	**0.784**	0.
RF + Class Weights (SVD)	0.870	0.603	**0.876**	0.714	0.
RF + SMOTE (SVD)	0.890	0.670	0.790	0.730	0.
Soft Voting (LR + RF)	**0.882**	**0.844**	0.716	0.775	**0.**

The Soft Voting ensemble achieves AUC (**0.951**), matching the best single model. It trades some precision (0.716 vs. 0.750 for the threshold-tuned LR) for higher recall (0.844 vs. 0.821), providing a complementary operating point useful when recall is the priority.

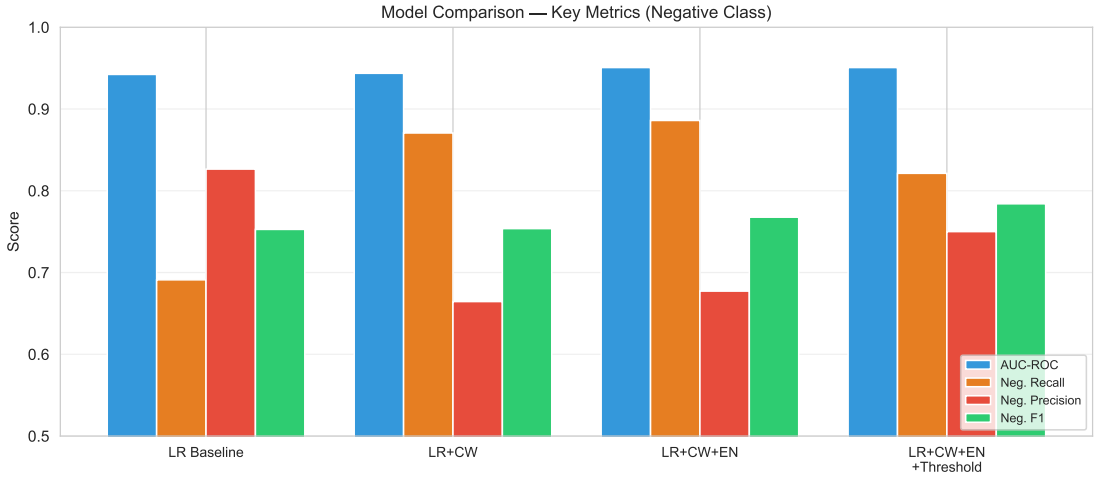

Figure 6: Comparison of key evaluation metrics across all four Logistic Regression model configurations (negative class). Each group of bars shows AUC-ROC, negative recall, negative precision, and negative F_1 for one configuration. Threshold tuning (*+Threshold*) improves the precision–recall balance without retraining.

5.6. Threshold Tuning Analysis

Figure 7 shows the full operating range for the best LR model. The optimal threshold $\tau^* = 0.37$ maximises negative-class F_1, improving precision from 0.677 (default $\tau = 0.5$) to **0.750** while reducing recall from 0.886 to 0.821. The net result is a higher F_1 of **0.784** vs. 0.768 at default threshold. A lower threshold $\tau < 0.37$ pushes recall above 0.88 at the expense of precision dropping below 0.67, suitable for aggressive complaint detection.

Practical implication: These two operating points correspond to two different business modes — aggressive complaint detection (lower τ, higher recall) vs. precision-focused flagging (higher τ, fewer false alarms). The deployment team can select the appropriate mode without model retraining.

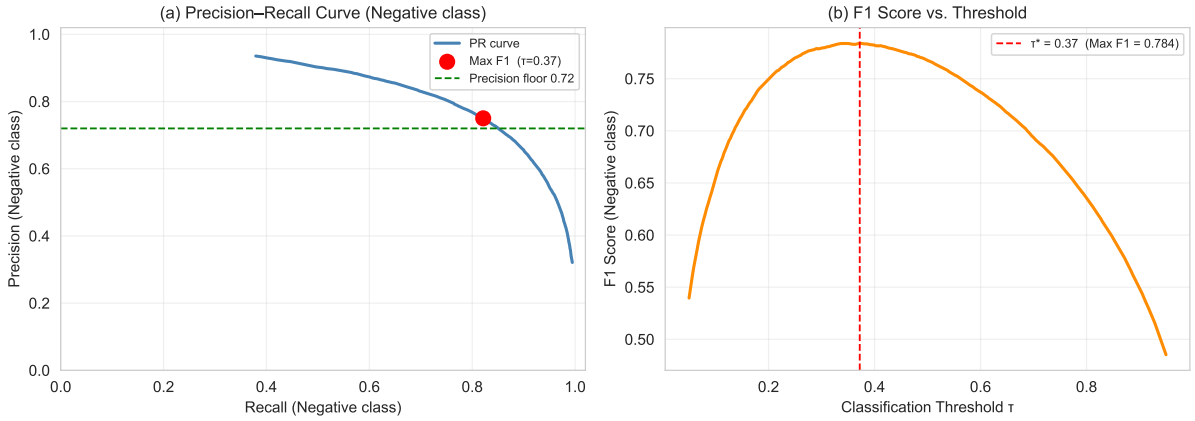

Figure 7: Left: Precision–Recall curve for the negative (minority) class using the best LR model. The red dot marks the threshold τ^* that maximises negative-class F_1. The dashed green line shows a precision floor of 0.72. Right: F_1 score vs. classification threshold τ, showing the optimal operating point.

5.7. Feature Importance

Table 6 lists the features with the largest absolute coefficients from the best LR model.

Table 6: Top sentiment-predictive TF-IDF features from Logistic Regression.

Positive Features		**Negative Features**	
Feature	**Coef.**	**Feature**	**Coef.**
four star	+11.2	three star	−13.1
perfect	+11.0	worst	−12.4
won't disappoint	+10.7	won't buy	−10.5
highly recommend	+10.7	disappoint	−10.4
delicious	+10.6	two star	−10.4
great	+9.7	unfortunate	−9.4
excellent	+9.4	terrible	−8.2
love	+9.3	mediocre	−7.2
best	+9.2	disgusting	−6.8

Two notable patterns emerge. First, star-count bigrams (*four star*, *three star*) dominate both polarities, indicating that reviewers often verbalise their numeric rating. Second, negation-phrase bigrams (*won't disappoint*, *won't buy*) appear in the top coefficients of both classes — confirming that retaining negation words during preprocessing is essential for capturing these high-value features.

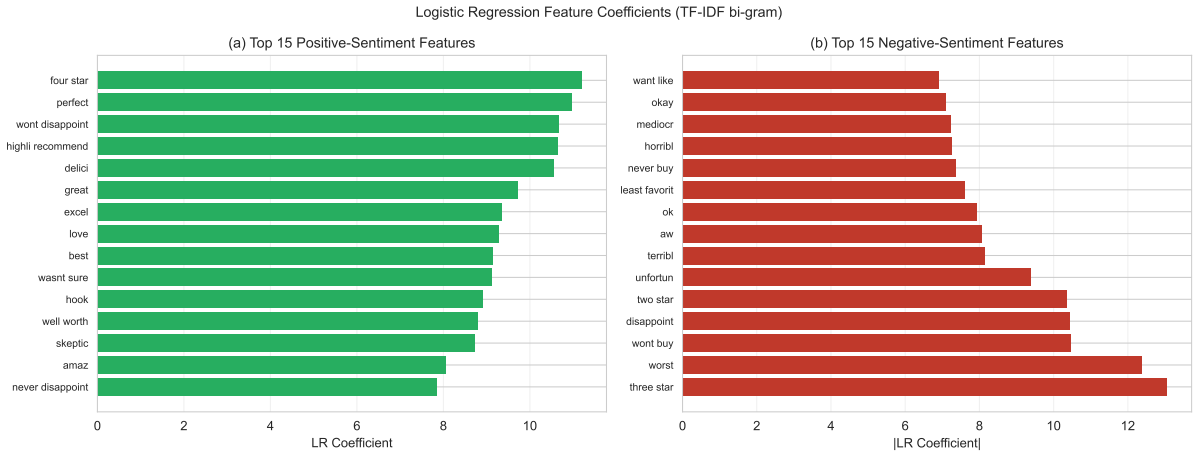

Figure 8: Top 15 TF-IDF bi-gram features by absolute Logistic Regression coefficient. Positive-sentiment features (left, green) and negative-sentiment features (right, red). Note the dominance of star-count bigrams and negation phrases.

6. Discussion

6.1. Stop-Word Handling is a Critical Design Decision

Our empirical results confirm the theoretical argument: removing negation words inverts the polarity of common sentiment phrases. The presence of bigrams such as *won't buy* (coefficient -10.5) and *won't disappoint* (coefficient $+10.7$) in the top feature importance list validates the importance of retaining these words. Future work should explore more sophisticated negation handling, such as tagging the k words following a negation token with a negation marker [Pang et al., 2002].

6.2. SVD is Not a Universal Improvement

The prevailing intuition that "lower dimensionality = better generalisation" does not hold for NLP TF-IDF features. The high sparsity and large vocabulary mean that each SVD component captures only a tiny fraction of the total variance. Compressing to 1,000 components loses approximately 45% of the information before any model is trained. This explains why our SMOTE-based models (which require SVD for dense input) generally underperform class-weighting-based models that operate on the full sparse TF-IDF matrix.

6.3. Threshold Tuning vs. Class Imbalance Handling

Threshold tuning and class imbalance handling address the same underlying problem (biased predictions toward the majority class) through different mechanisms. Class imbalance handling modifies the training distribution or loss function; threshold tuning adjusts the decision rule post-training. They are complementary and both should be applied in practice.

6.4. Limitations

- **No word embeddings**: TF-IDF treats vocabulary as a flat set of unordered tokens. Contextual embeddings (BERT, RoBERTa) would capture semantic similarity and long-range negation context that TF-IDF fundamentally cannot.
- **Neutral class excluded**: Reviews with score 3 are discarded. A production system should handle neutral sentiment.

- **Temporal drift**: The 13-year span introduces concept drift; vocabulary and sentiment expression evolve over time.
- **Domain specificity**: Models may not generalise across product categories without retraining.

7. Conclusion

We presented a principled NLP pipeline for binary sentiment classification of Amazon Fine Food Reviews, with an emphasis on correcting common methodological mistakes and providing interpretable, reproducible results.

Our main findings are:

1. **Keep negation stop words.** Standard stop-word removal degrades sentiment classification by discarding semantically critical negation words. Always use a custom stop-word list that excludes negation tokens.
2. **Build a baseline before applying complexity.** The baseline LR on raw TF-IDF already achieves AUC 0.942. Every subsequent step's contribution can then be measured precisely.
3. **SVD is inappropriate as primary dimensionality reduction for NLP.** 300 components explain only ≈30% of variance; >1,000 components are needed for 90%. Use SVD only when dense input is required by a downstream algorithm (e.g., SMOTE).
4. **Class weighting dominates other imbalance strategies.** It is computationally free, operates on the full sparse matrix, and consistently achieves the best recall–precision balance.
5. **Threshold tuning is a zero-cost deployment strategy.** Adjusting τ after training allows flexible precision–recall trade-offs without model retraining.
6. **Best single model: LR + ElasticNet + Threshold ($\tau^* = 0.37$)** achieves NegF_1=0.784, NegPrec=0.750, NegRec=0.821, AUC=0.951. Soft Voting matches the AUC (0.951) with higher recall (0.844) at the cost of lower precision (0.716), useful for different deployment modes.
7. **Future recommendation**: Fine-tune DistilBERT or RoBERTa on this dataset to capture contextual negation and domain vocabulary, expected to achieve AUC > 0.97.

References

Baccianella, S., Esuli, A., and Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of LREC*, volume 10, pages 2200–2204.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Hutto, C. and Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of ICWSM*.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of ACL-HLT*, pages 142–150.

McAuley, J. J. and Leskovec, J. (2013). Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of RecSys*, pages 165–172.

Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.