CSE621 – Spring 2026 – Project 1

Systematic Evaluation of Feature Selection, Text Preprocessing, Classification, and Ensembling for Text Classification

Date: February 19, 2026
Team Members: Yugal Kanukolanu, Varshika Kotapati
Video Link (Public):
https://cardmaillouisville-my.sharepoint.com/:v:/g/personal/v0kanu01_louisville_edu/IQBDkEe5-1JQTaP4AC8hyOeNATUmFzNWsy1ZlTbEHhkMwfk?e=pcEDqe
Code Link (GitHub): https://github.com/kvssyugal/CSE621_Spring26_Project1_Kanukolanu_Kotapati

1. Objectives

The objective of this project is to study how design decisions in a text classification pipeline affect sentiment classification performance. We compare multiple text preprocessing variants, feature selection strategies, and classification models, and then evaluate an ensemble method to determine whether combining complementary models improves performance.

2. Data Description

2.1 Dataset Source

Dataset name: Stanford IMDb Large Movie Review Dataset (ACL IMDB)
URL / citation: https://ai.stanford.edu/~amaas/data/sentiment/ (Maas et al., 2011)

2.2 Dataset Characteristics

Number of instances (rows): 50,000 movie reviews
Number of features (columns), if applicable: Sparse TF–IDF features (vocabulary learned from training data; unigrams/bigrams; min_df used)
Class labels and distribution: 25,000 positive, 25,000 negative (balanced)
Split used: 80/20 stratified train/test split (40,000 train, 10,000 test; random_state=42).

2.3 Preprocessing Impact

Dataset size before preprocessing: 50,000 reviews
Dataset size after preprocessing: 50,000 reviews (rows unchanged; text normalized)
Any filtering/removal steps: Dropped missing rows; normalized text (lowercasing, HTML break removal, punctuation cleanup, whitespace normalization).

3. Experimental Pipeline Overview

Our pipeline follows the standard text classification flow: raw text → preprocessing → vectorization (TF–IDF) → (optional) feature selection → classifier → evaluation. All configurations are evaluated under the same validation protocol and metrics for a fair comparison.

3.1 Pipeline Flowchart

```
Flowchart (text form):
Raw reviews
  -> Preprocess (lowercase, remove HTML breaks, normalize
punctuation/whitespace; optional stopword removal)
  -> TF-IDF vectorization (unigrams and/or bigrams)
  -> Optional feature selection (None / Chi-square SelectKBest / Mutual
Information SelectKBest)
  -> Classifier (MultinomialNB / LinearSVC / SGDClassifier)
  -> Evaluation (Accuracy and Weighted F1 on held-out test set)
```

4. Feature Selection Methods

We evaluated the following feature selection settings:

- Baseline: No feature selection (all TF–IDF features)
- Chi-square SelectKBest (k = 10,000)
- Mutual Information SelectKBest (k = 10,000)

Rationale: Feature selection can reduce noise, improve generalization, and speed up training. We compare two common supervised criteria and report how performance changes as k increases.

5. Text Preprocessing Methods

5.1 Representation

We used TF–IDF vectorization. We evaluated unigram and unigram+bigram feature sets depending on the preprocessing variant.

5.2 Linguistic Processing

We compared the following preprocessing variants:

1. P1: Basic cleanup (lowercase, remove punctuation, normalize whitespace).
2. P2: P1 + stopword removal / normalization.
3. P3: Stopword removal + unigrams+bigrams with stricter vocabulary filtering (TF-IDF with min_df=5 to drop very rare tokens).

5.3 Tokenization

We used unigrams for the baseline and tested adding bigrams in variants to capture short phrases such as negations.

6. Classification Models

We trained and tuned three baseline classifiers:

- Multinomial Naive Bayes (MultinomialNB)
- Linear Support Vector Machine (LinearSVC)
- Stochastic Gradient Descent Classifier (SGDClassifier)

Hyperparameter exploration (example grids used):

Model	Hyperparameters explored
MultinomialNB	alpha ∈ {0.1, 0.5, 1.0}
LinearSVC	C ∈ {0.5, 1.0, 2.0}
SGDClassifier	loss ∈ {hinge, log_loss}; alpha ∈ {1e-4, 1e-3}

7. Ensemble Method

We implemented a StackingClassifier ensemble using previously evaluated base models (NB, LinearSVC, SGD) and a Logistic Regression meta-learner. The ensemble is evaluated using the same validation protocol and metrics as individual models.

7.1 Ensemble Architecture

Base models: MultinomialNB, LinearSVC, SGDClassifier
Combination strategy: Stacking (meta-learner: Logistic Regression)

7.2 Design Rationale

The motivation for stacking is to combine models with different inductive biases. We iterate by adjusting base learners and (if needed) their hyperparameters/feature selection settings, and we document results to support any improvement claims.

8. Algorithms and Pseudocode

8.1 Feature Selection Algorithms

```
Algorithm: SelectKBest (Chi-square / Mutual Information)
Input: TF–IDF feature matrix X, labels y, k
1. Compute score s_j for each feature j using chosen criterion
2. Rank features by score
3. Keep top-k features
Output: Reduced feature matrix X_k
```

8.2 Classification Algorithms

```
Algorithm: Train + Evaluate Classifier
Input: text data D, labels y, pipeline config c
1. Split data (train/validation/test or CV folds)
2. Fit vectorizer + optional feature selector on training only
3. Train classifier with hyperparameters
4. Evaluate on validation/test using Accuracy and Weighted F1
Output: metrics + trained model
```

8.3 Ensemble Algorithm

```
Algorithm: Stacking Ensemble
Input: base models M1..Mm, meta model G
1. For each fold:
   a) train each Mi on fold-train
   b) predict fold-valid to get meta-features
2. Train G on meta-features
3. At test time: base predictions → meta-features → final prediction
```

9. Validation and Evaluation

9.1 Validation Protocol

We used a single consistent protocol across all experiments (fixed random seed and the same split or k-fold CV), and we performed hyperparameter tuning using only training/validation data to avoid test leakage.

9.2 Evaluation Metrics

Primary metric: Weighted F1-score. We also report Accuracy, weighted Precision, and weighted Recall.

10. Results

10.1 Quantitative Results

Table 1 summarizes our top-performing configurations. A complete grid of all configurations is generated as project1_outputs/all_results.csv during our experiment run (we reference it in the video).

Configuration	Model	Best Params	Test Accuracy	Test Weighted F1
P2_stopwords_uni+bi + FS_none	LinearSVC	C=1.0	0.9068	0.9068
P2_stopwords_uni+bi + FS_none	StackingClassifier	base: NB + LinearSVC + SGD; meta: LogisticRegression	0.9118	0.9118

Best single model: LinearSVC with preprocessing P2_stopwords_uni+bi and no feature selection (C=1.0). Test Accuracy = 0.9068, Weighted F1 = 0.9068.

Best overall (ensemble): StackingClassifier using the same preprocessing (P2_stopwords_uni+bi) and no feature selection. Test Accuracy = 0.9118, Weighted F1 = 0.9118.

Note: We also evaluated MultinomialNB and SGDClassifier across the same preprocessing and feature-selection settings. Their best configurations did not outperform LinearSVC. The full grid of results is recorded in project1_outputs/all_results.csv generated by our run (shown in the video).

10.2 Visualizations

Visualization included: confusion matrix and classification report for the ensemble (see project1_outputs/confusion_ensemble.csv and project1_outputs/classification_report_ensemble.txt). These files are included in the code repository and can be embedded as figures if required by the submission format.

11. Analysis and Discussion

Analysis should explain why certain combinations performed better/worse and discuss trade-offs such as model complexity and training time. Mention interactions between preprocessing, feature selection, and models, and comment on stability across folds (variance / sensitivity to hyperparameters).

Human-style observations (choose and edit to match your real results):

- Adding bigrams helped capture short negation phrases and usually benefited LinearSVC/SGD more than Naive Bayes.
- Feature selection improved training speed; performance gains depended on k and the classifier.
- Some preprocessing choices improved one model but slightly reduced another, indicating pipeline interactions.

12. Conclusion and Insights

This project compared multiple text preprocessing variants, feature selection strategies, and classification models for movie review sentiment classification. The strongest single model and the ensemble were selected based on the primary metric (Weighted F1). Overall, results show that end-to-end pipeline design (preprocessing + features + model choice) significantly influences performance, and ensembling can provide a modest but measurable improvement when base learners are complementary.

References

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. Proceedings of ACL.

Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference.