Detecting Opinion Spam Using Classical Machine Learning Methods: Assignment

Renato Mutavdzic (4773918)

Jacob Bae (0732567)

Jose Lizana (7162642)

Abstract

This project explores opinion spam detection using classical machine learning on the Ott et al. (2011) Deceptive Opinion Spam Corpus v1.4. We classify hotel reviews as truthful or deceptive and compare five algorithms: Multinomial Naive Bayes, L1-regularized Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting. All are trained under unigram and unigram+bigram bag-of-words features. Models were trained with four-fold cross-validation (folds 1-4) and assessed on a held-out test fold (fold 5). Performance was measured by accuracy, precision, recall, and macro-averaged F1-score.

Across both sets, Naive Bayes achieved the best performance, with Random Forest and Logistic Regression following. Gradient Boosting and Decision Tree showed the worst performance of all models. Bigrams yielded small gains on cross-validation for Naïve Bayes, Random Forest and Logistic Regression, but did not have the same impact on testing. Overall, Naive Bayes emerges as an efficient baseline for text-based deception detection with simple n-gram features.

1. Introduction

Online reviews heavily influence consumer behavior and business reputation, yet deceptive reviews can hurt trust in digital platforms. Detecting such opinion spam is then an important problem in data mining and natural language processing, trying to identify defective content using linguistic signals.

We apply supervised machine learning to the Ott et al. (2011) Deceptive Opinion Spam Corpus v1.4, which provides folds of truthful and deceptive hotel reviews suitable for controlled comparison. We evaluate five models: Multinomial Naive Bayes, L1-regularized Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting. Each model is trained and validated using four-fold cross-validation (folds 1-4) and then evaluated on fold 5 as a test set. We study two feature representations (unigrams and unigram+bigram bag-of-words) and report cross-validated mean accuracy, precision, recall, and macro-F1. Same metrics will be applied on the test set.

2. Data Description

2.1. Dataset Overview

This project uses the Deceptive Opinion Spam Corpus v1.4, created by Ott et al. (2011) at the University of Chicago. The dataset consists of 1,600 hotel reviews written in English, evenly split between 800 truthful and 800 deceptive reviews. Each review describes a visit at one of 20 Chicago hotels, which ensures consistency across classes. Truthful reviews were collected from TripAdvisor, while deceptive reviews were generated by Amazon Mechanical Turk workers.

The dataset is pre-divided into five non-overlapping folds, each containing 320 reviews (160 truthful, 160 deceptive). This enables standardized cross-validation and holdout evaluation. Folds 1–4 were used for training, while fold 5 served as the final test set.

2.2. Data Preprocessing

Before model training, each review went through several preprocessing steps to normalize the text and reduce noise:

- Lowercasing: All words were converted to lowercase to ensure consistency.
- Tokenization: Text was split into word tokens using scikit-learn's CountVectorizer.
- Stopword Removal: Common English stopwords (e.g., the, and, is) were excluded to focus on better terms.
- Vectorization: Reviews were transformed into numerical feature vectors using a bag-of-words approach with two configurations:
    - Unigrams: single-word tokens
    - Unigrams + Bigrams: includes two-word sequences to capture short contextual patterns (e.g., very clean, highly recommend)
- Vocabulary Filtering: Words appearing in fewer than two reviews or in more than 90% of all reviews were discarded to prevent sparsity or redundancy, respectively.

2.3. Feature Representation

The resulting feature matrices were high-dimensional and sparse, with approximately 3,000–4,000 features in the unigram setup and 8,000–9,000 features in the unigram+bigram (depending on fold composition). Each feature represents a term frequency count within the review. These representations were given directly to the learning algorithms.

2.4. Data Splitting Protocol

The predefined fold structure was as follows:

- Cross-validation: Performed on folds 1–4 to tune hyperparameters and estimate model variance.
- Final evaluation: Conducted on fold 5 to assess generalization to unseen data.

This consistent data partitioning ensures fair comparisons between models.

3. Methods

3.1. Model Selection

To evaluate the effectiveness of different learning algorithms for opinion-spam detection, five supervised models were implemented with scikit-learn:

- Multinomial Naive Bayes: A probabilistic model assuming conditional independence between words. Well suited for text data. It estimates class-conditional word distributions and typically performs strongly on high-dimensional and sparse inputs.
- L1-Regularized Logistic Regression: A linear classifier modeling the log-odds of class membership. The L1 penalty allows sparsity in learned weights, improving interpretability by selecting only the most discriminative terms.
- Decision Tree: A non-linear model that recursively partitions the feature space to minimize impurity. It captures feature interactions but is prone to overfitting on small datasets.
- Random Forest: An ensemble of decision trees trained on bootstrap samples and random feature subsets. Averaging across trees improves generalization and provides feature-importance estimates.
- Gradient Boosting: An ensemble that sequentially builds trees to correct residual errors, often achieving strong accuracy.

These models represent well different trade-offs between bias, variance, and interpretability. This enables a broad comparison across probabilistic, linear, and ensemble-based approaches.

3.2. Feature Representation

Text data were represented using bag-of-words vectors generated with scikit-learn's CountVectorizer in two configurations:

- Unigram model: individual tokens as features
- Unigram + Bigram model: single and consecutive two-word sequences (e.g., very clean, highly recommend) to capture short contextual patterns

Stopwords were removed using the default English list, and all text was lowercased before vectorization. Tokens were filtered using document-frequency thresholds (min_df = 2, max_df = 0.95) to exclude extremely rare or overly common terms. The resulting matrices contained several thousand dimensions per configuration.

3.3. Evaluation Procedure

Experiments followed the dataset's predefined structure:

- Training phase: folds 1-4 for model training and validation
- Cross-validation phase: four-fold CV within these folds to tune hyperparameters and estimate performance variance
- Testing phase: fold 5 left for final evaluation

Each model was trained twice, once using unigrams and once using unigram + bigram features. In total, we have ten configurations in total.

3.4. Hyperparameter Optimization

Hyperparameters were tuned with Randomized Search Cross-Validation within the training folds. This approach samples from a specified parameter space, providing tuning quality comparable to grid search while reducing computation time. The tuned parameters are summarized in Table 1.

Table 1. Model selection and their tuned parameters

Model	**Tuned Parameters**
Naïve Bayes	None (default smoothing α=1)
Logistic Regression	C (inverse regularization strength), penalty='l1', solver='liblinear'
Decision Tree	max_depth, ccp_alpha
Random Forest	n_estimators, max_depth, max_features
Gradient Boosting	n_estimators, learning_rate, max_depth

3.5. Evaluation Metrics

Performance was assessed using four evaluation metrics, each macro-averaged across both classes:

- Accuracy: Proportion of correctly classified reviews.
- Precision: Model's ability to avoid false positives.
- Recall: Model's ability to detect all relevant instances.
- F1-score: Harmonic mean of precision and recall, providing a balanced measure of performance.

Cross-validated mean metrics (folds 1–4) were evaluated using F1, and test-set metrics (fold 5) were reported with full metrics. Confusion-matrix plots and histograms were also generated for qualitative inspection of classification errors.

3.6. Implementation Details

All experiments were implemented in Python 3.11 using scikit-learn, NumPy, Pandas, and Matplotlib. The project structure included modular scripts:

- main.py – orchestrates training, testing, and result storage
- cv_model.py – handles cross-validation and hyperparameter tuning
- model2.py – defines model wrappers and training functions
- preprocessing.py – performs vectorization and cleaning
- utils.py – manages data loading and fold organization

Results were automatically saved as CSV files along with confusion matrices in the /CM directory.

4. Results

4.1. Cross-Validation Performance

Cross-validation (CV) was performed on folds 1–4 for all five models under both unigram and unigram + bigram representations.
Tables 2 and 3 summarize the macro-averaged F1 metrics across folds.
Overall performance remained consistent across folds, with Naive Bayes and Random Forest showing the highest mean F1-scores.

Table 2. Cross-Validation results (Unigram model)

Model	**CV F1 (Unigram)**
Naïve Bayes	0.831
Logistic Regression (L1)	0.793
Decision Tree	0.703
Random Forest	0.823
Gradient Boosting	0.777

Table 3. Cross-Validation results (Unigram + Bigram model)

Model	**CV F1 (Bigram)**
Naïve Bayes	0.865
Logistic Regression (L1)	0.803
Decision Tree	0.703
Random Forest	0.841
Gradient Boosting	0.774

Across folds, Naive Bayes achieved the best mean F1 on both representations, followed by Random Forest and Logistic Regression.
Adding bigrams slightly improved CV performance for Naive Bayes (+0.034), Random Forest (+0.018), and Logistic Regression (+0.010), while having no effect on Decision Tree and a marginal drop for Gradient Boosting.

4.2. Holdout (Fold-5) Results

Evaluation on fold 5 confirmed the same ranking observed in cross-validation. Naive Bayes remained the best performer, with the highest accuracy (0.881) and F1 (0.881) on unigrams, while Random Forest and Logistic Regression followed closely. Gradient Boosting showed moderate improvement with bigrams, whereas other models saw minor declines.

Table 4. Test results

Model	**Features**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Naïve Bayes	Unigram	0.881	0.883	0.881	0.881
Logistic Regression (L1)	Unigram	0.813	0.813	0.813	0.812
Decision Tree	Unigram	0.644	0.644	0.644	0.644
Random Forest	Unigram	0.831	0.831	0.831	0.831
Gradient Boosting	Unigram	0.731	0.734	0.731	0.730
Naïve Bayes	Unigram + Bigram	0.850	0.851	0.850	0.850
Logistic Regression (L1)	Unigram + Bigram	0.800	0.800	0.800	0.800
Decision Tree	Unigram + Bigram	0.644	0.644	0.644	0.644
Random Forest	Unigram + Bigram	0.794	0.796	0.794	0.793
Gradient Boosting	Unigram + Bigram	0.763	0.767	0.763	0.762

While bigrams increased CV scores, they did not bring higher test performance for most models, except for a small gain for Gradient Boosting (+0.032 F1).

4.3. Confusion-Matrix Analysis

Inspection of the confusion matrices (Appendix A) revealed that false positives—truthful reviews predicted as deceptive—occurred slightly more frequently than false negatives. This pattern suggests that emotionally charged but genuine reviews occasionally resemble deceptive writing.

Naive Bayes and Random Forest demonstrated the most balanced confusion matrices, whereas Decision Tree exhibited greater bias toward one class.

4.4. Top-Feature Analysis

Feature importance inspection on Logistic Regression and Naïve Bayes showed distinctive linguistic cues:

Class	**Top 5 Indicative Terms**
Deceptive	Wonderful, definitely, experience, highly recommend, amazing
Truthful	Room, location, staff, price, clean

Deceptive reviews tend to rely on emotional adjectives, while truthful review reference specific factual details such as facilities and service.

4.5. Summary of Findings

Our best overall model is Naive Bayes (CV F1 = 0.865, Test F1 = 0.881), with Random Forest right behind, closely followed by Logistic Regression (L1).
The weakest model is Decision Tree, limited by overfitting and low generalization.
Implementing bigrams show small improvements in cross-validation but minimal or negative effect on the held-out test set, except for Gradient Boosting.

In summary, simple probabilistic models outperformed more complex ensembles, demonstrating that interpretable linear methods with unigram features remain highly effective for deception-detection tasks on this dataset.

5. Analysis & Discussion

5.1. Comparison Between Naïve Bayes and Logistic Regression

The comparison between Multinomial Naive Bayes and L1-regularized Logistic Regression highlights the classic trade-off between probabilistic simplicity and discriminative modeling. Naive Bayes assumes conditional independence among words, which allows very fast training and strong generalization on sparse text data. In contrast, Logistic Regression directly optimizes the decision boundary and learns weighted feature importance through regularization.

In both unigram and bigram settings, Naive Bayes outperformed Logistic Regression by roughly 4-6 percentage points in F1-score on cross-validation and by about 7-8 points on the held-out test set (0.881 against 0.812 for unigrams).

These results indicate that the independence assumption provides a robust bias well suited for small, balanced corpora. Logistic Regression remained competitive and interpretable, but its gains from discriminative learning were offset by slightly higher variance and sensitivity to feature sparsity.

5.2. Comparison Between Linear and Ensemble Models

Ensemble models (Random Forest and Gradient Boosting) achieved comparable, but not superior performance to the best linear and probabilistic approaches.

Random Forest ranked second overall (Test F1 = 0.831), only marginally below Naive Bayes, while Gradient Boosting performed poorly compared to other models (Test F1 = 0.730 unigram; 0.762 bigram). This pattern suggests that for short texts with thousands of sparse features, ensemble methods provide little added benefit. Tree-based models can capture non-linear interactions, but in bag-of-words representations such interactions contribute minimally when strong individual terms dominate prediction.

Consequently, Naive Bayes and Random Forest offer the best trade-off between interpretability, efficiency, and generalization.

5.3. Effect of Adding Bigrams

Introducing bigrams produced mixed effects across models.

Cross-validation showed small F1 gains for Naive Bayes (+0.034), Random Forest (+0.018), and Logistic Regression (+0.010), while Decision Tree was unchanged and Gradient Boosting slightly decreased (–0.003). However, on the held-out test fold, most models performed slightly worse with bigrams, except Gradient Boosting, which improved (+0.032 F1). This discrepancy indicates that bigrams may overfit the limited training data rather than generalize better. While bigrams capture short phrases like “not clean” or “very helpful”, unigrams already encode most discriminative cues in this dataset.

Overall, the benefit of bigrams is minor, suggesting that unigram features are good enough for small deception-detection corpora.

5.4. Most Important Features

Feature-importance analysis provides qualitative insights into how models distinguish between truthful and deceptive reviews. The top-weighted terms in Logistic Regression and Naive Bayes suggest that deceptive reviews rely heavily on emotional language, while truthful reviews focus on specific and concrete details. In case of deceptive language, representative terms are words like “wonderful” and “experience”, while truthful have words like “room”, “location” and “staff”. This shows that deceptive reviews often use positive adjectives and adverbs, while truthful ones tend to include facts about facilities or service quality.

5.5. Broader Interpretation

From a data-mining perspective, these results demonstrate that linguistic regularities in deceptive text are learnable with classical machine-learning methods.

The strong showing of Naive Bayes (achieving up to 0.881 F1), illustrates that interpretable, low-variance models can rival more complex ensembles on small text corpora.

The narrow performance gaps among algorithms suggest unfavorable returns from model complexity when the dataset is balanced and feature rich.

While deep learning could model richer semantics, its benefits would likely appear only on larger, domain-diverse datasets.

5.6. Limitations

Several limitations should be acknowledged. The dataset is domain-specific (Chicago hotel reviews), which restricts generalization to other domains. Deceptive samples may not reflect genuine malicious spam, since they were artificially written. Reviews are short and uniform, limiting feature diversity. Only surface-level lexical features were used; syntactic, semantic, or psychological cues. Ensemble models were tuned via randomized search but not exhaustively optimized due to computation limits.

5.7. Key Takeaways

Naive Bayes achieved the highest overall performance across all metrics. Random Forest was close second, offering strong results with higher computational costs. Logistic Regression remained competitive and interpretable but was not close to Naive Bayes. Gradient Boosting showed inconsistent gains despite higher complexity. In terms of bigrams, they offered small, inconsistent improvements and are optional rather than essential.

6. Conclusion

This project explored the detection of deceptive online reviews using five classical machine learning models on the Deceptive Opinion Spam Corpus v1.4. Through experiments comparing Multinomial Naive Bayes, L1-regularized Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting, the study evaluated both unigram and unigram + bigram feature representations under cross-validation (folds 1–4) and a held-out test fold (fold 5).

The results demonstrate that Naive Bayes achieved the highest overall performance (Test F1 ≈ 0.88), confirming the effectiveness of simple probabilistic approaches for sparse text data. Random Forest followed, while Logistic Regression remained competitive but did not surpass Naive Bayes. Gradient Boosting and Decision Tree fell last, showing that additional model complexity did not yield clear benefits on this dataset.

The inclusion of bigrams produced minor, inconsistent effects, slightly improving cross-validation scores for some models but generally reducing test performance. This indicates that most discriminative information is already captured by single-word features.

Feature analysis showed that deceptive reviews relied on emotional adjectives (e.g., wonderful, definitely, amazing), while truthful reviews focused on factual details such as room, location, and staff.

Overall, the project confirms that classical machine-learning models remain highly effective for deception detection when supported by robust preprocessing and cross-validation. Future work could extend this foundation by incorporating TF-IDF weighting, semantic embeddings, or domain adaptation techniques to improve generalization across different review types and online platforms.

7. API & Use Disclosure

In the process of writing code, no outside assistance was required, other than the one from the source, to understand the dataset as best as possible.

In writing the report, QuillBot was used for grammar checking, as well as integrated Word assistance with spelling and wording. In addition, thesaurus.com was used to choose better synonyms for certain sentence structures. Large language models (such as ChatGPT) were used at the very end of the writing process, to check for redundancy or repetitiveness, as well as accuracy of wording for each section.

8. Appendix

8.1. Appendix A. Confusion Matrices for All Models

This appendix presents the confusion matrices generated for each model configuration (unigram and unigram + bigram) during both cross-validation and final hold-out evaluation.

Each matrix visualizes the classification outcomes for truthful and deceptive reviews, with darker colors representing higher counts.

Model	**Features**	**Figure**
Naïve Bayes	Unigram	A1
Naïve Bayes	Bigram	A2
Log Regression (L1)	Unigram	A3
Log Regression (L1)	Bigram	A4
Decision Tree	Unigram	A5
Decision Tree	Bigram	A6
Random Forest	Unigram	A7
Random Forest	Bigram	A8
Gradient Boosting	Unigram	A9
Gradient Boosting	Bigram	A10

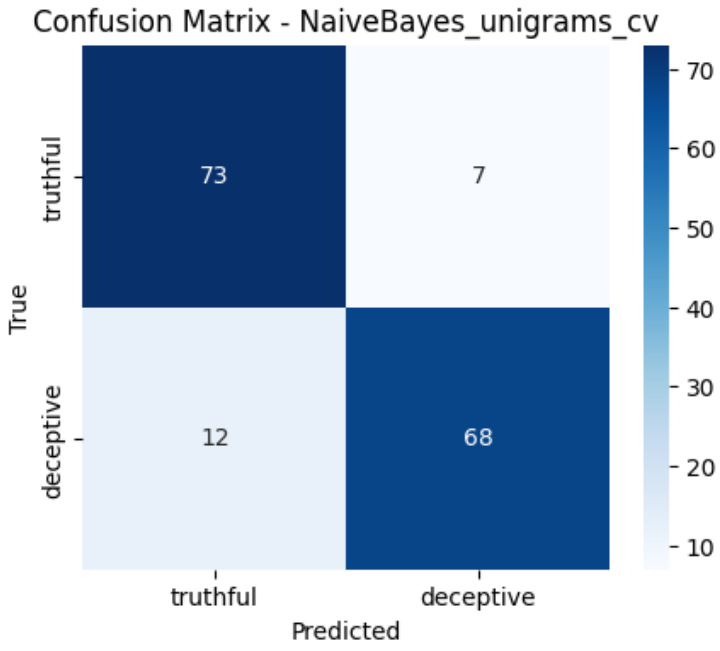

Figure A1

Naïve Bayes Confusion Matrix for Unigram Features

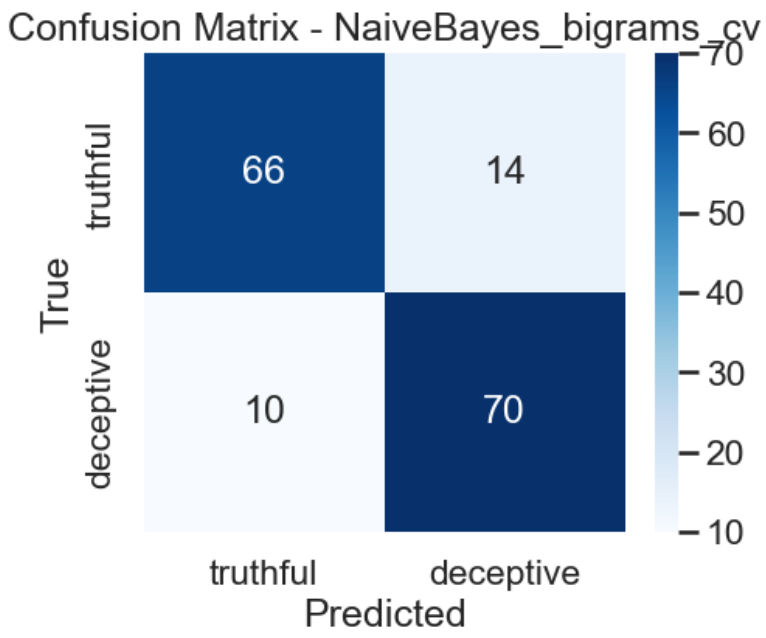

Figure A2

Naïve Bayes Confusion Matrix for Bigram Features

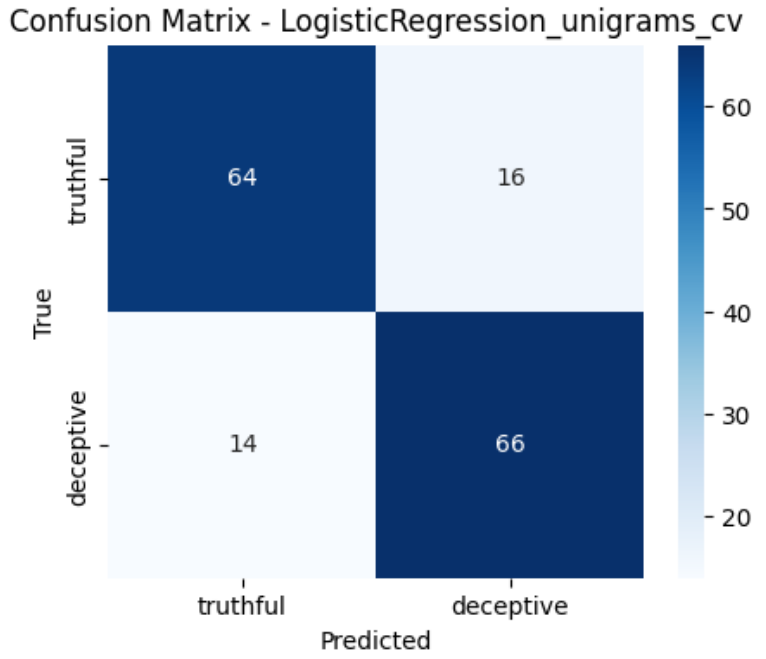

Figure A3

Logistic Regression (L1) Confusion Matrix for Unigram Features

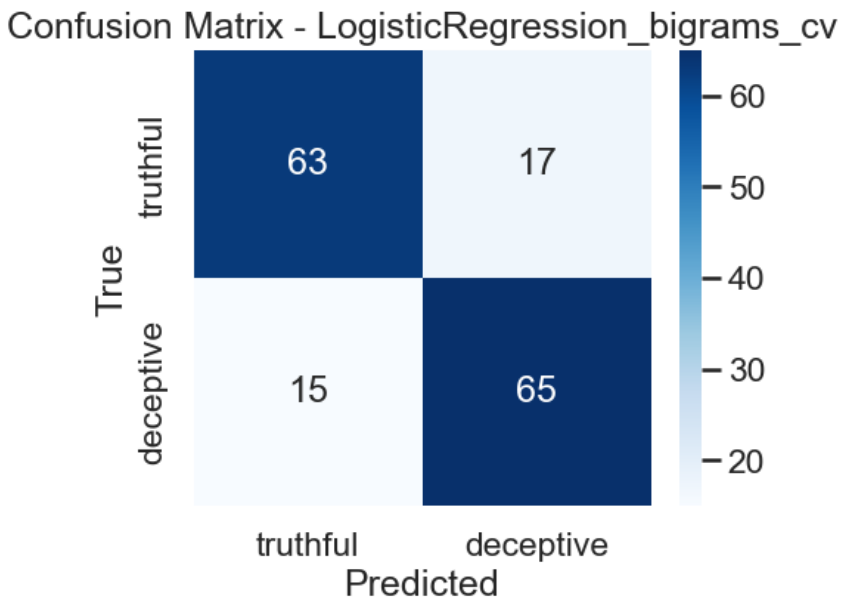

Figure A4

Logistic Regression (L1) Confusion Matrix for Bigram Features

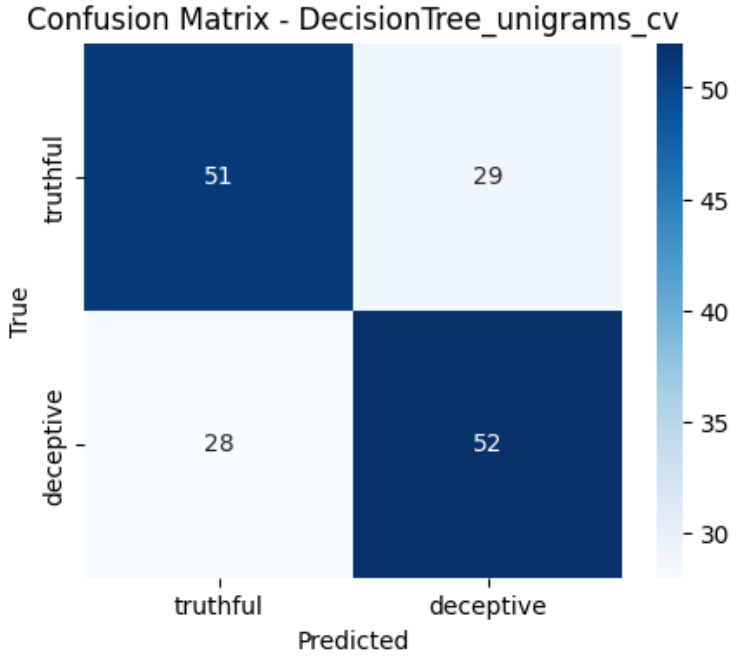

Figure A5

Decision Tree Confusion Matrix for Unigram Features

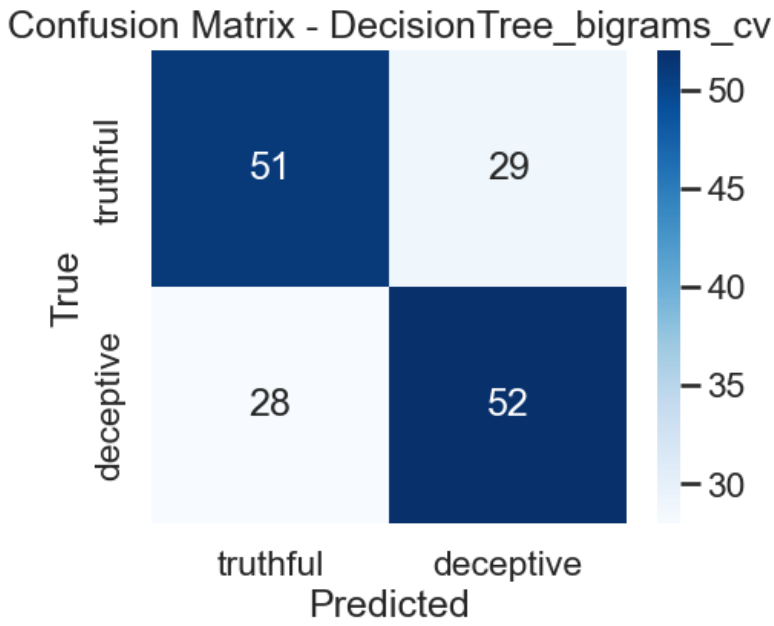

Figure A6

Decision Tree Confusion Matrix for Bigram Features

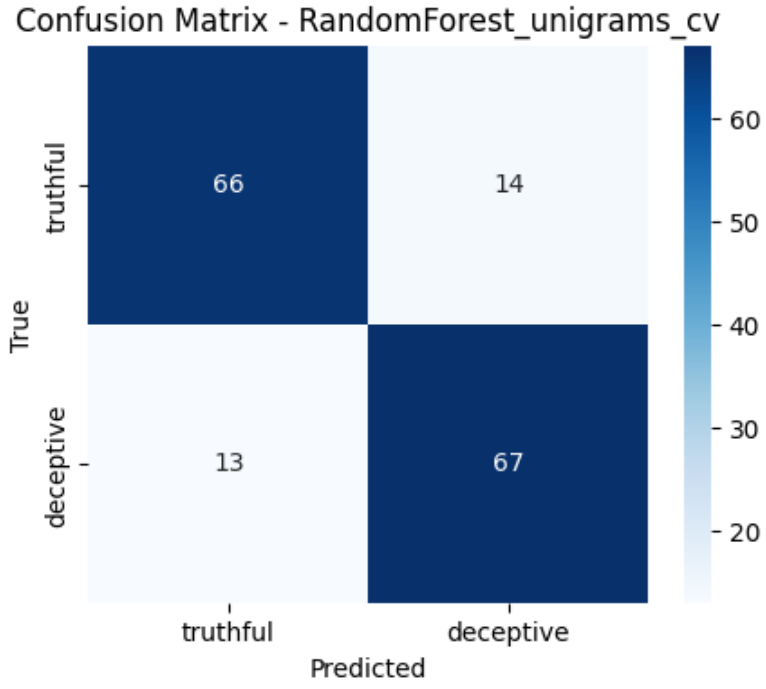

Figure A7

Random Forest Confusion Matrix for Unigram Features

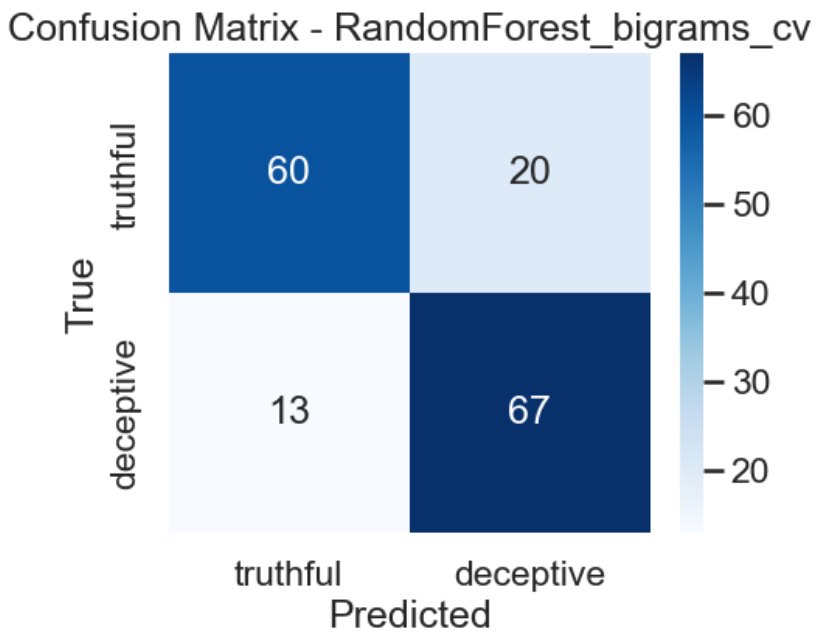

Figure A8

Random Forest Confusion Matrix for Bigram Features

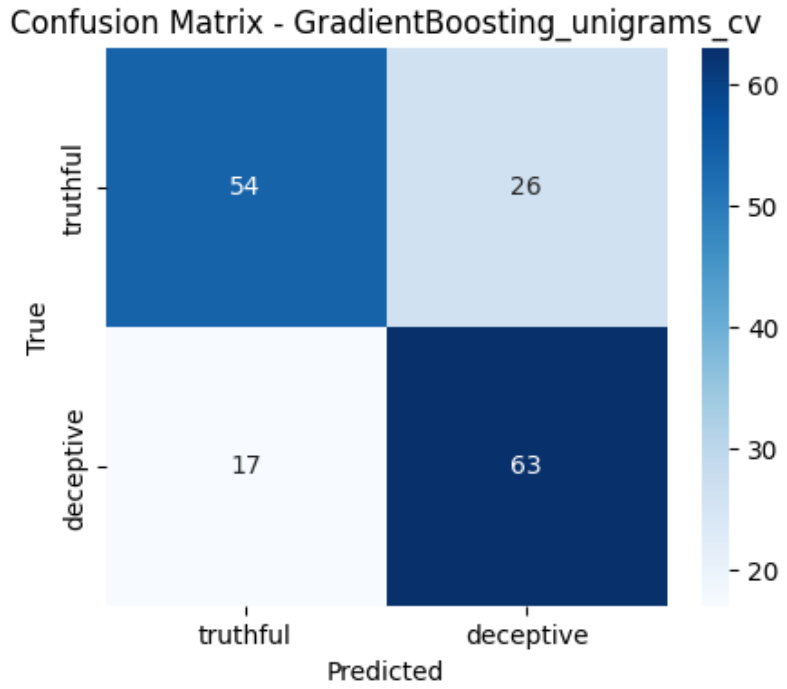

Figure A9

Gradient Boosting Confusion Matrix for Unigram Features

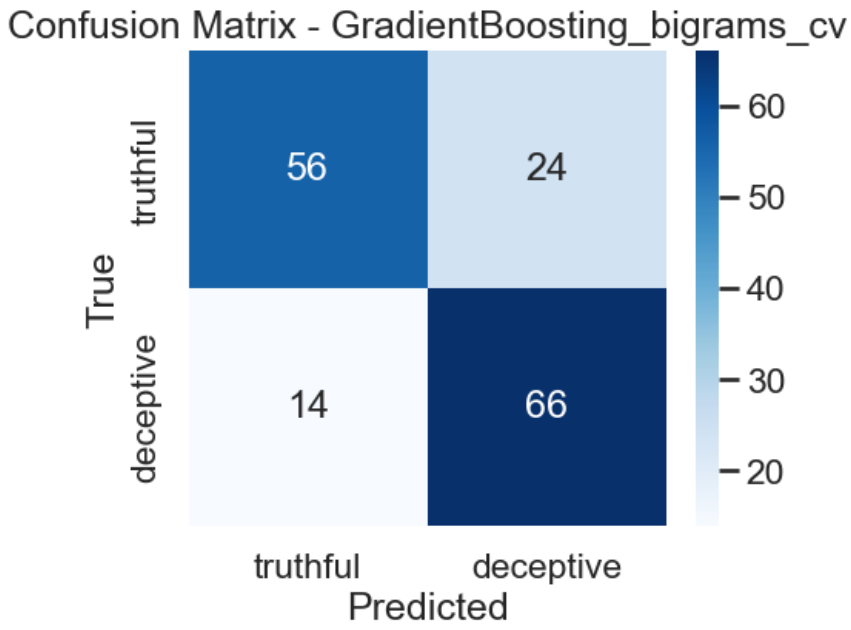

Figure A10

Gradient Boosting Confusion Matrix for Bigram Features

8.2. Appendix B. Aggregate Model Performance Visualizations

This appendix presents aggregated performance distributions across models for both unigram and unigram + bigram feature configurations. Each histogram illustrates the distribution of metric values (Accuracy, Precision, Recall, F1) across models. Red dashes represent the mean, while green dotted lines represent the median.

8.2.1 Appendix B.1 – Unigram Models

Description	**Figure**
CV F1 Mean Across Unigram Models	B1
Test Accuracy Across Unigram Models	B2
Test F1 Macro Across Unigram Models	B3
Test Precision Macro Across Unigram Models	B4
Test Recall Macro Across Unigram Models	B5

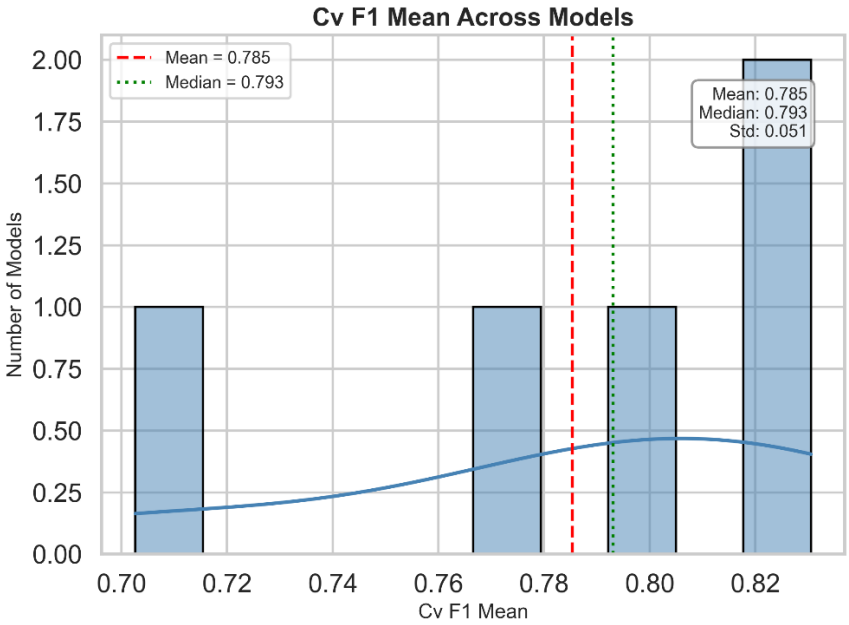

Figure B1

CV F1 Mean Across Unigram Models

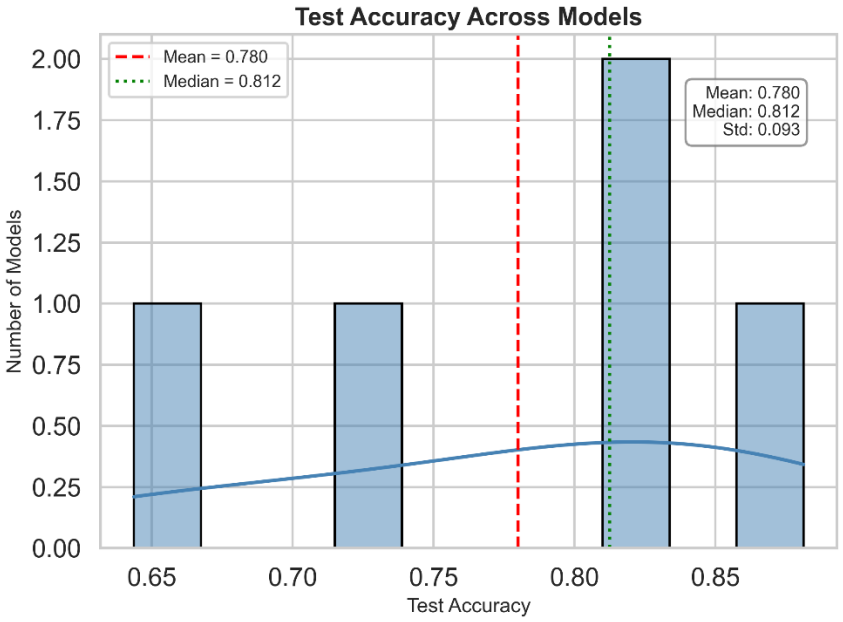

Figure B2

Test Accuracy Across Unigram Models

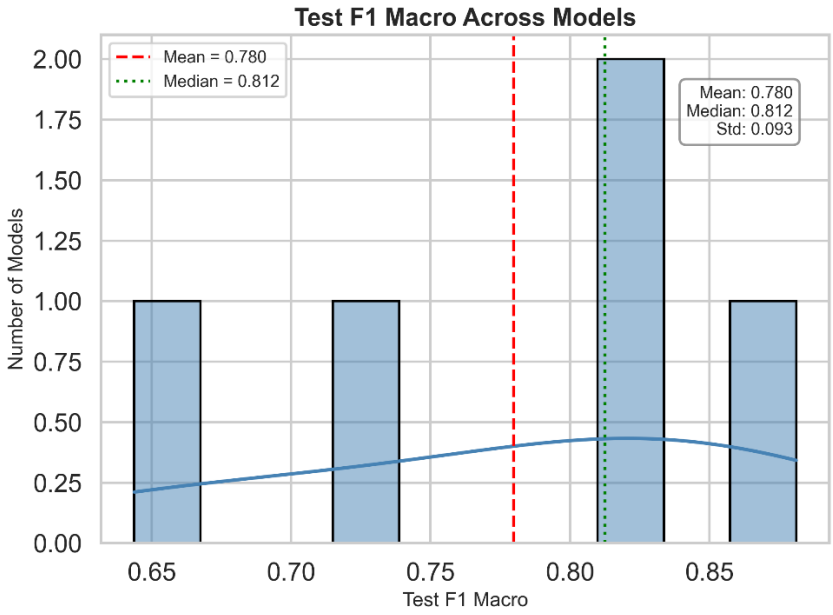

Figure B3

Test F1 Macro Across Unigram Models

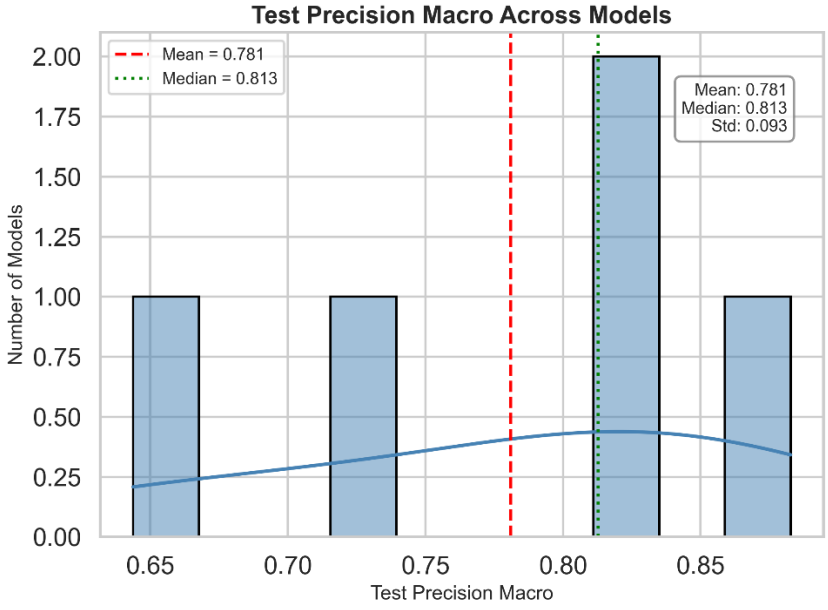

Figure B4

Test Precision Macro Across Unigram Models

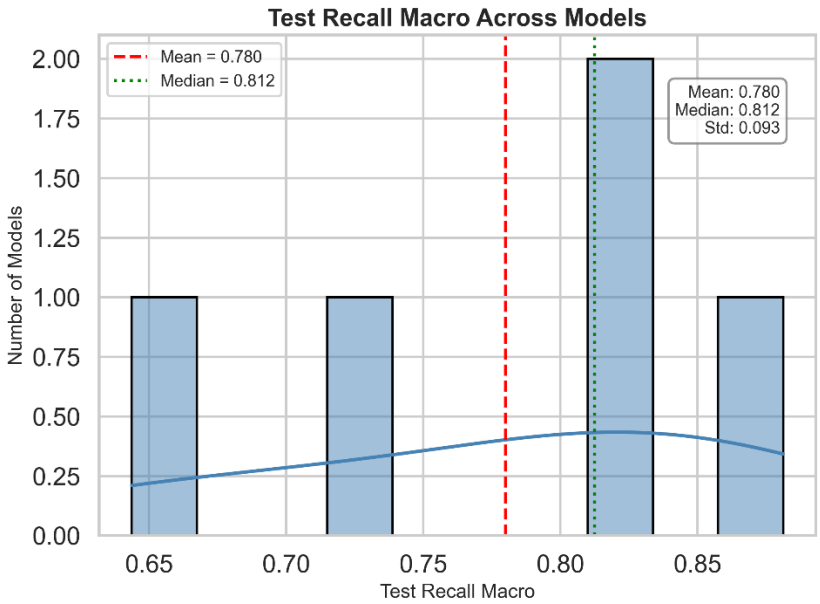

Figure B5

Test Recall Macro Across Unigram Models

8.2.2 Appendix B.2 – Bigram Models

Description	**Figure**
CV F1 Mean Across Bigram Models	B6
Test Accuracy Across Bigram Models	B7
Test F1 Macro Across Bigram Models	B8
Test Precision Macro Across Bigram Models	B9
Test Recall Macro Across Bigram Models	B10

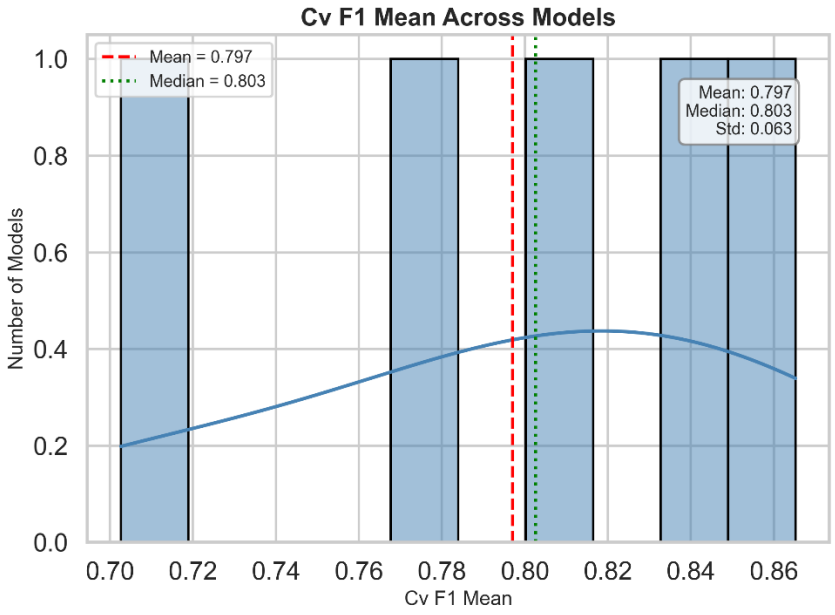

Figure B6

CV F1 Mean Across Bigram Models

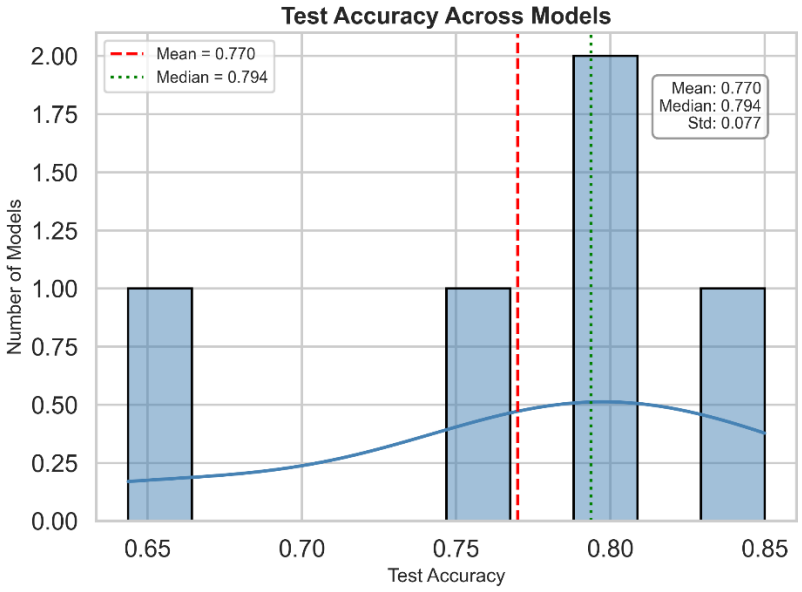

Figure B7

Test Accuracy Across Bigram Models

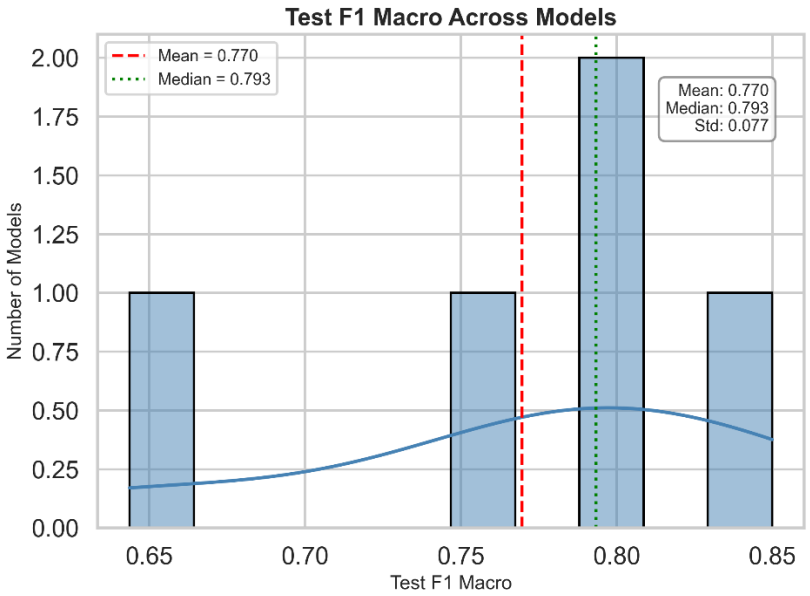

Figure B8

Test F1 Macro Across Bigram Models

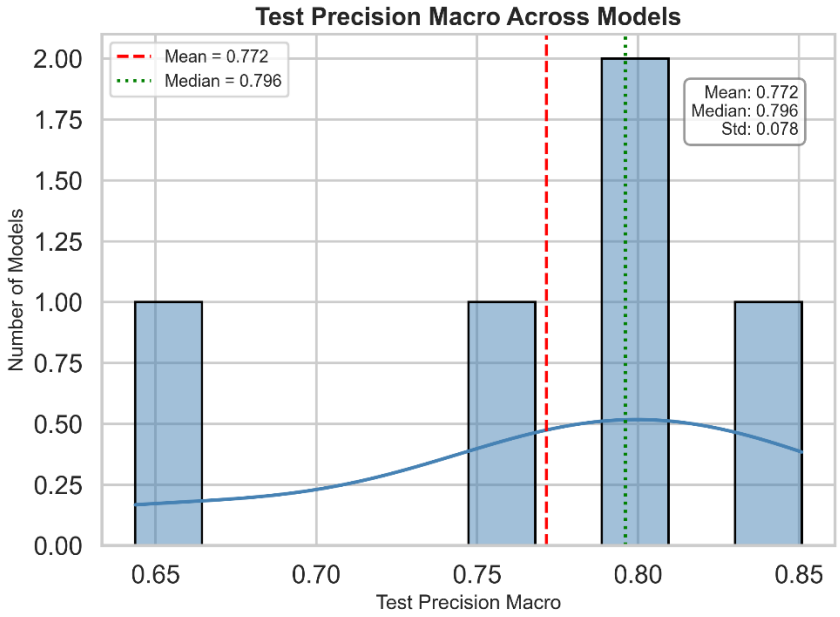

Figure B9

Test Precision Macro Across Bigram Models

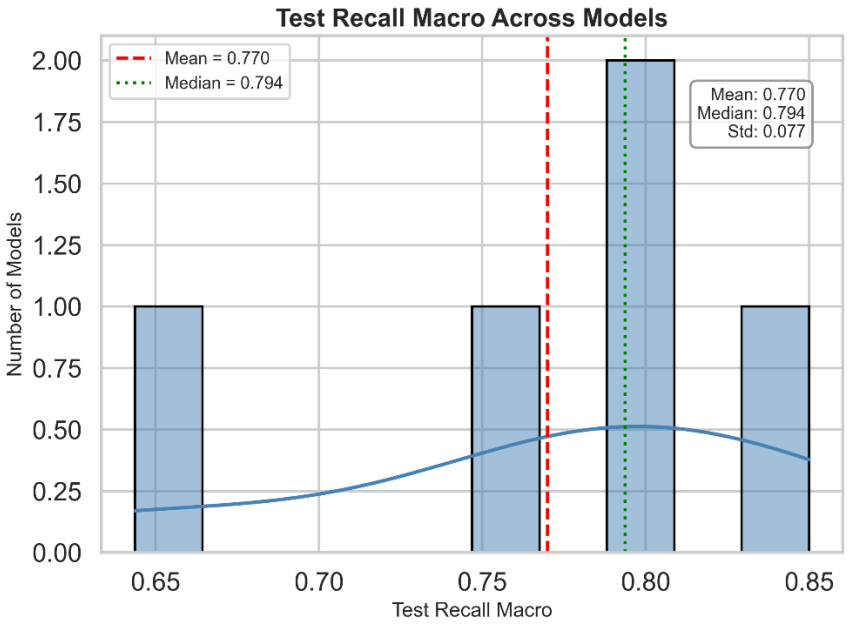

Figure B10

Test Recall Macro Across Bigram Models