Assignment Data Mining 2025: Classification for the Detection of Opinion Spam

Alexia Ntantouri **Lenny Pam** **Yating Qu**
Student Numbers: 2958481, 7128584, 2145189
Utrecht University
{a.ntantouri, l.g.pam, y.qu1}@students.uu.nl

1 Introduction

Online reviews play a central role in influencing consumer decisions. However, the rise of **opinion spam**, meaning fake reviews designed to mislead potential customers, poses a serious challenge to trust and transparency. So, detecting deceptive reviews automatically is an important problem in text mining and machine learning.

This project investigates the task of classifying hotel reviews as truthful or deceptive using a variety of supervised machine learning algorithms.

The main research questions are:

1. How does the performance of a generative linear model (Multinomial Naive Bayes) compare to that of a discriminative linear model (Lasso-regularized Logistic Regression)?

2. Do non-linear ensemble models (Random Forests and Gradient Boosting) outperform the linear models?

3. Does performance improve when using bigram features instead of unigrams?

4. What are the most important linguistic features indicative of deceptive and truthful reviews?

To investigate these research questions, we use a curated dataset of deceptive and truthful hotel reviews, which we describe in the next section.

2 Dataset

The dataset we use consists of 800 negative hotel reviews of 20 popular Chicago (20 reviews per hotel) hotels collected by Ott et al. (2011, 2013), split evenly between 400 truthful negative reviews (mined from six platforms: Expedia, Orbitz, Priceline, TripAdvisor, Hotels.com,) and 400 deceptive reviews (crowdsourced fake reviews via Mechanical Turk). The (400 HITS) deceptive negative reviews were written within 30 minutes by U.S workers under the instructions to post a fake negative review of a competitors hotel . The workers were paid 1 dollar per review. These reviews were sampled to match the length of the distribution of the deceptive set.

Each review includes only text, the labels are binary (truthful vs. deceptive) and the average length of the deceptive negatives are 178 words. On average the truthful reviews are longer than the deceptive reviews.

The data were preprocessed and pre-divided into five folds. We used folds 1–4 (640 reviews) for training and hyperparameter tuning, and fold 5 (160 reviews) for the final evaluation, ensuring an unbiased test. In the next section, we describe our methodology for training and evaluating the classification models in detail.

3 Methodology

We experimented with five model families: Multinomial Naive Bayes, Logistic Regression, Classification Trees, Random Forests, and Gradient Boosting.

3.1 Overall Workflow

The experimental workflow follows a modular structure, designed for reproducibility and comparability across multiple models and feature extraction methods. The general process consists of four main stages:

1. **Data Loading.** The raw text data from the `op_spam_v1.4` corpus is loaded using `load_data.py`. Individual `.txt` files are merged into `train.pkl` and `test.pkl` DataFrames.

2. **Preprocessing.** The script `preprocessing.py` applies standard text preprocessing operations: lowercasing, punctuation removal, tokenization, stopword removal, and lemmatization. The processed datasets are saved as `train_preprocessed.pkl` and `test_preprocessed.pkl`.

3. **Feature Extraction and Model Training.** Each classifier is implemented in two variants: one using TF–IDF features and one using Bag–of–Words (BoW). However, in this report we focus on the TF–IDF representations, as they generally provide more informative and discriminative features by weighting terms according to their relative importance across documents.

   For both representations, experiments are conducted using unigrams and bigrams. Each model is trained via 5-fold cross-validation on the training data (640 reviews), with hyperparameter tuning performed using grid search. The best model configuration is then retrained on the entire training set and evaluated on the held-out test set (160 reviews).

4. **Evaluation and Statistical Testing.** For each model, test set predictions, confusion matrices, and detailed classification reports are saved under the `results/` and the `results_extra/` directory. Finally, the script `mcnemar_test.py` performs pairwise McNemar significance tests across models and feature sets.

3.2 Model-Specific Details

For all models, a `TfidfVectorizer` was included in the pipeline, allowing us to also explore the most effective vocabulary size (`max_features`) for each model.

Grid search was performed separately for each model twice: once using unigrams ($n\_gram = (1, 1)$) and once using unigrams+bigrams ($n\_gram = (1, 2)$), to evaluate the impact of including bigrams on model performance.

3.2.1 Multinomial Naive Bayes

The following hyperparameters were tuned in the pipeline using grid search:

- `tfidf__max_features`: $\{3000, 5000\}$
- `nb__alpha`: $\{0.075, 0.1\}$ (additive smoothing parameter)

3.2.2 Logistic Regression

The logistic regression classifier was trained with L1 regularization using `LogisticRegressionCV` ($random\_state = 42$) and the `liblinear` solver.

The regularization strength C was optimized automatically via 5–fold cross-validation over:

$$C \in \{0.001, 0.01, 0.1, 1, 10, 100\}.$$

Feature extraction was performed with `TfidfVectorizer(max_features = 5000)`. The top five positive and negative coefficients were extracted to interpret the most influential features for deceptive versus truthful reviews.

The logistic regression model estimates the probability that a review is deceptive as:

$$P(y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \sum_{j=1}^{p} \beta_j x_j)}},$$

where $\mathbf{x}$ is the TF–IDF feature vector, β_0 is the intercept, and β_j are the feature weights. The model minimizes the regularized negative log-likelihood:

$$\min_{\boldsymbol{\beta}} \left(-\frac{1}{n} \sum_{i=1}^{n} [y_i \log P_i + (1 - y_i) \log(1 - P_i)] + \lambda \sum_{j=1}^{p} |\beta_j| \right),$$

where $\lambda = 1/C$ controls the strength of the L1 penalty. The L1 regularization encourages sparsity, automatically selecting the most informative features.

After training, positive coefficients ($\beta_j > 0$) indicate words typical of truthful reviews, while negative coefficients ($\beta_j < 0$) indicate words typical of deceptive reviews. The magnitude $|\beta_j|$ reflects feature importance. The five most positive and five most negative coefficients were selected as the top terms for truthful and deceptive reviews, respectively. These terms are analyzed and discussed in Section 5.4.

3.2.3 Classification Tree

Decision trees were trained using `DecisionTreeClassifier` ($random\_state = 42$) within a grid search pipeline:

- `vec__max_features`: $\{3000, 5000\}$
- `dt__max_depth`: $\{\text{None}, 10, 20, 30\}$
- `dt__min_samples_split`: $\{2, 5, 10\}$
- `dt__min_samples_leaf`: $\{1, 2, 5\}$
- `dt__ccp_alpha`: $\{0.0, 10^{-4}, 10^{-3}, 10^{-2}\}$ (cost-complexity pruning parameter)

3.2.4 Random Forest

Random forests were trained with `RandomForestClassifier` ($random\_state = 42$), tuning the following parameters in the pipeline:

- `vec__max_features`: $\{3000, 5000\}$
- `rf__n_estimators`: $\{100, 200\}$
- `rf__max_depth`: $\{\text{None}, 10, 20\}$
- `rf__min_samples_split`: $\{2, 5\}$
- `rf__min_samples_leaf`: $\{1, 2\}$

3.2.5 Gradient Boosting

The gradient boosting model was implemented with `GradientBoostingClassifier` ($random\_state = 42$). The following hyperparameters were explored in the pipeline:

- `tfidf__max_features`: $\{3000, 5000\}$
- `gb__n_estimators`: $\{100, 200\}$
- `gb__learning_rate`: $\{0.01, 0.05, 0.1\}$
- `gb__max_depth`: $\{3, 4, 5\}$

3.3 Evaluation Metrics and Statistical Testing

For each model, the following metrics were reported on the held-out test set: accuracy, precision, recall, and F1-score.

To assess statistical significance between models and feature sets, pairwise McNemar tests were conducted using `statsmodels`. Since all classifiers were evaluated on the same test set, predictions are paired for each review. The McNemar test checks whether differences in correct versus incorrect predictions between two models are statistically significant, providing a non-parametric measure of performance difference.

Specifically, McNemar tests were performed across:

- Naive Bayes vs. Logistic Regression,
- Linear vs. Tree-based models,
- Unigram vs. Bigram representations.

Results were summarized in `mcnemar_summary.txt`. In the next section, we present the results in detail.

4 Results

Table 1 shows the results in terms of test accuracy for each model and feature representation (unigrams or bigrams) using a TF-IDF vectorizer. We can see that for the unigram representation, the top-performing models are the Naive Bayes and the Random Forest with 82.50% test accuracy. For the bigram representation, Naive Bayes has the highest test accuracy of 85%.

Model	Unigrams	Bigrams
Naive Bayes	**0.8250**	**0.8500**
Logistic Regression	0.8000	0.8250
Classification Tree	0.6750	0.6500
Gradient Boosting	0.7438	0.7937
Random Forest	**0.8250**	0.8250

Table 1: Test set accuracy for each model and feature representation (unigrams or bigrams) using a TF-IDF vectorizer.

4.1 Multinomial Naive Bayes

Table 2 shows the results of the Multinomial Naive Bayes model using a TF-IDF vectorizer. For both the unigram and bigram representations, the grid search selected `nb__alpha` $= 0.1$ and `tfidf__max_features` $= 3000$ as the best hyperparameters for the pipeline.

4.2 Logistic Regression

Table 3 shows the results of the Logistic Regression model using a TF-IDF vectorizer. For the unigram representation, the grid search selected a regularization strength of $C = 100.0$, while for the bigram representation, the optimal value was $C = 10.0$. Feature extraction was performed with a maximum of 5000 TF-IDF features in both cases.

4.3 Classification Tree

Table 4 shows the results of the Classification Tree model using a TF-IDF vectorizer. For both unigram and bigram representations, the grid search

Features	Accuracy (CV)	Accuracy (Test)	Deceptive			Truthful		
			P	R	F1	P	R	F1
Unigrams	82.81%	82.50%	0.84	0.80	0.82	0.81	0.85	0.83
Bigrams	83.75%	85.00%	0.85	0.85	0.85	0.85	0.85	0.85

Table 2: Multinomial Naive Bayes performance on negative hotel reviews (800 reviews total). Results are based on 5-fold cross-validation (640 training reviews) and evaluation on a held-out test set (160 reviews). Precision (P), Recall (R), and F1-score are reported on the test set. A TF-IDF vectorizer was used in the pipeline.

Features	Mean CV Accuracy	Accuracy (Test)	Deceptive			Truthful		
			P	R	F1	P	R	F1
Unigrams	80.47%	80.00%	0.83	0.75	0.79	0.77	0.85	0.81
Bigrams	82.34%	82.50%	0.85	0.79	0.82	0.80	0.86	0.83

Table 3: Logistic Regression performance on negative hotel reviews (800 reviews total). Results are based on 5-fold cross-validation (640 training reviews) and evaluation on a held-out test set (160 reviews). Precision (P), Recall (R), and F1-score are reported on the test set. The mean CV accuracy corresponds to the average 5-fold accuracy during training with TF-IDF features.

selected the following hyperparameters: cost-complexity pruning `dt__ccp_alpha` = 0.01, no restriction on tree depth (`dt__max_depth` = None), minimum samples per leaf `dt__min_samples_leaf` = 5, minimum samples for a split `dt__min_samples_split` = 2, and a maximum vocabulary size `vec__max_features` = 5000.

4.4 Random Forest

Table 5 shows the results of the Random Forest model using a TF-IDF vectorizer. For both unigram and bigram representations, the selected hyperparameters were: maximum tree depth `rf__max_depth` = 20, minimum samples per leaf `rf__min_samples_leaf` = 2, minimum samples for a split `rf__min_samples_split` = 2, and maximum vocabulary size `vec__max_features` = 3000. The main difference between the two settings is the number of trees (`rf__n_estimators` = 200 for unigrams, 100 for bigrams)

4.5 Gradient Boosting

Table 6 shows the results of the Gradient boosting model using a TF-IDF vectorizer. For both unigram and bigram representations, the number of estimators was set to `gb__n_estimators` = 200, the learning rate was `gb__learning_rate` = 0.1, and the maximum vocabulary size was `tfidf__max_features` = 5000. The differences between the two n-gram settings are in the maximum tree depth (`gb__max_depth` = 5 for unigrams, 3 for bigrams).

5 Research Questions

5.1 How does the performance of the generative linear model (multinomial naive Bayes) compare to the discriminative linear model (regularized logistic regression)?

The results for unigrams. The test accuracy for the Naive Bayes (0.825) and the Logistic Regression (0.800) were quit close. The difference of 0.025 in test accuracy is hard to explain. An explanation for the small difference could be that the dataset is relatively small. A Naive Bayes classifier could perform better because it doesn't focus on learning complex relations between words it just uses likelihood. Logistic regression on the other hand learns a weight for every word. This could lead to slight overfitting. But again, the difference is small. Other factors could be at play.

The results for bigrams. The test accuracy for the Logistic Regression classifier with bigrams is 0.825, while the accuracy of the Multinomial Naïve Bayes classifier increases to 0.85 (an improvement of 0.025 over unigrams). The Multinominal Bayes classifier outperforms the logistic regression classifier by 0.025. One explanation for this improvement is that in Naive Bayes, Laplace (additive) smoothing allows every bigram to contribute some probability mass, even if it never appeared in the training data. This prevents zero probabilities and lets the model still make use of rare or unseen bigrams that occur in the test data. Logistic re-

Features	Accuracy (CV)	Accuracy (Test)	Deceptive			Truthful		
			P	R	F1	P	R	F1
Unigrams	68.13%	67.50%	0.68	0.65	0.67	0.67	0.70	0.68
Bigrams	68.28%	65.00%	0.67	0.59	0.63	0.63	0.71	0.67

Table 4: Classification Tree performance on negative hotel reviews (800 reviews total). Results are based on 5-fold cross-validation (640 training reviews) and evaluation on a held-out test set (160 reviews). Precision (P), Recall (R), and F1-score are reported on the test set.

Features	Accuracy (CV)	Accuracy (Test)	Deceptive			Truthful		
			P	R	F1	P	R	F1
Unigrams	84.69%	82.50%	0.86	0.78	0.82	0.80	0.88	0.83
Bigrams	84.38%	82.50%	0.88	0.75	0.81	0.78	0.90	0.84

Table 5: Random Forest performance on negative hotel reviews (800 reviews total). Results are based on 5-fold cross-validation (640 training reviews) and evaluation on a held-out test set (160 reviews). Precision (P), Recall (R), and F1-score are reported on the test set. A TF-IDF vectorizer was used in the pipeline.

Features	Accuracy (CV)	Accuracy (Test)	Deceptive			Truthful		
			P	R	F1	P	R	F1
Unigrams	78.28%	74.38%	0.77	0.70	0.73	0.72	0.79	0.75
Bigrams	78.59%	79.37%	0.82	0.75	0.78	0.77	0.84	0.80

Table 6: Gradient Boosting performance on negative hotel reviews (800 reviews total). Results are based on 5-fold cross-validation (640 training reviews) and evaluation on a held-out test set (160 reviews). Precision (P), Recall (R), and F1-score are reported on the test set. A TF-IDF vectorizer was used in the pipeline.

gression does not use smoothing; therefore, rare bigrams that appear only a few times provide too little statistical evidence for the model to assign them meaningful weights. But again, the difference (0.025) is small. Other factors could be at play.

Statistical significance (McNemar's test). To evaluate whether the observed differences between Multinomial Naive Bayes and Logistic Regression were statistically significant, McNemar's tests were performed for both unigram and bigram features. For unigrams, the test yielded a p-value of 0.61772, and for bigrams, $p = 0.54126$ (both $p > 0.05$). These results indicate that the differences in accuracy between the two models are not statistically significant, meaning that the observed improvements of Naïve Bayes over Logistic Regression are likely due to random variation rather than a true underlying performance gap.

Table 7 shows the p-values from McNemar's tests comparing Multinomial Naïve Bayes and regularized Logistic Regression for both unigram and bigram features.

Overall, both the generative (Multinomial Naïve Bayes) and discriminative (Logistic Regression) linear models demonstrate comparable performance. Naïve Bayes shows slightly higher accuracy for both unigram and bigram representations, but these differences are not statistically significant.

Feature Representation	p-value
Unigrams	0.6177
Bigrams	0.5413

Table 7: McNemar's test p-values: NB vs LR.

5.2 Are random forests and gradient boosting able to improve on the performance of the linear classifiers?

Based on the experimental results, the random forest and gradient boosting models did not outperform the linear classifiers (Naive Bayes and Logistic Regression). Notably, the top-performing model was Multinomial Naive Bayes with bigram features, achieving 85.00% test accuracy.

In terms of test accuracy, the results show that for unigram features, Naive Bayes and Random Forest achieved the best performance (82.50%), followed by Logistic Regression at 80.0%, while Gradient Boosting lagged behind at 74.38%. Similar patterns are observed across precision, recall, and F1-scores, where Gradient Boosting stays slightly

behind.

For bigram features, Naive Bayes remained the best-performing model with 85.0% accuracy, followed by Random Forest (82.50%) and Logistic Regression (82.50%), while Gradient Boosting reached 79.37%. Similar trends are observed across precision, recall, and F1-scores.

To assess whether these differences were statistically significant, McNemar's tests were conducted for all model pairs using both unigram and bigram feature representations. None of the comparisons between Random Forest or Gradient Boosting and the linear classifiers yielded significant p-values (all $p > 0.05$). This indicates that any observed performance differences are likely due to chance rather than true model superiority.

Table 8 reports the p-values from McNemar's tests comparing linear classifiers (MNB, LR) against tree-based models (RF, GB) using both unigram and bigram features.

Linear	Tree-based	Unigrams p	Bigrams p
MNB	RF	1.0000	0.5716
MNB	GB	0.0919	0.1996
LR	RF	0.5847	1.0000
LR	GB	0.1996	0.4869

Table 8: McNemar's test p-values for linear vs tree-based classifiers.

Overall, ensemble tree-based methods did not provide a statistically significant improvement over linear classifiers on this deceptive review detection task. This outcome is consistent with expectations for text classification problems, where the high-dimensional, sparse nature of bag-of-words representations (e.g., TF–IDF weighted features) tends to favor linear models such as Naive Bayes and Logistic Regression. Tree-based models are better suited for lower-dimensional, continuous feature spaces and may not effectively capture the subtle lexical patterns that distinguish deceptive from truthful reviews in this dataset.

5.3 Does performance improve by adding bigram features, instead of using just unigrams?

In terms of performance, adding bigram features generally had a positive effect across models. For Multinomial Naive Bayes, test accuracy increased from 82.50% with unigrams to 85.00% with bigrams, accompanied by improvements in precision, recall, and F1-scores for both deceptive and truthful reviews. This indicates that bigrams capture additional local context that unigrams alone miss.

For Logistic Regression, test accuracy increased slightly from 80.00% with unigrams to 82.50% with bigrams, indicating that incorporating bigram features provided a modest gain in discriminative power for this linear model.

In contrast, the Classification Tree experienced a slight drop in performance when using bigrams, from 67.50% with unigrams to 65.00% with bigrams (despite a minor increase in cross-validation accuracy (68.13% with unigrams to 68.28% with bigrams). This suggests that the model may have overfit to the higher-dimensional, sparse bigram feature space, capturing patterns in the training folds that do not generalize to the test set.

The Random Forest classifier achieved identical test accuracy with unigrams and bigrams (82.50%), suggesting that the ensemble's averaging effect mitigates any potential benefit from the additional bigram features.

Gradient Boosting improved from 74.38% with unigrams to 79.37% with bigrams, suggesting that this model could leverage the richer context of bigrams, although it still did not surpass the linear classifiers.

McNemar's tests comparing unigram and bigram representations for each model found no statistically significant differences (all p-values $p > 0.05$), indicating that the observed numerical improvements or drops are likely due to chance. Table 9 shows the p-values from McNemar's tests comparing unigram and bigram representations for each model.

Model	**p-value**
Multinomial NB	0.4545
Logistic Regression	0.3877
Decision Tree	0.6271
Random Forest	1.0000
Gradient Boosting	0.1153

Table 9: McNemar test results comparing unigrams and bigrams for each model. No comparisons were statistically significant.

Overall, bigram features improved test accuracy for Multinomial Naive Bayes, Logistic Regression, and Gradient Boosting, but slightly reduced performance for the Classification Tree and had no effect on the Random Forest. Notably, the top-performing

model was Multinomial Naive Bayes with bigrams, achieving 85.0% test accuracy.

5.4 What are the five most important terms (features) pointing towards a fake review, and what are the five most important terms pointing towards a genuine review?

The Logistic Regression model with TF–IDF features was used to identify the most informative terms for classifying reviews as deceptive or truthful.

For the unigram representation, the top features associated with *fake* reviews were *chicago*, *turned*, *smelled*, *recently*, and *relax*. The top features for *genuine* reviews were *star*, *adult*, *world*, *clothing*, and *location*.

When using bigram features, the most indicative terms for fake reviews changed slightly to *chicago*, *relax*, *homewood suite*, *turned*, and *hotel chicago*. For genuine reviews, the top bigram features were *star*, *booked hotel*, *adult*, *elevator*, and *location*.

Unigram features focus on single words, which highlight generic or emotional terms (e.g., *chicago*, *relax*) typical of fake reviews, and factual or descriptive words (e.g., *star*, *location*) typical of genuine ones.

Bigrams add short contextual information; phrases such as *hotel chicago* or *homewood suite* suggest attempts to sound credible by repeating hotel names, whereas *booked hotel* and *elevator* describe specific experiences. Although bigrams capture limited context, they did not lead to a statistically significant improvement over unigrams (Table 9).

Overall, deceptive reviews tend to use more generic and promotional language, while truthful reviews contain more concrete, experience-based details.

Term	Coefficient
chicago	−52.4534
turned	−27.6678
smelled	−27.6425
recently	−26.5458
relax	−25.0329

Table 10: Top five unigram features most indicative of *fake* reviews.

Term	Coefficient
star	32.3197
adult	26.5508
world	25.7308
clothing	25.1779
location	21.4243

Table 11: Top five unigram features most indicative of *genuine* reviews.

Term	Coefficient
chicago	−26.4863
relax	−17.4281
homewood suite	−15.3310
turned	−15.2472
hotel chicago	−15.2141

Table 12: Top five bigram features most indicative of *fake* reviews.

Term	Coefficient
star	18.0724
booked hotel	17.2052
adult	15.3691
elevator	13.9929
location	12.3243

Table 13: Top five bigram features most indicative of *genuine* reviews.

6 Conclusion

In conclusion, the Multinomial Naive Bayes model with bigram features gave the best results, which reached an accuracy of 85.0%. Although this score was slightly higher than the other models, statistical tests showed that the difference was not significant. Logistic Regression showed similar accuracy but was faster and easier to train. This makes it an attractive option for larger datasets or real-time applications.

Tree-based models such as Random Forest and Gradient Boosting did not outperform the linear models. This result suggests that for sparse and high-dimensional text data, linear models are more suitable and efficient. Also, Gradient Boosting showed some potential improvement when using bigram features, which could be explored further in future work.

It was also observed that bigram features slightly improved performance for some models. The improvement was small, and the added complexity may not justify the extra computation in similar

small-scale text datasets.

In future work, it would be useful to test more features, such as punctuation or sentiment. Deep learning models could also be tried to capture more context. Overall, simple models like Naive Bayes are still strong and reliable for detecting fake reviews.

A Appendix

A.1 Code Repository

All code used in this study is available in the project repository: GitHub Repository. This includes scripts for training all models, both with TF-IDF and Bag-of-Words (BoW) feature representations, as well as saved predictions, confusion matrices, and detailed classification reports for each model configuration. The repository provides full reproducibility of the experiments presented in this report.

A.2 Use of AI

We used ChatGPT (GPT-5) as a learning aid for this assignment.

Specifically, it was used to:

- Verify correct implementation of cross-validation and hyperparameter tuning.
- Suggest improvements in code reproducibility and model comparison (e.g., adding McNemar tests).
- Help polish and structure the final written report.
- Help understand statistical concepts and classifiers .

All code and analyses were written, run, and interpreted by the group. ChatGPT was not used to generate the final code or results automatically.

References

Myle Ott, Claire Cardie, and Jeffrey T. Hancock. 2013. Negative deceptive opinion spam. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 497–501, Atlanta, Georgia. Association for Computational Linguistics.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 309–319, Portland, Oregon, USA. Association for Computational Linguistics.