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

Group 23

Evita – Evdoxia Dologlou

9522786, e.dologlou@students.uu.nl

Zoi Dimitriou

8666334, z.dimitriou@students.uu.nl

Melissa Rueca

8463603, m.rueca@students.uu.nl

1 ABSTRACT

Online reviews play a crucial role in consumer decision-making, but the increasing amount of deceptive content presents significant challenges to reliability and trust. This study explores machine learning methods for detecting deceptive hotel reviews using text-based features. Five classification algorithms: Multinomial Naive Bayes, Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting, were trained and evaluated with unigram and bigram representations. Performance metrics and McNemar's statistical test were used to assess accuracy and significance across models. Results indicate that incorporating bigram features significantly improves performance, with the Multinomial Naive Bayes model achieving the highest accuracy at 88.1%. Overall, the study shows that simple probabilistic models, enhanced through thoughtful feature design, can effectively and transparently detect deceptive reviews.

2 INTRODUCTION

Nowadays, consumers rely heavily on online reviews before making a purchase or booking a service. At the same time, review platforms depend on the honesty of user-generated content to maintain trust and credibility. However, there is always the risk that some of these reviews are fake, written with the intention of misleading potential customers. In this study, we focus on deceptive opinion spam, which is fabricated reviews deliberately written to appear genuine but designed to deceive readers [2]. Such reviews can either promote a company or service through falsely positive feedback or damage its reputation through intentionally negative comments. Detecting these deceptive reviews is crucial, as they can mislead customers and influence their decisions. Yet, this task is particularly challenging for humans, whose ability to spot deception is often not much better than random guessing [1]. Therefore, data-driven and machine learning approaches are needed to

tackle this problem. The goal of this study is to develop and compare several machine learning models for classifying negative hotel reviews as either truthful or deceptive. Using the dataset introduced by Ott et al. [2], which contains both truthful and deceptive negative reviews, we evaluate the performance of different classifiers, including a Multinomial Naive Bayes model, regularized Logistic Regression, a Decision Tree, Random Forest, and Gradient Boosting.

3 DATA

The dataset used in this study consists of 800 negative hotel reviews from Chicago, evenly split between deceptive and truthful samples, and was introduced by Ott et al [2]. The 400 deceptive reviews were collected using Amazon Mechanical Turk (MTurk), a crowdsourcing platform. In this process, each participant was asked to imagine working for a hotel's marketing department and to write a false negative review about a competitor. These reviews covered 20 popular Chicago hotels, with 20 deceptive reviews per hotel, and were manually screened to ensure appropriate length, clarity, and relevance. Truthful reviews were gathered from six major online review platforms, Expedia, Hotels.com, Orbitz, Priceline, TripAdvisor, and Yelp, where users rate and comment on hotels they have visited. To ensure negative sentiment, only 1- or 2-star reviews were used. Once again, 20 truthful reviews were selected for each hotel, 400 in total, matching the number of fake ones.

4 ANALYSIS

4.1 Preprocessing of Data

Before model training, the data underwent several preprocessing steps. Initially, all reviews were converted to lowercase to eliminate case sensitivity issues. HTML tags, punctuation, and numeric characters were removed, as they provide limited meaningful information for distinguishing between truthful and deceptive reviews. [3] The cleaned text was then tokenized, a fundamental step that splits the text into individual words or tokens. Subsequently, common English stopwords, including articles, prepositions, and conjunctions (e.g., "the", "and", "is"), were removed due to their minimal semantic contribution. Negation words such as "not", "no", and "n't" were retained to preserve sentiment polarity and meaning, as these terms frequently appear in negative reviews and play a crucial role in distinguishing deceptive from truthful content. Finally, lemmatization was performed to reduce words to their base or dictionary forms, ensuring that different versions of the same word (e.g., "running", "ran") were treated uniformly by the models. Following preprocessing, the dataset was split according to the original fold structure. Folds 1–4, consisting of 640 reviews, were used for training and hyperparameter tuning, while fold 5, containing 160 reviews, was reserved as a test set for final model evaluation. [4]

4.2 Model Development and Evaluation Procedure

We conducted all model development and evaluation using the scikit-learn1 library, which provides a comprehensive suite of tools for text analysis and machine learning. Because machine learning models cannot process raw text directly, we converted the preprocessed text into numerical representations using Term Frequency-Inverse Document Frequency (TF-IDF) as our feature extraction technique. TF-IDF was chosen because it captures not only how frequently a word appears in a document but also how unique that word is across the entire corpus. This dual weighting helps the models focus on meaningful patterns rather than common, less informative words. [5]

Feature extraction was fully integrated into our cross-validation pipeline to prevent data leakage and ensure that no information from the test folds influenced the training process. During feature extraction, we excluded words that were either too frequent or too rare. Specifically, words that appeared in more than 95% of the reviews were excluded because they offer little discriminative power, while words appearing in fewer than two reviews were removed to eliminate noise from uncommon terms. This filtering step reduces noise and ensures that the resulting features generalize better across unseen data.

To evaluate robustness, we implemented stratified cross-validation and experimented with 5-fold, 8-fold, and 10-fold splits to determine which configuration provided the most stable and accurate performance for each model. Our development process followed a systematic approach to capture various linguistic patterns. We explored two n-gram configurations:

- Unigrams, representing individual words such as “room,” “dirty,” or “excellent”
- Unigrams combined with bigrams, which include two-word sequences such as “front desk” or “very clean.”

The inclusion of bigrams helps capture short phrases and local context that might better distinguish deceptive from truthful language.

Each model underwent hyperparameter tuning using grid search combined with cross-validation, allowing us to identify the optimal hyperparameters. Once the best hyperparameters were identified, we retrained each model on the training dataset (folds 1-4) to maximize learning from the available data. The test dataset (fold 5) was used to obtain unbiased performance estimates. We assessed each model using multiple metrics, such as accuracy, precision, recall, and F1-score, to capture different aspects of classification performance. Additionally, we produced confusion matrices to visualize classification outcomes and identify error patterns between truthful and deceptive reviews, providing insight into where and how each model struggled or outperformed. Finally, we used McNemar’s test to compare their performance.

4.3 Models

Our dataset was analyzed using various types of models to compare their performance. The models used in the analysis are Multinomial Naive Bayes (generative linear classifier), Logistic Regression with Lasso penalty (discriminative linear classifier), Single Classification Trees (non-linear classifier), Random Forest (ensemble

1 https://scikit-learn.org/stable/

of trees), and Gradient Boosting (ensemble of trees). Each model was trained and evaluated using the two feature configurations described above.

4.3.1 Multinomial Naïve Bayes (Generative Linear Classifier)

We applied Multinomial Naive Bayes (MNB) as our first approach due to its proven effectiveness in text classification. MNB assumes that features (words) are conditionally independent given the class label, making it computationally efficient while often achieving competitive results in document classification. To handle the high dimensionality of text data and improve model efficiency, we applied feature selection using the *chi-squared (χ^2)* statistic, which measures the dependence between each feature and the target class. This allowed us to retain only the most discriminative terms, speeding up training and reducing the risk of overfitting by focusing on the most important words. [6]

Hyperparameters were tuned via 5-fold cross-validation on the training set to ensure robust model selection. For each configuration, we optimized three hyperparameters, including the maximum number of TF-IDF features, the number of features to retain after chi-squared selection (*k*), and the Laplace smoothing parameter (*alpha*), which prevents zero probabilities for unseen words during prediction and addresses sparsity differences between unigram and bigram representations. The hyperparameter search was designed to balance model complexity and regularization for our 640 training samples. For TF-IDF, we tested unigram vocabularies from 2,000 to 10,000 words and bigram vocabularies from 5,000 to 12,000, since bigrams create a larger feature space. The chi-squared parameter (*k*) was set to retain 25% to 100% of features, letting the model find the optimal balance between noise reduction and keeping useful information. The Laplace smoothing parameter (*alpha*) ranged from 0.01 to 1.0.

The performance metrics of both models are shown in Table 4.1. The unigram model achieved 85.6% accuracy, meaning that approximately 86% of all reviews were correctly classified as either truthful or deceptive. The model attained a precision of 82.8%, indicating that when it predicted a review as deceptive, it was correct 82.8% of the time. The recall of 90% shows that the model successfully identified 90% of all actual deceptive reviews in the test set. The F1-score of 86.2%, which balances precision and recall through their harmonic mean, demonstrates solid overall performance in detecting deceptive content. The optimal configuration used 2,000 TF-IDF features, reduced to 1,500 after chi-squared selection and Laplace smoothing value of 0.5. T This feature set suggests that a focused vocabulary of highly discriminative terms is sufficient for capturing meaningful patterns. The confusion matrix (Figure 4.1 - left diagram) reveals that the unigram model correctly classified 65 truthful reviews and 72 deceptive reviews, but misclassified 15 truthful reviews as deceptive (false positives) and 8 deceptive reviews as truthful (false negatives). The high recall (90%) reflects the model's strength in identifying deceptive reviews, though this came at the cost of slightly more false positives.

Including bigrams further improved performance, increasing the accuracy to 88.1%, precision to 89.6%, recall to 86.3%, and F1-score to 87.9%. The optimal configuration used up to 8,000 TF-IDF features, reduced to 1,000 after selection, with a Laplace smoothing of 0.5. The larger initial feature set and aggressive dimensionality reduction (retaining only 12.5% of features) suggest that bigrams introduced considerable noise, requiring more selective feature filtering to maintain performance. The improvement demonstrates that word pairs capture useful contextual patterns that single words miss. The confusion matrix (Figure 4.1 - right

diagram) shows that bigrams improved classification balance, correctly identifying 72 truthful reviews and 69 deceptive reviews. Notably, false positives decreased significantly from 15 to 8, indicating improved precision in avoiding incorrectly flagging truthful reviews as deceptive. However, false negatives increased from 8 to 11, meaning the bigram model missed slightly more deceptive reviews. This trade-off reflects a shift toward higher precision at a minor cost to recall, with the bigram model achieving better overall balance and higher F1-score (87.9% vs. 86.2%), demonstrating more consistent performance across both classes.

	Accuracy	**Precision**	**Recall**	**F1-Score**
Unigrams	0.856	0.828	0.9	0.862
Unigrams + Bigrams	0.881	0.896	0.862	0.879

Table 4.1: Optimal Hyperparameters and Performance Metrics of Multinomial Naive Bayes models.

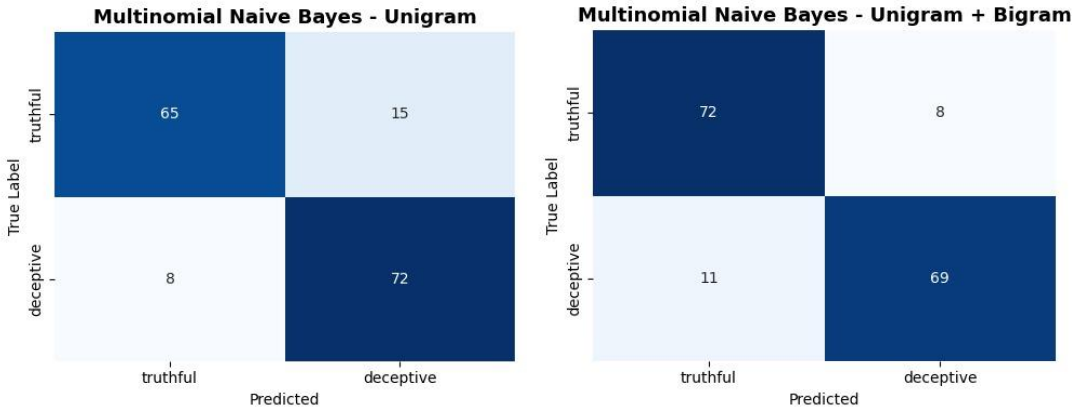

Figure 4.1: Confusion Matrices of Multinomial Naive Bayes Unigram model (left) and Unigram and Bigram model (right).

4.3.2 *Logistic Regression with Lasso penalty (Discriminative Linear Classifier)*

The next model used was Logistic regression with a Lasso penalty, which belongs to the class of discriminative linear classifiers. This classifier tries to predict the probability of a review being fake or genuine by learning how the input features, such as words or phrases, relate directly to the class label, rather than trying to model how are the data generated like Naïve Bayes does. Once again, both unigram and unigram and bigram feature representations were used to examine whether including short phrases, rather than single words alone, improved the classifier's performance.

To reduce the potentially large feature space, as each word or bigram can become a feature and to avoid overfitting, we restricted the models to a maximum of 1000 features (after multiple trials) and used L1

regularization, also known as the Lasso penalty. This penalty adds a constraint that discourages the model from assigning large weights to too many features. Consequently, it causes many unimportant feature weights to shrink to zero, effectively keeping only the most important words and performing feature selection. The strength of this penalty is controlled by a parameter called lambda (λ), or equivalently by C, which is the inverse of λ. A larger λ (or smaller C) means stronger regularization and more coefficients shrinking to zero, which translates to a simpler model. In contrast, a smaller λ (or larger C) allows the model to fit the data more closely but with a higher risk of overfitting.

To tune the logistic regression model with Lasso penalty, we used 5-fold cross-validation on the training set to find the best value of the regularization parameter C. We tested five values of C on a logarithmic scale, ranging from 0.1 to 10, and selected the value that achieved the highest cross-validated performance which was $C = 3.162$ for the unigram model and $C = 10$ for the unigram and bigram model. This approach helped balance the trade-off between model simplicity and predictive accuracy while preventing overfitting.

The performance metrics for both models are shown in Table 4.2. The logistic regression model using unigram features achieved an accuracy of 85%, precision of 88.9%, recall of 80% and an F1-score of 84.2%. When bigram features were added alongside unigrams, the overall performance improved slightly, reaching an accuracy 86.9%, precision of 87.3%, recall of 86.2%, and an F1-score of 86.8%. While the precision dropped slightly compared to the unigram model, both recall and overall accuracy increased, leading to a higher F1-score.

The confusion matrices in Figure 4.2 provide a more detailed view of the classifier's performance. For the unigram model, most reviews were correctly classified, with 72 truthful and 64 deceptive reviews correctly identified. However, 8 truthful reviews were misclassified as deceptive, and 16 deceptive reviews were incorrectly labeled as truthful. When bigram features were added, the model correctly classified 70 truthful and 69 deceptive reviews, showing fewer false negatives for deceptive reviews but a slight increase in false positives for truthful ones. This pattern aligns with the precision and recall results, where the inclusion of bigrams led to a small decrease in precision but an improvement in recall. These results indicate that the combined unigram–bigram model showed measurable improvement and achieved a better balance between identifying truthful and deceptive reviews, capturing more deceptive instances that the unigram model missed.

	Accuracy	**Precision**	**Recall**	**F1-Score**
Unigrams	0.850	0.889	0.800	0.842
Unigrams + Bigrams	0.869	0.873	0.862	0.868

Table 4.2: Performance Metrics of Logistic Regression models.

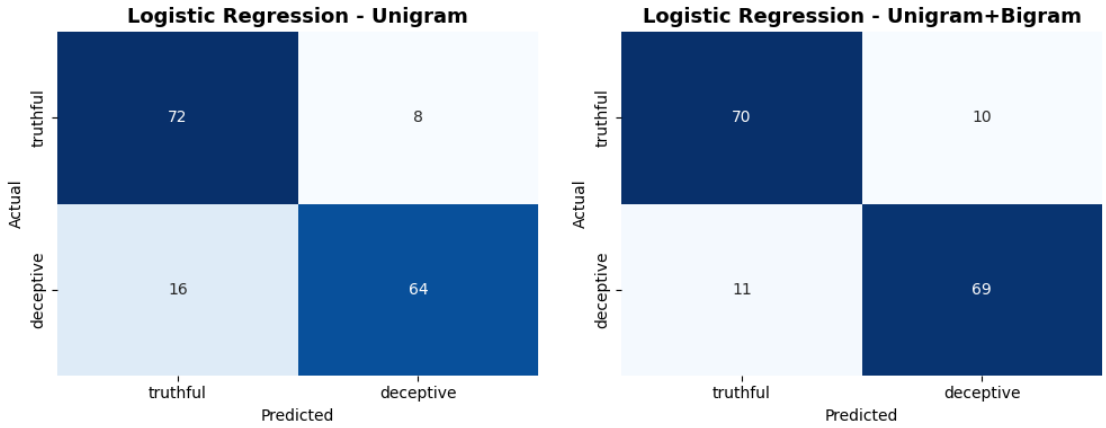

Figure 4.2: Confusion Matrices of Logistic Regression Unigram model (left) and Unigram and Bigram model (right).

Important Features Indicating Deceptive and Truthful Reviews:

To identify the most influential terms distinguishing fake from genuine reviews, feature importance was examined using the regularized logistic regression (L1) model trained on unigram and bigram features, which achieved the best overall performance among the logistic regression variants. After fitting the model, the signed coefficients (weights) were extracted. Each coefficient represents the strength and direction of association between a term and the predicted class. Positive weights indicate words that increase the likelihood of a review being classified as fake, while negative weights indicate words that increase the likelihood of a review being classified as genuine. The top five weighted terms pointing towards a fake review were “booked hotel”, “world”, “star”, “elevator” and “street”. The top five terms pointing towards a genuine review were “chicago”, “hotel chicago”, “relax”, “turned” and “finally”. These coefficients show that deceptive reviews favor broad, promotional language (e.g., “booked hotel,” “five star”), whereas truthful reviews use more location-specific or experiential language (e.g., “chicago,” “relax,” “finally”). The observed pattern aligns with the findings of Ott et al. [1].

Comparison with Generative Linear Model (Multinomial Naive Bayes):

The Multinomial Naive Bayes (MNB) and regularized logistic regression (L1) models achieved comparable performance for our task. With unigram features, both classifiers reached an accuracy of about 85%, confirming that linear text models can effectively distinguish fake from genuine reviews. When bigram features were added, performance improved for both models; however, MNB achieved the highest overall scores (accuracy = 88.1%, F1 = 88.2%), slightly surpassing logistic regression (accuracy = 86.9%, F1 = 86.8%). These results suggest that, given a relatively small corpus of 800 documents, the generative Naive Bayes model benefits from its fast convergence and strong independence assumptions, allowing it to reach near-optimal performance with limited data. In contrast, the discriminative logistic regression model, while theoretically superior for large datasets due to its lower asymptotic error, may require more training examples to realize this advantage. This outcome is consistent with the “two-regime” phenomenon described by Ng and Jordan where generative models such as Naive Bayes tend to outperform discriminative models on smaller

datasets by converging more rapidly, but discriminative models like logistic regression eventually overtake them as sample size increases. [10]

4.3.3 Decision Tree (non-linear Classifier)

The next model used was the Decision Tree classifier, which belongs to the class of non-linear models. Unlike linear classifiers such as Logistic Regression or Naïve Bayes, decision trees can split the feature space using non-linear decision boundaries and hierarchical decision rules. This allows them to capture non-linear relationships in the data, which can be useful for classifying reviews as truthful or deceptive. In our experiments, we used the Gini impurity criterion, which evaluates how well a potential split separates truthful from deceptive reviews.

Once again, after trying several different values for both unigram and unigram and bigram models, we constrained the feature space to 1000 features. Hyperparameter tuning was performed using 10-fold stratified cross-validation on the training set. Two key parameters were optimized through grid search. The first one was the cost-complexity pruning parameter (α), which controls post-pruning by penalizing overly complex trees by removing unnecessary branches, and the second one was the maximum depth of the tree which limits how deeply the model can grow. We tested 20 α values evenly spaced between 0.0 and 0.05 and explored maximum depths of *None, 10, 20, and 30.* The best configuration for the unigram model was found at $\alpha = 0.011$, while the unigram and bigram model achieved optimal performance with $\alpha = 0.018$. We should note that in both cases the tree depth remained unrestricted.

The confusion matrices in Figure 4.3 illustrate the classifier's performance in detail. For the unigram model, the decision tree correctly identified 55 truthful and 50 deceptive reviews, but misclassified 25 truthful as deceptive and 30 deceptive as truthful. When bigram features were included, the overall accuracy slightly improved from 65.0% to 65.6% as shown in Table 4.3, with precision = 66.7%, recall = 62.5% and F1-score = 64.5%. Although the effect of adding bigrams as input was limited, this small improvement suggests that including short phrases helped the model capture additional patterns.

Overall, the Decision Tree achieved moderate classification accuracy compared to the linear models. One possible reason could be that decision trees make local, rule-based choices, such as whether a word appears more or less than a certain number of times. Text meaning, however, is usually spread across many words, so linear models can capture those broader patterns more effectively. In addition, our dataset can be considered relatively small, which limits how well a non-linear model like a tree can learn and generalize. So, even though the Decision Tree is easy to interpret and helps show which words influenced predictions, its accuracy was limited by the sparse, noisy and small-scale nature of the data. More advanced ensemble methods such as Random Forests or Gradient Boosting should be explored to test whether they can improve performance by combining many trees to get more stable and accurate predictions.

	Accuracy	**Precision**	**Recall**	**F1-Score**
Unigrams	0.650	0.646	0.663	0.654

Unigrams + Bigrams	0.656	0.667	0.625	0.645

Table 4.3: Optimal Hyperparameters and Performance Metrics of Decision Tree models.

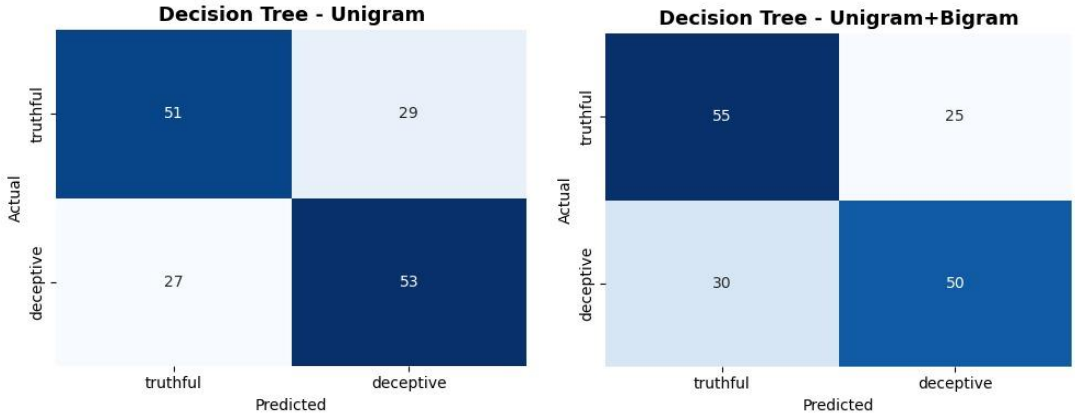

Figure 4.3: Confusion Matrices of Decision Tree Unigram model (left) and Unigram and Bigram model (right).

4.3.4 Random Forests (ensemble of trees)

We used the Random Forest classifier because of its nature as a non-linear ensemble of trees, which makes it useful in text classification. This model combines the predictions of multiple decision trees built on random subsets of both samples and features, which helps reduce overfitting and increase model stability. By averaging the predictions of many weak or noisy decision trees, Random Forest produces a more accurate and robust overall model. It can be viewed as an extension of the bagging technique that aims to reduce the correlation of the predictions of individual trees by randomly selecting a subset of features at each split, which lowers the correlation between individual trees and improves generalization capability [7].

Hyperparameter tuning was performed using five-fold stratified cross-validation, which considers all feature combinations using the F1-score of the "deceptive" class as the scoring metric. We used the F1-score because it provides a balanced measure of precision and recall and reflects the model's ability to correctly identify deceptive reviews.

Choosing appropriate hyperparameters was crucial for achieving good generalization and stability. The number of trees in the forest was tested with values of *300, 400*, and *500*, as increasing this number generally improves model stability by averaging across more trees, thereby reducing variance. The maximum tree depth was tested at unrestricted depth, *20*, and *30*, allowing the grid search to compare models that can grow fully with those that have limited depth, which helps control complexity and prevent overfitting. The minimum number of samples required to split an internal node was set to either *2* or *5*, and the minimum number of samples required to form a leaf node was set to *1* or *2*. These low thresholds allowed individual trees to grow deep enough to capture subtle linguistic patterns, while the ensemble averaging in Random Forest helped mitigate potential overfitting. Finally, the number of features considered at each split was set to follow either

the *square root* or the *logarithm base two* of the total number of available features in order to further reduce overfitting by introducing diversity among the trees. [14].

After performing grid search and cross-validation, the optimal configurations were found to be similar across both feature representations. In both cases, the best models used an unrestricted tree depth, considered logarithm-based feature selection, required a minimum of one sample per leaf and five samples to split a node, but differed slightly in the number of trees. The unigram model performed best with 300 trees, while the unigram–bigram model achieved better results with 400 trees.

In addition to the tuned hyperparameters, we also enabled the out-of-bag (OOB) evaluation to obtain an internal estimate of the model's generalization performance as shown in Table 4.4. This method leverages the data not used in the training of individual trees, offering a reliable, complementary assessment to cross-validation [7]. As shown in Table 4.4, including bigrams helped the model capture additional linguistic patterns.

	OOB-score
Unigrams	0.8391
Unigrams + Bigrams	0.8578

Table 4.4. OOB-score of Random Forest models.

The Random Forest classifier achieved high performance across both unigram and unigram+bigram representations as shown in Table 4.5. When using only unigrams, the model has an accuracy of 86.9% with a precision of 89.3% and a recall of 83.8%, resulting in an overall F1-score of 86.5%. After adding bigrams to the feature representation, the accuracy slightly decreased to 84.4%. The precision increased to 92.3% and this shows that the model made fewer mistakes when predicting deceptive reviews. However, the recall decreased to 75% meaning it found fewer of the deceptive reviews overall. Because of this drop in recall, the overall F1-score fell to 82.8% and we can say that the addition of bigrams did not give an improvement in performance.
The confusion matrix in Figure 4.4 shows that for the unigram model, most of the reviews were correctly classified, with 72 truthful and 67 deceptive reviews identified correctly, and only 8 truthful and 13 deceptive reviews misclassified. In the unigram+bigram le remodel, the number of correctly predicted truthful reviews increased to 75, while deceptive predictions slightly decreased to 60, showing the previous observation that the inclusion of bigrams improved precision but reduced recall.
The unigram model maintained a more balanced performance across both classes, while the bigram model became more conservative in detecting deception. Adding bigrams helped the model see some useful word pairs, but it also made the data more complex, and the unigram model stayed more balanced and gave more stable
Overall, the Random Forest with unigrams features achieved the best results, with the highest accuracy and a strong ability to find deceptive reviews.

	Accuracy	**Precision**	**Recall**	**F1-Score**

Unigrams	0.8688	0.8933	0.8375	0.8645
Unigrams + Bigrams	0.8438	0.9231	0.75	0.8276

Table 4.5: Optimal Hyperparameters and Performance Metrics of Random Forest models.

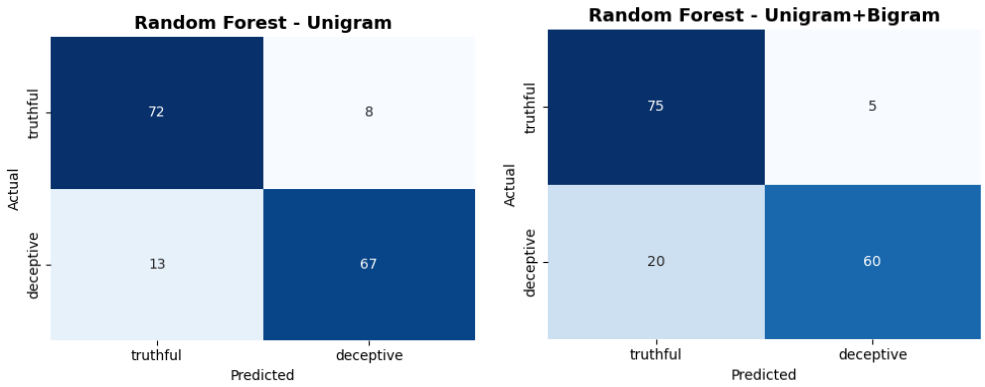

Figure 14: Confusion Matrices of Random Forest Unigram model (left) and Unigram and Bigram model (right)

4.3.5 Gradient Boosting (ensemble of trees)

The final model applied was the Gradient Boosting classifier, another non-linear ensemble method based on decision trees. Unlike Random Forest, which builds trees independently, in Gradient Boosting, trees are added one at a time sequentially, where each successive tree is trained to correct the errors made by the previous tree. This iterative approach helps to reduce bias without increasing variance, making it effective for text classification [7, 8].

As with the Random Forest model, the dataset was split using k-fold cross-validation to ensure class balance across folds. However, in this case, we used eight folds instead of five to obtain a more reliable estimate of model performance. Since Gradient Boosting builds trees sequentially, it's more sensitive to changes in the training data, and using more folds helps make the results smoother and reduces overfitting [9]. During cross-validation, the F1-score was used because it offers a balanced measure of precision and recall, suitable for detecting deceptive content.

Choosing the right value for each hyperparameter played a crucial role in determining the model's ability to generalize effectively. The number of boosting stages, or trees, was tested with values of *200* and *300*, as increasing this number generally improves model stability and reduces variance by allowing the ensemble to correct errors over more iterations. The maximum depth of each tree was set to either *3* or *5*, enabling the model to compare the performance of shallower and moderately deep trees. The learning rate, which controls how much each new tree contributes to correcting the mistakes of the previous ones, was tested at *0.05* and *0.1* to balance training speed and accuracy, given that smaller learning rates make training slower but typically lead to more precise results. Finally, the subsample rate was set to either *0.7* or *0.85*. We chose these two

values because a too small value might reduce accuracy, but with a too high value, there's the risk of overfitting [14].

The optimal configuration of the Gradient Boosting model differed slightly between the unigram and unigram–bigram feature sets. In both cases, the models performed best when trained with 300 boosting stages, a learning rate of 0.05, and by considering a square-root proportion of features when searching for the best split. However, some important differences were observed. The unigram model achieved its best performance with shallower trees of depth 3, a higher subsample rate of 0.85, and a smaller vocabulary size of 5,000 terms. In contrast, the unigram–bigram model performed better with deeper trees of depth 5, a lower subsample rate of 0.7, and a larger vocabulary size of 8,000 terms. These results suggest that the bigram configuration benefited from deeper trees and a larger vocabulary, allowing it to capture more complex interactions.

In terms of performance, the Gradient Boosting classifier achieved strong results for both representations, as shown in Table 4.6. With unigrams, the model reached an accuracy of 83.1%, precision of 89.6%, recall of 75% and an F1-score of 81.6%. When bigrams were included, performance improved across all the metrics: accuracy increased to 87.5%, precision went to 85.4%, recall to 78.8% and the overall F1-score rose to 86.3%. All this improvement shows that bigrams added useful context and helped the model understand word relationships better.
The sequential nature of Gradient Boosting allows it to iteratively correct the errors made by previous trees, step by step. This process is particularly beneficial when dealing with more complex features such as bigrams, as it helps the model capture richer linguistic patterns. In the end, with bigram features, the model became significantly better at recognizing complex patterns of deception, improving all evaluation metrics.

The confusion matrices in Figure 4.5 show that for the unigram model, most of the reviews were correctly classified, with 73 truthful and 60 deceptive reviews identified correctly, while 7 truthful and 20 deceptive reviews were misclassified. Including the bigrams, the number of correctly predicted truthful reviews increased to 77, and the correctly identified deceptive ones to 63, while misclassifications decreased to only 3 truthful and 17 deceptive reviews.

The bigrams showed a clear improvement in correctly identifying truthful reviews, while also maintaining good performance in detecting deceptive ones. This confirms that Gradient Boosting benefited from the additional contextual information provided by bigrams, improving precision and recall.

	Accuracy	**Precision**	**Recall**	**F1-Score**
Unigrams	0.831	0.896	0.750	0.816
Unigrams + Bigrams	0.875	0.955	0.788	0.863

Table 4.6: Optimal Hyperparameters and Performance Metrics of Gradient Boosting models.

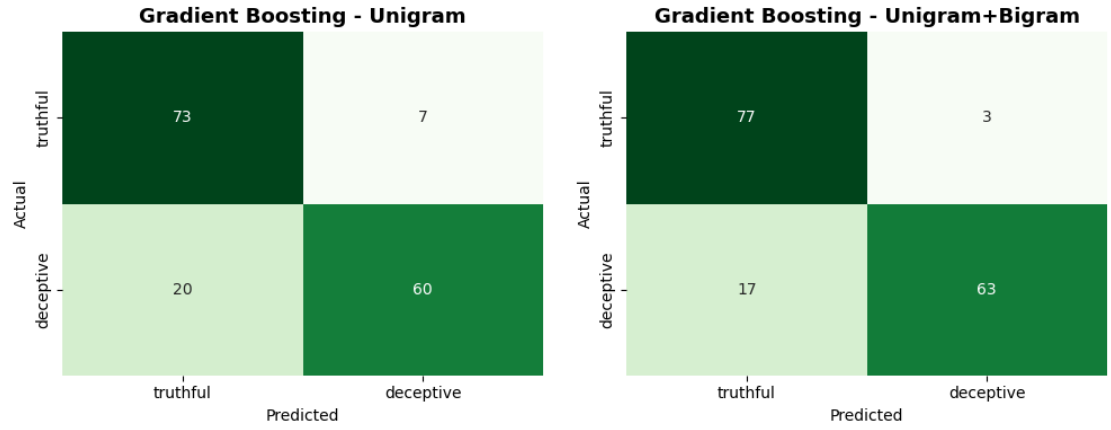

Figure 4.5: Confusion Matrices of Gradient Boosting Unigram model (left) and Unigram and Bigram model (right).

Comparison between Logistic Regression and Random Forest/Gradient Boosting:

Random Forest and Gradient Boosting differ substantially from Logistic Regression in both their structure and learning logic. Logistic regression is a linear model that assumes a direct relationship between the features and the target labels [8], while Random Forest and Gradient Boosting are non-linear ensembles of tree models made up of multiple decision trees that can learn more complex patterns in the data [7][9]. For this reason, we expect differences in performance, especially when bigrams are included.

When comparing their performance, Logistic Regression showed strong and consistent results, especially when using bigrams. Random Forest performed slightly better with unigrams but did not improve after adding bigrams, and Gradient Boosting took advantage of the richer bigram features and achieved the best overall performance among the three models.

Although Random Forest and Gradient Boosting are generally expected to outperform linear models, in this case, the difference was not very large because the dataset is relatively small and the TF-IDF features already capture most of the useful information linearly [13], allowing Logistic regression to remain highly competitive.

Overall, Gradient Boosting achieved the best results, showing that more advanced models can work better when the features contain more information, like with bigrams.

4.4 Statistical Comparison of Model Performance

While accuracy and other performance metrics provide valuable insights into individual model performance, they do not indicate whether observed differences between models are statistically significant or merely due to chance. To compare classifiers and determine if one model genuinely outperforms another, we used McNemar's test, a non-parametric statistical test specifically designed for comparing paired classifiers on the same test set.

McNemar's test evaluates whether two models make systematically different errors by examining instances where one model correctly classifies an example while the other does not. To do this, it constructs a 2×2

contingency table based on two critical counts: cases where Model A is wrong and Model B is correct and cases where Model A is correct and Model B is wrong. [11]

The test statistic follows a chi-squared distribution with one degree of freedom, and the resulting p-value indicates whether the difference in error patterns is statistically significant. A p-value below 0.05 suggests that the performance difference between models is unlikely to be due to random chance, providing evidence that one model genuinely performs better than the other. [12]

We implemented McNemar's test to perform pairwise comparisons between all trained models, including both unigram and bigram configurations for each classifier. This comprehensive approach allowed us to assess not only differences between model architectures but also the statistical significance of improvements gained by incorporating bigram features.

The results of all pairwise comparisons are visualized in Figure 4.6 as a p-value heatmap, where each cell represents the McNemar test p-value for a pair of models. Lower p-values (darker purple colors) indicate statistically significant differences in performance, while higher p-values (yellow colors) suggest that observed differences could be due to random variation.

The heatmap reveals several important patterns. First, all bigram models significantly outperformed their unigram counterparts ($p < 0.05$), as shown by the dark purple blocks along the diagonal, confirming that bigram features provided statistically meaningful improvements across all architectures. Second, Classification Tree models showed distinct performance differences from ensemble methods like Random Forest and Gradient Boosting ($p \approx 0$), indicating fundamentally different classification strategies when used individually versus in ensembles.

However, among top-performing models, differences were often not statistically significant. Comparisons between Gradient Boosting and Random Forest ($p = 0.27$), and between Logistic Regression and Gradient Boosting ($p = 0.82$) suggest these models perform comparably despite numerical accuracy differences. Notably, Multinomial Naive Bayes with bigrams remained competitive with more complex models ($p = 0.65$ vs. Logistic Regression), validating its efficiency for text classification tasks.

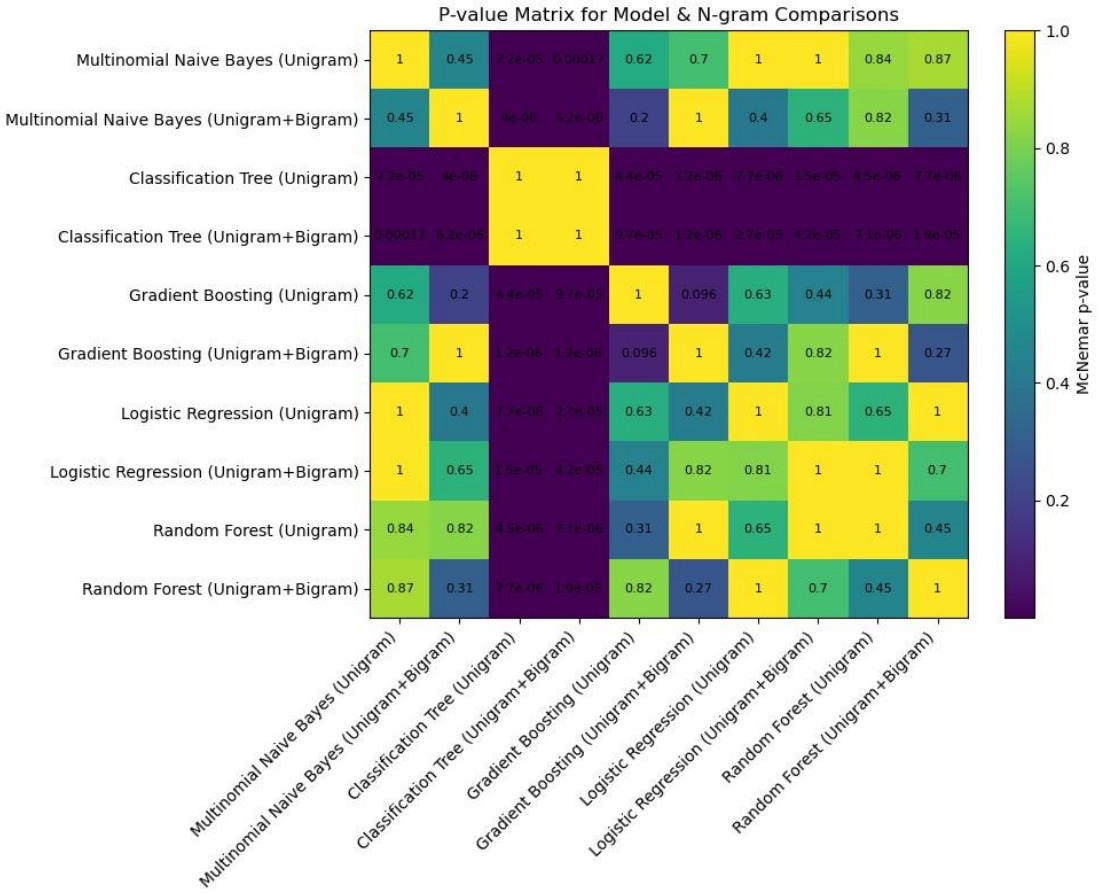

Figure 4.6: P-value Matrix for Model & N-gram Comparisons

5 CONCLUSIONS

This study successfully demonstrated the effectiveness of machine learning approaches for detecting deceptive hotel reviews, achieving strong performance across multiple models. The systematic comparison of five classifiers (Multinomial Naive Bayes, Logistic Regression, Classification Tree, Random Forest, and Gradient Boosting) revealed that incorporating bigram features significantly improved detection accuracy across all models ($p < 0.05$). Multinomial Naive Bayes with bigrams emerged as the optimal choice, achieving 88.1% accuracy with 88.2% F1-score while maintaining computational efficiency and interpretability. Statistical testing via McNemar's test confirmed that while MNB performed comparably to more complex models like Logistic Regression and ensemble methods ($p > 0.05$), it offered practical advantages in terms of training speed and model simplicity. The results show that effective deception detection can be achieved with simple models and well-designed features. Lightweight, efficient models can flag suspicious reviews accurately. Future work could explore deep learning, include reviewer metadata, or apply the approach to other domains.

6 GENERATIVE AI TOOLS

We utilized generative AI tools (ChatGPT) to support our learning process by seeking explanations of machine learning concepts, guidance on appropriate statistical tests (e.g., McNemar's test), and suggestions for structuring our analysis. AI assistance was also used to improve the clarity and flow of our written report. However, all code development, experimental design, data analysis, and interpretation of results were conducted independently by the authors.

7 REFERENCES

[1] Myle Ott, Yejin Choi, Claire Cardie and Jeffrey T. Hancock, Finding deceptive opinion spam by any stretch of the imagination. Proceedings of the 49th meeting of the association for computational linguistics, pp. 309-319, 2011

[2] Myle Ott, Claire Cardie and Jeffrey T. Hancock, Negative deceptive opinion spam. Proceedings of NAACL-HLT 2013, pp. 497-501, 2013.

[3] Rajesh K. P., Sherlin Paul P. S., Manikanda Prabu Nallasivam, Hari Kumar S., Sakthi Gokul Rajan C., Dharum V. S., Detection of Fake Hotel Reviews Using ANFIS and Natural Language Processing Techniques, 2024 International Conferen on Circuit Power and Computing Technologies (ICCPCT).

[4] Anusuya Baby Hari Krishnan, Unmasking Falsehoods in Reviews: An Exploration of NLP Techniques, College of IT, United Arab Emirates University, UAE, July 2023

[5] Harris Drucker, Donghui Wu, Vladimir N. Vapnik, Support Vector Machines for Spam Categorization, IEEE Transactions on Neural Networks, Vol. 10, No. 5, September 1999

[6] Novi Yusliani, Syechky Al Qodrin Aruda, Mastura Diana Marieska, Danny Mathew Saputra, Abdiansah, The effect of Chi-Square Feature Selesction on Question Classification using Multinomial Naive Bayes, Universitas Sriwijaya Indonesia, Oct 2022

[7] Leo Breiman, Random Forests. Machine Learning, 2001

[8] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.).

[9] Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, 29(5), 1189–1232.

[10] Andrew Y. Ng, Michael I. Jordan, On Discriminative vs Generative classifiers: A comparison of logistic regression and Naive Bayes.

[11] Ayodele Marvellous, Bamidele Matthew, Mauro Pezze, Silivia Abrahao, Birgit Penzenstadler, Ashis kumar mandal, Md Nadim, Ulrik Schultz, Statistical Significance Testing in ML Model Comparisons: Beyond p-values and t-tests, June 2025

[12] Matilda Q. R. Pembury Smith, Graeme D. Ruxton, Effective use of the McNemar test, 2020

[13] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval

[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research