American University of Armenia

College of Science and Engineering

Fake News Detector By Naive Bayes Algorithms

Machine Learning

Group Project

Nane Mambreyan, Hasmik Sahakyan, Gayane Ohanjanyan

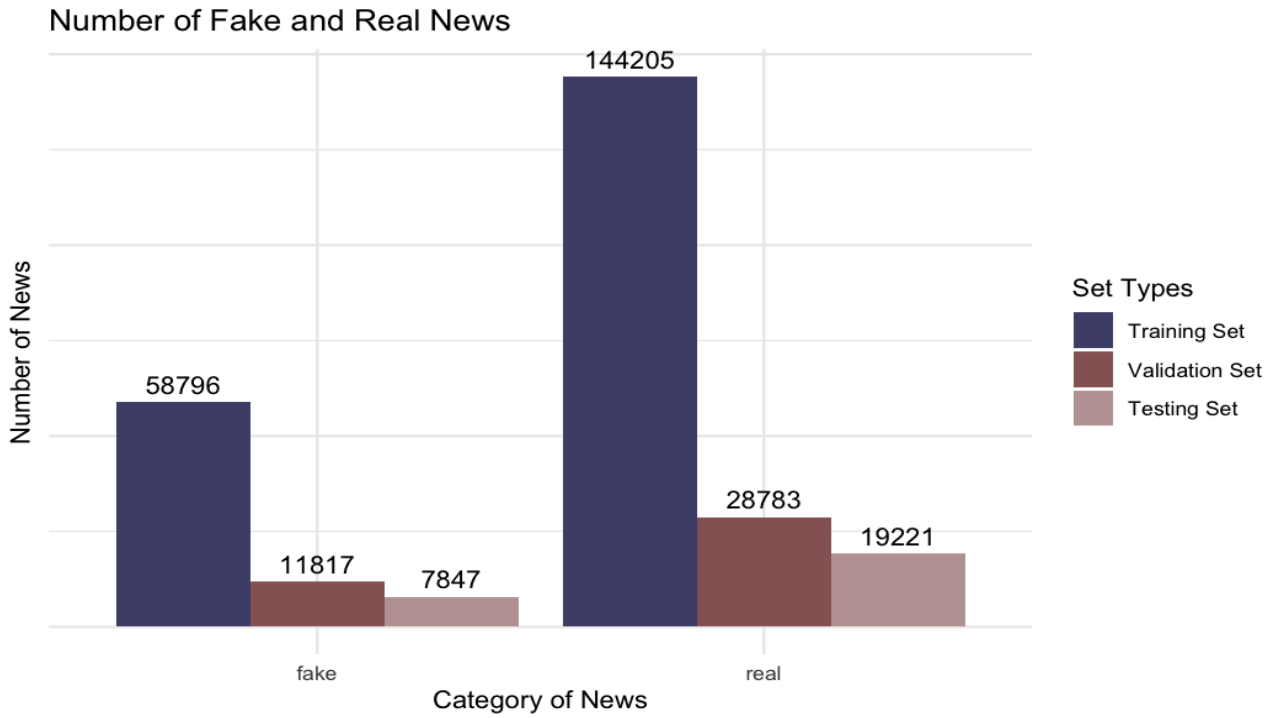
Introduction

Fake news detection has become a critical challenge in today's information age, where misinformation spreads rapidly through various online platforms. In this study, we create and explore the efficacy of machine learning algorithms for detecting fake news. The data was chosen from Kaggle. The dataset comprises around 270,669 pieces of text selected from various news articles from the web pages and texts generated by Chat GPT. The chosen dataset has been split into train, validation, and test sets.
We employ a diverse set of techniques, including Multinomial Naive Bayes (MNB) and Gaussian Naive Bayes (GNB), leveraging both TF-IDF and Count Vectorizer for feature extraction. Additionally, we implement custom Gaussian Naive Bayes and Multinomial Naive Bayes classifiers to further investigate their performance. The dataset comprises labeled news articles, and extensive preprocessing techniques are applied to enhance the data quality. Through rigorous experimentation, we evaluate the performance of each model on both validation and test datasets, employing resampling methods to mitigate class imbalance.
Our results show that some classifiers and feature extraction methods perform better than others for fake news detection. We also compare our custom-made models with built-in models from Sklearn library.

Key words: Fake news detection, Machine Learning, Gaussian Naive Bayes(GNB), Multinomial Naive Bayes(MNB), Term Frequency-Inverse Document Frequency(TF-IDF), Count Vectorizer, Hyperparameter Tuning

Data distribution

In the first stage of the project, we divided our dataset into training, validation, and testing sets by following a systematic approach. Initially, we split the data into a training set and a temporary subset using the train_test_split function, allocating 75% of the data for training. Next, we further divide the temporary subset into validation and test sets, with 40% designated for validation and the remaining 60% for testing. This ensures distinct subsets for training, fine-tuning model parameters, and evaluating performance. By setting a random state, we ensure reproducibility, facilitating consistent results across executions.



The bar chart illustrates the distribution of fake and real news articles across training, validation, and test sets. This visual depiction provides a comprehensive overview of the dataset's composition and facilitates comparisons between different subsets. By color-coding each subset, we highlight the relative proportions of fake and real news within each category.
As we can see, the amount of fake and real news is not equally distributed, for which we will use resampling methods to address class imbalance. We will discuss this in more detailed later.

Data Processing

In this section, we will go through the preprocessing pipeline applied to the textual data before training the machine learning models. The preprocessing aims to clean and standardize the text data, making it suitable for feature extraction, and is critical for enhancing the efficacy of the Naive Bayes algorithms.
The function preprocess_text() systematically transforms the raw text, encompassing several steps as follows.

Text Cleaning
We removed URLs from the text data using regular expressions to eliminate any web links present in the news articles. Eliminating URLs from the text data removes irrelevant information that may not contribute to distinguishing between fake and real news. This can enhance the accuracy of the models by reducing noise in the dataset. Then common contractions (e.g., "can't" to "cannot") were expanded to their complete forms to ensure consistency in the text. Expanding contractions ensures uniformity in language usage, which improves model performance by reducing the variability in textual representations. Our next step was to remove any HTML tags or markup present in the text using regular expressions. Stripping HTML tags removes formatting elements that do not carry semantic meaning, thus simplifying the text and improving model interpretability.
We optionally suggest stopword removal and abbreviation handling. Although not implemented in the final model, we prepared code snippets to handle abbreviations and acronyms, expanding them to their full meanings for better comprehension.

Tokenization and Lowercasing
We tokenized the cleaned text into individual words using the word_tokenize function from the NLTK library and converted all tokens to lowercase to standardize the text and reduce the vocabulary size. This way, we ensure consistency in text representation. Breaking down the text into individual tokens facilitates subsequent analysis by treating each word as a separate feature. This step is crucial for feature extraction and modeling.
Lowercasing standardizes text representation, potentially improving accuracy by reducing the sparsity of the feature space.

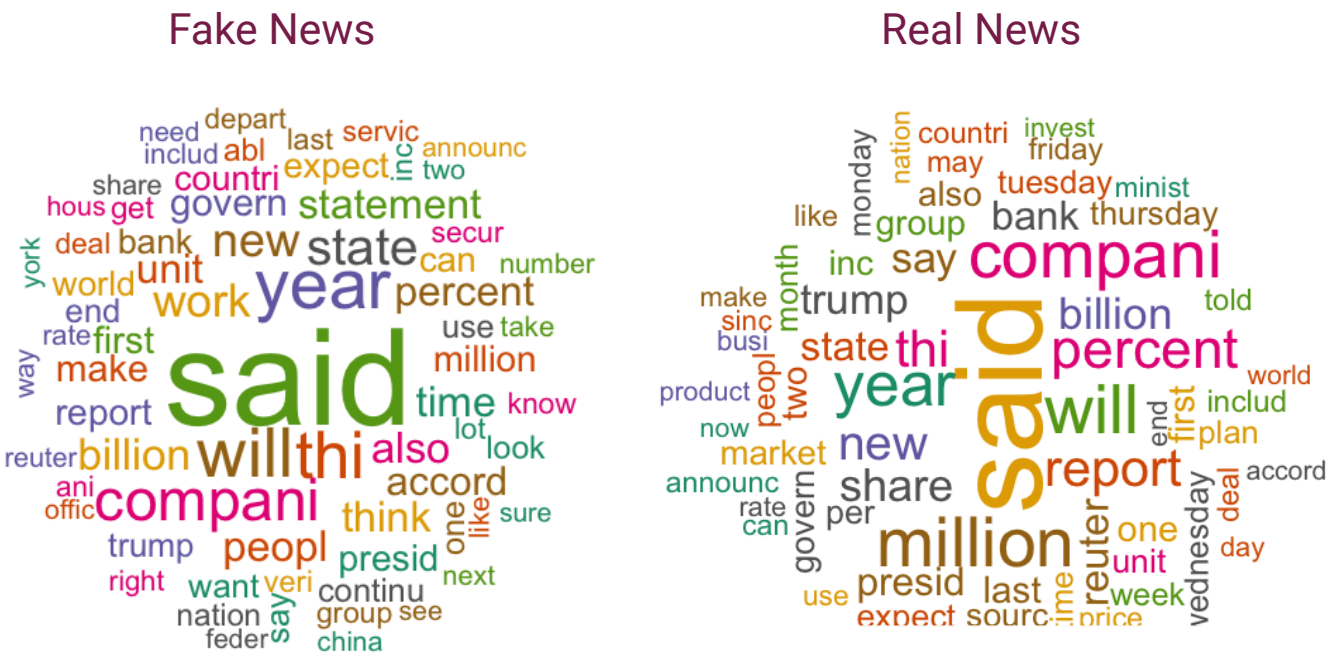
Removing Punctuation, Numbers, and Special Characters
Regular expressions were utilized to remove any non-alphabetic characters, including punctuation marks, numbers, and special symbols, from the tokenized text. This step

helps to remove noise from the text data, focusing the model's attention on relevant linguistic features and patterns.

Lemmatization and Stemming
We applied lemmatization using the WordNetLemmatizer, and Porter stemming was employed. Lemmatization and stemming reduce words to their base forms, standardizing the text representation and consolidating the vocabulary, resulting in model accuracy improvement by reducing the variability in word forms and capturing semantic similarities between related words. However, they increase computational complexity and memory usage.

By implementing these preprocessing steps, we ensured that the text data was appropriately formatted and ready for feature extraction and subsequent model training. This preprocessing pipeline helped improve the quality and consistency of the input data, contributing to the overall effectiveness of the fake news detection models by enhancing the accuracy.

Results of Preprocessing



The visualization depicts the frequency of words in a processed textual training dataset, essential for discerning fake from real news using the Naive Bayes algorithm. Notably, the word "said" emerges as the most common in both categories, indicating its ubiquitous presence in news articles. Additionally, words like "year," "compani," "thi,"

"will," and "percent" recur frequently across both labels, underscoring their significance in news content. However, certain terms exhibit distinct patterns: "work," "state," and "billion" are prevalent in fake news, whereas "million," "report," and "trump" feature prominently in real news.

Methodology

In this section, we outline the techniques and methods employed to detect fake news using machine learning algorithms. We utilized machine learning algorithms, including Multinomial Naive Bayes and Gaussian Naive Bayes, for fake news detection. These algorithms analyze patterns in the text data to classify news articles as either fake or real based on their linguistic features.

Feature Extraction Techniques

Feature extraction techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) and Count Vectorizer were employed to convert the textual content of news articles into numerical representations that machine learning models can understand. TF-IDF assigns weights to words based on their frequency in a document relative to their frequency in the entire corpus, while Count Vectorizer counts the occurrences of each word in a document.

Resampling Methods for Class Imbalance

To address class imbalance, where the number of fake news articles may be significantly lower than the number of real news articles, we employed resampling methods. These techniques ensure that the machine learning models are trained on a balanced dataset by either oversampling the minority class (fake news) or undersampling the majority class (real news), thus preventing bias towards the dominant class. This helps improve the performance and fairness of the models in detecting both fake and real news articles.

Hyperparameter Tuning

Hyperparameter tuning is essential for optimizing model performance. We focus on tuning the alpha hyperparameter for our Multinomial Naive Bayes model and var_smoothing for Gaussian Naive Bayes. This parameter serves as a smoothing factor, addressing situations where certain words are absent in the training data but appear in

the test data. By tuning alpha and var_smoothing, we aimed to optimize the models' ability to generalize to unseen data.

How is Hyperparameter Tuning Done?

Define Grid of Hyperparameters: We define a grid of alpha/var_smoothing values to search over. These values are chosen based on a range of possible values that are likely to yield good performance.
Initialize Model: We initialize a Multinomial/Gaussian Naive Bayes model without specifying any hyperparameters.
Grid Search with Cross-Validation: We perform a grid search with cross-validation (5-fold) using the defined grid of hyperparameters. Each combination of hyperparameters is evaluated using cross-validation to estimate the model's performance.
Select Best Hyperparameter: We select the best-performing hyperparameter based on a chosen evaluation metric (in this case, accuracy). The best hyperparameter is the one that maximizes the model's performance on the validation set.
Use Best Hyperparameter: Finally, we use the best hyperparameter obtained from the grid search to train the final model on the entire training dataset.

Multinomial Naive Bayes Model

For our fake news detection task, we opted to utilize the Multinomial Naive Bayes (MNB) model due to its suitability and effectiveness in handling text classification tasks. Given the textual nature of our dataset, where news articles are represented as sequences of tokens, MNB emerges as a natural choice. In our custom Multinomial Naive Bayes (MNB) model, we implemented the fundamental principles of the algorithm to classify news articles as either fake or real based on their textual content.

Why Multinomial Naive Bayes?

Textual Data Handling: Given that our dataset primarily comprises textual data, where each news article is represented as a sequence of tokens, Multinomial Naive Bayes is a natural choice. It excels in processing and classifying text data efficiently.

Multiclass Classification Capability: Fake news detection involves categorizing news articles into distinct classes: fake or real. This multiclass classification scenario aligns well with MNB's capabilities, as it can efficiently handle multiple classes. Despite its

simplicity, MNB's assumption of feature independence, where words are treated as conditionally independent given the class label, often holds in practice for text classification tasks. This assumption simplifies the modeling process while yielding reasonable performance.

Feature Independence Assumption: While simplistic, Multinomial Naive Bayes assumes conditional independence among features (tokens) given the class label. This assumption often holds well for text classification tasks. Even though words in news articles may be related, their occurrences can be considered independent when conditioned on the article's class.

Hyperparameter Tuning for Multinomial Naive Bayes Model

The alpha hyperparameter is a smoothing parameter used in Multinomial Naive Bayes to handle cases where certain words are absent in the training data but appear in the test data. Tuning this hyperparameter allows us to find the optimal value that improves the generalization ability of the model.
The steps for choosing an alpha hyperparameter are defined above.

Multinomial Naive Bayes Model Custome Implementation

The Multinomial Naive Bayes (MNB) classifier, a variant of the Naive Bayes algorithm, is utilized for fake news detection. It calculates class prior probabilities and conditional probabilities of features given each class, incorporating a smoothing parameter to address data sparsity issues. By operating on logarithms of probabilities, it prevents numerical instability during computation. Ultimately, based on the computed log probabilities, the model predicts the class label for input data, distinguishing between fake and real news articles.

Gaussian Naive Bayes Model

GNB is an effective choice for fake news detection, particularly when dealing with continuous features or data that aligns with a Gaussian (normal) distribution. While Multinomial Naive Bayes (MNB) is commonly employed for text classification tasks, GNB shines when handling numeric features, as seen in various word representation techniques discussed earlier.

Why Gaussian Naive Bayes?

Multiclass Classification: GNB excels at multiclass classification tasks like categorizing news articles as "fake" or "real" by leveraging the assumption of Gaussian-distributed features. In our scenario, aggregated word counts often approximate a Gaussian distribution, particularly with extensive vocabularies. Techniques such as TF-IDF bolster this Gaussian-like behavior through normalization and weighting.

Assumption of Feature Independence: Similar to MNB, GNB assumes feature independence given the class label, simplifying modeling and enhancing computational efficiency crucial for large datasets. Despite articles focusing on specific topics, this assumption suggests that the presence of terms depends more on the article's authenticity than on interactions between terms.

Robustness to Noisy Data: GNB exhibits robustness to noisy data and gracefully handles missing values. Its performance remains stable even when some assumptions of the underlying data distribution are violated.

Hyperparameter Tuning for Gaussian Naive Bayes Model

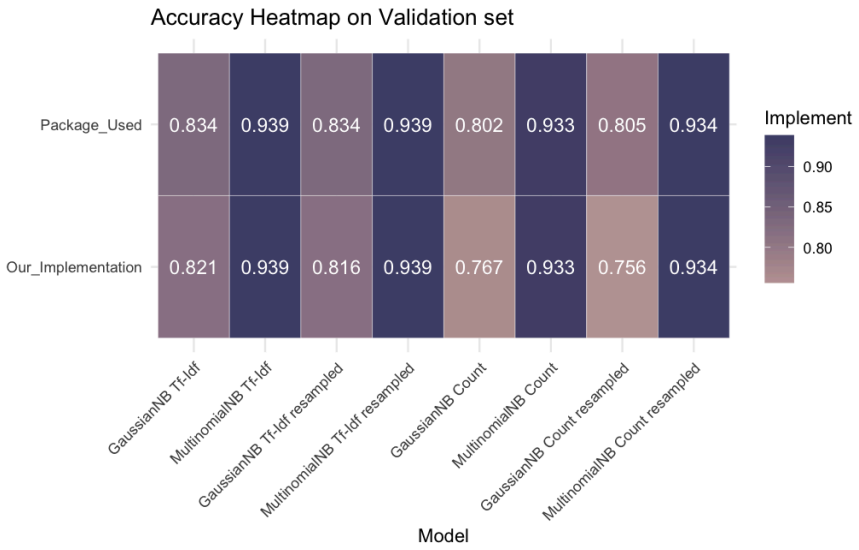
The var_smoothing hyperparameter is a smoothing parameter used to stabilize the computation of probabilities, especially when dealing with features with zero variance. By tuning this hyperparameter, we aim to find an optimal value that enhances the model's generalization capability.
The steps for choosing var_smoothing hyperparameter are defined above.

Gaussian Naive Bayes Model Custome Implementation

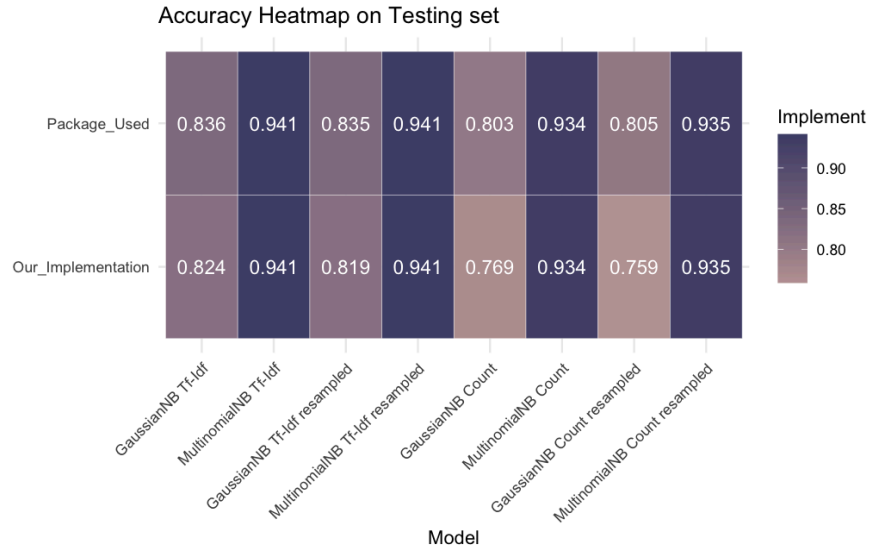
The Gaussian Naive Bayes (GNB) classifier is designed based on the principles of the Naive Bayes algorithm, similar to the Multinomial Naive Bayes model. However, instead of assuming a multinomial distribution for the features, GNB assumes a Gaussian distribution. The Gaussian Naive Bayes classifier calculates the log-likelihood of the data given to each class using the Gaussian probability density function. It then combines the log-likelihood with the log prior probability to compute the log posterior probability. Finally, it assigns the class with the highest log probability as the predicted class label for each example.

Accuracy Matrices

Below are the heatmaps illustrating the comparison of accuracies between our implemented models and already built-in models.



The heatmap shows the accuracies tested on the validation set. As we can observe, all the versions of the Multinomial NB model perform better compared to the Gaussian NB models. In addition, the built-in Gaussian NB models perform better than our implemented ones. Moreover, the accuracies of our Implemented and built-in Multinomial NB are almost the same (numbers are rounded).



The accuracy results on testing sets are almost identical to the validation set. The models neither overfit nor underfit the data and perform very well. Surprisingly, in some cases, the accuracy is higher on the testing set compared to the validation set.

Conclusion

In our investigation of machine learning algorithms for fake news detection, Multinomial Naive Bayes (MNB) emerged as the clear winner over Gaussian Naive Bayes (GNB). By leveraging techniques like TF-IDF and Count Vectorizer for feature extraction, along with custom implementations of MNB and GNB, we rigorously assessed their performance on validation and test datasets. Our results consistently favored MNB, demonstrating its superior ability to categorize news articles into fake or real classes.

MNB's strength lies in its adept handling of text classification tasks, leveraging the feature independence assumption given the class label. Despite its simplicity, MNB outperformed GNB, which struggled to capture the nuances of textual data effectively. Through hyperparameter tuning and rigorous experimentation, we fine-tuned the MNB model for optimal performance, showcasing its reliability in fake news detection.

In summary, Multinomial Naive Bayes stands out as the preferred choice for fake news detection, offering a robust and efficient solution to combat misinformation in the digital era.

References

https://www.geeksforgeeks.org/multinomial-naive-bayes/

https://www.analyticsvidhya.com/blog/2022/06/stemming-vs-lemmatization-in-nlp-must-know-differences/#:~:text=Stemming%20is%20a%20process%20that,form%2C%20which%20is%20called%20Lemma

https://www.kaggle.com/datasets/bjoernjostein/fake-news-data-set

https://monkeylearn.com/blog/what-is-tf-idf/