Classifying Political Bias in News Articles: Exploring ML Algorithms, Text Processing, and Model Performance

By: Amber Swain

Advisors: Bill Breslin and Chris Lane

Spring 2025

Abstract

As the internet and digital media continue to shape public discussion, detecting political bias in news articles has become increasingly important, given that much of the information people encounter may be misleading or inaccurate. This project focuses on developing a machine learning algorithm to classify news articles as either left-leaning or right-leaning. Motivated by the need to address the spread of misinformation and its broader societal consequences, this research seeks to provide tools that help readers better assess the credibility and accuracy of the information they consume. Additionally, it explores how effectively traditional machine learning algorithms can perform a task typically handled by more complex models like deep learning neural networks. The dataset used includes approximately 18,000 articles from an AllSides news roundup published in November 2022, along with an additional 150 recent articles per political side, party platforms, and political documents such as Project 2025. Traditional machine learning models including Logistic Regression, Naïve Bayes, Gradient Boosting, and Support Vector Machines (SVM), were trained using two different text vectorization techniques: Count Vectorization and TF-IDF. Additionally, the BERT deep learning model was implemented for comparison. While BERT achieved the highest performance overall, the traditional models were not far behind in accuracy, demonstrating that simpler algorithms remain viable for bias classification tasks. Unlike websites that simply display a bias score, this project provides a comparison of different machine learning approaches, highlighting their strengths and weaknesses. The implications of this research could extend to broader efforts to detect political bias in media and promote more balanced journalism by identifying one-sided narratives.

1 Introduction

As digital media continues to grow, political bias in news has become a big concern. Tools that can detect bias can help fight misinformation, help people think more critically about what they read, and support research on how social media shapes opinions. While newer deep learning models are often used today, this project looks at how well traditional machine learning models can do at telling apart left-leaning and right-leaning articles. The main goals of this were to compare different models and test different ways of turning text into numbers.

2 Related Works

Several recent studies have explored how machine learning can be used to detect political bias in news articles. Baly et al. [1] created a large dataset of news articles labeled by political bias and trained machine learning models to predict an article's bias based on its content, rather than just the news source. Hajare et al. [3] studied political bias on social media by labeling posts based on political speeches and used machine learning to track how bias appeared in online conversations. Other work by Nadeem and Raza [4] compared different machine learning models for detecting bias in news articles, including traditional methods and newer deep learning models. They found that deep learning models performed better, but simpler models like logistic regression were still effective. These projects show that while complex models can achieve high performance, traditional machine learning approaches remain useful. Our project builds on this research by comparing traditional models such as Logistic Regression, Naïve Bayes, Gradient Boosting, and SVMs, and also testing a modern deep learning model (BERT) to better understand the trade-offs between model complexity and performance.

3 Data Description

The dataset for this project was built by combining several sources of politically relevant text. The largest portion came from the AllSides November 2022 news headline roundup, which included thousands of headlines from a variety of left-leaning and right-leaning news outlets. Additional recent articles were manually collected from sources such as Fox News and Vox to ensure coverage through 2024 and 2025.

In addition to news articles, political documents were also included in the dataset. Full text pages from Project 2025's *Mandate for Leadership* were extracted using PDF reading tools, and political party platforms, such as the 2024 GOP platform and the President's Advisory 1776 Commission report, were included to diversify the data and strengthen the classification of right-leaning material. Each document or headline was labeled either `left` or `right` according to its political leaning. In the case of the AllSides data, articles were already labeled by bias; for manually added documents, classification was based on the publisher or the source (e.g., Fox News classified as right-leaning, Washington Post as left-leaning).

Preprocessing steps included but were not limited to:

- Lowercasing all text
- Removing punctuation and special characters
- Filtering out entries with missing text
- Resetting indices after combining all sources

The final dataset contained approximately 18,400 entries, with a noticeable amount more of left-leaning articles compared to right-leaning articles due to the original AllSides data distribution.

4 Methods

4.1 Text Vectorization

The purpose of text vectorization is to convert raw text into numerical values that machine learning models can actually understand and "read" in a sense. This project tests two vectorization methods: Count Vectorization and TF-IDF (Term Frequency–Inverse Document Frequency).

When vectorizing text, we assign numbers to individual words or combinations of words (called n-grams) up to a specified length. The main difference between the two methods is how they treat word importance:

- **Count Vectorization** simply counts how many times each word or phrase appears in a document, without considering how common that word might be across the entire dataset.

- **TF-IDF Vectorization** adjusts the word counts by also considering how rare or informative a word is. Words that appear frequently across many documents (like "the" or "and") are given lower weights, while words that are more unique to specific documents are given higher importance.

The TF-IDF score for a given word is calculated as the product of two terms:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

where:

- $\text{TF}(t, d)$ (Term Frequency) is the number of times term t appears in document d.
- $\text{IDF}(t)$ (Inverse Document Frequency) measures how rare term t is across all documents:

$$\text{IDF}(t) = \log\left(\frac{N}{n_t}\right)$$

where N is the total number of documents, and n_t is the number of documents containing the term t.

Simple Example Using Count Vectorization:
Python Dictionary of Words:

```
{"free":0, "win":1, "money":2, "meeting":3, "schedule":4}
```

Sentence: `Win free money now and win prizes`
Translation using Count Vectorization: [1, 2, 1, 0, 0]

The sentence includes "free" and "money" once each, and "win" appears twice. The words "meeting" and "schedule" do not appear, so their counts are zero. With TF-IDF, these counts would be adjusted to account for how common each word is across all documents, giving more weight to rarer, more informative terms.

4.2 Traditional Machine Learning Models

The following models were trained on both types of vectorized data:

- **Logistic Regression:** A linear model that calculates a weighted sum of the input features (words) and applies a sigmoid function to predict probabilities between 0 and 1.
- **Naive Bayes:** Uses Bayes' Theorem to calculate the probability that a document belongs to a class based on the frequency of each word. It assumes all words/phrases are independent, which is why it is called "naive."
- **Gradient Boosting:** Builds a sequence of decision trees where each tree corrects the errors of the previous ones.
- **Support Vector Machines (SVM):** A linear model similar to logistic regression, except instead of predicting probabilities, it focuses on finding the best hyperplane that separates the two classes and maximizing the margin between the closest points from each class.

4.3 Deep Learning Model: BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a deep learning model designed to understand language by considering the context of words in both directions. Unlike traditional machine learning models that rely on simple vectorization techniques, BERT uses a process called tokenization, where text is broken into smaller pieces called tokens. Each token is then assigned a number based not just on the word itself, but also its meaning within the sentence. The key advantage of BERT is that it considers the full context of a word by looking at the words before

and after it simultaneously. This allows BERT to understand language in a much deeper and more nuanced way compared to traditional models that treat each word as independent.

BERT was introduced in 2019 by Devlin et al. [2] as a new method for pre-training deep bidirectional language representations. The model was trained on very large datasets, including Wikipedia (2,500M words) and the BookCorpus (800M words), using two tasks: Masked Language Modeling (predicting missing words in a sentence) and Next Sentence Prediction (determining if one sentence logically follows another). These training steps helped BERT learn not only the meaning of individual words but also how full sentences fit together, allowing it to perform well on many different tasks like answering questions, understanding sentiment, and classifying political bias.

5 Evaluation Metrics

Each model was trained on 70% of the dataset and tested on the remaining 30%. Models were evaluated based on the following metrics:

- **Accuracy:** Measures the overall proportion of correct predictions. It is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision:** Measures the proportion of predicted positive cases that were actually positive. It is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** Measures the proportion of actual positive cases that were correctly identified. It is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score:** Provides a single score that balances precision and recall, which is especially useful for imbalanced datasets. It is calculated as:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

Because the dataset was slightly imbalanced toward left-leaning articles, the F-1 score was a very useful metric in addition to accuracy. F1 balances the trade-off between precision (correctness of positive predictions) and recall (completeness of positive predictions), making it a more reliable metric than accuracy alone.

For each of the traditional machine learning models (Logistic Regression, Naive Bayes, Gradient Boosting, and SVM), a hyperparameter grid search was performed to optimize model performance. For BERT, no hyperparameter tuning was applied; the model was used with default settings due to the computational cost of it.

6 Results and Discussion

The results of this project are summarized in Figure 1. Among the traditional machine learning models, Gradient Boosting achieved the best performance based on F1 scores and overall accuracy. However, it came at a much higher computational cost, requiring significantly longer training time and more memory. Logistic Regression produced nearly equivalent results but was much faster to run and required far fewer computational resources.

BERT achieved the highest performance overall, even though no additional fine-tuning was applied. Its advantage likely comes from the fact that it was pre-trained on massive text datasets, giving it a built-in understanding of language before being applied to the classification task. However, this came at a substantial computational cost, as BERT requires more memory and processing power to run.

When comparing text vectorization methods for the traditional models, Count Vectorization and TF-IDF had very similar accuracies across all models, with no major differences. This suggests that either vectorization approach was sufficient for representing political bias language in this dataset.

Overall, while BERT achieved the highest accuracy, traditional models like Logistic Regression and Gradient Boosting showed that simpler, less resource-intensive approaches can still produce reliable and meaningful predictions well above random guessing. These results are consistent with findings from previous academic work on political bias detection, where models typically achieved accuracies between 60% and 75%, with few models exceeding 78% accuracy [1, 3, 4].

Model	Accuracy	F-1 Score Left	F-1 Score Right	Vectorization Method
Logistic Regression	0.65	0.71	0.54	Count
Logistic Regression	0.64	0.72	0.50	TF-IDF
Naive Bayes	0.60	0.70	0.39	Count
Naive Bayes	0.61	0.72	0.32	TF-IDF
Gradient Boosting	0.65	0.74	0.45	Count
Gradient Boosting	0.65	0.73	0.47	TF-IDF
Support Vector Machines	0.64	0.74	0.41	Count
Support Vector Machines	0.64	0.73	0.44	TF-IDF
BERT	0.70	0.76	0.59	Tokenization

Figure 1: Comparison of model performance based on accuracy and F1 scores.

6.1 Best and Worst Performing Models

Below are the confusion matrices for the best and worst performing traditional models. The best was Gradient Boosting using TF-IDF and the worst was Naive Bayes using Count. In the confusion matrices, each square represents a different classification outcome:

- **Top-left square**: Left-leaning articles correctly classified as left (true positives).
- **Top-right square**: Right-leaning articles incorrectly classified as left (false positives).
- **Bottom-left square**: Left-leaning articles incorrectly classified as right (false negatives).
- **Bottom-right square**: Right-leaning articles correctly classified as right (true negatives).

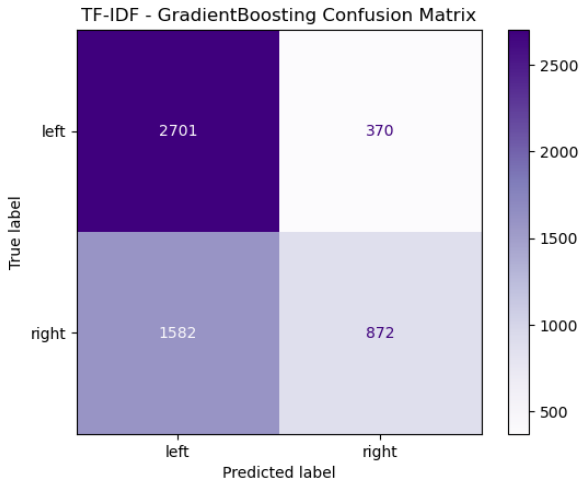

Figure 2: Confusion matrix for Gradient Boosting with TF-IDF vectorization.

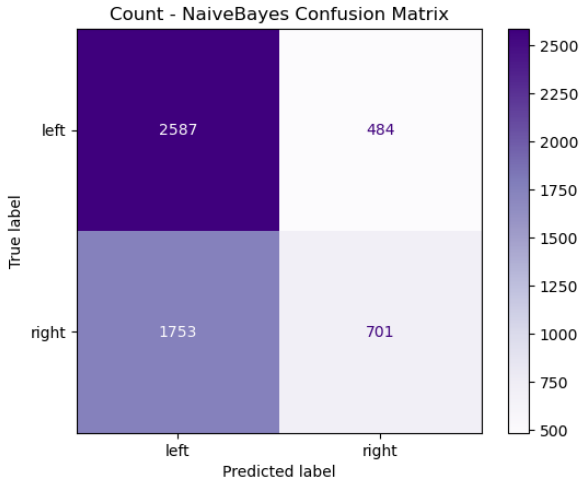

Figure 3: Confusion matrix for Naive Bayes with Count Vectorization.

6.2 Feature Importances in Gradient Boosting

The top 20 most important features used by the Gradient Boosting model are shown in Figure 4. Feature importance refers to how much a word contributed to the model's decisions. Terms such as "mandate," "leadership," and "president trump" had the highest importance scores, suggesting they were especially informative for distinguishing between left-leaning and right-leaning articles. Some redundancy in the important words (e.g., "mandate" appearing twice) reflects different n-gram patterns or word contexts captured by the model.

Interestingly, although the model was allowed to consider phrases up to three words long, nearly all of the top features selected were single words. This suggests that individual keywords carried most of the distinguishing information, and that longer multi-word phrases were either too rare or too specific to consistently influence model decisions. Certain features such as "fox" and "cnn" likely appeared because the web scraping process pulled not just the article text, but also metadata or page elements that included the names of news sources.

Finally, some less expected words like "com", "2023", and "act" were also ranked as important. Although these words seem random at first glance, they may have represented underlying patterns in the dataset, such as references to legislation ("act"), web addresses ("com"), or the timing of political discussions ("2023") that indirectly showed political leaning.

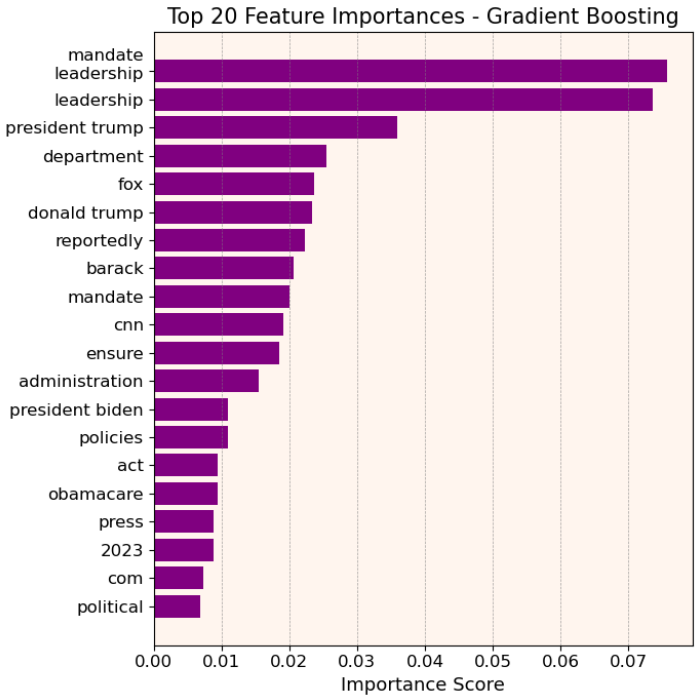

Figure 4: Top 20 most important features for Gradient Boosting.

7 Conclusion

This project developed and compared machine learning models to classify news articles as left-leaning or right-leaning. The traditional models used consisted of Logistic Regression, Naive Bayes, Gradient Boosting, and SVM, and each one used with both Count Vectorization and TF-IDF. Gradient Boosting achieved the best results among traditional models, though Logistic Regression offered similar accuracy with greater efficiency. BERT achieved the best results because of its pre-existing knowledge it came in with but was more costly. These findings align with previous research, showing that traditional models remain effective for bias detection tasks despite the advantages of newer deep learning models. Improving political bias detection tools can contribute to reducing misinformation and encouraging more balanced media consumption.

References

[1] Ramy Baly, Carlos Castillo, Firoj Alam, Georgi Karadzhov, Preslav Nakov, and Yifan Zhang. We can detect your bias: Predicting the political ideology of news articles. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4358–4372, 2020.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186, 2019.

[3] Tanay Hajare, Brandon Butler, and Dongwon Lee. A machine learning pipeline to examine political bias with congressional speeches. In *Proceedings of the 15th International AAAI Conference on Web and Social Media (ICWSM)*, pages 123–134, 2021.

[4] Mohammad Nadeem and Ali Raza. Detecting bias in news articles using nlp models. In *Proceedings of the 2022 International Conference on Frontiers of Artificial Intelligence and Machine Learning (FAIML)*, pages 89–95, 2022.