News Bias Classification Using Machine Learning

By: Amber Swain Advisors: Bill Breslin and Chris Lane

Pacific University Oregon

Objective

Develop a machine learning algorithm to classify news articles as either left-leaning or right-leaning. The dataset includes about 18,000 articles from an AllSides news headline roundup published in November 2022 [2]. An additional 150 recent articles per political side were added, along with party platforms and similar political documents (ex. Project 2025). For the traditional models, the text is vectorized using two different methods, and each is used to train four different machine learning models. This project was inspired by a desire to address the spread of misinformation and to explore how well traditional machine learning algorithms can perform a task typically handled by more complex models like neural networks.

Key Questions:

- What words or phrases are the most helpful in differentiating left from right?
- Is there a difference in accuracy between each of the text vectorization methods and/or models?

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 any text, we assign numbers to single words or phrases of multiple words up to a specified length. The difference between the Count and TF-IDF methods is that Count tallies how often each word or phrase appears, while TF-IDF also considers how common or rare a word is across the entire dataset. TF-IDF gives more weight to words that are potentially more helpful while reducing the influence of common ones like "the" or "and".

Simple Example:

Python Dictionary of Words:

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

Sentence: *Win free money now and win prizes*

Translation: [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.

The Traditional Models

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.

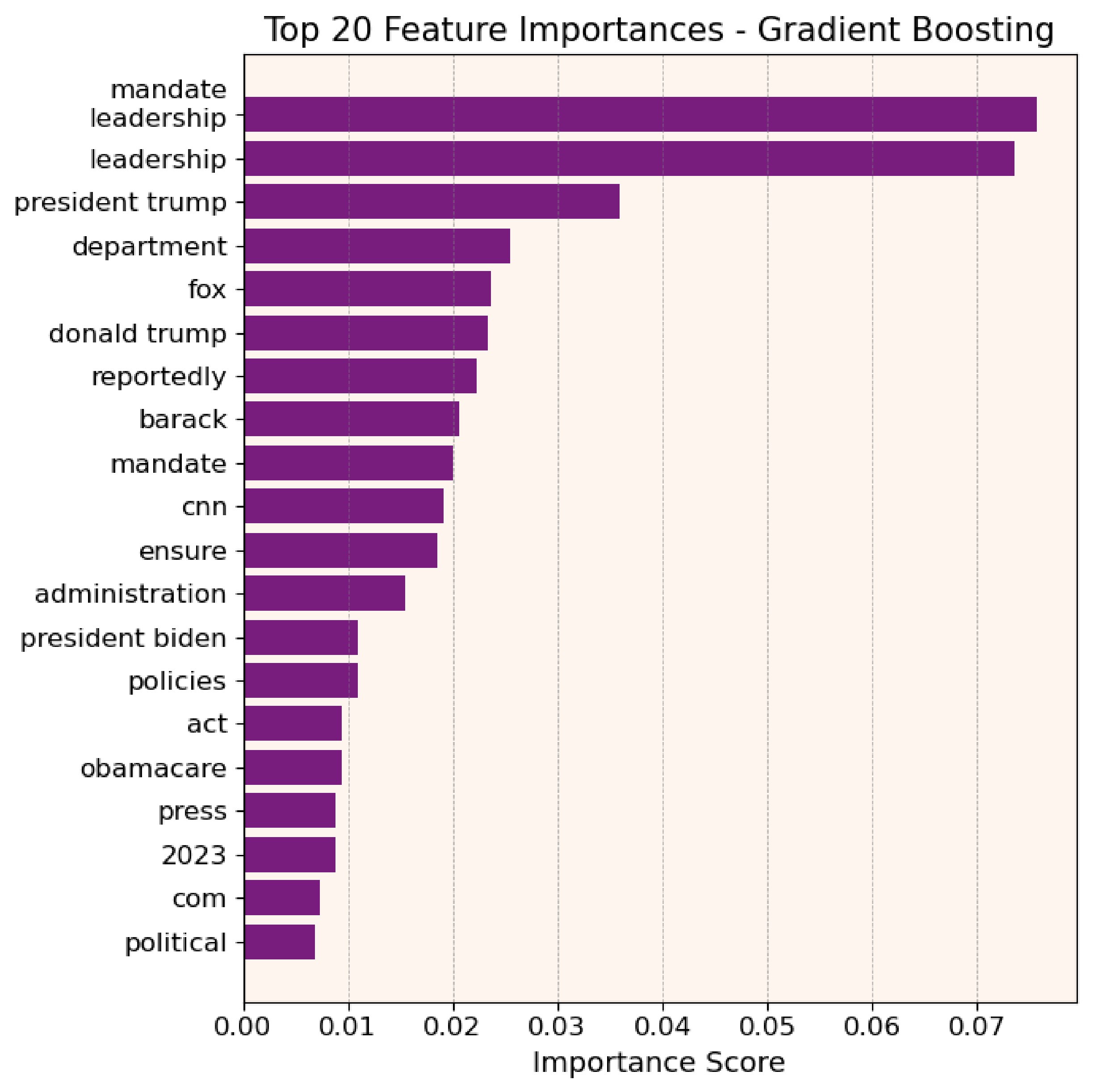
Support Vector Machines (SVM): A linear model similar to logistic regression, except instead of predicting probabilities, it focuses on finding the best hyperplane that separates the two classes and maximizing the margin between the closest points from each class.

Results

Model	Accuracy	F-1 Score Left	F-1 Score Right	Best Vectorization Method
Logistic Regression	0.65	0.71	0.54	Count
Naive Bayes	0.61	0.72	0.32	TF-IDF
Gradient Boosting	0.65	0.74	0.48	TF-IDF
Support Vector Machines	0.64	0.73	0.44	TF-IDF
BERT	0.70	0.76	0.59	Tokenization

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

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$



The F_1 score balances precision and recall, where precision is the proportion of predicted positive cases that are actually positive, and recall is the proportion of actual positive cases that were correctly identified. Thus, F_1 gives a single metric that accounts for both false positives(FP) and false negatives(FN).

All of the models were better at predicting the left-leaning articles because the dataset somewhat imbalanced towards the left.

BERT stands for Bidirectional Encoder Representations from Transformers. It's a deep learning model that understands language by looking at words in both directions. Instead of using simple vectorization, it uses tokenization, which breaks text into smaller pieces called tokens and assigns each one a number based on its meaning. Unlike the traditional models used in this project, BERT considers all the words in a sentence, both before and after, so each token is understood in full context. This model comes pre-trained, so it performed the best out of all the models in this project likely because of its pre-existing understanding of language.

Conclusion

Out of the traditional models, Gradient Boosting had the best results, but it took a lot longer to run and used more memory. Logistic Regression had almost the same performance but was much faster and had far lower computational cost. The Count vs TF-IDF Vectorizations did not have much difference in accuracy. BERT outperformed the other models due to its pre-trained understanding of language, but is also very costly. The simpler models were still able to produce reliable predictions better than randomly guessing. The results from this project are in line with what others have achieved in similar academic work [1]. The comparable projects we looked at also saw accuracy scores in the 60–75% range, with few models surpassing 78%.

[1] Baly, et. al. We can detect your bias: Predicting the political ideology of news articles. *Proceedings of the 2020 Conference on EMNLP*, pages 4982–4991, 2020. Association for Computational Linguistics.

[2] Haak, Fabian et. al. Qbias - A Dataset on Media Bias and Search Queries and Query Suggestions. *Proceedings of the 15th ACM Web Science Conference*. pages 239-244, 2023. Association for Computing Machinery.