Automated Fact-Checking Prediction Using Machine Learning

Introduction
In an era dominated by online information, the spread of misinformation has become a pressing social concern. News travels faster than ever before, and with it, false or misleading claims can influence public opinion and policy decisions. To address this issue, the present project explores the development of an automated fact-checking system using machine learning. The system is trained on a dataset of public claims that have already been labeled by professional fact-checkers at PolitiFact.com, assigning each claim to one of six categories: True, Mostly True, Half True, Mostly False, False, and Pants on Fire.

The core idea of this project is to evaluate different machine learning models and determine which is most effective at predicting the credibility labels assigned by PolitiFact. The statements were scraped using BeautifulSoup, along with associated metadata such as the source of the claim and the date of publication. The resulting dataset contains not only the text of each claim but also structured contextual features that help inform the model. These claims span a wide range of topics—including politics, health, social issues, and viral content—making the dataset a diverse and realistic foundation for training automated fact-checking models.

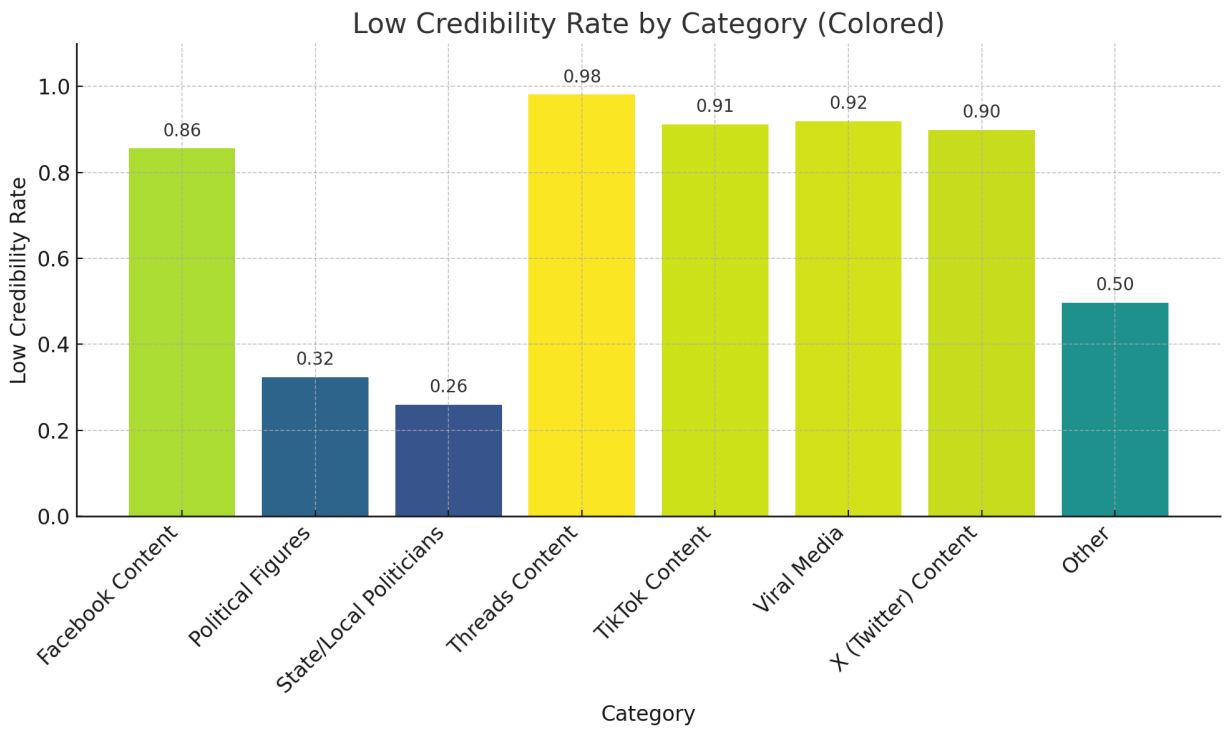
More specifically, the model is trained to predict the fact-check label based on three key inputs:
(1) the **statement** itself,
(2) the **source** of the claim (e.g., Instagram, X/Twitter, a public figure),
and (3) the **date** the statement was made.
By learning from this combination of textual and contextual information, the model attempts to replicate the fact-checking decisions made by experts—enabling automated credibility assessment for new, unseen claims.

Data Preprocessing
Preprocessing was a critical step in the pipeline. Each statement was normalized through lowercasing and the removal of punctuation and special characters. Standard English stopwords were filtered out using scikit-learn's `CountVectorizer`with the `stop_words='english'` parameter. However, it quickly became apparent that some frequent words, such as "say," "said," and "says," were not included in the default stopword list and were appearing consistently across all truthfulness categories. While such words may carry weak predictive power, their prevalence often drowns out more meaningful distinctions between classes. To improve interpretability and reduce noise, we manually extended the stopword list to exclude these terms. This allowed the model to focus on words with clearer associations to specific truth labels.

To better understand the sources associated with misinformation, we calculated the Low Credibility Rate for each content category. This rate represents the proportion of statements rated as False, Full Flop, or Pants on Fire by PolitiFact.com. The results showed that Threads Content, TikTok Content, and Viral Media had extremely high low credibility rates (above 90%), suggesting that most fact-checked statements from these sources were judged as misleading or false. Facebook Content also ranked high with a low credibility rate of 85.6%, aligning with concerns about misinformation on the platform.

X (Twitter) Content showed a high low credibility rate of 89.9%, supporting concerns over relaxed moderation and algorithmic amplification of controversial posts. Political Figures and State/Local Politicians had lower low credibility rates (32.4% and 26.0%, respectively), indicating more varied truthfulness and higher scrutiny. The "Other" category displayed a moderate rate of 49.7%, capturing sources not covered in primary categories. These findings highlight how social media and viral content platforms are frequently associated with misinformation. They also suggest that source category is a powerful predictive signal for truthfulness classification and should be carefully considered in model development.

Based on this insight, we incorporated the source category into the feature set using **one-hot encoding**. Each content type was converted into a binary indicator feature, enabling the model to detect patterns linked to specific sources without assuming any ordinal relationship. Including these encoded source features not only improved model performance but also enhanced interpretability by explicitly modeling the influence of content origin on credibility classification.



Additionally, we extracted structured features from the **publication date** of each statement. These included the year, month, and day of the week, allowing the model to capture temporal patterns such as election cycles, policy events, or changes in platform moderation. Since dates are inherently structured and numeric, these derived features were directly usable by the model without further transformation. Incorporating temporal data provided valuable context for when misinformation spikes may occur, thereby strengthening the model's predictive capabilities.

Model Training and Tuning

A variety of machine learning models were trained and evaluated in this project, including **Multinomial Naive Bayes, Logistic Regression, Random Forest, Support Vector Machine (SVM), and XGBoost**. These models were selected based on their popularity and effectiveness in text classification tasks, as well as their trade-offs in interpretability, scalability, and performance.

Each model was embedded into a **scikit-learn pipeline** and optimized using **GridSearchCV** to systematically search for the best hyperparameter combinations. For example, we tuned parameters such as the regularization strength `C` in SVM, `max_depth` in XGBoost, and the smoothing parameter `alpha` in Naive Bayes. Grid search with cross-validation allowed us to evaluate model performance more reliably and mitigate overfitting by averaging results across different data folds.

Performance was evaluated using both **classification accuracy** and **macro-averaged AUC (Area Under the Curve)**. Macro AUC was chosen to account for the multi-class nature of the problem and to ensure balanced performance across all six credibility categories, including underrepresented ones like "Full Flop" or "Pants on Fire."

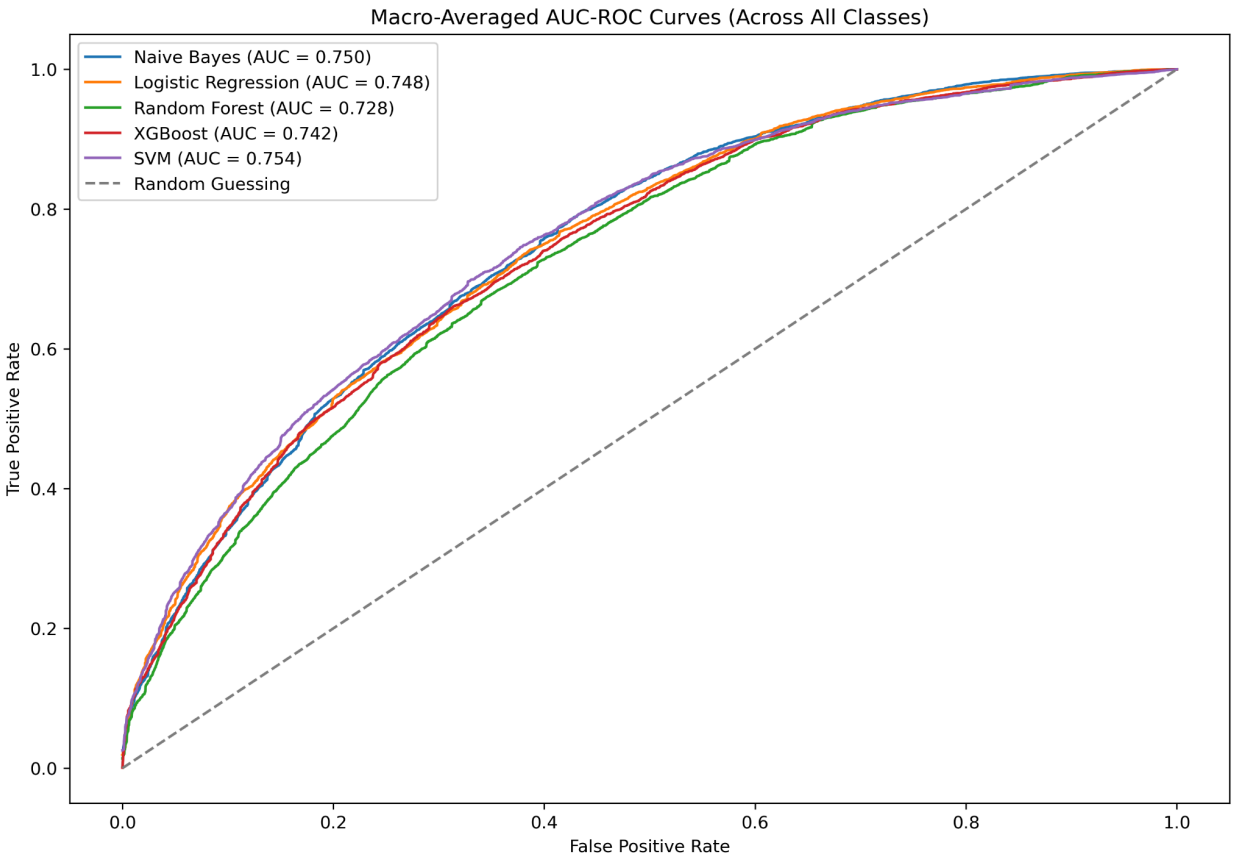
This approach enabled a fair comparison between models, allowing us to select the best-performing algorithm not only in terms of predictive accuracy but also interpretability and robustness across diverse claim types.

Model Comparison and Results

Among the models tested, **Logistic Regression emerged as the best overall choice**, offering a compelling balance between performance, simplicity, and interpretability. While **Naive Bayes** and **SVM** achieved slightly higher AUC scores, Logistic Regression demonstrated **greater consistency across validation folds**, as well as clear interpretability through its feature coefficients.

The table below summarizes the key evaluation metrics across models:

- **Naive Bayes**: Accuracy = 0.3920, Macro AUC = 0.7496
- **Logistic Regression**: Accuracy = 0.3930, Macro AUC = 0.7485
- **Random Forest**: Accuracy = 0.3420, Macro AUC = 0.7284
- **XGBoost**: Accuracy = 0.3820, Macro AUC = 0.7417
- **SVM**: Accuracy = 0.3827, Macro AUC = 0.7539



Although the differences in performance were subtle, Logistic Regression was ultimately selected for deployment because it offered several practical advantages. Most notably, it provided strong interpretability, as its feature weights could be easily analyzed to understand which inputs were most influential in shaping predictions. Additionally, Logistic Regression delivered fast inference times, making it well-suited for real-time applications. Its performance across cross-validation folds was also stable and consistent, which added to its reliability as a production-ready solution.

The final model was deployed using **FastAPI** and hosted on **Google Cloud Run**. It is now available as a web-based interface where users can input statements and receive automated fact-checking predictions:

https://fastapi-app-273008876300.us-central1.run.app/

Predictive Word Analysis

To provide insight into what the model learned, we visualized the most predictive words for each class. By calculating the smoothed conditional probabilities P(word | class, α), we were able to generate visualizations (including heatmaps and bar charts) that highlight which words were most influential for each truth label. These probabilities were computed using a smoothing parameter α = 0.01 to ensure robust estimation even for less frequent terms.

To calculate these probabilities, we applied the following version of Bayes' Theorem with Laplace smoothing:

$$P(word_i \mid class_j, \alpha) = (count(word_i \text{ in } class_j) + \alpha) / (\sum count(word_k \text{ in } class_j) + \alpha * V)$$

Where:
- count(word_i in class_j) is the number of times word i appears in class j,
- V is the total number of unique words (the vocabulary size),
- α is the smoothing hyperparameter.

This approach helps avoid assigning zero probability to rare or unseen words and leads to more stable and interpretable probability estimates across all classes.

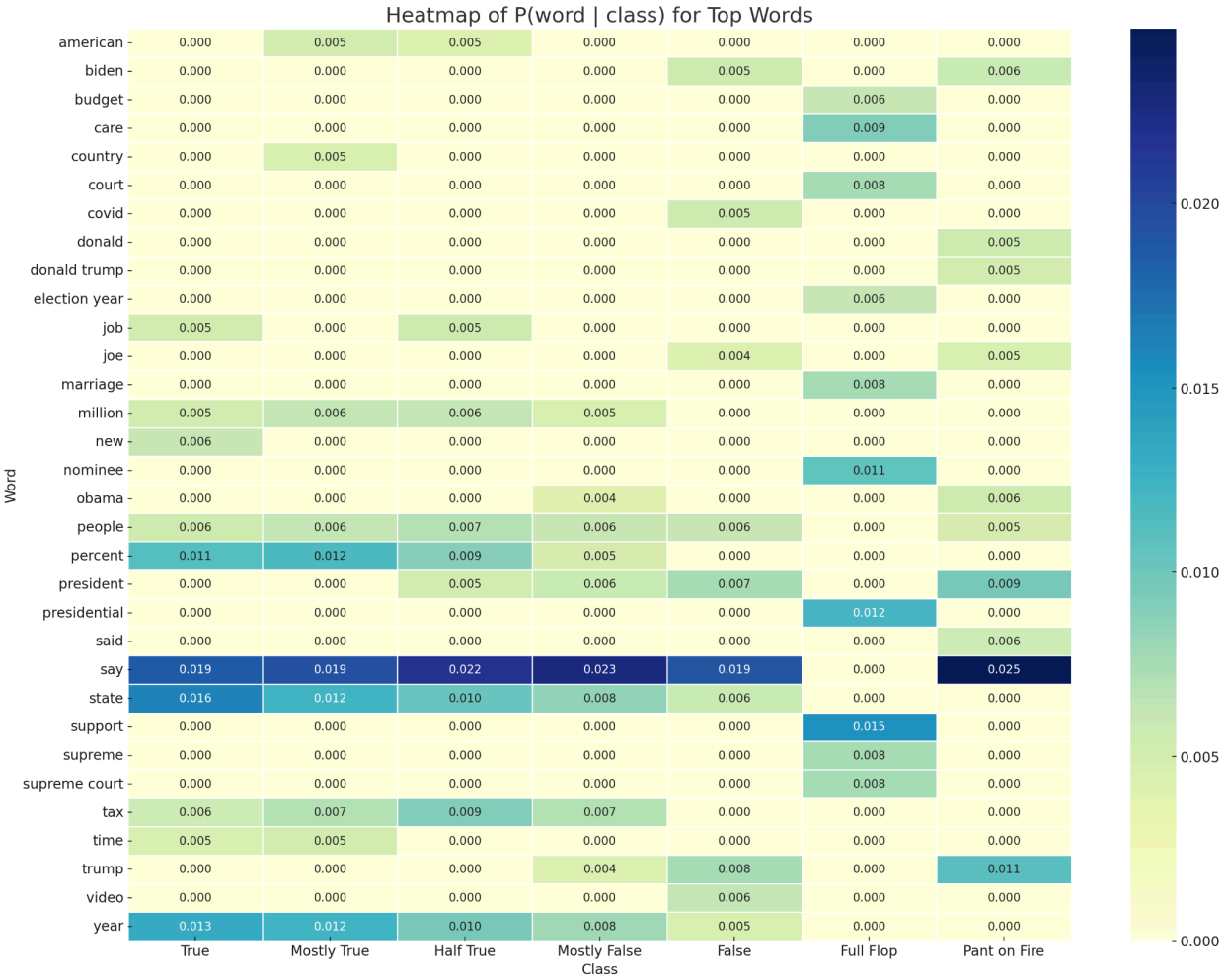
For example, in the "Pants on Fire" class, the word “trump” had a probability of 0.011, while “biden” had a probability of 0.006. This indicates that “trump” was nearly twice as predictive for the Pants on Fire class compared to “biden.” Likewise, the word “say” had an even higher probability of 0.025, suggesting that many disputed claims contained a quoted or paraphrased assertion.

In contrast, for the "True" class, words such as “state” (0.016), “year” (0.013), and “percent” (0.011) were more prominent. These terms are often associated with statistical or factual reporting, suggesting that claims grounded in measurable data were more likely to be labeled as true. Even the word “trump” appeared in the True class, but with a much lower probability (0.0037) compared to its weight in the Pants on Fire class, reflecting nuanced use across categories.

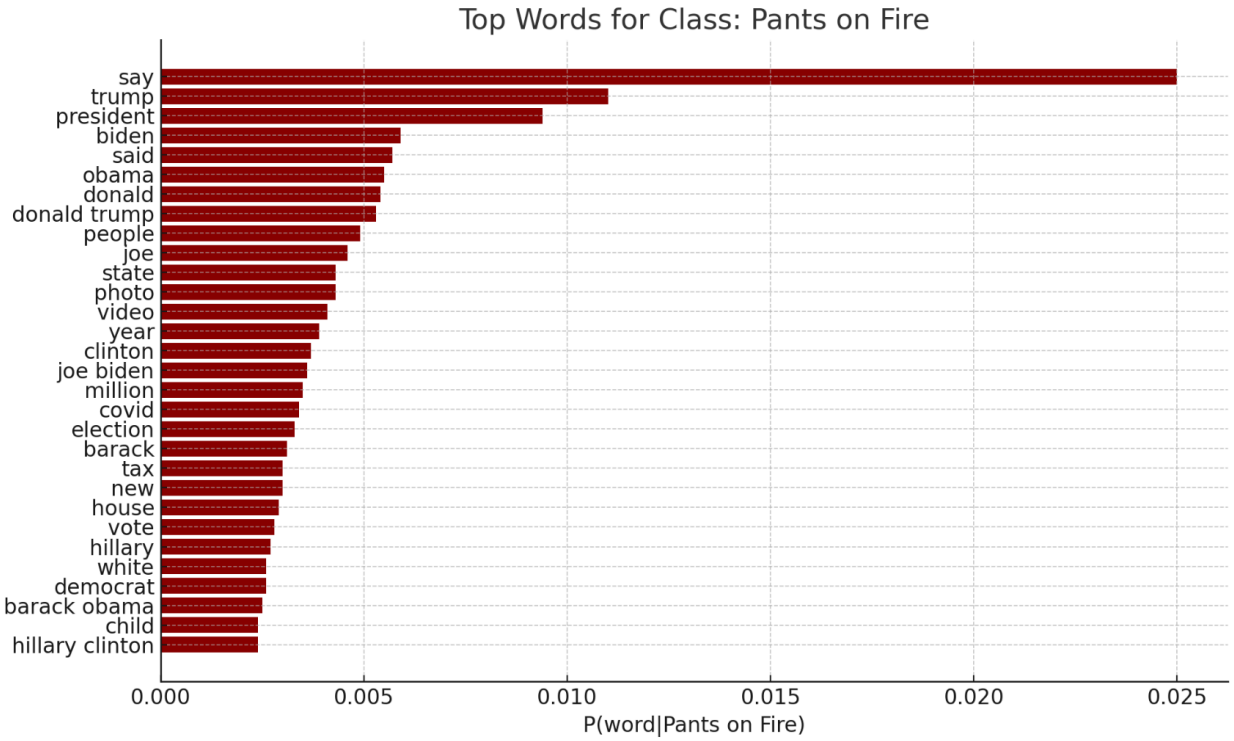
Bar charts further illustrate these differences, revealing how thematic content (e.g., "covid", "election", "vote") and named entities (e.g., "donald trump", "joe biden", "hillary clinton") contribute to the classification process. Notably, while some names appeared across multiple classes, their relative weights differed significantly, providing insight into how the model distinguishes contextually similar terms based on usage patterns.

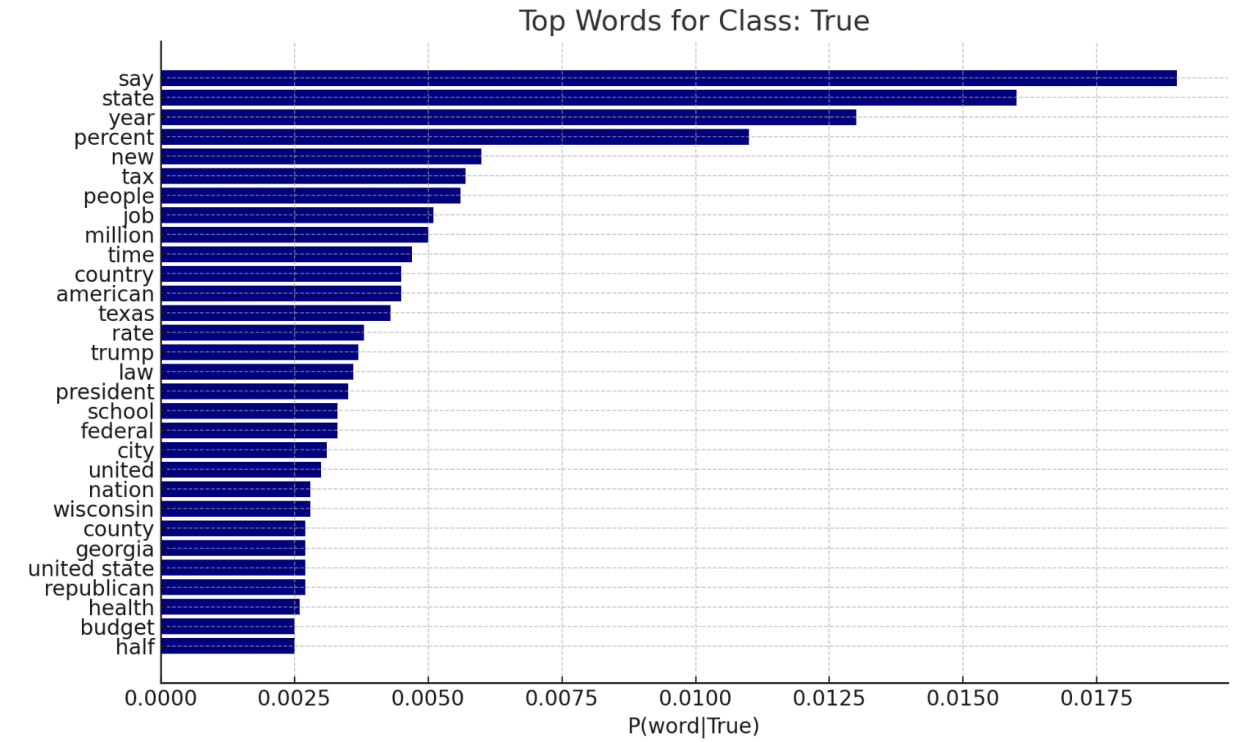
It’s important to emphasize that these probabilities do not imply that these individuals or terms are inherently associated with truth or falsehood. Rather, the model has learned correlations based on observed data—specifically, how often and in what contexts certain words appeared in statements labeled with different degrees of factual accuracy. These predictive patterns reflect both the linguistic framing and political discourse embedded in the dataset.

In essence, this analysis reveals how word-level cues serve as thematic signals, helping the model differentiate between factual and misleading claims. It underscores the importance of interpretability in understanding how machine learning systems learn from—and potentially amplify—patterns in real-world data.


Heatmap of P(word | class) for Top Words
Word
american 0.000 0.005 0.005 0.000 0.000 0.000 0.000
biden 0.000 0.000 0.000 0.000 0.005 0.000 0.006
budget 0.000 0.000 0.000 0.000 0.000 0.006 0.000
care 0.000 0.000 0.000 0.000 0.000 0.009 0.000
country 0.000 0.005 0.000 0.000 0.000 0.000 0.000
court 0.000 0.000 0.000 0.000 0.000 0.008 0.000
covid 0.000 0.000 0.000 0.000 0.005 0.000 0.000
donald 0.000 0.000 0.000 0.000 0.000 0.000 0.005
donald trump 0.000 0.000 0.000 0.000 0.000 0.000 0.005
election year 0.000 0.000 0.000 0.000 0.000 0.006 0.000
job 0.005 0.000 0.005 0.000 0.000 0.000 0.000
joe 0.000 0.000 0.000 0.000 0.004 0.000 0.005
marriage 0.000 0.000 0.000 0.000 0.000 0.008 0.000
million 0.005 0.006 0.006 0.005 0.000 0.000 0.000
new 0.006 0.000 0.000 0.000 0.000 0.000 0.000
nominee 0.000 0.000 0.000 0.000 0.000 0.011 0.000
obama 0.000 0.000 0.000 0.004 0.000 0.000 0.006
people 0.006 0.006 0.007 0.006 0.006 0.000 0.005
percent 0.011 0.012 0.009 0.005 0.000 0.000 0.000
president 0.000 0.000 0.005 0.006 0.007 0.000 0.009
presidential 0.000 0.000 0.000 0.000 0.000 0.012 0.000
said 0.000 0.000 0.000 0.000 0.000 0.000 0.006
say 0.019 0.019 0.022 0.023 0.019 0.000 0.025
state 0.016 0.012 0.010 0.008 0.006 0.000 0.000
support 0.000 0.000 0.000 0.000 0.000 0.015 0.000
supreme 0.000 0.000 0.000 0.000 0.000 0.008 0.000
supreme court 0.000 0.000 0.000 0.000 0.000 0.008 0.000
tax 0.006 0.007 0.009 0.007 0.000 0.000 0.000
time 0.005 0.005 0.000 0.000 0.000 0.000 0.000
trump 0.000 0.000 0.000 0.004 0.008 0.000 0.011
video 0.000 0.000 0.000 0.000 0.006 0.000 0.000
year 0.013 0.012 0.010 0.008 0.005 0.000 0.000
True
Mostly True
Half True
Mostly False
False
Full Flop
Pant on Fire
Class
0.020
0.015
0.010
0.005
0.000

Even though this section focuses primarily on the “Pants on Fire” and “True” classes for illustration, the same analysis





was applied to **all six fact-checking categories**, allowing the model to detect distinctive linguistic patterns associated with each level of credibility. Each class revealed its own set of indicative terms, further highlighting the model's ability to capture nuanced variations in how statements are framed across the truthfulness spectrum.

Challenges and Future Work

Future work may involve a more thorough analysis of model limitations and failure modes. For example, evaluating confusion between closely related classes—such as "Half True" and "Mostly False"—could help identify where the model tends to struggle. Additionally, since the dataset labels are assigned by human fact-checkers, there may be inherent bias in the source labels. Future versions of this system should include mechanisms for auditing or addressing potential biases to ensure fair and balanced predictions.

Another key challenge emerged during data preprocessing—particularly when categorizing sources. While major media platforms such as TikTok, Facebook, and Threads were easy to assign based on keywords, it was far more difficult to categorize statements made by individuals who were not immediately recognizable. As a result, many entries were grouped under the 'Other' category, even though they may have come from public figures. This limited the model's ability to learn from source-related credibility patterns. Future improvements could involve using named entity recognition (NER) with tools like SpaCy, or leveraging pretrained transformer models to automatically identify and classify these ambiguous sources more accurately.

Conclusion

This project involved developing and deploying an end-to-end machine learning pipeline for fact-checking using labeled claims from PolitiFact.com. From data scraping and preprocessing to feature engineering, model evaluation, and deployment, the entire process was implemented with the goal of predicting the truthfulness of public statements. The final system integrates both textual and contextual features—such as the content of the statement, its source, and the date of publication—demonstrating how credibility can be influenced by both language and metadata.

Logistic Regression was selected as the final model due to its balance between accuracy and interpretability. The system was deployed using FastAPI and made accessible via a web interface, allowing users to input claims and receive immediate predictions.

Future enhancements include automating the data collection pipeline so that the model remains current as new fact-checked statements are published. Integrating this automation into the existing web application would enable continuous updates and re-deployment of the model without manual intervention.

Ultimately, this project has shown me how data science can be used to address pressing societal challenges like misinformation. As the public continues to navigate an increasingly complex information ecosystem, I believe systems like these—grounded in transparency, up-to-date data, and interpretability—can offer valuable support to journalists, researchers, and concerned citizens alike.