Misinformation Detector

Andrew Herman and Thomas McLinden

CMPSC 445

Janghoon Yang

December 7th, 2025

Description of the Project:

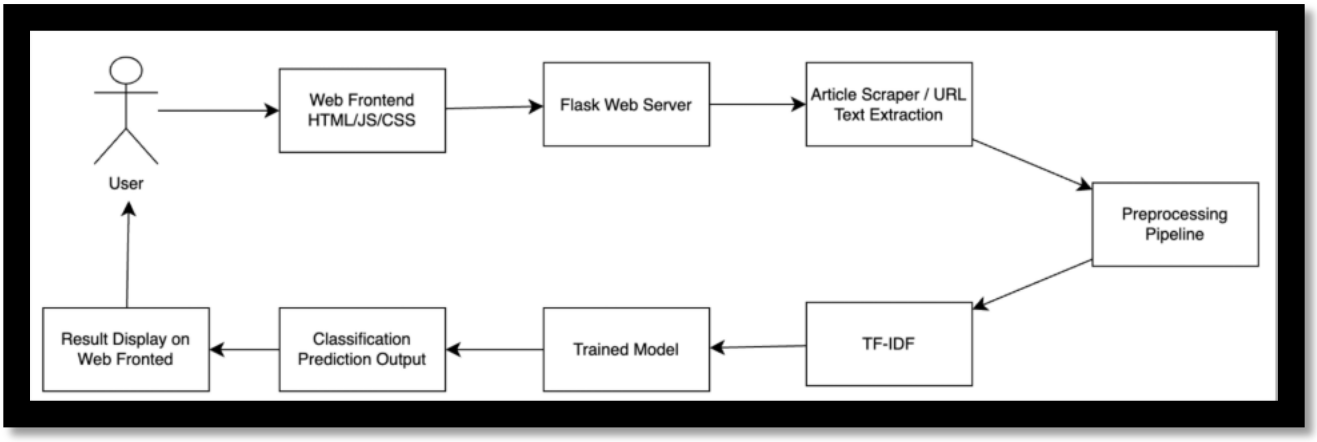
Misinformation is defined as false and inaccurate information. These false claims can be spread across the internet faster than ever, commonly without intent to deceive. Detecting false information can be difficult when the content seems realistic upon first glance. The more misinformation that is spread online, often unknowingly, it's more likely to be believed by some readers. To combat the spread of misinformation, our web app was designed to detect and classify which websites hold credible information. The user can insert any valid URL, in which our model will analyze its content, classify its credibility, and present the user with its final prediction. This application is useful to anybody who needs to verify credibility in news articles or online content they read, such as student research papers. It is vital that the public can identify credible sources, which will prevent misinformation, scams, and defamation.

Significance of the Project:

This project was developed to slow down the rate of misinformation spread on the internet. People of all ages are susceptible to believing false information, with older generations being more likely to believe in misinformation. By using our model, the number of scams, deceptive media, and false narratives will decrease.

Our model implements the use of Natural Language Processing (NLP) to analyze content of online sources. The content flows through a full machine learning pipeline to preprocess and produce real-time predictions based on the user's input. What makes our project unique is the live feedback from any given valid URL, redisplaying the extracted text, and producing an output in one tool. A user has access to their test data and our models' predictions on the same web page, providing them with the necessary detection.

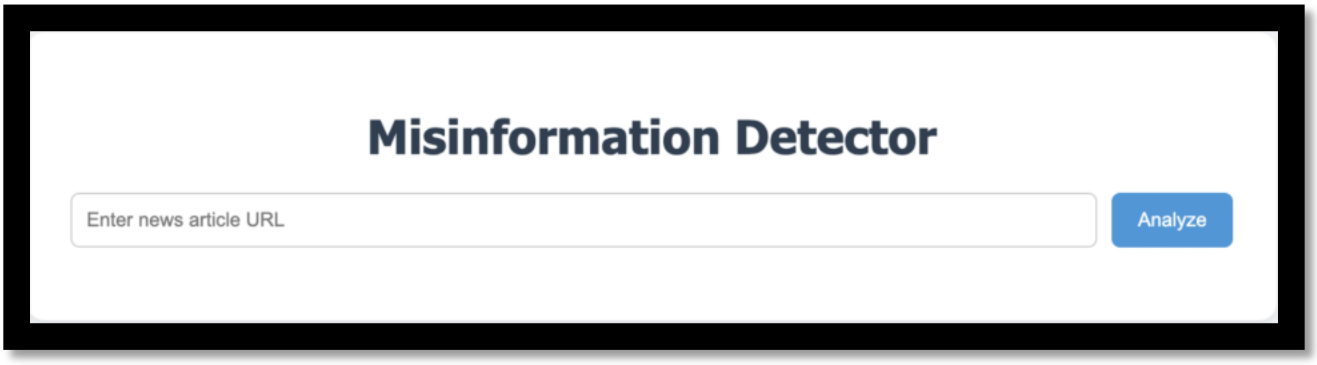
Code Structure:

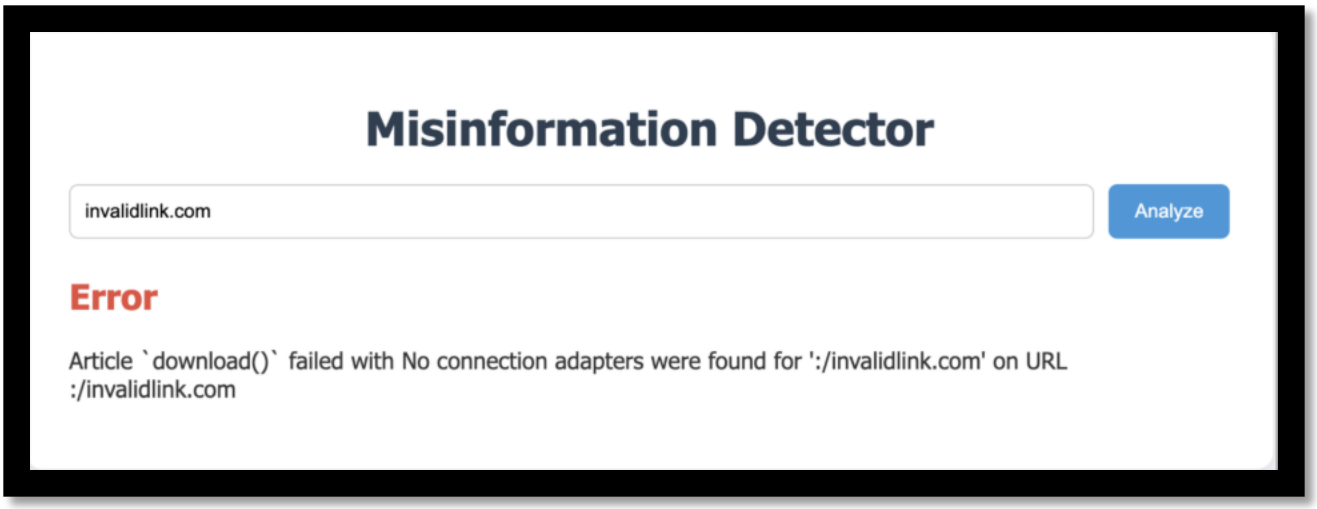


1. The user inserts a website or article URL into our Web Frontend.
2. Text is extracted from the article.
3. Preprocessing Pipeline to clean up text.
4. Term Frequency-Inverse Document Frequency to give words numeric values.
5. Content inserted into trained model for classification.
6. Final decision is made and model report created.
7. Results are output back to the user.

Functionalities and Test Results:

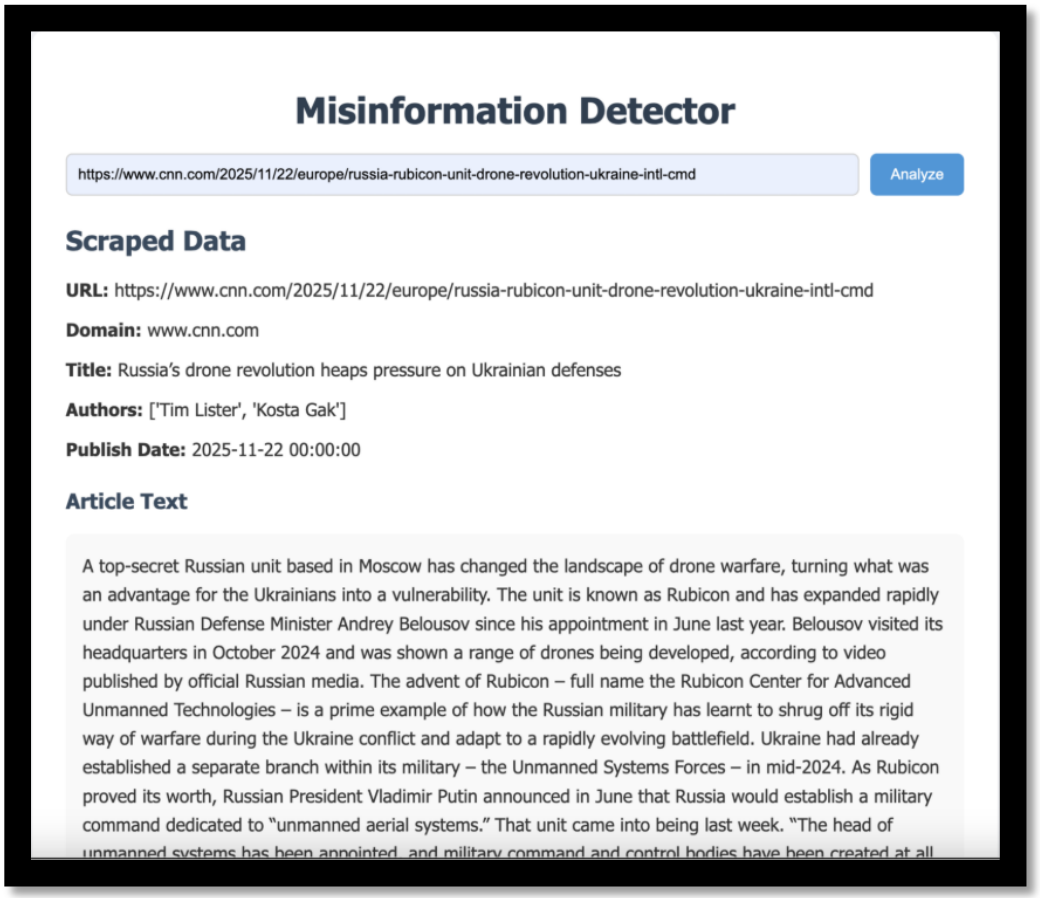
URL Submission: The text-box input allows the user to insert the URL. Invalid links will return and display an error message; valid links will be given to the system.





Content Preprocessing: Extracted text is cleaned by making all characters lower case and removing punctuation and stop words. Then, it is converted into machine readable values.

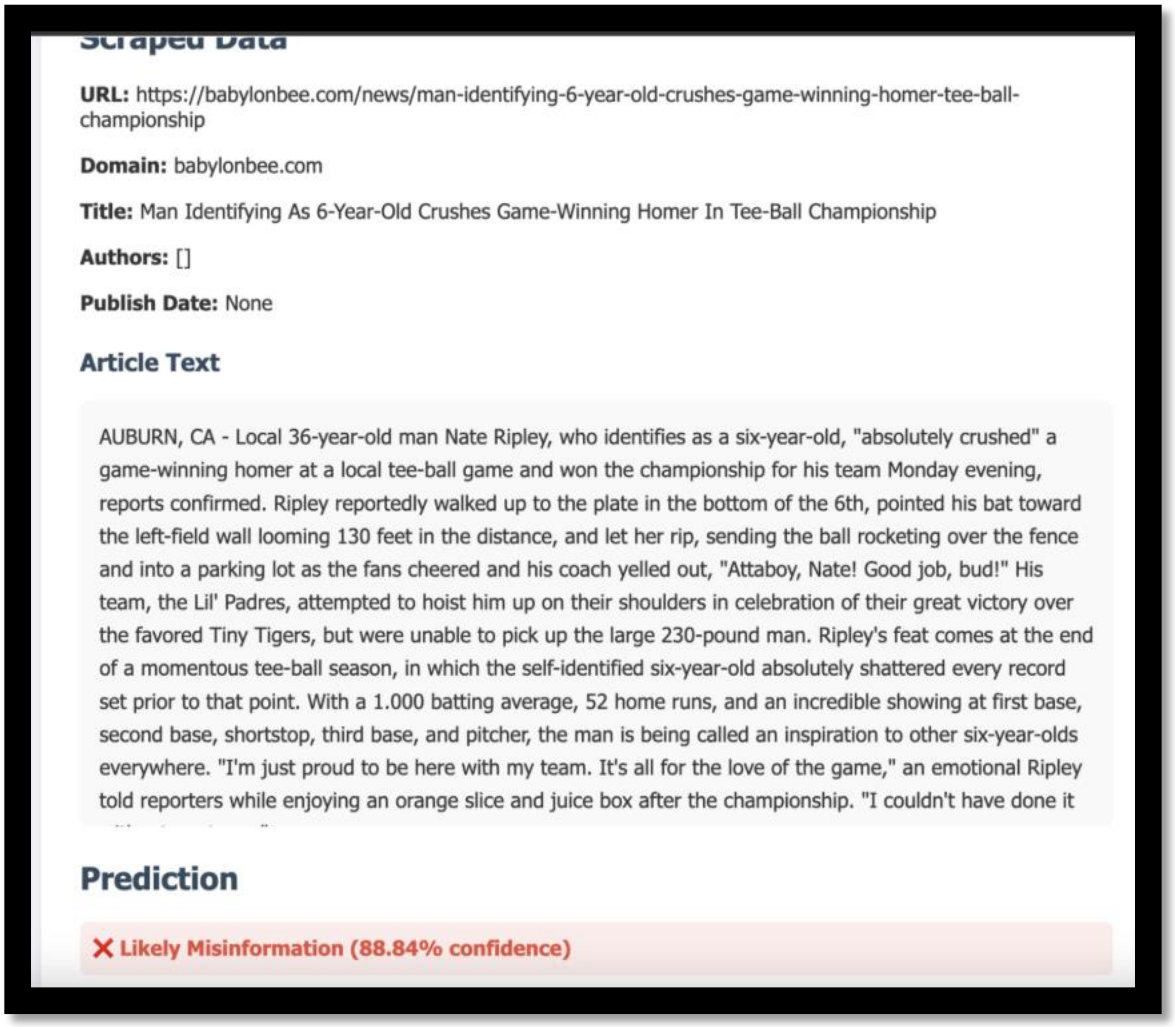
Data Confirmation: Article information like domain, title, authors, and content are displayed for confirmation that the correct data was scraped.



Classification and Visualization: Display of the model's prediction and output directly on web interface

- Credible Source (CNN article)

- Satire Article (Babylon Bee)



The system functionality was tested and verified by inserting multiple links, from both credible and non-credible sources. The final prediction given sources like CNN and 6ABC

correctly identified they were valid, while satire websites like The Onion and Babylon Bee were classified as misinformation.

Data Collection:

Raw Datasets:

- LIAR-Dataset - Sourced from Kaggle, the LIAR-Dataset stores meta-data such as statements, subjects, speakers, and a label to identify the level of misinformation. There are three tables, "train," "test," and "valid," with the training table containing over 10,000 rows. This table is meaningful for training a model so it can learn and identify which statements and words lead to misinformation.
- Fake News Detection Dataset - Also sourced from Kaggle, the Fake News Detection Dataset contains real and fake news from numerous political topics, such as World News, US News, and partisan left and right-leaning news. The meta-data includes title, text, and subject. The real news table contains over 21,000 articles and the fake news tables contains over 23,000 articles. These tables are useful for training our model because it's also vital to provide the model with credible data to be able to classify the difference.

Data Processing:

Once all the data was gathered, the final step was to merge the tables into one. The most important columns from both tables were *Title, Text, Domain,* and *"Has Information."* Since the LIAR dataset was a spectrum of six different truth levels ranging from "Pants on Fire" to "True." The three levels closest to "Pants on Fire" were given a binary value of 1 into the merged "Has Information" column. The Fake News Detection Dataset was already split into two tables of True and False, merging defaulted to 0 or 1 depending on the table it was sourced from.

To clean the data, four preprocessing steps were performed to ensure the model will train as optimally and consistently as possible. First, any null values in the table were converted into an empty string to avoid any None-value errors when calling the next operations. The second step was detecting and removing any emoticons and punctuation to prevent noise and our models' learning patterns. Next, all characters were concerted to lower case to prevent duplicates of the same word but with different capitalization. Lastly, stop words were removed from statements to eliminate words with little meaning that would slow down efficiency.

To prepare the cleaned data for model development, train-test-split and Term Frequency-Inverse Document Frequency (TF-IDF) were applied for strong model predictions. We used an 80/20 split with stratified sampling to preserve distribution from both sets. The text data was finally converted into meaningful numeric values using TF-IDF. Although stop words were already removed, we included the ‘stop_words’ parameter to confirm that only optimal words were kept. The vocabulary limit was set to 15,000 words, which kept enough informative words without running into a Curse of Dimensionality issue. The N-gram value was bounded between one and three words, allowing short phrases to be considered when determining the relevance of a word. Lastly, the minimum document frequency threshold was set to two, which filtered out rare tokens that provided our model with little value. After TF-IDF, a simple chi-squared calculation was applied to find dependencies between features and the target variable. No features were removed, but the top two features had high statistical scores, helping the model better generalize data. This full pipeline converted raw text into meaningful numerical values that the model can understand to make realistic predictions.

Model Development:

The model inputs are the TF-IDF features applied to the articles' title and content, represented by the 15,000 unigrams, bigrams, and trigrams. The target output is the binary labels classifying if the row has misinformation or not. The output is a binary value, with 0 representing a credible source and 1 indicating the model detected misinformation.

To develop the model, we built a full voting classifier ensemble with four different estimators. The estimators were a Logistical Regression, MultinomialNB, Support Vector Machine, and a SGDClassifier. The ensemble uses soft voting to average the predicted probabilities and select the class with the highest average probability.

After building the model, a classification report was built to determine its overall strength. The test dataset of 11,028 samples achieved an accuracy of 91%. There is a good balance between True and False classes. True achieved a precision value of 90%, recall value of 92%, and F1-score of 91%, while False achieved a precision of 92%, recall of 90%, and F1-score of 91%. The confusion matrix shows a total of 4,984 true negatives and 5,061 true positives, indicating a strong performance after analyzing the 11,000 unseen samples.

Classification Report:

	precision	recall	f1-score	support
0	0.90	0.92	0.91	5434
1	0.92	0.90	0.91	5594
accuracy			0.91	11028
macro avg	0.91	0.91	0.91	11028
weighted avg	0.91	0.91	0.91	11028

Confusion Matrix:

```
[[4984  450]
 [ 533 5061]]
```

Discussion and Conclusion:

This project displays a full machine learning ensemble, combined with multiple preprocessing steps. The final model achieved a 91% accuracy and an ROC-AUC score of 97.8%, meaning it is a reliable source for classifying misinformation and true content. These

high scores taught us how important and useful it is to combine feature engineering with ensemble models to build a reliable classifier. Although project issues were limited, calculating optimal thresholds and parameters took time. For example, TF-IDF had a vocabulary limit of 15,000 words. Although a larger threshold could improve accuracy, it risked computational cost and risked overfitting. The N-gram range was originally set between 1 and 2 but missed key misinformation phrases. Within classifiers, we had to alter the C-value parameter a few times in the support vector machine to find the proper line between bias and variance.

Throughout this project, we applied various topics that we discussed in the lecture. These topics include web scraping feature extraction, supervised learning pipelines, ensemble modeling, and classification reporting for evaluation of metrics. We further understood what was necessary for machine learning models to understand and break down text versus numerical data only. Overall, this project successfully challenged our machine learning knowledge with hands-on experience of integrating a model to detect real world issues like the spread of misinformation.