Fake News Detection Report

Desheng Liu
Vrinda Pandey
COE 379L, Software Design for Intelligent Systems

Data Preparation and Preprocessing

In today's digital age, where information spreads quickly and often without verification, the spread of fake news has become a serious concern. Particularly in the political and social spheres, misleading or false information can influence elections, create confusion, and damage public trust. With the rise of generative AI and the need for fast-paced content creation to gain viewership by sensationalization, identifying misinformation is very crucial.

For this project, We wanted to explore the use of machine learning and natural language processing (NLP) techniques to detect fake news articles based purely on the text. The goal was to build a model that can accurately classify a news article as either **true** or **fake**, using textual features and sentiment cues. We thought to chose this project because it felt interesting to me, but also a pretty good representation or demonstration of my progress and learning of all the AI / ML topics that we learned / covered in class :). Furthermore / more on the application of this project, motivation stems from this – fake news has tangible effects on the real world, and building a detection model felt like a meaningful application of what we've learned.

Also in addition to those previous motivations, We also saw this project as a good opportunity to compare different kinds of modeling strategies. We worked with both classical machine learning models (like logistic regression, Naive Bayes, and random forests), and newer transformer-based models that we learned near the end of the semester(like DistilBERT). My goal was to understand which approach performed better on this kind of NLP task, and what insights We could learn from patterns across fake and real articles.

Data Sources and Technologies Used

For this project, We used a dataset from Kaggle titled "Misinformation Fake News Text Dataset", which contains two CSV files: one for real (referenced as TRUE) news articles and one for fake (referenced as FAKE)news articles. Each file contains roughly 40,000 entries, which makes the dataset quite balanced and large enough for both classical and transformer-based modeling.

Each article in the dataset includes the text content and the index of the datapoint. To work with the data, We used a variety of Python libraries and machine learning tools:

- **Pandas** and **NumPy** for data loading, manipulation, and analysis.
- **Matplotlib** and **Seaborn** for visualizing things like sentiment scores and word frequencies.
- **Scikit-learn** for classical machine learning pipelines and model training
    - TF-IDF vectorization
    - TfidfVectorizer for turning the text into numerical features
    - LogisticRegression, MultinomialNB, LinearSVC, and RandomForestClassifier for classification
    - VotingClassifier to build a soft-voting ensemble using multiple models
- **TextBlob** for sentiment analysis — We used this to compare polarity (positive/negative tone) and subjectivity between fake and real articles.
- **WordCloud** and **CountVectorizer** to help visualize and analyze word frequency distributions across the two classes.
- **Transformers (DistilBERT)** — a lightweight transformer model for NLP tasks.

Methods Employed

Classical Machine Learning Models

We focused on building out a strong pipeline using classical machine learning models. This involved cleaning the data, transforming the raw text into numerical features, and training several classifiers to detect whether a news article was real or fake based purely on its content.

First, We combined the TRUE and FAKE datasets into a single DataFrame and added a binary label column — 1 for true and 0 for fake. We removed any rows with missing or empty text and dropped duplicates to avoid data leakage. We also shuffled the dataset to ensure a good distribution of both classes.

For feature extraction, We used TF-IDF (Term Frequency–Inverse Document Frequency) vectorization. This converts the raw text into numerical vectors that reflect the importance of each word or phrase within a document relative to the rest of the dataset. We used a max document frequency of 0.7 to ignore overly common words, and included both unigrams and bigrams (ngram_range=(1, 2)), which allowed the model to learn short sequences of words, not just individual terms.

Once the text was vectorized, We trained and evaluated the following models using Scikit-learn:

- **Logistic Regression**: a simple linear classifier that often performs well in text classification tasks
- **Multinomial Naive Bayes**: a common choice for NLP problems due to its speed and probabilistic nature
- **Linear Support Vector Machine (SVC)**: optimized for text data
- **XGBoost (XGBClassifier)**: a more advanced boosting method that does good with feature interactions
- **VotingClassifier (Soft Voting)**: a model that combines predictions from Logistic Regression, Naive Bayes, and XGBoost using soft voting (averaging predicted probabilities)

All models were trained on 80% of the data and tested on the remaining 20%. We evaluated them using precision, recall, F1-score, accuracy, and ROC AUC. We also visualized their confusion matrices and ROC curves to better understand their strengths and weaknesses.

Throughout this process, We noticed some interesting differences in model behavior. Logistic Regression and SVM performed well overall, especially in terms of recall and F1. XGBoost gave a small boost in precision but took longer to train. Naive Bayes was the fastest, though slightly weaker in terms of generalization. The VotingClassifier (soft) helped blend the strengths of each model, leading to the best overall ROC AUC in most of my runs.

Non-Classical Machine Learning (Transformer – the DistilBERT model)

We wanted to try a more modern NLP approach using transformers – Specifically with DistilBERT, since it is a lightweight, faster version of BERT from Hugging Face (reputable company that develops models) that is designed for efficient text classification tasks. My goal was to train / fine-tune this model on the same binary classification task to see the differing results and what was ultimately the best model for this project task.

To prepare the pipeline for the kaggle dataset, We loaded the pre-trained distilbert-base-uncased tokenizer and applied it to the news text using padding and truncation with a maximum length of 512 tokens. This ensured that all input sequences were of equal length and fit within the model's token limit. We then loaded the DistilBERTForSequenceClassification model with two output labels (real vs fake) and used Hugging Face's Trainer API to set up the training loop. My training configuration included:

- 3 epochs
- batch size of 16 for training and 64 for evaluation
- AdamW optimizer with weight decay
- evaluation strategy set to run every epoch

The specifics of each result will be further discussed in the "results" portion of this paper, but overall the DistilBERT model performed very well and generalized better than We expected. Even without a lot of fine-tuning and testing, the model achieved strong results on both precision and recall. One thing We speculate was that it was more performant in its ability to detect subtle phrases that the classical models may have missed, most likely in more subjective or emotionally charged pieces of fake news. The model didn't overfit, and the test set performance stayed consistent with what We saw during training.

Results

Classical Machine Learning Model Results

We trained and evaluated five different classical models using an 80/20 train-test split. The table below includes metrics for both training and testing sets, so We could directly compare performance and look for signs of overfitting or underfitting. We evaluated each model using precision, recall, F1-score, and accuracy, and also visualized ROC AUC to understand the models' overall ability to distinguish between true and fake news.

Model	Dataset	Accuracy	Precision	Recall	F1-Score
Logistic Regression	**Train**	0.95	0.95	0.95	0.95
	Test	0.94	0.94	0.94	0.94
Naive Bayes	**Train**	0.91	0.91	0.91	0.91
	Test	0.90	0.90	0.90	0.90
Linear SVM	**Train**	0.97	0.97	0.97	0.97
	Test	0.96	0.96	0.96	0.96
XGBoost	**Train**	0.95	0.95	0.95	0.95
	Test	0.94	0.94	0.94	0.94
Voting Ensemble (Soft Voting)	**Train**	0.96	0.96	0.96	0.96
	Test	0.95	0.95	0.95	0.95

Classical Learning Model Results

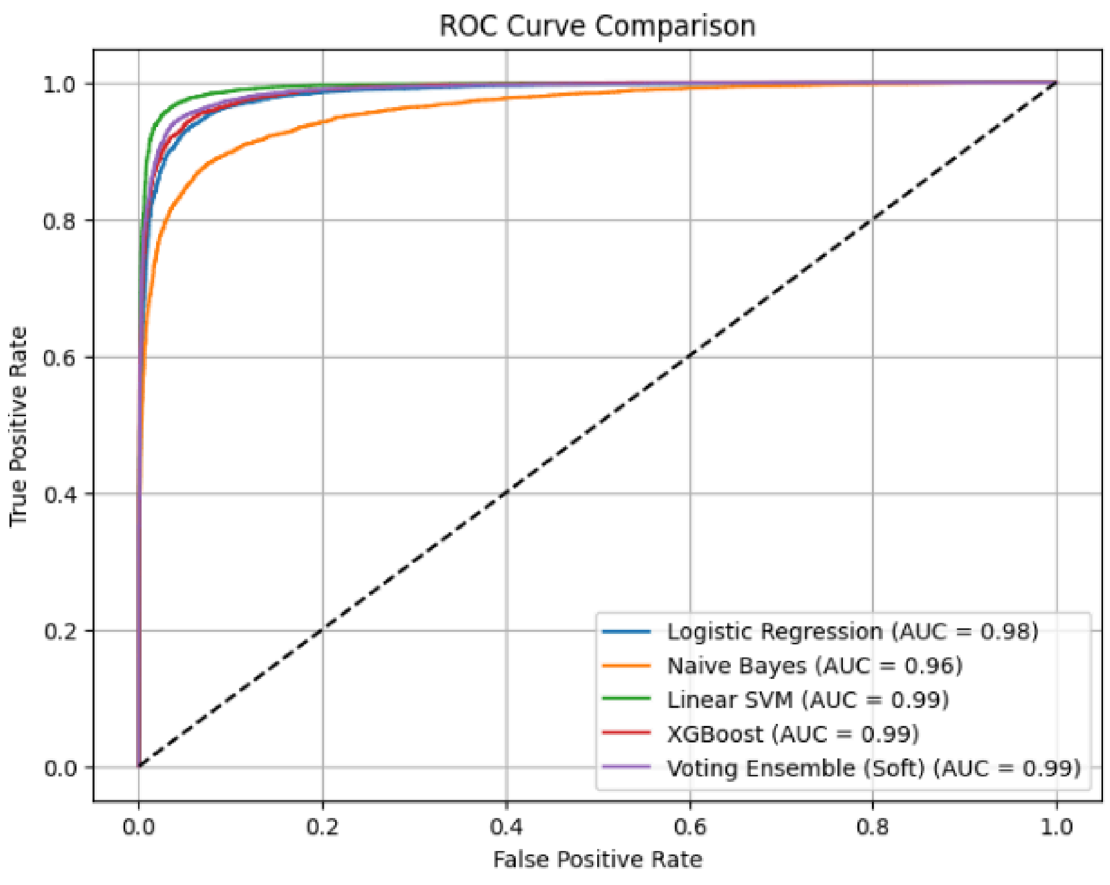
Overfitting Discussion

Most of the classical models performed consistently between training and testing, which is a good sign of generalization. The Linear SVM achieved the highest overall scores but also showed the largest gap (roughly 2%) between train and test, which suggests slight overfitting. The Voting Ensemble did a good job balancing accuracy and generalization, staying close across both datasets.

Overall, none of the models showed extreme overfitting. The small drops in test performance compared to training are expected and acceptable given the dataset size and complexity of the models.

ROC Curve Comparison

The ROC curve below shows the AUC for each classical model. As expected, the Linear SVM, Voting Ensemble, and XGBoost models each achieved an AUC of 0.99, indicating excellent performance. Logistic Regression followed at 0.98, and Naive Bayes finished at 0.96.



ROC Curve Comparison for Classical Machine Learning Models

Transformer Learning Results:

Conclusion and Final Thoughts

IOverall, this project pushed us to explore both classical and transformer-based approaches to fake news detection using raw text. What made this project interesting was seeing how well simpler models held up compared to a modern transformer, and how preprocessing and vectorization directly impacts model performance. If our team had more time, we suggested focusing on trends by issue (e.g., comparing tone in fake vs. real news about elections, health, or immigration), which might be a future self-project this summer!

References

1. Steven Peutz. *Misinformation Fake News Text Dataset (79k)*. Kaggle. https://www.kaggle.com/datasets/stevenpeutz/misinformation-fake-news-text-dataset-79k
2. Hugging Face Transformers Documentation. https://huggingface.co/docs/transformers
3. Hugging Face DistilerBERT Documentation. https://huggingface.co/docs/transformers/model_doc/distilbert
4. TextBlob Documentation. https://textblob.readthedocs.io/en/dev/