FAKE NEWS

FAKE NEWS DETECTION

COMPARATIVE MACHINE LEARNING PROJECT

In this project, I built and evaluated a machine learning pipeline to classify news articles as real or fake. I compared two preprocessing approaches and six machine learning models to understand how feature engineering affected performance.

Masood BAI

Project case study

Project overview

What I built

I created a binary text classification pipeline that predicts whether a news article is real or fake. The project focused on improving model performance through better preprocessing and feature engineering rather than changing the dataset itself.

Why I structured it this way

I used the same merged dataset across both approaches so that any performance difference came from the preprocessing pipeline. This made the comparison fair and helped me evaluate the practical value of each improvement.

Tools

Python, Scikit-learn, Pandas, TF-IDF, lemmatisation, chi-squared feature selection, confusion matrix analysis

99.72%

Best accuracy (Random Forest)

6

Models tested

2

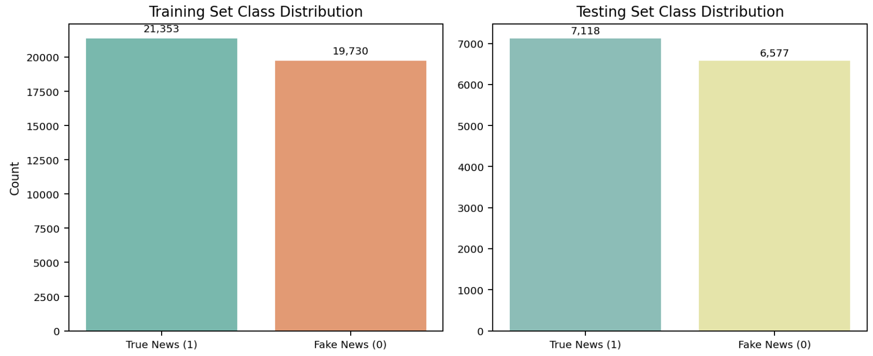
Approaches

54,778

Total articles used

75 / 25

Stratified train-test split



Balanced class distributions recreated from the project counts

Data preparation and workflow

Dataset and preparation

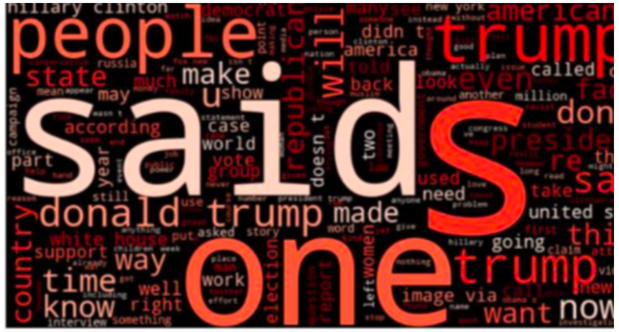
- I merged two Kaggle fake news datasets into one combined dataset with consistent binary labels.
- I cleaned the raw text by lowercasing, removing punctuation, URLs, HTML, extra spaces and other noise.
- I used lemmatisation in the stronger approach to reduce word variation and improve consistency.
- I kept the class distribution balanced through a stratified train-test split.

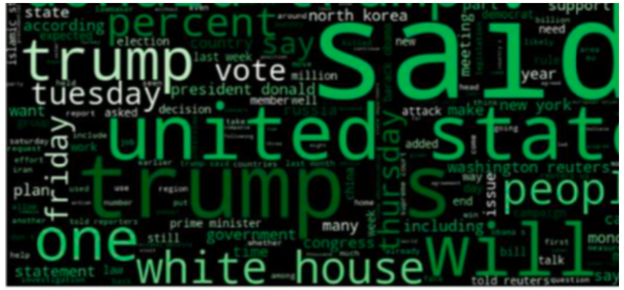
Approach 1

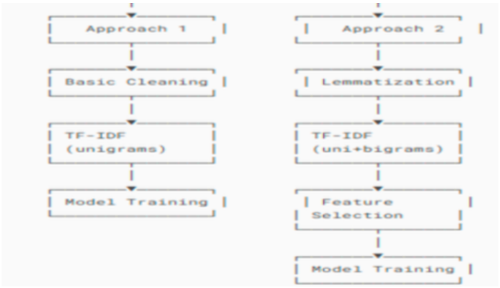
Basic cleaning + TF-IDF using unigrams.

Approach 2

Lemmatisation + TF-IDF using unigrams and bigrams + chi-squared feature selection (top 10,000 features).

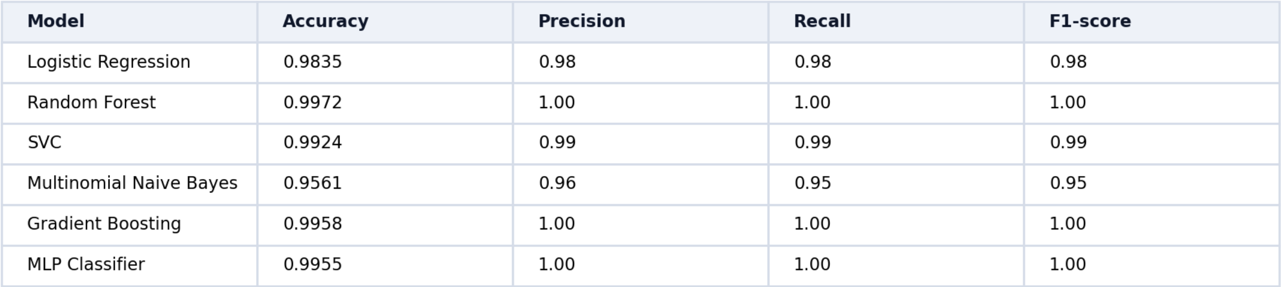




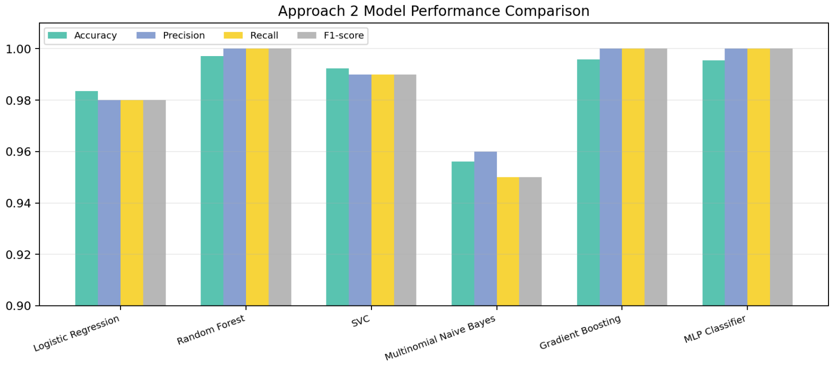


Workflow diagram and word clouds taken from the project output

Model evaluation and key results



Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.9835	0.98	0.98	0.98
Random Forest	0.9972	1.00	1.00	1.00
SVC	0.9924	0.99	0.99	0.99
Multinomial Naive Bayes	0.9561	0.96	0.95	0.95
Gradient Boosting	0.9958	1.00	1.00	1.00
MLP Classifier	0.9955	1.00	1.00	1.00



What the results showed

I trained six models in both approaches: Logistic Regression, Multinomial Naive Bayes, SVC, Random Forest, Gradient Boosting and MLP. The strongest overall result came from Random Forest in Approach 2.

99.72%

Best model: Random Forest (Approach 2)

Precision: 1.00   Recall: 1.00   F1-score: 1.00

My interpretation

- Approach 2 improved performance across most models because it used stronger text preprocessing and feature selection.
- Random Forest and Gradient Boosting produced the most reliable results across the evaluation metrics.
- Even the simpler models performed better when the text was cleaned properly and stronger features were used.

Conclusion

Why this project matters

This project shows how I approach a classification problem from both an analytical and engineering perspective. I compared two preprocessing pipelines, evaluated six models, and used summary metrics to justify the final choice.

Key points from the project

- Clear experimentation across two approaches
- Strong NLP preprocessing and feature engineering
- High-performing Random Forest result
- A well-structured evaluation using accuracy, precision, recall and F1-score

Final outcome: I built a well-documented fake news classification project and identified Random Forest with Approach 2 as the strongest final model.