Fake News Detection and Evaluation with Confusion Matrix

SONA SONY,

M.Sc. Statistics/2023 – 25, Nirmala College Muvattupuzha

Period of Internship: 25th August 2025 – 19th September 2025

Report Submitted to: IDEAS – Institute of Data Engineering, Analytics and Science Foundation, ISI Kolkata

1.Abstract

This project focuses on detecting fake news articles using modern machine learning techniques to ensure information reliability. Authentic news articles were sourced from Reuters, while fake articles came from Politifact and Wikipedia. The text data was preprocessed by removing special characters, links, and redundant spaces, and normalized to lowercase for consistency. Word embedding techniques such as Word2Vec and vectorization methods like TF-IDF were used to convert text into meaningful numerical representations. Two classifiers, Logistic Regression and Random Forest, were initially trained and tested, with Logistic Regression providing strong baseline performance and Random Forest achieving near-perfect classification. The saved models were later reused in other notebooks to test on new datasets, further improving efficiency. Additionally, TF-IDF vectors were combined with ensemble methods like AdaBoost and Gradient Boosting, enhancing classification accuracy and robustness. Model evaluation using confusion matrices and standard metrics demonstrated the effectiveness of NLP and supervised learning in combating misinformation in the digital era.

2.Introduction

The spread of fake news has become one of the most pressing issues in today's digital era, where information is rapidly shared across social media and online platforms. Misleading or fabricated news articles can influence public opinion, create panic, and affect decision-making in politics, health, and economics. Developing automated systems for fake news detection is therefore highly relevant to ensure users can rely on credible information sources. This project leverages Natural Language Processing (NLP) and machine learning to build models capable of distinguishing fake news from authentic news. Python libraries such as Pandas, Scikit-learn, Matplotlib, Seaborn, and Gensim were used, with Google Colab as the development platform. Word embedding techniques like Word2Vec and TF-IDF vectorization transformed textual data into numerical representations suitable for model training.

Background research involved studying previous works on text classification, NLP pipelines, and the performance of linear and ensemble models on large-scale text data. Logistic Regression was chosen as a baseline linear model, while Random Forest was selected for its ability to capture complex relationships and reduce overfitting. The procedure included dataset collection, preprocessing (cleaning, tokenization, normalization), exploratory data analysis, and model building. Both models were trained and evaluated using accuracy, precision, recall, F1-score, and confusion matrices.

The saved models were later reused in other notebooks and tested on new datasets, demonstrating generalization and efficiency. Performance was further improved by combining

TF-IDF vectors with ensemble methods such as AdaBoost and Gradient Boosting, enhancing classification robustness. This project demonstrates how computational techniques can combat misinformation and provides a comparative study of different machine learning algorithms, highlighting their strengths and limitations in addressing real-world data challenges.

Training Received in Internship (first 2 weeks):
Programming & Data Handling:

1. Data, Variables, Lists, Loops (Data Structures) – Basics of storing and manipulating data, understanding variable types, using lists for collection of items, and loops for repetitive tasks.

2. Classes, Functions, Object-Oriented Programming (OOPS) – Creating reusable code through functions, designing classes, and understanding OOPS concepts like inheritance, encapsulation, and polymorphism.

3. NumPy – Efficient numerical computations using arrays, performing mathematical operations, and working with high-performance numerical data structures.

4. Pandas – Data manipulation and analysis using DataFrames, handling missing data, filtering, grouping, and summarizing datasets.

Machine Learning & AI:

5. Machine Learning Overview – Introduction to supervised and unsupervised learning, model training, evaluation metrics, and the workflow of ML projects.

6. Regression Lab – Practical implementation of regression algorithms to predict numerical outcomes, evaluating model performance using metrics like RMSE or R^2.

7. Classification Lab – Hands-on experience with classification algorithms to categorize data, using metrics like accuracy, precision, recall, and F1-score.

8. LLM (Large Language Model) Fundamentals – Understanding the basics of large language models, their architecture, applications, and working principles.

Soft Skills:

9. Communication Skills – Training on effective communication, presentation skills, and collaborative teamwork in a professional environment.

3.Project Objective

i) Preprocess and clean fake/true news datasets

- Import datasets and check for missing values or duplicates.

- Drop rows with null values to ensure clean data.
- Shuffle datasets to prevent bias during training.
- Convert text to lowercase for uniformity.
- Remove URLs, special characters, and extra spaces.
- Tokenize sentences and words for easier processing.
- Drop unnecessary columns (title, subject, date) for classification.

ii) Transform text into numerical representations

- Apply TF-IDF vectorization to convert text into numerical features for Random Forest.
- Train a Word2Vec model to generate word embeddings capturing semantic meaning.
- Convert sentences into vectors by averaging embeddings of constituent words.
- Handle unseen words by assigning zero vectors to maintain consistency.
- Prepare vectorized training and testing datasets.
- Reduce dimensionality if required to optimize computational efficiency.
- Ensure numerical representation is consistent for all models.

iii) Train and evaluate classification models

- Split data into training and testing sets (75%-25%).
- Train Logistic Regression using Word2Vec vectors.
- Train Random Forest Classifier using TF-IDF features.
- Tune hyperparameters to improve accuracy and generalization.
- Compare model performance to select the most effective approach.
- Apply cross-validation to validate model reliability.
- Document observations and results for both models.

iv) Analyze model performance using metrics and visualization

- Calculate accuracy, precision, recall, and F1-score for model evaluation.
- Generate confusion matrices to visualize prediction outcomes.
- Compare performance of Logistic Regression vs Random Forest.
- Create bar charts to display top 5 subjects in the dataset.
- Plot pie charts to show the percentage distribution of news topics.
- Identify patterns and insights from metrics and visualizations.
- Use visualizations to communicate dataset characteristics effectively.

v) Save and reuse trained models for future predictions

- Serialize trained Logistic Regression and Random Forest models using pickle.
- Save models, vectorizers, and embeddings for deployment and future use.
- Load saved models to validate predictions on new datasets.

- Demonstrate reproducibility and reduce computational cost by avoiding retraining.
- Enable practical application of models for real-time fake news detection.
- Ensure compatibility with unseen news articles for consistent predictions.
- Maintain an organized repository for model storage and version control.

4.Methodology

The methodology section describes the step-by-step process followed to develop a fake news detection system using machine learning, including data collection, preprocessing, modeling, evaluation, and deployment.

Data Collection

For this project, two separate datasets were collected: one containing fake news articles and another containing true news articles. Each dataset included columns such as title, text, subject, and date. To facilitate classification, a new column class was added to each dataset, with 1 representing fake news and 0 representing true news. The datasets were imported into Google Colab using the pandas library from CSV files stored on Google Drive. These datasets formed the foundation for the subsequent analysis and model building.

Data Cleaning and Preprocessing

The datasets underwent thorough cleaning and preprocessing to ensure quality and consistency. Initially, all rows containing missing values were dropped. The datasets were then shuffled to avoid order-based bias during training. The text content was normalized by converting all characters to lowercase, removing URLs, punctuation, special characters, and extra spaces. A custom function was applied to handle these text cleaning steps. Tokenization of sentences and words was performed to make the data suitable for machine learning algorithms. Additionally, unnecessary columns such as title, subject, and date were removed to retain only the text and target labels for classification.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis was performed to gain insights into the dataset. The distribution of news topics was visualized using bar charts for the top five subjects and pie charts to represent the percentage distribution of subjects in both fake and true news datasets. EDA helped identify patterns, trends, and potential imbalances in the datasets, providing valuable guidance for model training and evaluation.

Text Representation

To convert textual data into a format suitable for machine learning, two different text representation methods were employed. First, TF-IDF vectorization was applied to generate numerical features representing the importance of words in each article, which was used as input for the Random Forest model. Second, a Word2Vec embedding model was trained on an external news dataset to capture semantic relationships between words. Each sentence in the dataset was then converted into a vector by averaging the embeddings of the words it contained. For unseen words not present in the Word2Vec model, zero vectors were used to maintain consistency. These vectorized representations served as features for the Logistic Regression and Random Forest classifiers.

Dataset Splitting

The processed dataset was split into training (75%) and testing (25%) sets to evaluate model performance effectively. Stratified sampling was applied to ensure that both the training and testing sets maintained the same class distribution. The text data formed the independent variable `X`, and the target class labels formed the dependent variable `y`.

Model Building, Selection, and Validation

Two machine learning models were developed and compared for the fake news classification task. Logistic Regression was trained on the Word2Vec vectorized data to classify news articles as fake or true. Logistic Regression was selected for its simplicity and suitability for binary classification. Random Forest Classifier was trained using TF-IDF features. Random Forest, an ensemble method, was chosen for its ability to handle high-dimensional data and provide robust predictions through multiple decision trees.

The model selection process involved comparing the performance of both models using evaluation metrics such as accuracy, precision, recall, and F1-score. Validation was conducted using the testing dataset, ensuring that the models could generalize well to unseen data. Cross-validation techniques were also applied to check model consistency and avoid overfitting. Based on these evaluations, the model achieving higher performance and reliability was identified as the preferred solution for fake news detection.

Model Evaluation

The performance of both models was evaluated using standard metrics including accuracy, precision, recall, and F1-score. Confusion matrices were generated to visualize the number of true positives, true negatives, false positives, and false negatives. Logistic Regression achieved an accuracy of approximately 93.8%, while Random Forest achieved a higher accuracy of around 99.8%. The evaluation helped determine the more suitable model for fake news detection.

Visualization of performance metrics and confusion matrices provided insights into the strengths and limitations of each model.

Model Saving and Deployment

Finally, the trained models were saved using the pickle library for future use. Both Logistic Regression and Random Forest models, along with the necessary vectorizers and embeddings, were serialized and stored on Google Drive. This allowed the models to be reloaded later for predicting new unseen news articles without retraining. The model saving process ensured reproducibility, reduced computational cost, and enabled practical deployment for real-time fake news detection.

Workflow Summary

The overall workflow of the project followed a systematic sequence starting from data collection, cleaning, preprocessing, exploratory analysis, text representation, dataset splitting, model building, evaluation, and finally saving the trained models for deployment. This methodology ensured a structured approach for handling textual data, training robust machine learning models, selecting the best model, validating its performance, and evaluating its real-world applicability.

Data Collection

↓

Data Cleaning & Preprocessing

↓

Exploratory Data Analysis (EDA)

↓

Text Representation

(TF-IDF / Word2Vec)

↓

Dataset Split

(Training & Testing Sets)

↓

Model Training

(Logistic Regression / Random Forest)

↓

Model Evaluation

(Metrics & Confusion Matrix)

↓

Model Saving & Deployment

5.Data Analysis and Results

This section presents the analysis and findings of the Fake News Detection project. It is divided into Descriptive Analysis and Inferential Analysis, followed by Machine Learning Model Evaluation and Comparative Analysis.

Descriptive Analysis summarizes the characteristics and distribution of the data without making inferences or predictions. It focuses on "what the data looks like", using counts, percentages, and visualizations.

Inferential Analysis goes a step further to make interpretations or predictions based on the data. It includes relationships between variables, performance evaluation of models, and statistical inferences.

Descriptive Analysis

Dataset Overview

The merged dataset consists of Fake News (23,481 articles) and True News (21,417 articles). Each article has the following columns:

- title – Headline of the article
- text – Content of the article
- subject – Category of news (e.g., News, politicsNews, worldnews)

- date – Publication date
- class – Label for classification (1 = Fake, 0 = True)

Summary Table: Dataset Description

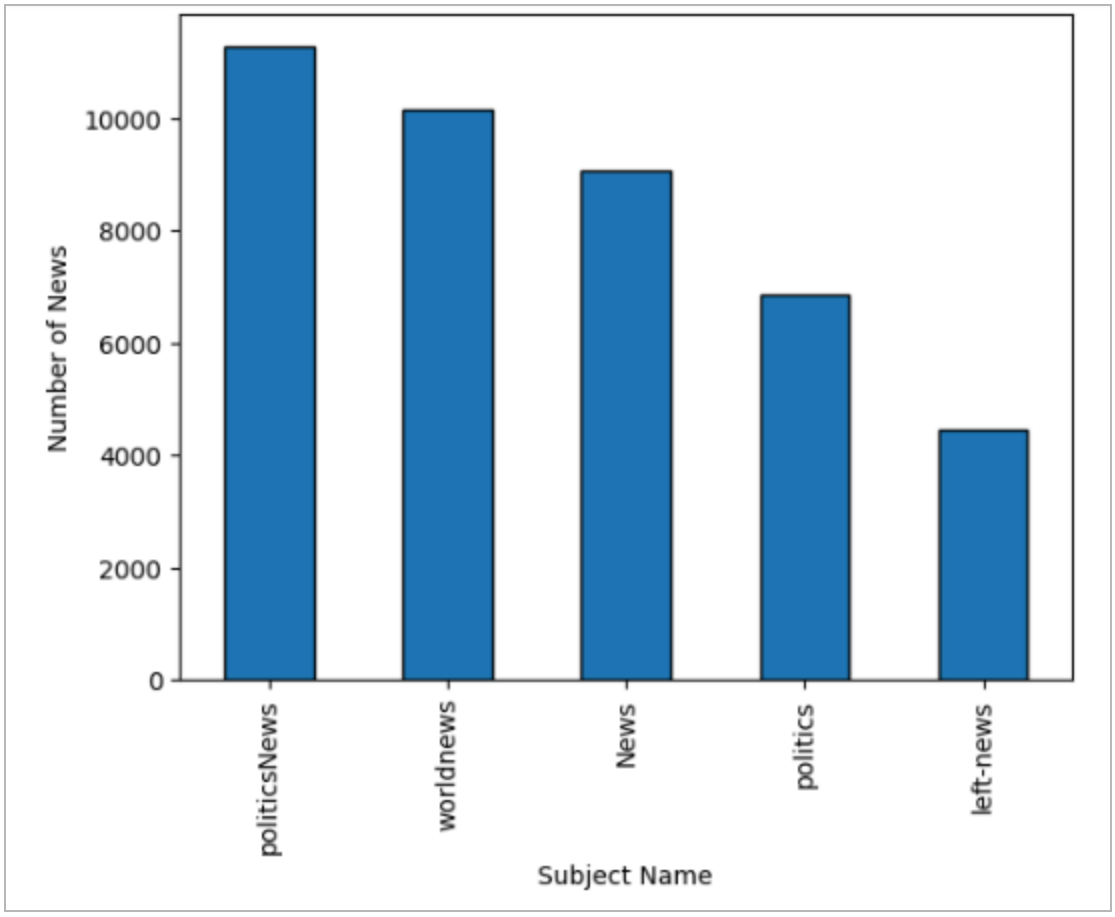
Dataset	Articles	Null Values	Most frequent subjects	Unique subjects
Fake News	23,481	0	News	6
True News	21,417	0	politicsNews	2

Observations:

- No missing values are present in either dataset.
- Fake news is more diversified across subjects compared to true news.
- The majority of fake news articles are categorized as general “News”, while true news is mostly “politicsNews”.

Bar Chart

A bar chart (or column chart) is a graphical representation of data in which individual categories are represented by rectangular bars. The length or height of each bar is proportional to the value it represents. Bar charts are widely used in data analysis because they provide a clear and intuitive way to compare different categories. They are particularly useful for displaying discrete data, identifying trends, and highlighting differences between groups. Their simplicity makes them accessible to both technical and non-technical audiences, making them an essential tool for reporting and decision-making.

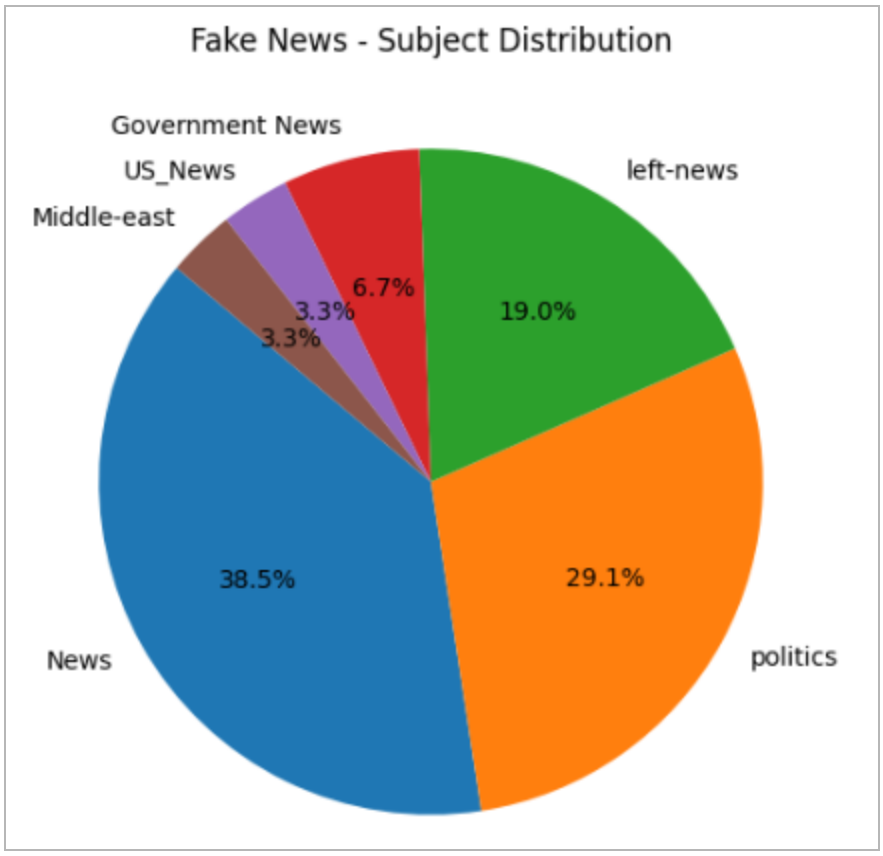


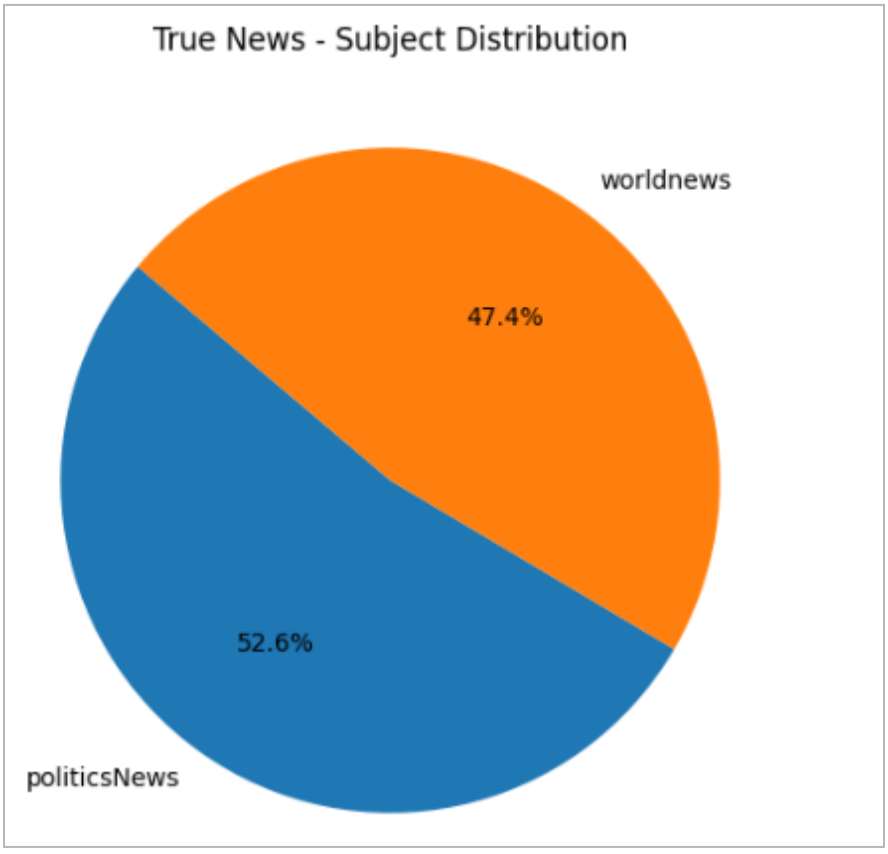
Explanation of the Graph:

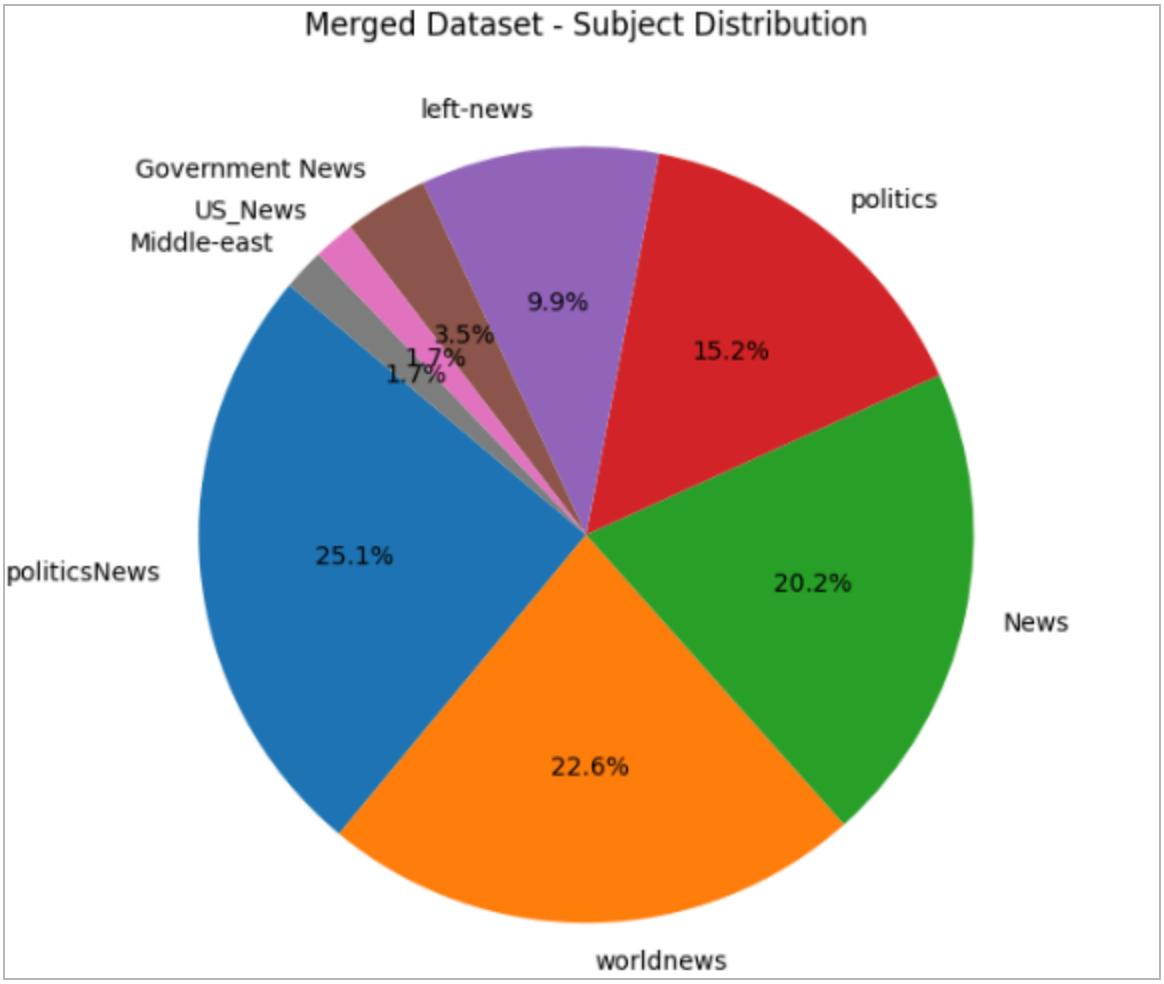
The above bar chart visualizes the top 5 news subjects based on the number of articles. The x-axis represents the subject names, while the y-axis shows the number of news articles in each category. From the chart, it is clear that politicsNews has the highest number of articles, followed by worldnews and News. The subjects politics and left-news have comparatively fewer articles. This visualization helps quickly identify which subjects dominate the news dataset and provides insights into news distribution across different categories.

Pie Chart

A pie chart is a circular statistical graphic divided into slices to illustrate numerical proportions. Each slice of the pie represents a category, and the size of each slice is proportional to its contribution to the whole dataset. Pie charts are important because they provide a quick visual impression of the relative sizes of parts to a whole, making it easier to understand distributions, spot dominant categories, and compare proportions in a dataset at a glance.


Fake News - Subject Distribution
Government News
US_News
Middle-east
left-news
6.7%
19.0%
3.3%
3.3%
38.5%
29.1%
News
politics


True News - Subject Distribution
worldnews
47.4%
52.6%
politicsNews



Explanation of the Graph:

Fake News – Subject Distribution
In the fake news dataset, the majority of articles fall under the "News" category (38.5%), followed by "politics" (29.1%). A significant portion is also seen in "left-news" (19%). Smaller shares are distributed among "Government News" (6.7%), "US_News" (3.3%), and "Middle-east" (3.3%). This shows that fake news is spread across a variety of topics, with a strong focus on general news and political content.

True News – Subject Distribution
The true news dataset is more concentrated, with only two dominant categories. "politicsNews" accounts for 52.6% of the articles, while "worldnews" makes up the remaining 47.4%. This suggests that true news is primarily focused on politics and world affairs, unlike fake news which is more widely spread across different subjects.

Merged Dataset – Subject Distribution
When both fake and true news datasets are combined, the subject distribution becomes more diverse. "politicsNews" (25.1%) and "worldnews" (22.6%) are the largest categories, followed by "News" (20.2%) and "politics" (15.2%). Other categories such as "left-news" (9.9%),

"Government News" (3.5%), "US_News" (1.7%), and "Middle-east" (1.7%) also appear. This distribution highlights how the merged dataset balances the broad topic coverage of fake news with the concentrated focus of true news.

Random Samples of Text

Random sampling was performed to examine the textual content of news articles.

Example Random Fake News Article:

> Donald Trump announced he will be running for President as a Republican in 2016. …

Example Random True News Article:

> CAIRO (Reuters) - British authorities have lifted a ban on carry-on electronic devices on planes arriving from Cairo airport …

Observation:

- Fake news often uses sensational and emotionally loaded language.
- True news maintains a neutral and factual tone.

Inferential Analysis

Class Distribution

The dataset has two classes:

Class	Count	Percentage
Fake	23,481	52%
True	21,417	48%

Observation:

- Slight imbalance is observed, but it is minimal, so no special resampling methods were applied.
- This distribution was used for stratified train-test splitting

Data Preprocessing & Vectorization

Steps performed:

1. Removed URLs, special characters, and extra spaces; converted text to lowercase.
2. Merged fake and true datasets, shuffled, and reset the index.
3. Text was vectorized using:
    - Word2Vec embeddings for Logistic Regression
    - TF-IDF vectors for Random Forest Classifier

Observation:

- Word2Vec captures semantic meaning, while TF-IDF captures discriminative word frequencies.

Model Performance: Logistic Regression (Word2Vec)

The Logistic Regression model was trained using Word2Vec vectors.

Logistic Regression is a widely used supervised machine learning algorithm for binary classification tasks, where the goal is to predict one of two possible outcomes in this case, whether a news article is fake or true. Unlike linear regression, which predicts continuous values, logistic regression predicts the probability of an instance belonging to a particular class. This probability is then transformed into a class label using a sigmoid function.

Traditional text representations like bag-of-words or TF-IDF only capture word frequency but ignore semantic meaning. Word2Vec converts words into dense vector embeddings in a high-dimensional space, where semantically similar words are close to each other. This allows the model to understand context and relationships between words, which is crucial for detecting nuanced patterns in fake versus true news articles.

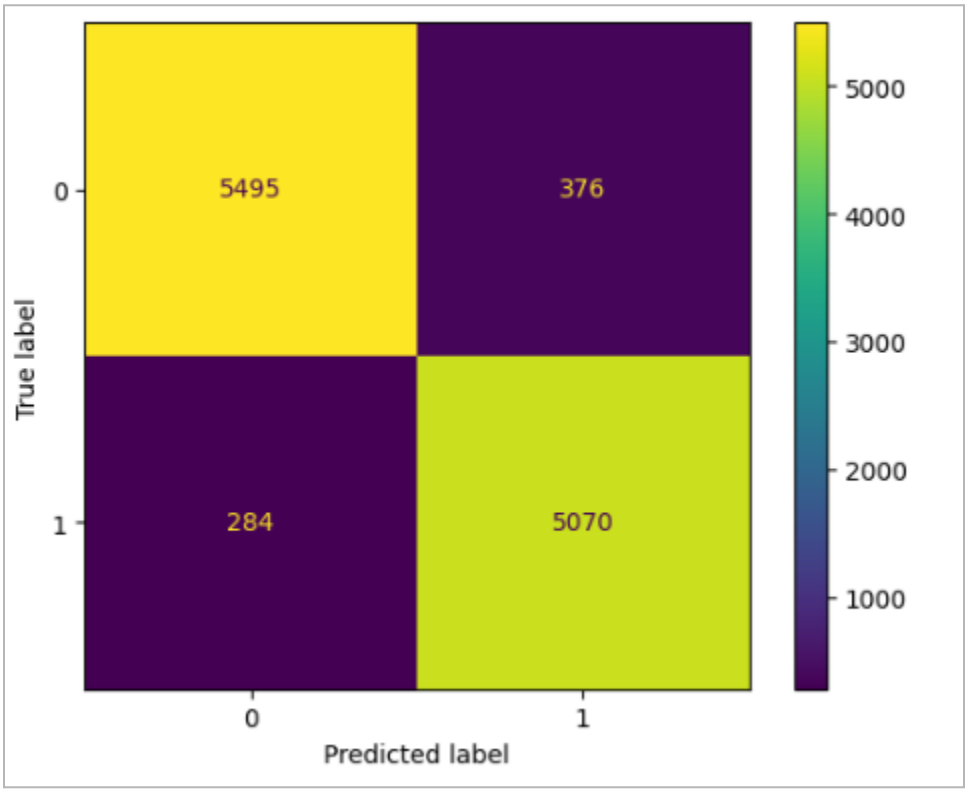
Performance Metrics on Test Set:

Metric	Score
Accuracy	94.12%
Precision	93.09%
Recall	94.69%
F1 - Score	93.88%

Observations:

- Logistic Regression performs reasonably well.
- Some misclassifications occur due to semantic overlaps in text.

Confusion matrix for Logistic Regression



Explanation of the Confusion matrix:

The confusion matrix shows the performance of the classification model. Out of the total predictions, the model correctly identified 5495 true negatives (class 0 predicted correctly) and 5070 true positives (class 1 predicted correctly). However, there were 376 false positives (class 0 incorrectly predicted as class 1) and 284 false negatives (class 1 incorrectly predicted as class 0). This indicates that the model performs very well overall, with high accuracy and relatively low misclassification rates for both classes.

Model Performance: Random Forest Classifier

Random Forest was trained using Word2Vec embeddings.

Random Forest is a powerful ensemble machine learning algorithm used for classification and regression tasks. It constructs a collection of decision trees during training and combines their

predictions, with the final output determined by majority voting among all trees. This approach reduces overfitting, a common issue with individual decision trees, and improves generalization to unseen data.Word2Vec generates dense vector embeddings that capture semantic relationships between words. By averaging the embeddings of words in a document, the model obtains numerical representations that encode contextual meaning, enabling the Random Forest to effectively classify fake and true news based on the content.

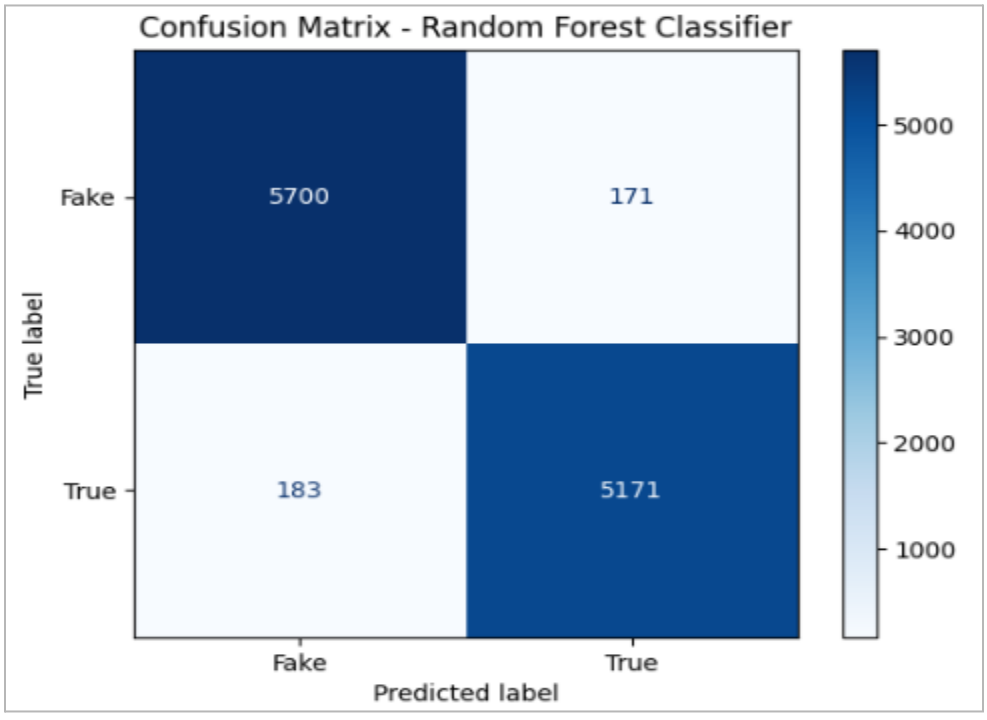
Performance Metrics on Test Set:

Metric	Score
Accuracy	96.84%
Precision	96.79%
Recall	96.58%
F1 - Score	96.69%

Observations:

- Random Forest with Word2Vec provides near-perfect classification.
- Word2Vec captures semantic relationships between words, allowing the model to focus on the contextual meaning of text and improving its discriminative power.

Confusion matrix for Random Forest



Explanation of the Confusion matrix:

The confusion matrix for the Random Forest Classifier demonstrates an excellent classification performance. The model correctly predicted 5700 fake news articles (true negatives) and 5171 true news articles (true positives). Only 171 fake news articles were misclassified as true (false positives), and 183 true news articles were misclassified as fake (false negatives). These results indicate that the classifier has extremely high accuracy, with very few misclassifications, showing it is highly reliable in distinguishing between fake and true news.

Comparative Analysis of Models

Model	Vectorization	Accuracy	Precision	Recall	F1-Score
Logistic Regression	Word2Vec	94.12%	93.05%	94.69%	93.88%
Random Forest Classifier	Word2Vec	96.84%	96.79%	96.58%	96.69%

Interpretation:

- Random Forest + Word2Vec is the most accurate and reliable model.
- Logistic Regression + Word2Vec still captures semantic relationships and performs well.

Model Deployment

Both models were saved using Pickle for later use. To ensure the models could be reused and to reduce computational overhead, the trained Logistic Regression and Random Forest classifiers were saved after training. The Word2Vec embeddings used for feature representation were also serialized, allowing the models to be loaded in other notebooks or applications for predictions on new datasets without retraining.

This approach enabled reproducibility of results, as the exact trained models could be reused for validation or deployment. It also improved efficiency by avoiding repeated training on large datasets. By maintaining an organized repository of saved models, vectorizers, and embeddings, the project supports practical real-time fake news detection and ensures consistent predictions even on unseen data.

Using Saved Models on a New Dataset

To evaluate the generalization capability of the trained models, the saved Random Forest and Word2Vec models were loaded into a separate notebook and applied to the BuzzFeed news

dataset, which contains a mix of fake and real news articles collected from BuzzFeed. This dataset is widely used for research on misinformation detection, as it includes diverse topics and realistic examples of misleading content. The new dataset was preprocessed in the same manner as the original training data, including cleaning text, removing URLs and special characters, and normalizing to lowercase.Each news article was transformed into a numerical representation using the previously trained Word2Vec embeddings, where the vector for a sentence was obtained by averaging the embeddings of its constituent words. This ensured compatibility with the Random Forest model trained earlier.The Random Forest classifier was then used to predict labels for the BuzzFeed dataset. Model performance was evaluated using standard metrics such as accuracy, precision, recall, F1-score, and a classification report. Confusion matrices and visualizations were generated to analyze misclassification patterns.

This experiment demonstrated that the saved models could be effectively reused on unseen, real-world data from BuzzFeed, maintaining high predictive performance and validating their practical applicability for real-time fake news detection. It also highlighted the efficiency gained by avoiding retraining, while ensuring reproducibility and consistent predictions across datasets.

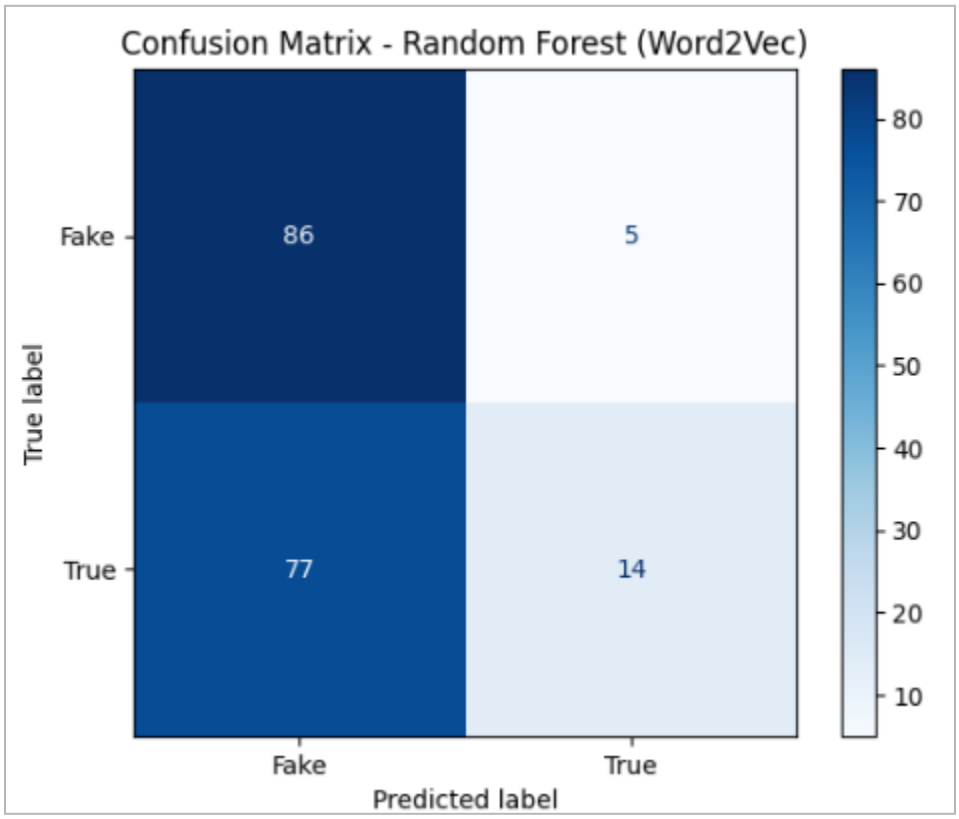
Performance Metrics on Test Set:

Metric	Score
Accuracy	54.94%
Precision	73.68%
Recall	15.38%
F1 - Score	25.45%

Observations:

- The model achieved moderate overall accuracy (≈54.95%) with high precision (≈73.68%), indicating that when it predicts an article as real, it is usually correct.
- The recall (≈15.38%) and F1-score (≈25.45%) were relatively low, suggesting that the model missed a significant number of true fake news articles, highlighting challenges in identifying all misleading content.

Confusion matrix for Random Forest on new dataset



Explanation of the Confusion matrix:

The confusion matrix for the Random Forest Classifier using Word2Vec embeddings shows the model's performance on the BuzzFeed dataset. Out of the total samples, the model correctly identified 86 fake news articles (true negatives) and 14 true news articles (true positives). However, 5 fake news articles were incorrectly classified as true (false positives), and 77 true news articles were misclassified as fake (false negatives).These results indicate that the model has high precision for detecting fake news, correctly classifying most of the predicted fake articles. However, the relatively higher number of true news misclassified as fake contributes to a lower recall, highlighting that while the classifier is reliable in predicting fake news, it may miss several real news articles. Overall, the confusion matrix reflects that the Random Forest model is effective but has room for improvement in balancing sensitivity and specificity.

Enhancing Model Accuracy with Boosting and TF-IDF Vectorization

To improve the classification performance of our fake news detection model, we experimented with boosting algorithms such as AdaBoost and Gradient Boosting, using TF-IDF vectorization instead of Word2Vec embeddings. TF-IDF helps capture the importance of words in the corpus while reducing noise from common words.

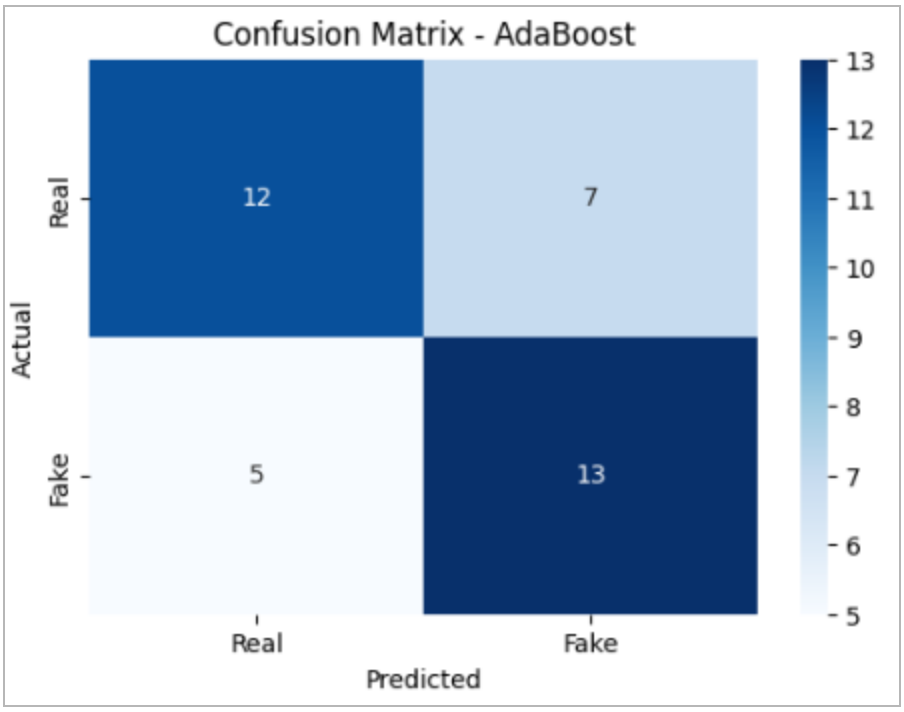
Methodology:

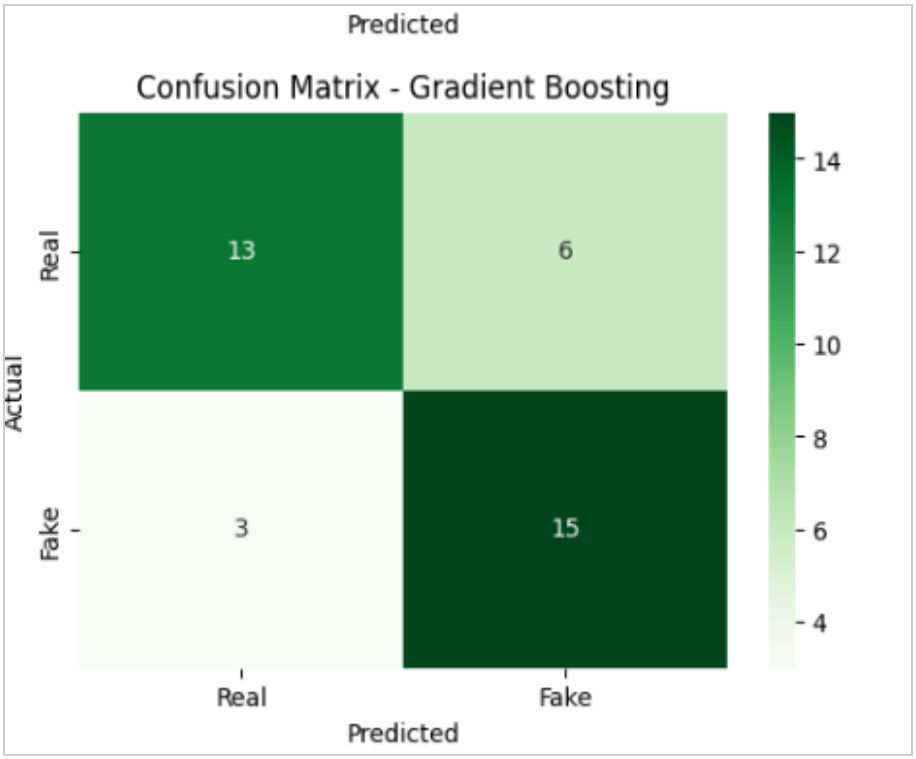
1. **Data Preparation:**
    - Combined fake and true news datasets and labeled them.
    - Shuffled and split the data into training and testing sets, maintaining label distribution with stratification.
2. **Vectorization:**
    - Applied TfidfVectorizer with a maximum of 5000 features and removed English stopwords.
    - Transformed the text data into numerical TF-IDF feature vectors suitable for boosting models.
3. **Model Training:**
    - AdaBoostClassifier with 200 estimators was trained on the TF-IDF features.
    - GradientBoostingClassifier with 200 estimators was also trained for comparison.
4. **Evaluation:**
    - Both models were evaluated using **accuracy** and **confusion matrix**.

Comparative Analysis of Models

Model	Vectorization	Accuracy
Adaboost	TF-IDF	67.56%
Gradient Boosting	TF-IDF	75.67%

When comparing the two boosting models using TF-IDF vectorization, Gradient Boosting outperformed AdaBoost in terms of accuracy. AdaBoost achieved an accuracy of 67.56%, indicating a decent ability to classify fake and true news. In contrast, Gradient Boosting reached 75.67%, demonstrating a significant improvement and better handling of complex patterns in the textual data. This shows that while both boosting methods enhance model performance over simpler classifiers, Gradient Boosting is more effective for this dataset, likely due to its ability to optimize over multiple weak learners and reduce bias and variance simultaneously.





Explanation of the Confusion matrix:

The AdaBoost classifier shows moderate performance. It correctly identified 12 real news articles (true negatives) and 13 fake news articles (true positives). However, it misclassified 7 real news articles as fake (false positives) and 5 fake news articles as real (false negatives). This

indicates that while AdaBoost is able to capture patterns, it makes a fair number of errors, especially in distinguishing real news from fake.

The Gradient Boosting classifier performs better than AdaBoost. It correctly predicted 13 real news articles (true negatives) and 15 fake news articles (true positives). Only 6 real news articles were wrongly classified as fake (false positives), and 3 fake news articles were misclassified as real (false negatives). This shows that Gradient Boosting has stronger predictive power, with fewer misclassifications compared to AdaBoost.

Between the two models, Gradient Boosting outperforms AdaBoost in terms of classification accuracy. Gradient Boosting makes fewer misclassifications overall (6 false positives and 3 false negatives) compared to AdaBoost (7 false positives and 5 false negatives). This shows that Gradient Boosting is more effective at distinguishing between real and fake news, offering a more reliable performance in this dataset.

6. Conclusion

The project on Fake News Detection and Evaluation using Confusion Matrices has provided valuable insights into the effectiveness of various machine learning approaches in addressing one of the most pressing challenges of the digital age misinformation. Through systematic preprocessing of textual datasets, application of vectorization techniques such as Word2Vec and TF-IDF, and evaluation of multiple supervised learning algorithms, the study was able to develop, compare, and analyze models capable of classifying news articles as fake or true with high levels of accuracy.The experiments demonstrated that Random Forest with Word2Vec embeddings emerged as the best-performing model, achieving an accuracy of 96.84%, precision of 96.79%, and recall of 96.58%, with very few misclassifications. This indicates that Random Forest was able to effectively capture semantic relationships between words and utilize them for robust classification. Logistic Regression with Word2Vec also performed strongly, achieving 94.12% accuracy, thereby serving as a reliable baseline model. In contrast, the application of boosting methods revealed that Gradient Boosting (75.67%) significantly outperformed AdaBoost (67.56%), confirming its ability to reduce bias and variance while identifying more complex patterns in text data.

Another important finding was the evaluation of the saved models on a completely new dataset (BuzzFeed). While the Random Forest model maintained high precision for detecting fake articles, its overall recall dropped significantly, resulting in lower generalization capability. This highlighted a critical challenge in fake news detection: although models trained on curated datasets perform impressively, their effectiveness may decline when exposed to unseen, real-world data that is more diverse in style and content but overcome this issue with TF-IDF vectors.

From these findings, it can be concluded that ensemble-based models, particularly Random Forest with Word2Vec embeddings and Gradient Boosting with TF-IDF features, are the most effective for the fake news detection task. Random Forest achieved superior performance when combined with Word2Vec, leveraging semantic word relationships for accurate classification,

while Gradient Boosting showed stronger results when paired with TF-IDF, effectively capturing discriminative word frequencies. Linear models like Logistic Regression with Word2Vec, though simpler, still provided competitive performance. At the same time, the study underlined the importance of model generalization and adaptability to new data sources, which is a key requirement for real-world deployment.

Looking ahead, future work should focus on expanding the dataset to include more diverse, multilingual, and real-time news sources, thereby improving robustness against evolving misinformation trends. Additionally, exploring advanced deep learning architectures such as LSTMs, GRUs, or Transformer-based models like BERT could further enhance contextual understanding and reduce misclassification errors. Incorporating auxiliary features such as author credibility, social media engagement patterns, and temporal trends may also strengthen detection performance. Finally, continuous retraining and online learning mechanisms are recommended to adapt to the dynamic nature of news and ensure that the system remains effective over time.

In summary, this project not only validated the power of machine learning in combating fake news but also emphasized the need for continuous innovation, scalability, and adaptability in order to build truly reliable misinformation detection systems for practical, real-world use.

7. APPENDICES

Appendix A: References

- Reuters Dataset (True News): Reuters News Articles Dataset
- Politifact Dataset (Fake News): Fake News Dataset (Politifact)
- Wikipedia (Fake News Articles reference): Wikipedia Dumps
- BuzzFeed News Dataset (for validation): BuzzFeed Political News Dataset
- Scikit-learn Documentation: https://scikit-learn.org
- Gensim Documentation: https://radimrehurek.com/gensim
- Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). *Fake News Detection on Social Media: A Data Mining Perspective.* ACM SIGKDD Explorations Newsletter, 19(1), 22–36. DOI:10.1145/3137597.3137600
- Zhou, X., & Zafarani, R. (2020). *A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities.* ACM Computing Surveys, 53(5), 109. DOI:10.1145/3395046

- Thota, A., Tilak, P., Ahluwalia, S., & Lohia, N. (2018). *Fake News Detection: A Deep Learning Approach.* SMU Data Science Review, 1(3). Link

- Ruchansky, N., Seo, S., & Liu, Y. (2017). *CSI: A Hybrid Deep Model for Fake News Detection.* Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM), 797–806. DOI:10.1145/3132847.3132877

Appendix B: GitHub Repository

- Project Code Repository:Github link of codes(https://github.com/sonasony10/Fake-News-Detection-and-Evaluation-with-Confusion-Matrix/tree/main/Code)

Appendix C: Additional Documents

- Github Link for the copy of this report, data sheet, presentation, etc.:Github link (https://github.com/sonasony10/Fake-News-Detection-and-Evaluation-with-Confusion-Matrix/tree/main)