A Comparative Overview of Various Fake News Detection Models

Anish Parmar
Duke University
Durham, NC 27708
avp30@duke.edu

Rebecca Du
Duke University
Durham, NC 27708
rrd17@duke.edu

Abstract

In this project, we explore the performance of various machine learning models and feature extraction techniques for the task of fake news detection. We apply Logistic Regression, Random Forest, Naive Bayes, and Support Vector Machine (SVM) (using both linear and RBF kernels) classifiers to two datasets, using N-Gram TF-IDF variants and CountVectorizer as preprocessing techniques. Through our experiments, we demonstrate that simple linear classifiers, such as logistic regression and linear SVM, achieved the highest F1 scores when combined with fine-grained character-level and hybrid TF-IDF vectorization. These models performed well on both the larger ISOT dataset and the smaller second dataset, consistently attaining F1 scores above $94\%+$. Overall, the results show that with the correct choice of feature engineering, relatively simple linear models can accurately detect fake news.

1 Introduction and Problem Statement

The creation of the internet has enabled individuals to access an unprecedented stream of real-time information. However, not all of this information is genuine. The past decade has seen numerous scandals and crises fueled by the spread of fake news, making the need to distinguish fact from fiction more urgent than ever. Developing a machine learning model to address this challenge not only tackles the issue of fake news but also aids in mitigating its impact by identifying false articles.

In our project, we aim to **identify the combination of preprocessing technique and machine learning model that best performs binary classification of news articles** as either fake and genuine. To accomplish this, we examine the performance of CountVectorizer and N-Gram TF-IDF as preprocessing techniques applied to the following models:

- Logistic Regression
- Random Forest
- Naive Bayes
- Support Vector Machine (SVM) with linear and RBF kernels

The code for our project is available in a Github repository[1].

Our results indicated that a Linear SVM combined with hybrid (character- and word-level TF-IDF) vectorization achieved the best performance across both the initial and new datasets. This suggests that fake news data can often be effectively separated using a linear decision boundary and that fine-grained feature extraction techniques generalize better by recognizing subtle patterns in the fake news articles.

[1]Github Repository Link: https://github.com/rrdu/ECE-580-Project

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2 Literature Review

We conducted a literature review on the paper *Detecting Fake News using Machine Learning and Deep Learning Algorithms* by All-Tanvir et. al [1].

Contributions The main objective of the paper is to compare the performance of five well-known machine learning algorithms on the task of identifying fake news on Twitter related to the 2010 Chile earthquake. The algorithms evaluated were:

- Naive Bayes
- Logistic Regression
- Support Vector Machine (SVM)
- Recurrent Neural Network (RNN)
- Long Short-Term Memory (LSTM)

After conducting their analysis, the authors concluded that Naive Bayes and SVM achieved the best performance.

Methodology The researchers used a pre-labeled dataset of 20,360 entries from a previous study, where each Tweet was labeled as either H (for harassment/fake news) or N (for non-harassment/true news). They investigated the impact of different feature extraction techniques on the performance of each machine learning model. The techniques they examined are listed in the table below:

Table 1: Summary of Feature Extraction Techniques

Technique	Description
Word Count Feature	Based on research linking tweet length to fake news, a word length feature was added to each Tweet.
Count Vectorization	Represents text as a sparse matrix of token counts, where each row is a Tweet and each column is a unique word in the corpus.
TF-IDF	Weights tokens by term frequency and inverse document frequency. Applied at three levels: word-level, n-gram-level (for phrase detection), and character-level (for subword patterns).
Word Embeddings	Uses pretrained word vectors to capture semantic relationships. Applied using embedding layers for deep models like RNN and LSTM.

The researchers tested four main feature extraction methods: word count, count vectorization, TF-IDF (including word-, n-gram-, and character-level variants), and word embeddings. TF-IDF variants were evaluated independently to assess their impact on model performance.

The dataset was divided into a 60/20/20 split for training, testing, and validation for the RNN and LSTM models. For the traditional machine learning models (Naive Bayes, Logistic Regression, and SVM), 2-fold cross-validation was used.

Results The researchers compared the precision, recall, and F1 scores across the different models. They found that among the traditional classifiers, SVM with TF-IDF performed the best. Among the deep learning models, LSTM with kernel initialization showed the highest performance, although it still did not surpass SVM with TF-IDF. The results emphasized the significant impact of feature engineering on model performance, as different feature extraction techniques (such as CountVectorizer and TF-IDF) led to varying levels of accuracy.

Relevance to Our Work This paper is closely related to our project and its findings were instrumental in guiding our baseline experiment design. Like the researchers, we are addressing the task of fake news detection. In our midsemester report, we evaluated the performance of the following models using CountVectorizer and Unigram TF-IDF:

- Logistic regression

- Random Forest
- Naive Bayes
- SVM

Our midsemester results showed that Logistic Regression and SVM (both with TF-IDF) were the best performing models, consistent with the reviewed paper's finding that SVM with TF-IDF achieved the highest performance.

However, our problem differs slightly from that of the research paper. While their dataset focused on short-form Tweets (limited to 280 characters each) centered around a single event—the 2010 Chile earthquake—our dataset consists of long-form news articles covering a broad range of topics in American and world politics. Therefore, if our experiments yield similar results, it would suggest that these models can generalize effectively across different types of text data.

The broader scope of our dataset may introduce challenges to classifier performance. However, successfully fine-tuning our models to perform well on a diverse range of topics would result in a more robust and widely applicable fake news detection system.

Regarding feature extraction techniques, the paper's findings suggest that TF-IDF performs the best. We plan to investigate this further by assessing the impact of N-gram tokenization on TF-IDF and other preprocessing techniques across our models. Additionally, to better handle the broader scope of our data, we will apply further preprocessing steps such as stopword removal, helping models focus on key words, phrases, and character patterns indicative of fake news across various topics. Finally, we will test our top-performing Logistic Regression and SVM variants on a new dataset to evaluate their ability to generalize to slightly different data distributions.

3 Experiment Design

Midsemester Report We used the ISOT Fake News Kaggle Dataset (2016-2017)[2], which consists of 23,481 fake articles from unreliable websites and 21,417 articles from Reuters. For our task, we primarily focused on the title and full text body of the articles. We split the cleaned data into training, validation, and test sets, using a $70\%/15\%/15\%$ split chosen arbitrarily.

We tested four distinct models: Logistic Regression, Support Vector Machine (SVM), Random Forest, and Naive Bayes. Our goal was to determine which models performed best on the dataset. Additionally, we compared the performance of two preprocessing techniques–CountVectorizer and Unigram TF-IDF– across all four models.

Final Report We used the `fake_real_news_dataset`[3]. This dataset has an even 50/50 split between real news (randomly selected from 2015 and 2016 articles published by sources such as the New York Times, Wall Street Journal, and Bloomberg) and fake news (sourced from the Getting Real about Fake News[4] dataset). There is a total of $4,593$ entries, making the dataset significantly smaller than the ISOT dataset. As before, we split the cleaned data into training, validation, and test sets, again using a $70\%/15\%/15\%$ split.

We selected the two best-performing models from the midsemester report (Logistic Regression and SVM) and further analyzed their performance using various N-Gram TF-IDF configurations on the midsemester dataset. For SVM, we also compared the performance of an RBF kernel to the linear kernel used in the midsemester report. Finally, we selected the top-performing model configurations and evaluated whether these results generalized to the new dataset.

[2]ISOT Fake News Dataset Link: `https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset`

[3]Dataset Link: `https://github.com/GeorgeMcIntire/fake_real_news_dataset/tree/main?tab=readme-ov-file`

[4]Dataset Link: `https://www.kaggle.com/datasets/mrisdal/fake-news`

4 Baseline Methodology

Overview

In our midsemester report, we implemented our baseline methodology, which is reiterated below.

For our initial experiments, we aimed to accomplish two main objectives:

1. Identify which of the four selected models (Logistic Regression, Random Forest, Naive Bayes, SVM) performed the best.

2. Determine which feature engineering technique (CountVectorizer or Unigram TF-IDF) yielded the best performance.

To achieve this, we tested each combination of feature engineering technique and model, then evaluated their performance.

Preprocessing

The preprocessing steps we took are as follows:

- **Load, Label, and Combine Datasets:** Since the datasets were provided separately, we manually assigned labels (0 = Fake news, 1 = Real news) to the data points from the respective CSV files and consolidated them into a single dataframe for easy handling later on.

- **Combine Relevant Columns into 'Content':** We merged the 'title' and 'text' columns into a single 'content' column to simplify text processing.

- **Text Cleaning:** We cleaned the 'content' column by converting all text to lowercase, removing punctuation, and removing stopwords (according to the English NLTK stopwords list) to ensure that only key, non-filler words remained.

- **Train/Validation/Test Splits:** The cleaned data was then split into training, validation, and test sets using a $70\%/15\%/15\%$ split.

- **Apply TF-IDF Vectorization:** We applied TF-IDF Vectorization to the training, validation, and test data. TF-IDF (Term Frequency-Inverse Document Frequency) is a weighting technique that assigns higher importance to words that appear frequently within a document but are relatively rare across all documents. Term Frequency (TF) measures how often a term appears in a document, while IDF measures how unique a term is across the dataset. The final TF-IDF score is the product of TF and IDF.

- **Apply CountVectorization:** Separately, we also applied CountVectorization to another copy of the training, validation, test data. CountVectorization converts text data into numerical feature vectors by counting the frequency of each word, allowing the models to identify patterns based on raw word occurrences.

Model Architectures

The following table summarizes the models we implemented in our baseline design, along with details regarding their setup.

Table 2: Summary of Classifier Models and Setup

Model Name	Model Description	Model Setup Specifications
Logistic Regression	• Linear model using input features (e.g., word counts/frequencies). • Outputs probability between 0 (Fake) and 1 (Real).	• `LogisticRegression (scikit-learn)` • L2 regularization (default) • `max_iter=1000` • `random_state=42` • `C=1.0`
Random Forest	• Uses multiple decision trees built on random subsets. • Combines predictions through majority voting.	• `RandomForestClassifier (scikit-learn)` • `random_state=42` • `n_estimators=100` (default) • `max_depth=None` (default) • `criterion='gini'` (default)
Naive Bayes	• Calculates word likelihoods under independence assumptions. • Uses Bayes' Theorem to compute class probabilities.	• `MultinomialNB (scikit-learn)` • `alpha=1.0` (default) • `fit_prior=True` (default) • `force_alpha=False` (default) • `class_prior=None`
SVM*	• Finds the optimal boundary between classes using support vectors. • Maximizes the margin between class-separating hyperplanes.	• `SVC (scikit-learn)` • `kernel='linear'` • `C=1` • `random_state=42`

Training Procedure

We followed the training procedure outlined below:

1. Apply the appropriate feature extraction technique (Unigram TF-IDF or CountVectorization) to the training set.
2. Train each model on the training set, then validate it on the validation set.
3. Evaluate the model's performance on the test set, recording the results in a confusion matrix.

Evaluation Metrics

We implemented a function to compute the following evaluation metrics for every model's predictions on the test set.

In the context of our problem, the following definitions are used:

Table 3: Confusion Matrix Label Definitions

Label	Definition
True Positive (TP)	The article was **fake**, and the model predicted it as **fake**.
False Positive (FP)	The article was **real**, but the model predicted it as **fake**.
True Negative (TN)	The article was **real**, and the model predicted it as **real**.
False Negative (FN)	The article was **fake**, but the model predicted it as **real**.

For each model, the following evaluation metrics were used:

- **Precision:** Assesses the *accuracy of the model's fake news predictions*. High precision indicates that the model *doesn't falsely flag real news* as fake.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

- **Recall:** Assesses the *completeness of the model's fake news detection*. High recall indicates that the model *is good at catching fake news* and doesn't miss any.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

- **F1 Score:** Assesses the *balance between precision and recall*. It is the harmonic mean of precision and recall.

$$\text{F1 Score} = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}} \tag{3}$$

Additionally, a confusion matrix and classification report were printed for evaluation on both the test and validation sets. The confusion matrix provides a visual representation of how many data points were correctly or incorrectly classified, and what category (TP/FP/TN/FN) they belonged to.

5 Extended Experiment Methodology

Overview

In the final portion of our experiment, we focused on the two best-performing models from the previous phase (Logistic Regression and SVM), as well as the best-performing feature engineering method (TF-IDF).

For the second phase of our experiment, we aimed to accomplish the following tasks:

1. Investigate whether an SVM with an RBF kernel can outperform an SVM with a linear kernel.
2. Identify which N-Gram variant (bigram, trigram, character-level N-gram, or a hybrid vectorizer) of TF-IDF performs best on Logistic Regression and the ideal SVM kernel.
3. Assess the performance of the top two N-Gram TF-IDF variants and CountVectorizer on Logistic Regression and SVM using a new dataset.

We followed the same logical path as the baseline experiment by testing every combination of the N-Gram TF-IDF variants and CountVectorizer on Logistic Regression and Linear SVM. We also tested RBF SVM using our baseline setup.

Preprocessing

We followed the same general preprocessing steps as in the baseline experiment, with a few modifications to Text Cleaning and TF-IDF Vectorization:

1. **Load, Label, Combine Datasets:** For the original dataset, we retained the same loading, labeling, and combining procedure. The new dataset was already combined into a single CSV, so merging was not necessary.
2. **Combine Relevant Columns into 'Content':** As before, we consolidated the 'title' and 'text' columns into one 'content' column for easy processing.
3. **Text Cleaning:** During the N-Gram TF-IDF experimentation, we did NOT remove stopwords for the bigram vectorizer and one trigram vectorizer. Since bigrams and trigrams focus on word phrases, keeping common words such as stopwords can actually help capture meaningful combinations.

4. **Train/Validation/Test Splits:** We split the cleaned data into training, validation, and test sets using the same arbitrary $70\%/15\%/15\%$ split.

5. **Apply TF-IDF Vectorization:** While in the baseline experiment we only applied unigram TF-IDF, here we expanded our experimentation to include the following N-Grams configurations:

Table 4: Summary of Vectorization Techniques

Vectorization Type	Description
Unigram TF-IDF	Word-level single tokens.
Bigram/Trigram TF-IDF	Word pairs or triples (n-grams).
Character-level TF-IDF	3–5 character n-grams inside words.
Hybrid TF-IDF	Combination of word-level and character-level TF-IDF features.

Model Architecture

We retained the same Logistic Regression and SVM model architectures from the baseline experiment design. The only difference was that we also explored an SVM with an RBF kernel to evaluate how introducing a non-linear boundary to separate the data classes would impact performance.

Training Procedure

We followed a similar training procedure as outlined in the baseline methodology:

1. The appropriate feature extraction technique (N-Gram TF-IDF or CountVectorization) was applied to the training set.
2. The models (Logistic Regression, SVM with Linear Kernel) were trained on the training set. (Note: The SVM with RBF kernel was trained using data that had either Unigram TF-IDF and CountVectorizer applied to it, to allow for easy comparison with earlier results).
3. Their performance was then evaluated on the test set, and the outputs were recorded in confusion matrices.
4. This entire process was repeated on the new dataset, using the top-performing feature extraction and model pairs from the initial ISOT dataset assessment.

Evaluation Metrics

For consistency, we retained the same evaluation metrics–Precision, Recall, F1 score–as used in the baseline methodology. We also plotted confusion matrices for both the test and validation set to provide a visual assessment of classification performance.

6 Results

The following results are divided into two portions:

1. A summary of the results obtained from our baseline experiment design.
2. A comprehensive presentation of the results from our extended experiment design.

Baseline Experiment Design Results

Table 5 below displays the precision, recall, and F1 scorse obtained from each of our initial models and their feature engineering processes.

Table 5: Evaluation Metrics for Models

Model	Precision		Recall		F1	
	TF-IDF	CountVectorizer	TF-IDF	CountVectorizer	TF-IDF	CountVectorizer
Logistic Regression	98.48%	99.75%	98.94%	99.68%	98.71%	99.71%
Random Forest	99.03%	98.88%	99.28%	99.47%	99.16%	99.17%
Naive Bayes	93.29%	95.25%	95.29%	96.91%	94.28%	96.08%
SVM	99.40%	99.71%	99.47%	99.47%	99.43%	99.59%

Table 6 below shows the results of fine-tuning the C hyperparameter for the Linear SVM model. The results indicate that setting $C = 1$ led to the best performance across our evaluation metrics. As a result, we retained C = 1 for all future Linear SVM instantiations in the extended experiments.

Table 6: C Finetuning Evaluation Metric Results (Linear SVM)

C Value	Precision	Recall	F1
C = 0.01	99.7627%	99.7624%	99.7624%
C = 0.1	99.7625%	99.7624%	99.7624%
C = 1	99.7921%	99.7921%	99.7921%
C = 10	99.7921%	99.7921%	99.7921%
C = 100	99.7921%	99.7921%	99.7921%

Finally, Table 7 summarizes the best-performing models based on their F1 scores. From these results, we can see that CountVectorizer was the most effective feature engineering method, while Logistic Regression and SVM were the best-performing models.

Table 7: Model Vectorizer and F1 Score Summary

Model	Best Preprocessing Method	Best F1	Model Rank
Logistic Regression	CountVectorizer	99.71%	1
Random Forest	CountVectorizer	99.17%	3
Naive Bayes	CountVectorizer	96.08%	4
SVM	CountVectorizer	99.59%	2

Extended Experiment Design Results

Table 8 below summarizes the results of implementing SVM with RBF Kernel on the original ISOT dataset. We see that CountVectorizer outperforms Unigram TF-IDF, and that it slightly beats the F1 score from Linear SVM on the same dataset.

For the sake of conciseness, we did not include all the confusion matrices that were generated. However, they can found in the notebooks in the Github repository [5]

Table 8: Test Set Evaluation Metrics for RBF Kernel SVM on ISOT

Feature Engineering	Precision	Recall	F1
Unigram TF-IDF	99.34%	99.34%	99.34%
CountVectorizer	99.65%	99.40%	99.53%

Table 9 below shows the results of fine-tuning the C hyperparameter for the RBF SVM model. The results indicate that setting C = 100 led to the best performance across our evaluation metrics. Here,

[5]Github Repository Link: `https://github.com/rrdu/ECE-580-Project`

we observe that as C increases, precision, recall, and F1 scores improve; however, with Linear SVM, the scores peaked at C = 1. The Linear SVM's performance peaked at C = 1 due to its simpler, linear nature, while the RBF SVM kept improving as C increased, benefiting from its ability to model complex nonlinear boundaries by creating more flexible decision surfaces as the regularization decreased. As C grew higher, the RBF SVM was allowed to fit the training data more closely, capturing intricate patterns in the data that the Linear SVM couldn't. However, beyond a certain point, both models could start to overfit if C is set too high.

Table 9: C Finetuning Evaluation Metric Results (RBF SVM)

C Value	**Precision**	**Recall**	**F1**
C = 0.01	92.9528%	92.8731%	92.8614%
C = 0.1	98.0552%	98.0549%	98.0548%
C = 1	99.5843%	99.5843%	99.5843%
C = 10	99.6585%	99.6585%	99.6585%
C = 100	99.6883%	99.6882%	99.6882%

Table 10 presents the results of applying various N-Gram feature extraction methods with Logistic Regression on the original ISOT dataset. Based on the F1 scores, Character-level N-Gram and Hybrid N-Gram yielded the best performance.

Table 10: Test Set Evaluation Metrics for N-Gram Logistic Regression on ISOT

TF-IDF Variant	**Precision**	**Recall**	**F1**
Bigram (no stopwords)	98.24%	99.09%	98.66%
Trigram (no stopwords)	97.91%	99.25%	98.57%
Trigram (with stopwords)	97.57%	99.12%	98.34%
Character-level	98.54%	99.09%	98.82%
Hybrid	98.70%	99.31%	99.00%

Similar to Table 10, Table 11 summarizes the results of applying various N-Gram feature extraction techniques with Linear SVM on the original ISOT dataset. Once again, Character-Level N-Gram and Hybrid N-Gram achieved the best performance.

Table 11: Test Set Evaluation Metrics for N-Gram Linear SVM on ISOT

TF-IDF Variant	**Precision**	**Recall**	**F1**
Bigram (no stopwords)	99.25%	99.65%	99.45%
Trigram (no stopwords)	98.88%	99.68%	99.28%
Trigram (with stopwords)	98.91%	99.75%	99.33%
Character-level	99.44%	99.84%	99.64%
Hybrid	99.56%	99.84%	99.70%

Lastly, Table 12 presents the results of the top 3 feature extraction techniques (Character-Level N-Gram, hybrid N-Gram, and CountVectorizer) on the new dataset. We observe that Linear SVM with hybrid TF-IDF achieved the best overall performance, yielding an accuracy of (94.13%).

Table 12: Evaluation Metrics for Models on New Dataset

Model	Feature Engineering	Precision	Recall	F1
Logistic Regression	Character-level TF-IDF	89.91%	90.69%	90.30%
Logistic Regression	Hybrid TF-IDF	92.83%	90.04%	91.60%
Logistic Regression	CountVectorizer	94.18%	89.53%	91.80%
Linear SVM	Character-level TF-IDF	94.04%	91.86%	92.94%
Linear SVM	Hybrid TF-IDF	94.97%	93.31%	94.13%
Linear SVM	CountVectorizer	89.91%	90.69%	90.30%

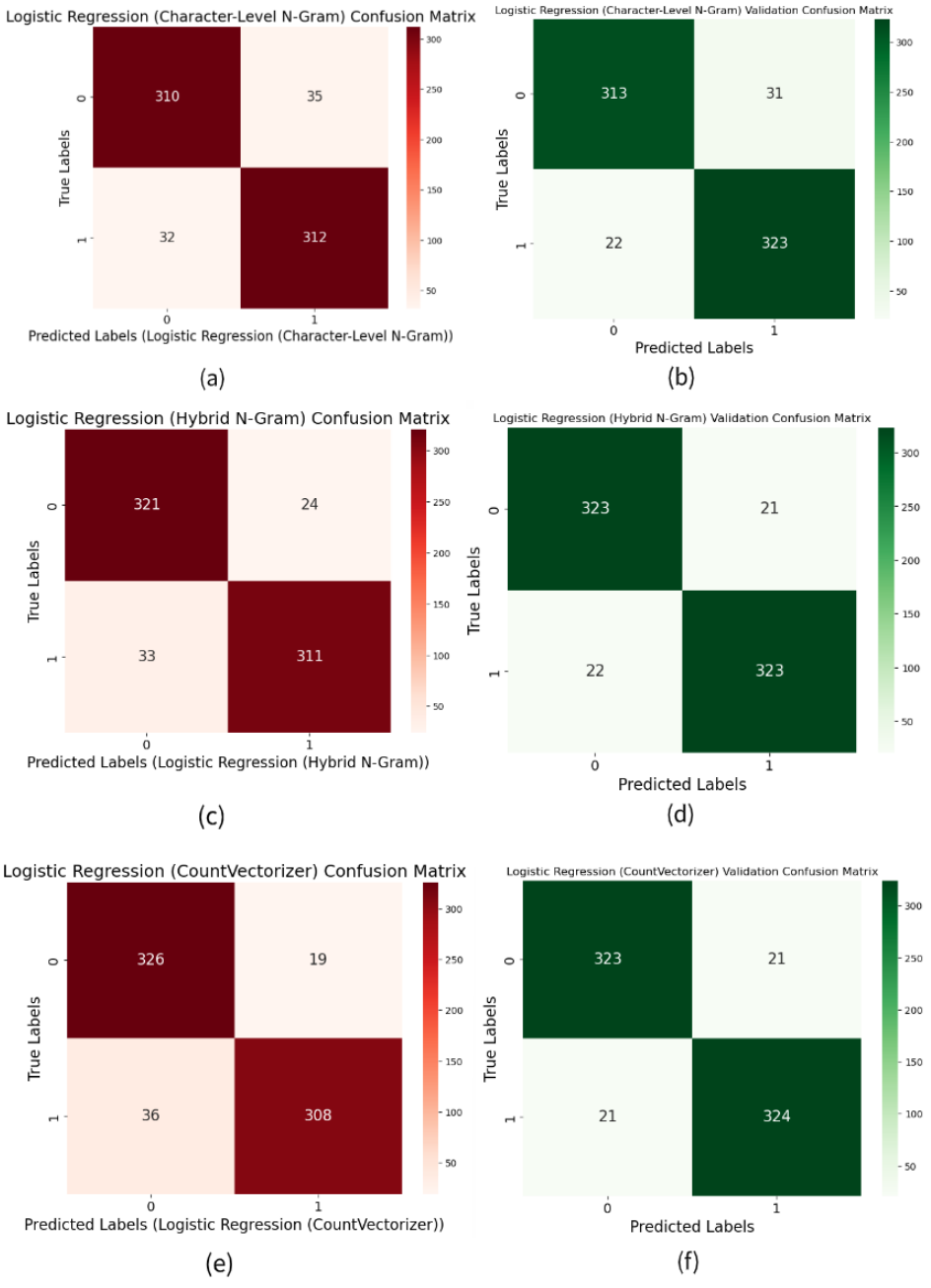

Figure 1: Confusion Matrices for Logistic Regression on New Dataset

Figure 1 above shows the confusion matrices for the Logistic Regression models trained with the top three techniques (Character-level N-Gram TF-IDF, Hybrid N-Gram TF-IDF, and CountVectorizer) on the new dataset. The red confusion matrices represent the test set, while the green ones correspond to the validation set.

Similarly, Figure 2 below displays the confusion matrices for the linear SVM models trained with the same top three techniques as the Logistic Regression models in Figure 1.

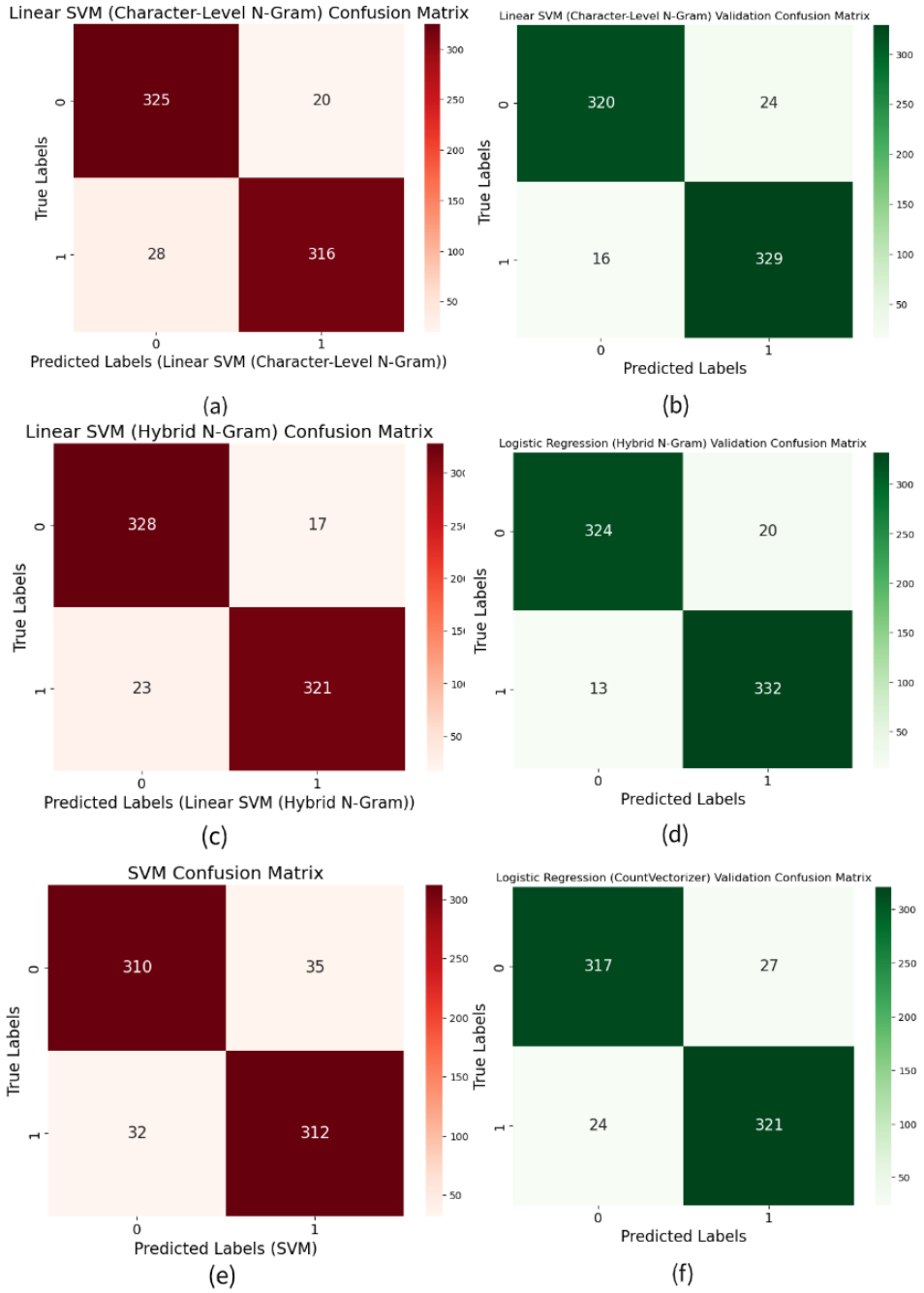

Figure 2: Confusion Matrices for Linear SVM on New Dataset

7 Analysis

The analysis is divided into three sections:

1. A brief analysis of the baseline results from the midsemester report.
2. A detailed analysis of the extended experiment results.
3. A comprehensive discussion of the general trends, successes, and limitations observed across our experiments.

Baseline Results Analysis

The baseline experiments revealed that among the models tested with Unigram TF-IDF and CountVectorizer, SVM and Logistic Regression emerged as the top performers based on F1 score. Notably, SVM with CountVectorizer achieved the highest F1 score of 99.71%, demonstrating the effectiveness of combining simple feature extraction with a linear decision boundary for fake news detection. The superior performance of CountVectorizer over Unigram TF-IDF suggests that a raw frequency-based representation was more capable of capturing the distinctive features between fake and real news articles in a manner that the models could effectively interpret.

In terms of hyperparameter tuning, we found that the optimal value for the regularization parameter (C) was 1, which aligns with the standard regularization setting. This indicates that a straightforward linear SVM was sufficient for accurate binary classification on the ISOT dataset.

Extended Experiment Results Analysis

In our extended experiments, we further explored the impact of N-Gram TF-IDF vectorization on model performance and the effect of using an RBF kernel on SVM. Among the tested N-Gram TF-IDF variants, character-level TF-IDF and hybrid TF-IDF representations outperformed the others. Logistic Regression with hybrid TF-IDF achieved a highest F1 score of 99.00%, while Linear SVM with hybrid TF-IDF achieved a highest F1 score of 99.70%. Since the hybrid vectorization format includes character-level TF-IDF, we can conclude that the fine-grained nature of character-level features effectively captures specific patterns in fake news that might be overlooked by larger features such as bigrams and trigrams.

Regarding the RBF kernel for SVM, we found that it resulted in a very slight improvement when paired with CountVectorizer, achieving an F1 score of 99.53%, compared to the 99.59% achieved by the Linear SVM. However, given the small magnitude of this improvement, it suggests that the data is likely already linearly separable, and the RBF kernel doesn't add substantial value in this case.

When testing on the new, smaller dataset (4,593 entries compared to ISOT's 44,898 entries), we observed more pronounced differences in model performance. The reduced size of the new dataset made it more challenging for the models, resulting in lower performance across the board compared to the ISOT dataset. This is reflected in the confusion matrices, which show more off-diagonal elements (indicating misclassifications). The off-diagonal elements suggest that models were more prone to incorrectly marking real news as fake due to the limited training data.

Despite the reduced dataset size, **Linear SVM with hybrid TF-IDF remained the best performing model**, achieving an F1 score of 93.13%. The strong performance of hybrid TF-IDF on both datasets suggests that combining character- and word-level feature extraction provides a more robust and generalizable approach to fake news detection, as it can capture patterns of varying lengths that are more representative of fake news content.

Overall Analysis

Overall, the strong performance of fine-grained feature extraction techniques, such as character-level and hybrid TF-IDF, combined with effective linear classifiers like logistic regression and linear SVM, resulted in a model that can accurately identify fake news.

We successfully demonstrated that machine learning models, particularly logistic regression and SVM, can significantly aid the task of fake news detection with high accuracy. The consistent

performance (above 90% scores for the best-performing models across all datasets) showcases the robustness of these models in handling fake news detection.

However, there were limitations in our experiments. One major issue was the small size of our second dataset, which constrained the models' ability to generalize effectively due to the limited training data. Additionally, while TF-IDF and CountVectorizer performed well on the datasets tested, they rely on surface-level features and may struggle to capture more complex patterns in fake news texts. Finally, the slight bias towards misclassifying real news as fake (as indicated by the confusion matrices) suggests that there is room for improvement, especially in handling borderline or ambiguous cases.

8 Conclusion

In our project, we successfully analyzed various machine learning models for the task of fake news detection. By testing them with various feature engineering techniques across multiple datasets, we gained insight into which classifiers performed best for this task.

Our results showed that linear classifiers, such as logistic regression and linear SVM, achieved the highest performance when combined with detailed feature extraction methods like character-level and hybrid TF-IDF. These models consistently demonstrated strong performance scores ($94\% - 99\%+$) on both the larger and smaller datasets. This suggests that even relatively simple models can effectively learn from the data and correctly classify the majority of fake news articles encountered during testing. However, we did observe a decline in performance on the second dataset, indicating some difficulty with more borderline cases.

Future work could explore whether linear classifiers maintain their effectiveness when applied to more nuanced fake news articles. Additionally, investigating the performance of more complex models, such as BERT, would be an interesting avenue for further research. Nevertheless, our findings demonstrate that simple linear classifiers, when paired with the right preprocessing techniques, can still achieve competitive results in fake news detection.

References

[1] Abdullah-All-Tanvir, Ehesas Mia Mahir, Saima Akhter, and Mohammad Rezwanul Huq. *Detecting Fake News using Machine Learning and Deep Learning Algorithms*. In 7th International Conference on Smart Computing & Communications (ICSCC), 2019.