Detecting Fake News with Machine Learning Techniques

This is a group project for CAPP 30254: Machine Learning for Public Policy

Done By: Big Brother Debunker
Team Member: Dingwei Liu & Qi Zhao

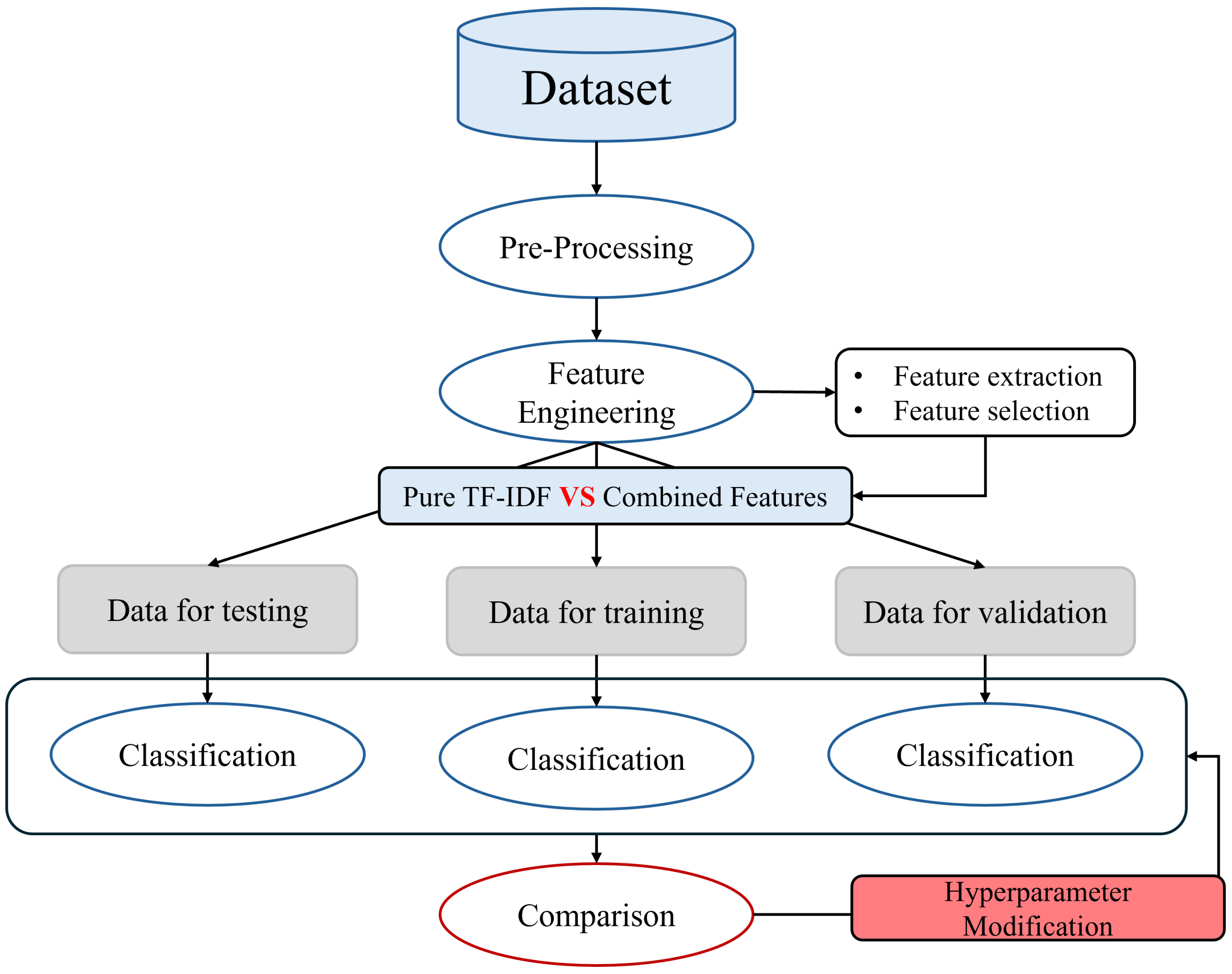
1. Introduction and research problem

Misinformation in the media, exemplified by events like the Arab Spring, can influence public opinion and incite social unrest. Traditional verification methods, reliant on human effort, struggle with the vast volume of online content.

This project aims to create a scalable tool that detects and classifies fake news, assisting users and tech companies in identifying reliable information and reducing the spread of misinformation. We proposes an automated solution to detect fake news by analysing textual features such as words, phrases, sources, and titles using machine learning. Utilizing a labelled dataset from Kaggle, we applied supervised machine learning algorithms and feature selection techniques to develop a predictive model. The model's accuracy is determined through testing on unseen data.

2. Methodology

We adhere to a structured approach consisting of data pre-processing, feature engineering, classification, and analysis. The detailed steps are as follows.



2.1. Data Pre-processing

- We begin by checking for missing values (NAs) across all columns and confirm that the label column contains no NAs. Since our goal is to extract additional attributes from other columns, there is no need to remove any columns at this initial stage.
  - From the label distribution, fake news and non-fake news are evenly distributed. We got balanced dataset and don't need to oversample.

  Shape
  - Rows: 20800
  - Columns: 5
    - id, title, author, text, label

Missing values

id	title	author	text	label
0	558	1957	39	0

Label distribution

label	0	1
count	10387	9855

- For the titles and text, we pre-process by removing non-English words, special characters, HTML tags, and stopwords. Additionally, we standardize text casing, perform lemmatization to reduce words to their base forms, strip extra spaces, and filter out common but uninformative words to enhance the clarity and effectiveness of our data for classification tasks.
- Split news source from the title. Source refers to the organization where the news is coming from. For example, The New York Times. However, only part of the title have source information. We believe containing source information or not is caused by crawling, which is irrelevant to whether one news is fake.

  For example, id=80:

  - Before:

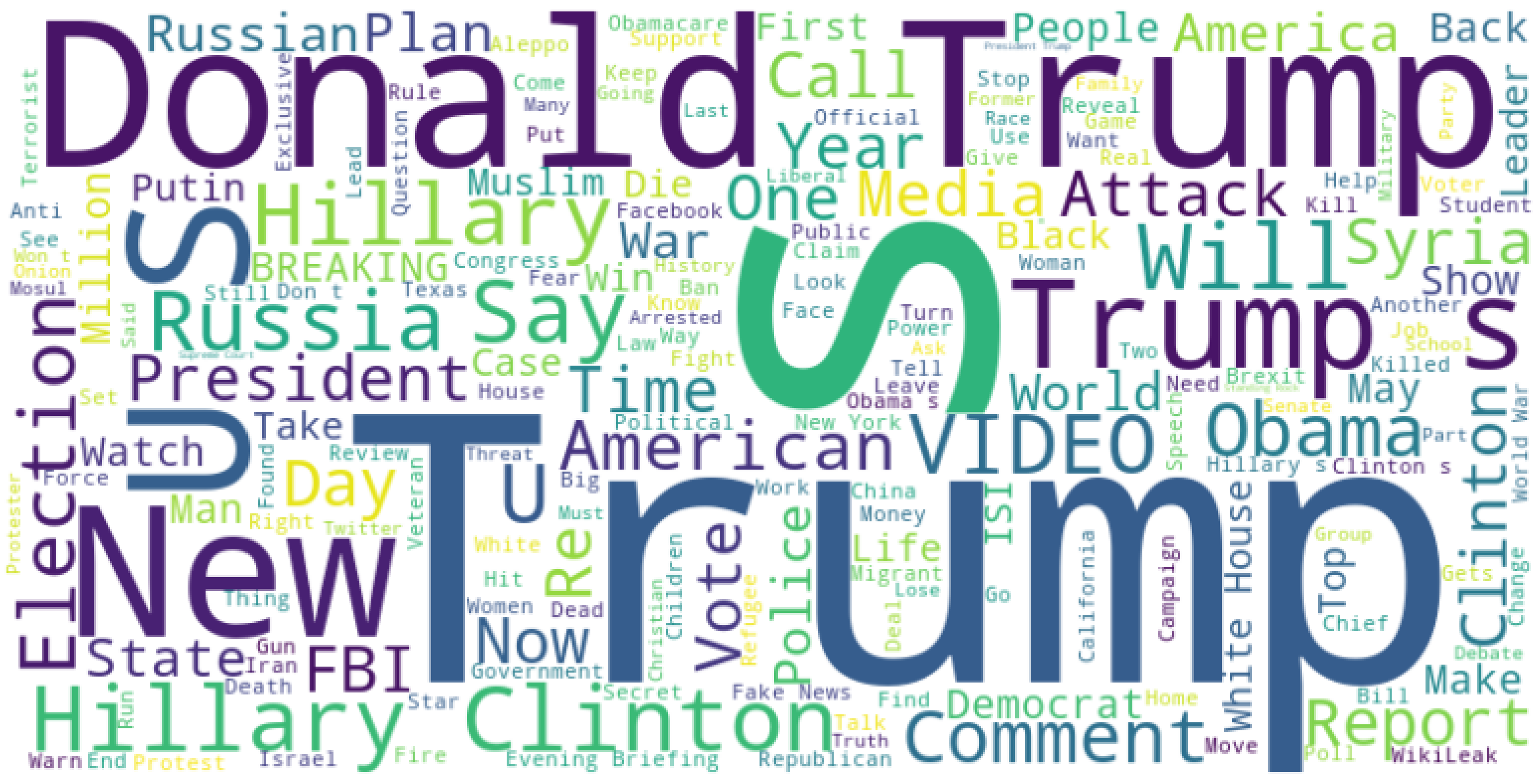
    ```
    Louisiana, Simone Biles, U.S. Presidential Race: Your Tuesday Evening Briefing - The New York Times
    ```

  - After:

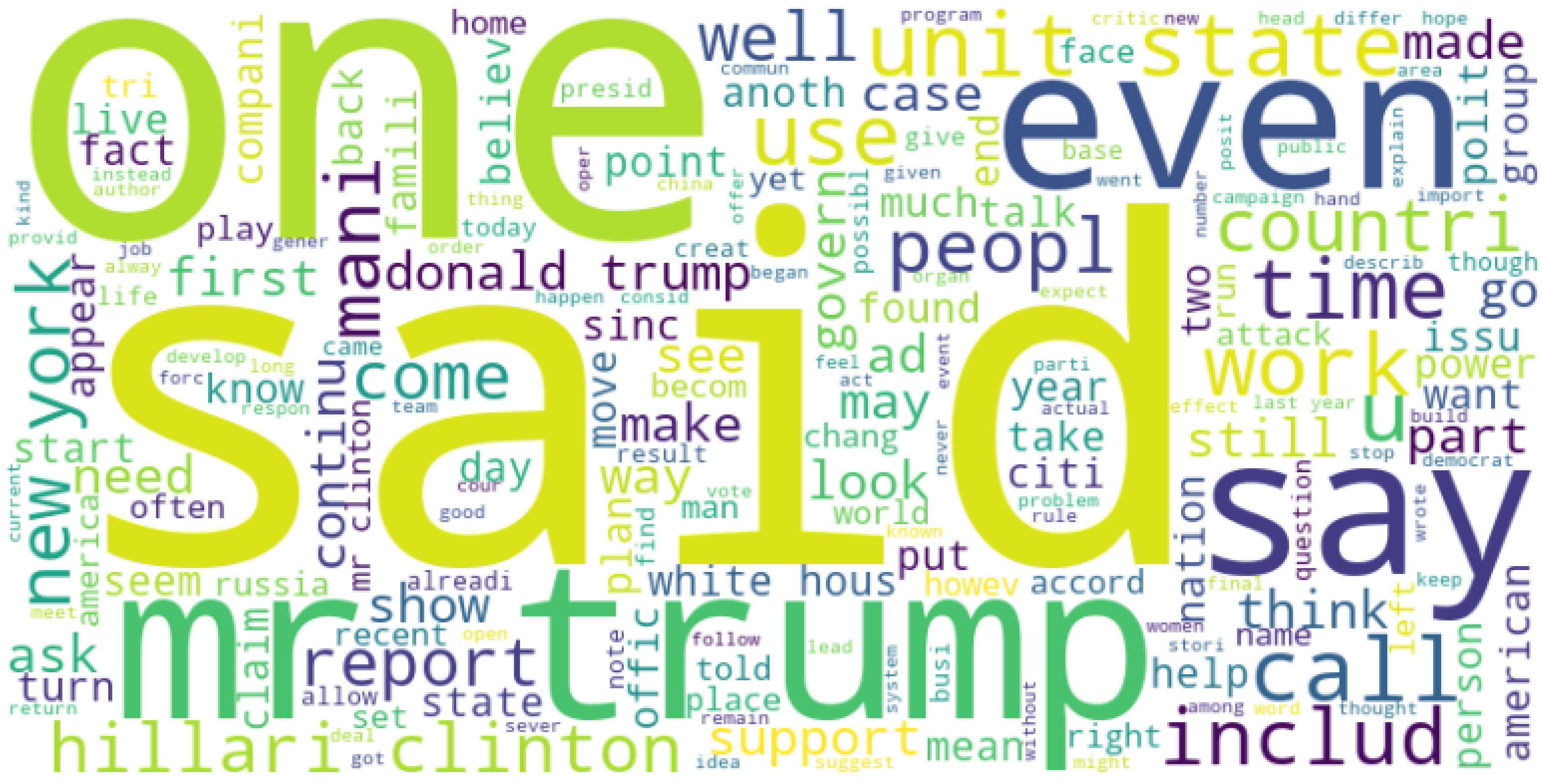
    ```
    Louisiana, Simone Biles, U.S. Presidential Race: Your Tuesday Evening Briefing
    ```

- Stemming: Stemming is performed to reduce words to their base form, thereby decreasing the number of word variations. This helps machine learning models better understand and process language data because different forms of the same root word (such as "run", "running", "runner") are semantically similar. With stemming, the model can treat these variations as the same word, reducing the complexity of the feature space and improving processing efficiency and model performance.
- Explore the word count:
  - **Word Cloud for Titles**: This image highlights the most prominent words used in titles or headings. Key terms include "Trump," "Hillary," "Clinton," and "Obama," indicating a political context. Other significant words such as "America," "President," "Russia," and "Media" suggest topics related to international relations, leadership, and news coverage. The size of the words "Trump" and "Hillary" suggests they are among the most frequently mentioned, pointing to a dataset possibly from the time around the 2016 U.S. Presidential election.
  - **Word Cloud for Stemmed Texts**: This cloud shows stemmed versions of words (base forms), which aids in grouping similar terms together, enhancing the text analysis by consolidating different forms of a word into a single representation. Dominant terms here include "Trump," "state," "presid" (likely stemming from "president"), and "peopl" (stemmed from "people"). This visualization emphasizes themes similar to the first cloud but from a broader textual analysis, potentially from article bodies or extensive texts discussing similar political themes.

Word Cloud for Titles



Word Cloud for Stemmed Texts



3. Feature Engineering

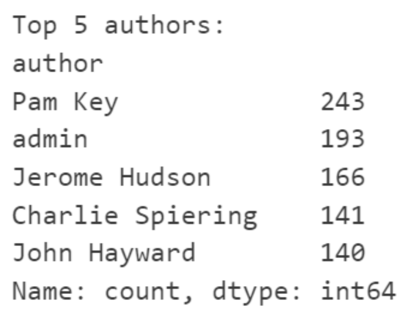
3.1. Feature Extraction

Based on our literature review and initial data exploration—where we discovered a significant correlation between the news source and the incidence of fake news—we have expanded our feature extraction beyond the traditional TF-IDF. This will allow for more effective classification.

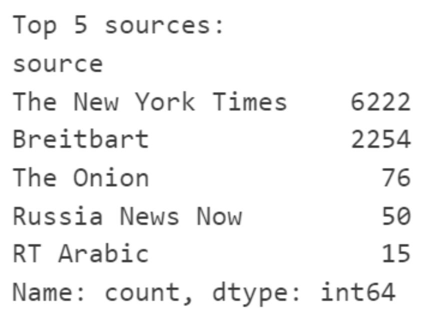
Initially, we construct our baseline model using **TF-IDF features**. We generate sparse vector representations of titles and article bodies by calculating the term frequency of each n-gram and normalizing these by their inverse document frequency. We begin by fitting a `TfidfVectorizer` to the concatenated texts of titles and bodies to establish the vocabulary. Subsequently, using this vocabulary, we separately fit and transform the n-grams of titles and bodies into sparse vectors. Additionally, we calculate the cosine similarity between the vectors of the titles and bodies.

Furthermore, we have extracted additional attributes such as:

- **check whether the news have title**
    - Ratio of no_title: 0.02599
- **check whether the news have author**
- Ratio of no_author: 0.0866



```
Top 5 authors:
author
Pam Key             243
admin               193
Jerome Hudson       166
Charlie Spiering    141
John Hayward        140
Name: count, dtype: int64
```



```
Top 5 sources:
source
The New York Times    6222
Breitbart             2254
The Onion               76
Russia News Now         50
RT Arabic               15
Name: count, dtype: int64
```

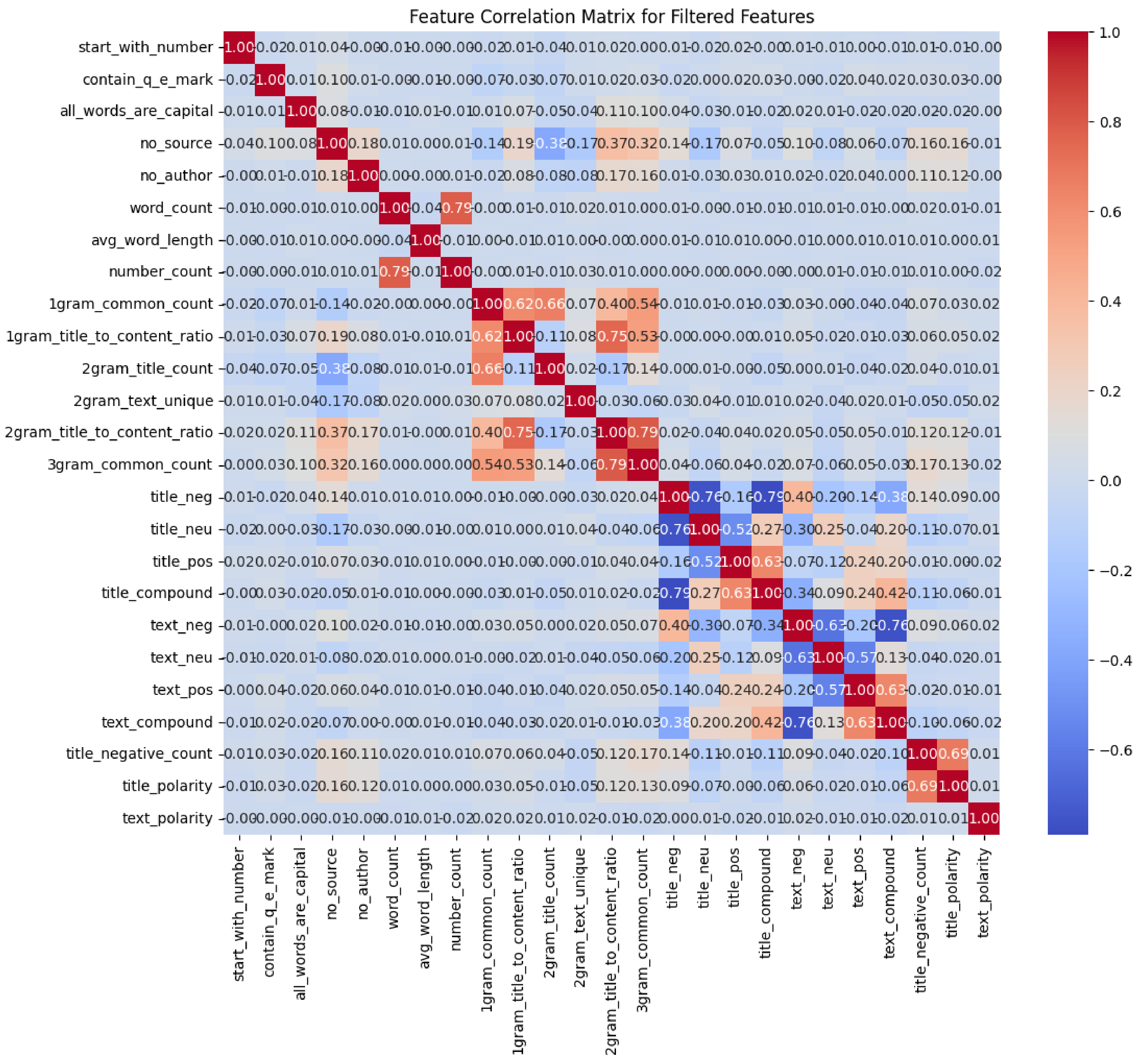
- **check whether the news have source**
    - Ratio of no_source: 0.5558
- **check whether the news' title starts with number**
    - Ratio of start_with_number: 0.0214
- **check whether title contains ? and/or ! marks**
    - Ratio of contain_q_e_mark: 0.0812
- **check whether all words are capital in title**
    - Ratio of all_words_are_capital: 0.0076
- Perform feature extraction by choosing lexical features, Such as **word count, average word length, length of article, number count, number of sections of speech (adjective).**
    - word count average: 746.7307
    - word length average: 4.83696
    - article length average: 4607.858
    - number count average: 15.5381
    - adjective word count average: 55.6835
- Basic count for title and text tuple. We takes the **uni-grams, bi-grams and tri-grams** and creates various counts and ratios which could potentially signify how a body text is related to a title. Specifically, it counts how many times a gram appears in the title, how many unique grams there are in the title, and the ratio between the two. The same statistics are computed for the body text, too. It then calculates how many grams in the title also appear in the body text, and a normalized version of this overlapping count by the number of grams in the title. The results are saved in the pickle file which will be read back in by the classifier.
    - 1-gram title to content ratio average: 0.7289
    - 2-gram title to content ratio average: 0.3902
    - 3-gram title to content ratio average: 0.2499
- **Sentiment features:** we uses the Sentiment Analyzer in the `NLTK` package to assign a sentiment polarity score to the headline and body separately. For example, negative score means the text shows a negative opinion of something. This score can be informative of whether the body is being positive about a subject while the headline is being negative. But it does not indicate whether it's the same subject that appears in the body and headline; however, this piece of information should be preserved in other features.
- **Polarity features:** The polarity of the title and text is determined by counting the number of negative sentiment words. Here, the NRC Word-Emotion Association Lexicon is used to obtain a list of negative emotion associated English words. The identity of a negative word is determined by whether the word is found on the negative word list or not. We assign the polarity of a corpus based on whether it has odd or even number of negative words.

3.2. Feature Selection

We employ a two-stage method for selecting features for our model, aiming to optimize the predictive power for detecting fake news.

First Stage: We begin by evaluating new features using information gain and Pearson correlation, both with the label and among the features themselves, to assess their predictive relevance. If two features demonstrate a correlation exceeding 90%, we retain the feature with the higher information gain and discard the other. Additionally, features that exhibit zero information gain or no correlation with the label are removed.

```
                      Feature  Info Gain  Pearson Correlation
0             start_with_number   0.000813             0.043109
1              contain_q_e_mark   0.011150             0.143954
2         all_words_are_capital   0.009137             0.093875
3                     no_source   0.368399             0.786671
4                     no_author   0.060549             0.321512
5                      no_title   0.000000                  NaN
6                    word_count   0.059461            -0.098273
7               avg_word_length   0.022774             0.006194
8             length_of_article   0.059085            -0.102989
9                  number_count   0.023913            -0.104081
10              adjective_count   0.040619            -0.043845
11            1gram_title_count   0.112539            -0.329969
12           1gram_title_unique   0.110651            -0.340548
13             1gram_text_count   0.053483            -0.109002
14            1gram_text_unique   0.057982            -0.138330
15           1gram_common_count   0.054153            -0.150776
16  1gram_title_to_content_ratio   0.165101             0.146686
17            2gram_title_count   0.110483            -0.329962
18           2gram_title_unique   0.113563            -0.336452
19             2gram_text_count   0.057592            -0.109002
20            2gram_text_unique   0.062287            -0.113465
21           2gram_common_count   0.081461             0.260135
22  2gram_title_to_content_ratio   0.212285             0.412378
23            3gram_title_count   0.107894            -0.329728
...
37          title_negative_count   0.026142             0.196959
38                title_polarity   0.031088             0.194181
39            text_negative_count   0.037503            -0.043459
40                 text_polarity   0.002642            -0.019876
```



Second Stage: In this phase, we integrate information gain, Pearson correlation, and Recursive Feature Elimination (RFE) to refine our feature selection. This approach combines the strengths of each method, enhancing the model's ability to discriminate between genuine and fake news by focusing on the most informative and statistically significant features.

Based on this rigorous selection process, we ultimately identified 10 features that, alongside our baseline TF-IDF features, are used to train our model. This strategic combination allows us to leverage both textual and contextual information, significantly boosting the model's accuracy and robustness.

3.3. Classification

We partition the dataset into training, testing and validation sets with ratios of 70%, 20%, and 10%, respectively, using Python's sklearn library. After applying the chosen algorithms, we document the processes and results in a Jupyter notebook (.ipynb file) that encapsulates the classification models.

- We will be using several models.
- SVC
- Logistic Regression
- Perceptron
- Random Forest
- Gaussian Naive Bayes

4. Analysis

We evaluate the precision of our model using the test portion of the dataset and generate a confusion matrix. We assess the model's performance by examining accuracy, precision, recall, F1-score, ROC curve, and AUC for both fake and real news categories. These metrics help in identifying not just the overall correctness (accuracy), but also the balance between detecting true positives and avoiding false positives (precision and recall), overall model's diagnostic ability (ROC curve and AUC), and the harmonic mean of precision and recall (F1-score), ensuring robustness in varied scenarios.

4.1. Benchmark

- Accuracy:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

  - Using %accuracy is acceptable as the label is **well-balanced**
- Precision

$$Precision = \frac{TP}{TP+FP}$$

- Recall

$$Recall = \frac{TP}{TP+FN}$$

- F1-score:

$$F_1\ Score = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP+1/2(FP+FN)}$$

- ROC AUC Score

$$\text{AUC} = \sum_{i=1}^{n-1} (\text{FPR}_{i+1} - \text{FPR}_i) \cdot \frac{\text{TPR}_{i+1} + \text{TPR}_i}{2}$$

4.2. Initial Result without Hyper Parameter Modifying

Here we report 5 benchmarks of 5 traditional machine learning methods, using default hyper-parameter from sklearn. As an example, for regularization parameter C, we use default C=1.0.

We report 2 types of metrix: TF-IDF and Combined Features.

Evaluation metrix for TF-IDF

	Accuracy	Precision	Recall	F1	AUC
SVC	94.96%	93.03%	96.39%	94.68%	95.05%
Logistic Regression	93.89%	92.14%	94.97%	93.53%	93.96%
Decision Tree	87.02%	85.93%	86.21%	86.07%	86.96%

	Accuracy	Precision	Recall	F1	AUC
Perceptron	93.84%	91.69%	95.40%	93.51%	93.94%
RandomForest	90.43%	93.11%	85.78%	89.29%	90.13%
GaussianNB	82.38%	81.07%	81.07%	81.07%	82.30%

Evaluation metrix for Combined features

	Accuracy	Precision	Recall	F1	AUC
SVC	96.02%	94.00%	97.70%	95.82%	96.14%
Logistic Regression	96.13%	94.39%	97.48%	95.91%	96.22%
Decision Tree	95.11%	94.95%	94.53%	94.74%	95.07%
Perceptron	95.11%	95.96%	93.44%	94.68%	95.00%
RandomForest	95.16%	97.01%	92.45%	94.68%	94.99%
GaussianNB	82.48%	81.53%	80.63%	81.08%	82.36%

Based on above tables, we can see that Support Vector Classifier (SVC) and Logistic Regression, particularly when trained with combined features, exhibit superior performance across nearly all metrics. SVC with TF-IDF features achieves an accuracy of 94.96%, a recall of 96.39%, and an F1-score of 94.68%, making it highly effective at correctly identifying fake news cases. This model demonstrates a remarkable capability in minimizing false negatives. In contrast, Logistic Regression shows slightly improved performance with combined features, reaching an accuracy of 96.13% and an F1-score of 95.91%. It also maintains a balance between precision (94.39%) and recall (97.48%), highlighting its robustness in handling both fake and real news categorization with reduced error rates.

Furthermore, the Random Forest model, when enhanced with combined features, exhibits the highest precision (97.01%) among the tested models, underscoring its reliability in predicting true positives with minimal false positives. However, Decision Tree and Perceptron models, despite their relative simplicity, perform comparably under the condition of combined features, achieving accuracies around 95%. This suggests that, under well-engineered features, even less complex models can achieve performance close to more sophisticated ones.

In conclusion, Logistic Regression and SVC stand out as the most effective models for fake news detection, especially when utilizing a combination of features. These models not only achieve high accuracy but also ensure a balanced sensitivity and specificity, making them suitable for practical applications where both types of errors (false positives and false negatives) carry significant consequences. Therefore, for critical applications, an ensemble approach involving Logistic Regression and SVC might offer enhanced performance by leveraging the strengths of both models to better generalize across different fake news scenarios.

4.3. Further Result with Selected Hyper-Parameter Modifying

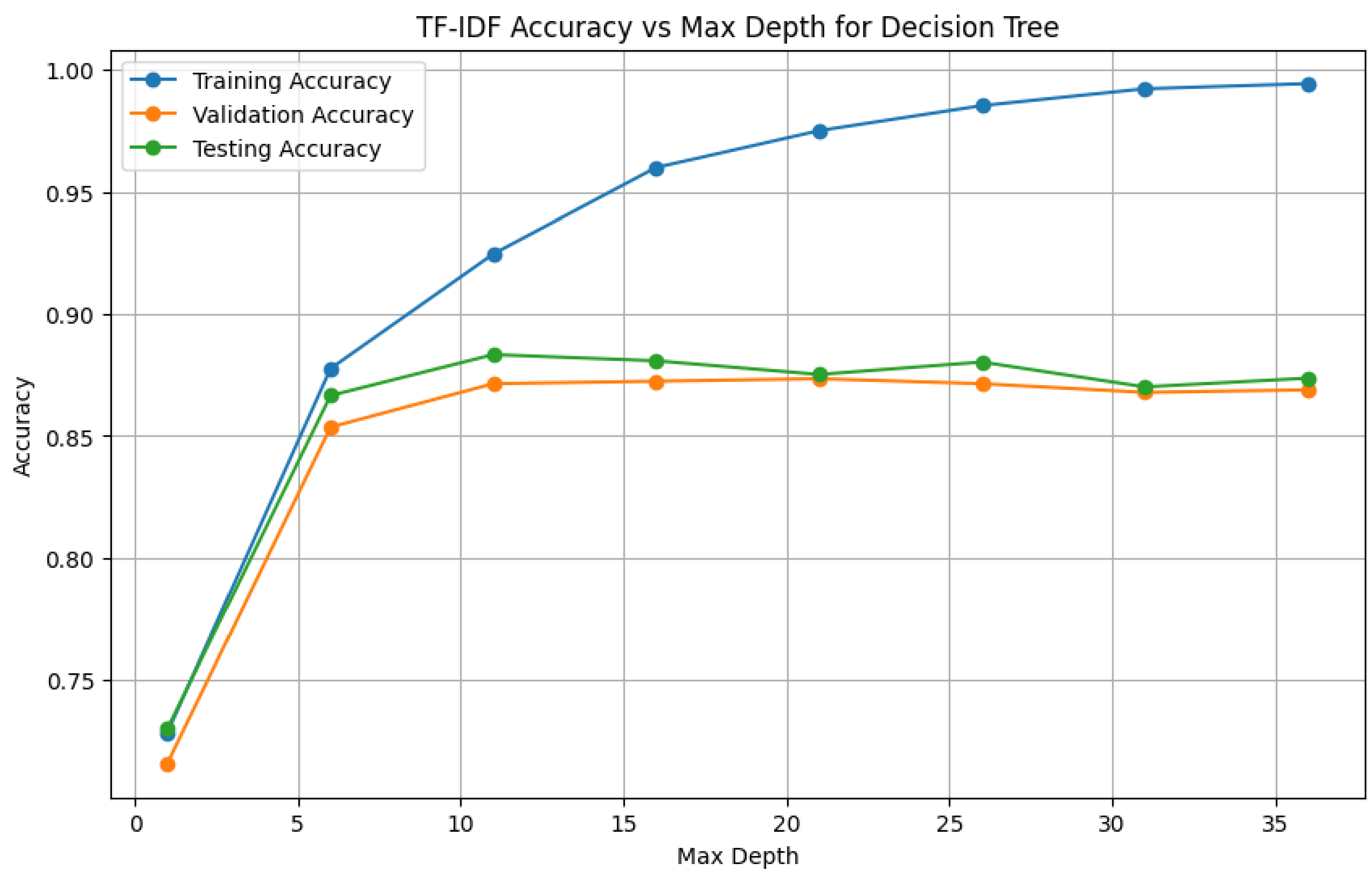
In this section, we try to modify the hyper-parameter of selected models.

4.3.1. Decision Tree

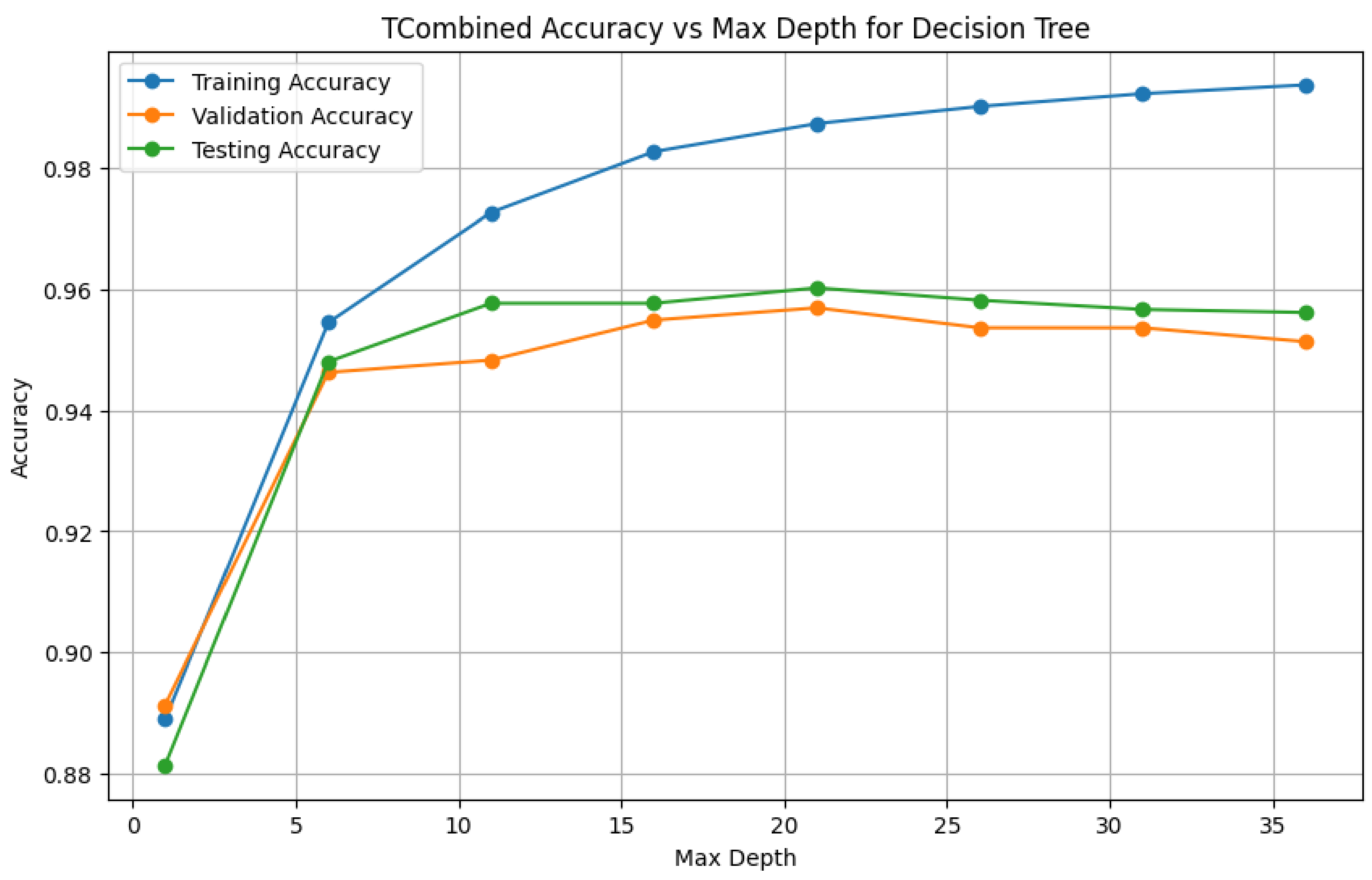
4.3.1.1. Max Depth

The `max_depth` parameter in a Decision Tree is crucial for fake news detection as it controls the tree's complexity and helps prevent overfitting. By limiting the depth, the model avoids becoming too complex and fitting noise in the training data, which improves its generalization to new articles. It also enhances training efficiency and interpretability, making it easier to understand and validate the model's decisions. Properly tuning `max_depth` ensures a balance between accuracy and practicality in detecting fake news.

TF-IDF



Combined



TF-IDF Accuracy vs Max Depth

- **Training Accuracy**: Increases steadily with increasing depth, reaching almost 1.0, indicating the model fits the training data perfectly at higher depths.
- **Validation Accuracy**: Peaks around a depth of 10-15, then slightly decreases or stabilizes, suggesting the model starts overfitting beyond this point.
- **Testing Accuracy**: Follows a similar pattern to validation accuracy, peaking around the same depth and then stabilizing or slightly decreasing.

Combined Accuracy vs Max Depth

- **Training Accuracy**: Also increases with depth, though slightly less steeply than TF-IDF.
- **Validation Accuracy**: Peaks around a depth of 10-15 and then shows a slight decrease or plateau, indicating potential overfitting beyond this point.
- **Testing Accuracy**: Again, follows the validation accuracy pattern closely, peaking around the same depth.

Choosing Max Depth

We choose max_depth = 20 as it generally performs the best as validation data.

TF-IDF

	Accuracy	Precision	Recall	F1	AUC
Decision Tree - No Limit	87.02%	85.93%	86.21%	86.07%	86.96%
Decision Tree - Max_depth = 20	87.73%	87.18%	86.32%	86.75%	87.64%

Combined

	Accuracy	Precision	Recall	F1	AUC
Decision Tree - No Limit	95.11%	94.95%	94.53%	94.74%	95.07%
Decision Tree - Max_depth = 20	96.33%	95.86%	96.28%	96.07%	96.33%

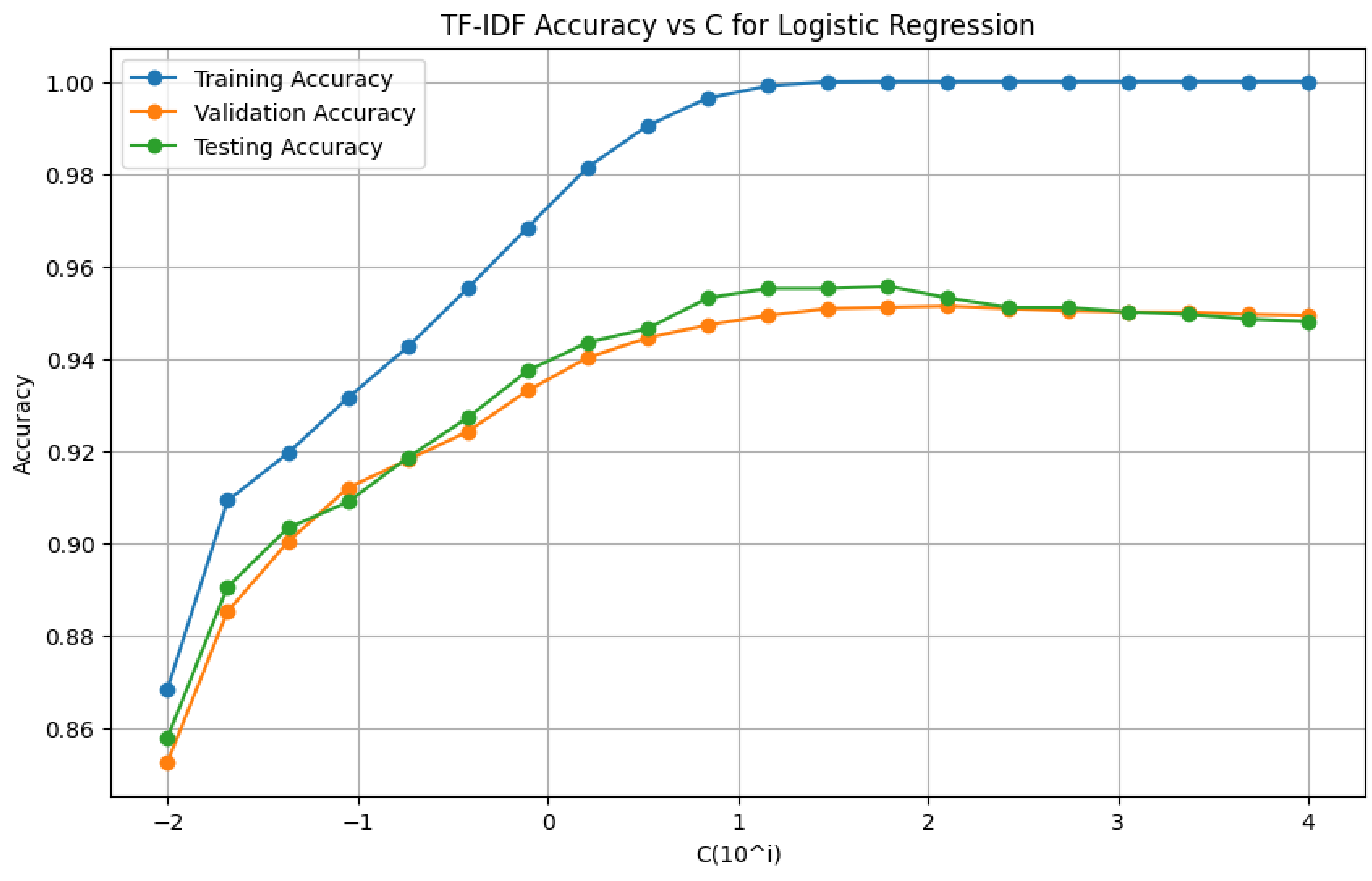
Limiting the depth of the decision tree (Max_depth = 20) improves performance metrics across the board compared to an unlimited depth decision tree, indicating better generalization and reduced overfitting.

4.3.2. Logistic Regression

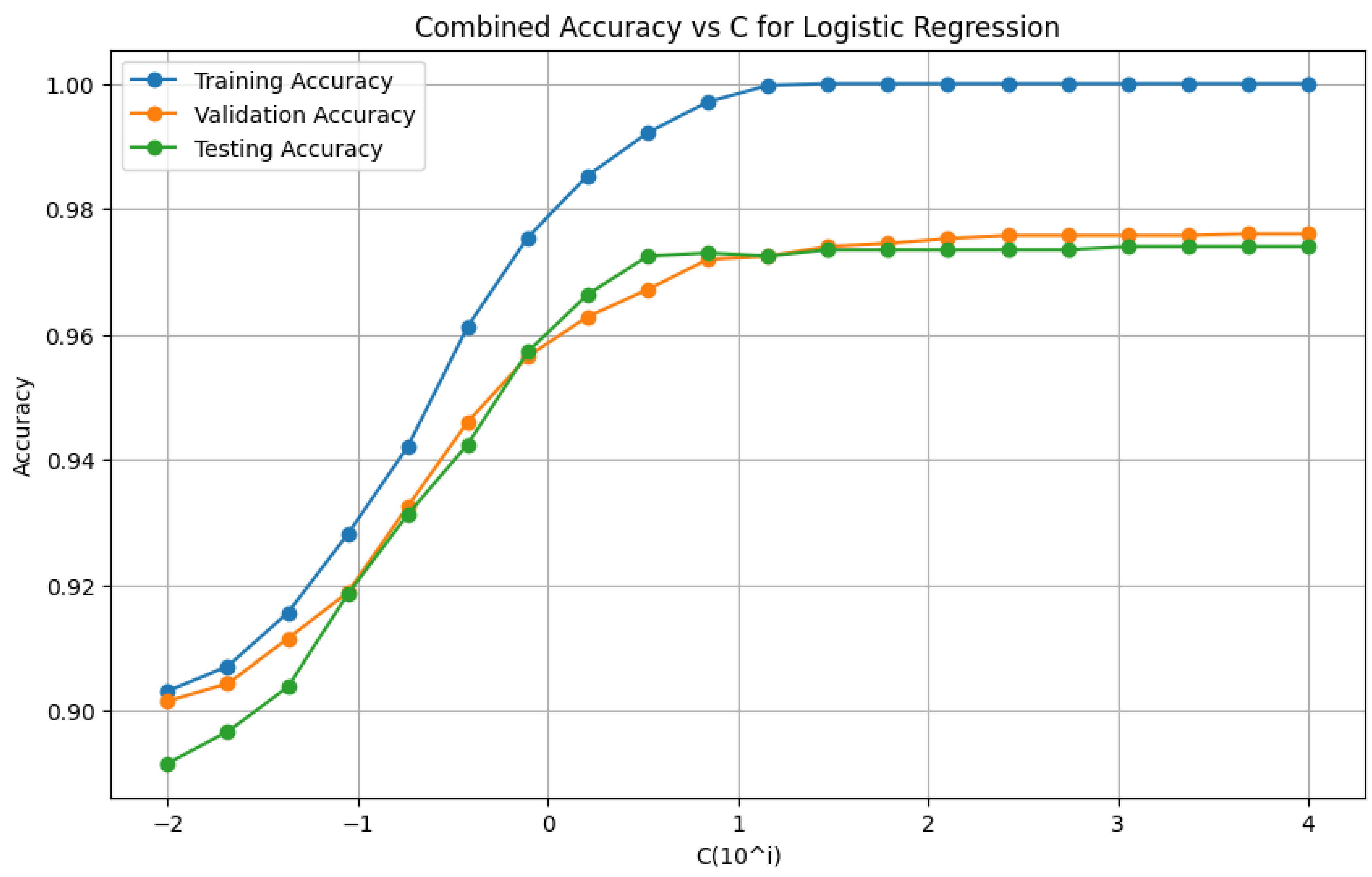
4.3.2.1. Regularization Parameter - C

The `C` parameter in Logistic Regression is crucial for fake news detection as it controls the regularization strength. Regularization helps prevent overfitting by penalizing large coefficients, thus making the model more generalizable to unseen data. A smaller `C` value implies stronger regularization, leading to simpler models that might underfit the data. Conversely, a larger `C` value reduces regularization, allowing the model to fit the training data more closely, but risking overfitting. Properly tuning `C` helps balance the trade-off between bias and variance, ensuring the model accurately classifies new articles as fake or real.

TF-IDF



Combined



TF-IDF Accuracy vs. C

- **Training Accuracy**: Increases steadily with higher values of `C`, eventually plateauing near perfect accuracy, indicating overfitting at high values.
- **Validation Accuracy**: Peaks around `C = 10^1` and then stabilizes, suggesting the optimal regularization strength.
- **Testing Accuracy**: Follows a similar pattern to validation accuracy, peaking around the same `C` value and then plateauing.

Combined Accuracy vs. C

- **Training Accuracy**: Increases with `C` and plateaus near perfect accuracy, showing overfitting at high `C` values.
- **Validation Accuracy**: Peaks around `C = 10^0` to `10^1` and then stabilizes, indicating the optimal range for `C`.
- **Testing Accuracy**: Mirrors the validation accuracy trend, peaking around the same `C` value and then stabilizing.

Choosing C

We choose C = 10^3 as it generally performs the best on validation data.

TF-IDF

	Accuracy	Precision	Recall	F1	AUC
Logistic Regression - C = 1	93.89%	92.14%	94.97%	93.53%	93.96%
Logistic Regression - C = 10^3	95.06%	92.86%	96.83%	94.80%	95.18%

Combined

	Accuracy	Precision	Recall	F1	AUC
Logistic Regression - C = 1	96.13%	94.39%	97.48%	95.91%	96.22%
Logistic Regression - C = 10^3	97.40%	96.55%	97.92%	97.23%	97.44%

Less regularization in Logistic Regression may be beneficial for fake news detection because it allows the model to capture the complex and subtle patterns present in fake news articles more effectively. This includes the ability to assign higher weights to important words or phrases indicative of fake news. By allowing more flexibility, the model can better utilize the rich feature set, such as combined text and metadata features, leading to improved performance in distinguishing fake news from real news.

4.4. A discussion about high training accuracy

Due to the unique characteristics of news text when using TF-IDF (Term Frequency-Inverse Document Frequency), where the number of attributes (features) is very high, traditional machine learning models tend to perform extremely well during training. This high-dimensional feature space allows models to capture the intricate patterns and nuances within the training data, often resulting in a training accuracy of 1 (100%).

For instance, in models like Decision Tree and Logistic Regression, the abundance of features enables these models to fit the training data perfectly. In the case of Decision Trees, the model can create very detailed branches that precisely classify each instance in the training set. Similarly, in Logistic Regression, the model can assign appropriate weights to a large number of features, allowing it to achieve perfect separation of classes in the training data.

However, while a training accuracy of 1 indicates that the model has learned the training data exceptionally well, it also raises concerns about overfitting. Overfitting occurs when the model captures not only the true patterns but also the noise and specific peculiarities of the training data. As a result, the model may fail to generalize to new, unseen data, leading to poor performance on validation and testing datasets. Therefore, achieving a perfect training accuracy highlights the need for careful hyperparameter tuning and regularization to ensure that the model generalizes well and performs reliably on new data.

5. Conclusion

The research in this paper tackles fake news detection through three steps, data pre-processing, feature engineering and classification.

5.1. Our finding for models

The evaluation of various machine learning models for fake news detection using metrics such as accuracy, precision, recall, F1-score, ROC curve, and AUC demonstrates that Logistic Regression and Support Vector Classifier (SVC) are the most effective. Particularly with combined features, these models excel in balancing high performance across all metrics, highlighting their robustness in minimizing both false positives and false negatives. Therefore, for critical applications where accurate detection is paramount, an ensemble approach that integrates the strengths of both Logistic Regression and SVC is recommended to enhance overall system reliability and efficiency in detecting fake news.

When adjusting parameters in a machine learning model, if the training accuracy reaches 1 (100%), it indicates that the model has perfectly fit the training data. This can have several implications for the training process. First, perfect training accuracy often suggests overfitting, where the model has learned the training data too well, including noise and minor details. As a result, the model may perform poorly on new, unseen data because it fails to generalize the learned patterns. Second, a model with 100% training accuracy may not generalize well to the validation or testing datasets. This means that while it performs perfectly on the training data, its performance on new data may drop significantly, leading to lower validation and testing accuracies.

Achieving perfect training accuracy might also indicate that the regularization parameter is too low (e.g., a very high `C` value in Logistic Regression), allowing the model to fit the data too closely. Increasing regularization can help prevent overfitting by penalizing large coefficients and promoting simpler models. Additionally, a training accuracy of 1 can imply that the model is too complex. Reducing model complexity, such as lowering the depth of a decision tree, decreasing the number of features, or simplifying the model architecture, can help improve generalization and prevent overfitting.

Perfect training accuracy indicates a low bias model with high variance. While the model fits the training data perfectly (low bias), it may be very sensitive to changes in the data (high variance). Adjusting parameters to achieve a balance between bias and variance is crucial for better performance on unseen data. To optimize hyperparameters effectively, various strategies can be employed, including cross-validation, grid search, random search, Bayesian optimization, regularization adjustments, learning rate tuning, and early stopping. By carefully tuning these hyperparameters and monitoring training, validation, and testing performance, you can achieve a well-balanced model that generalizes well to new data.

5.2. Limitation of data

- **Source Transparency:** The dataset is sourced from Kaggle; however, the origin of how the news articles were compiled is not transparent. It is essential to investigate whether the data was gathered from diverse news outlets or if it predominantly includes certain publishers, which could introduce bias towards a particular political or ideological slant.

- **Labeling Methodology:** There is a lack of clarity on how the labels (fake or real) were assigned to each news article. Were the labels manually curated by experts, or were they generated using automated methods? Understanding the labeling process is vital because inaccuracies in label assignment can lead to poor training outcomes and misclassifications by the model.
- **Political Bias**: We found that the articles we got are mostly centered in the political area raises concerns about ideological bias. This could affect the model's ability to generalize across different types of fake news that may not be politically oriented. Exploring how representative the dataset is of various news genres will help in assessing the model's robustness.
- **Baseline Accuracy**: Without a baseline accuracy, it is challenging to set expectations for our model's performance. Unfortunately, we don't get one from this dataset.
- **Feature Completeness**: Here, we only got three features at beginning - title, author, and text. But we lack more information, like detailed author information(like organization, age, race), publication date, comments and so on. Missing features could limit the model's ability to learn effective discrimination between fake and real news.

5.3. Further Investigate

1. **Feature Engineering**:
    - **Advanced Text Features**: Beyond simple TF-IDF, explore more nuanced text features that capture the underlying semantics and intent of the articles. For instance, extract the stance of an article or the sentiment expressed within the text, which can provide deeper insights into the authenticity of the information. Utilizing word embeddings, such as Word2Vec or GloVe, can also capture contextual relationships between words, providing a richer feature set.
    - **Author and Source Credibility**: Incorporate features related to the author's historical reliability and the credibility of the publication source. Metrics could include previous instances of publishing misleading information or the general reputation of the source.
    - **Structural Features**: Analyze the structure of the articles, such as the complexity of language, use of sensationalist headlines, or the presence of certain punctuation patterns, which are often indicative of fake news.
2. **Experiment with Advanced Models**:
    - **Transformer Models**: Implement state-of-the-art NLP models like BERT, GPT, or RoBERTa, which utilize transformer architectures known for their effectiveness in capturing complex language patterns and dependencies. These models can significantly outperform traditional machine learning approaches, especially in tasks involving deep semantic understanding.
    - **Temporal Dynamics**: For detecting fake news in an online platform, consider incorporating models that account for temporal dynamics, such as how narratives evolve over time. Recurrent Neural Networks (RNNs) or models with attention mechanisms can track changes in how stories are reported, which can be crucial for identifying coordinated disinformation campaigns.
3. **Implement Ensemble Techniques**:
    - **Stacking**: Use a stacking ensemble where the outputs of multiple models (like decision trees, SVMs, and neural networks) are used as inputs to a final meta-classifier. This approach leverages the strengths of diverse models, potentially leading to better generalization on unseen data.
    - **Model Blending**: Blend predictions from various models using a weighted average approach, where weights are assigned based on each model's performance on a validation set. This method helps in reducing variance and bias of the predictions.
4. **Cross-Validation and Hyperparameter Tuning**:
    - **Robust Validation Strategy**: Employ k-fold cross-validation to ensure that your model's performance assessment is reliable and not overly dependent on any particular split of the data.
    - **Optimize Model Parameters**: Utilize systematic approaches like grid search or Bayesian optimization to explore a wide range of parameter settings and identify the most effective combinations for each model, thereby enhancing their predictive capabilities.

By incorporating these advanced feature extraction techniques, leveraging cutting-edge model architectures, and employing robust ensemble and optimization strategies, you can significantly improve the accuracy and reliability of your fake news detection system.

6. Reference

6.1. Data Source

William Lifferth. (2018). Fake News. Kaggle. https://kaggle.com/competitions/fake-news

6.2. Articles

Aldwairi, M., & Alwahedi, A. (2018). Detecting fake news in social media networks. *Procedia Computer Science*, *141*, 215-222.

Ahmed, A. A. A., Aljabouh, A., Donepudi, P. K., & Choi, M. S. (2021). Detecting fake news using machine learning: A systematic literature review. *arXiv preprint arXiv:2102.04458*.

Ahmed, H., Traore, I., & Saad, S. (2017). Detection of online fake news using n-gram analysis and machine learning techniques. In *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments: First International Conference, ISDDC 2017, Vancouver, BC, Canada, October 26-28, 2017, Proceedings 1* (pp. 127-138). Springer International Publishing.

Baarir, N. F., & Djeffal, A. (2021, February). Fake news detection using machine learning. In *2020 2nd International workshop on human-centric smart environments for health and well-being (IHSH)* (pp. 125-130). IEEE.

Ghanem, B., Rosso, P., & Rangel, F. (2018, November). Stance detection in fake news a combined feature representation. In *Proceedings of the first workshop on fact extraction and VERification (FEVER)* (pp. 66-71).

Khanam, Z., Alwasel, B. N., Sirafi, H., & Rashid, M. (2021, March). Fake news detection using machine learning approaches. In *IOP conference series: materials science and engineering* (Vol. 1099, No. 1, p. 012040). IOP Publishing.

Lo, S. (2018). *Detecting Fake News Using Stance-Based Approach* (Doctoral dissertation, Fordham University).

Mahir, E. M., Akhter, S., & Huq, M. R. (2019, June). Detecting fake news using machine learning and deep learning algorithms. In *2019 7th international conference on smart computing & communications (ICSCC)* (pp. 1-5). IEEE.

Zhang, X., & Ghorbani, A. A. (2020). An overview of online fake news: Characterization, detection, and discussion. *Information Processing & Management*, 57(2), 102025.