university of groningen

faculty of science and engineering

Design and Development of Machine Learning Frameworks for Fake News Detection

Kevin Wang

university of groningen

faculty of science and engineering

University of Groningen

Design and Development of Machine Learning Frameworks for Fake News Detection

Bachelor's Thesis

To fulfill the requirements for the degree of
Bachelor of Science in Computing Science
at University of Groningen under the supervision of
Prof. dr. Fadi Mohsen (University of Groningen)
and
Prof. dr. George Azzopardi (University of Groningen)

Kevin Wang (s3470016)

June 28, 2023

Contents

Acknowledgments

First and foremost, I would like to express my deep gratitude to my project supervisor, Mr Fadi Mohsen, whose expertise, understanding, and patience, added considerably to my undergraduate experience. I appreciate his vast knowledge and skill in many areas, and his assistance in not only this thesis, but also in navigating the larger project of which this thesis was a part.

I would also like to extend my thanks to the staff of the University of Groningen, for their valuable insight and expertise that guided me through my studies, and especially for their assistance in this project.

My sincere thanks also go to my friends, who have given me their unequivocal support throughout this project. Your friendship made the journey more enjoyable.

Finally, I must express my profound gratitude to my parents and to my family for providing me with unfailing support and encouragement throughout my years of study. This accomplishment would not have been possible without them.

Abstract

This thesis explores advancements in the field of fake news detection, specifically aiming to enhance an existing application that compares various machine learning libraries for this purpose. It introduces an additional library to the application, enabling users to modify key parameters, which significantly augments the flexibility of the application. A series of experiments are conducted, which examine the performance of various models and the application of transfer and incremental learning techniques across diverse datasets. The results underscore the critical role of dataset alignment and balance in transfer learning. Moreover, they highlight the utility of incremental learning in maintaining high detection performance while substantially reducing runtime. The application currently supports English text and binary fake news labels, presenting opportunities for future work to extend its capabilities. The implications of the project are discussed, touching on the influence of fake news on public trust and the ethical considerations of deploying such an application.

Disclaimer: This thesis uses supervised machine learning methods for fake news detection, focusing on patterns such as writing style and keyword usage. While these techniques can effectively identify some elements common to fake news, they should not be misconstrued as a comprehensive fact-checking system. The methods discussed herein are not designed to verify the veracity of individual claims or facts within a news article, but rather to identify potential patterns that may be indicative of misinformation. We acknowledge the importance of human-driven fact-checking and critical analysis in discerning truth from falsehood in news media.

1 Introduction

In recent years, the global rise of fake news, aided by the internet and social media, poses serious societal challenges, influencing public opinion, political choices, and potentially threatening national security[1]. The need for effective automated fake news detection is urgent as online content volume increases.

Machine learning, a branch of artificial intelligence, is crucial in addressing this issue. Many algorithms are available for fake news detection, but selecting the appropriate one requires careful consideration due to performance variations across different datasets and use cases[2].

This thesis aims to improve an existing application designed to compare different machine learning libraries for fake news detection. The goal is to integrate an additional library, enable users to modify key parameters, and investigate the potential of transfer / incremental learning across diverse datasets.

This research seeks to bridge gaps in understanding, contributing to machine learning advancements. It will offer practical guidance for applications where limited labelled data is available or models are deployed to new domains, situations where transfer learning can enhance performance.

1.1 Background and Motivation

The motivation for this project stems from the increasing relevance of fake news detection in our society. As the digital age progresses, more and more individuals turn to online sources for information. This transition necessitates an urgent need to discern accurate information from false narratives.[3] The task, however, is far from simple due to the volume, velocity, and variety of data produced daily on the internet. Here, machine learning algorithms have demonstrated substantial promise, utilizing pattern recognition and statistical learning to identify and classify fake news effectively.

A multitude of machine learning libraries and tools are available to assist developers in this endeavor. However, the task of selecting an optimal algorithm and fine-tuning the corresponding parameters can be a daunting and time-consuming task. It often involves extensive experimentation, evaluation, and a solid understanding of the underlying principles of each algorithm.

In extending the existing application developed by previous master's students[4], this project aims to simplify this process. By integrating an additional machine learning library and allowing user customization of key parameters, the aim is to develop a more multifaceted tool. This enhanced functionality could prove essential for researchers, journalists, and other stakeholders engaged in the challenging task of mitigating the spread of fake news.

Moreover, this project serves as an excellent opportunity to explore the capabilities and limitations of various machine learning libraries. By comparing and contrasting their performance, I can contribute to ongoing research in this field. This endeavor is not only academically stimulating but also has practical implications. It assists in uncovering insights that could potentially lead to the development of more robust and efficient fake news detection systems in the future.

1.2 Overview of the Existing Master Project

The current master's project not only conducts a comparative analysis of various machine learning algorithms in the context of fake news detection, but also builds a unique framework for such comparisons. This work leverages Flask and Dash for an interactive frontend, and it includes the implementation of a novel approach, which is then evaluated through a dedicated comparison study.

The core of the application employs the scikit-learn library, offering four algorithms: Logistic Regression, Decision Tree, K-Nearest Neighbors (KNN), and Gradient Boosting, each with unique strengths for a comprehensive comparison.

The application also supports three text vectorization techniques: TF-IDF, feature extraction, and a novel hybrid model. The TF-IDF method assigns weights to words based on their frequency, useful for classification and information retrieval, including fake news detection[5]. Empath extracts a range of features from text, including psychological, morphological, and grammatical aspects[6]. A hybrid model combining TF-IDF and Empath shows superior performance in fake news detection, especially with fewer records datasets[4].

Feature selection uses Analysis of Variance (ANOVA) to select statistically significant features, reducing feature space dimensionality and enhancing model accuracy.

The hybrid approach and ANOVA selection outperform existing benchmarks, proving effective for text classification tasks, including sentiment analysis, authorship attribution, and topic modeling. This novel feature extraction approach will be further discussed in the Methodology section.

This project offers a robust platform for fake news detection, allowing users to choose from various machine learning algorithms and text vectorization techniques. It serves as the foundation for this thesis, which intends to extend these capabilities by incorporating additional machine learning libraries, enabling parameter customization, and exploring transfer learning and incremental learning within the Naive Bayes framework.

1.3 Objectives and Research Questions

The overarching goal of this thesis is to extend an existing application for fake news detection by incorporating additional machine learning libraries, enabling user customization of key parameters, and exploring the potential of transfer learning within the Naive Bayes framework. The intended outcome is to enhance the application's overall performance, versatility, and effectiveness.

To guide this research and ensure its focus and relevance, several research questions have been identified, each addressing a crucial aspect of the broader objective:

Q1. **What is the comparative effectiveness of different machine learning models and feature extraction methods in detecting fake news?** The first research question aims to understand the effectiveness of various machine learning models in detecting and classifying fake news with various feature extraction methods. The purpose is to identify the most suitable combinations for different types of data sources. By comparing the performance of different algorithms on the same datasets, this research will provide insights into which combinations are best suited for different types of data and feature sets. This knowledge can contribute to the development of more accurate and efficient fake news detection systems, as the selection of the most suitable model can significantly improve the overall performance.

Q2. **What is the potential of transfer and incremental learning in improving the efficiency of fake news detection?** The second research question delves into the untapped potential of transfer and incremental learning techniques of the Naive Bayes classifiers on fake news detection. Transfer learning, which involves applying knowledge from one task to another related task, has exhibited promising outcomes

in diverse machine learning applications. Nonetheless, its exploration within the context of fake news detection remains limited. This study will thoroughly examine the feasibility and effectiveness of integrating transfer learning into the Naive Bayes framework, with the objective of devising machine learning frameworks that are not only more efficient but also more potent in detecting fake news. Furthermore, the investigation will encompass incremental learning techniques, which allow systems to gradually acquire knowledge and adapt to evolving fake news patterns over time. By considering both transfer and incremental learning, this research seeks to pave the way for more advanced and adaptive fake news detection models.

Q3. **What are the ethical considerations and implications of using machine learning for fake news detection?** The final research question investigates the ethical considerations and implications of using machine learning frameworks for fake news detection. As machine learning algorithms become more prevalent in our society, concerns about bias, privacy, and potential misuse are growing.[7] This research aims to explore these issues in the context of fake news detection, seeking ways to ensure responsible and transparent use of these technologies.

These research questions aim to address a range of technical and ethical issues related to the use of machine learning for fake news detection. By providing answers to these questions, this thesis seeks to contribute to the development of more effective, efficient, and responsible fake news detection systems. The results of this research will not only enhance the existing application but also provide valuable insights for future work in this rapidly evolving field.

2 Background Literature

The literature review section presents a comprehensive survey of existing research relevant to the area of fake news detection, machine learning, and transfer learning. It aims to critically examine the current body of knowledge, identify gaps that this thesis seeks to fill, and set the present study within the context of the broader academic discourse.

In this section, I will examine pertinent research that delves into the comparative analysis of various Machine Learning (ML) models, along with the application of transfer and incremental learning methodologies in conjunction with Multinomial and Bernoulli Naive Bayes classifiers. Following this, in the 'Research Gaps' section, I will highlight the areas that this project uniquely addresses, and distinguish it from the existing body of research by pinpointing the variations and novel contributions.

2.1 Review of Existing Research

2.1.1 Applications for Comparing ML Models

Several studies have been instrumental in the evolving landscape of machine learning-based fake news detection. A noteworthy thesis undertakes an exhaustive assessment of conventional machine learning and deep learning models for this purpose. The work by Khan et al. sets itself apart by addressing a significant gap in the literature - the previous focus on specific news types, like political content, and the consequent potential dataset bias [8].

This study notably differs from mine and the earlier student's work in several ways. While I have used fewer datasets, the authors employed three distinct datasets, thus substantially mitigating the risk of dataset bias. As for machine learning algorithms, they examined a range of traditional machine learning approaches and deep learning models. Compared to the four algorithms explored by Albahr et al. [9], my research, similar to the previous student's work, investigates a broader selection of models. Furthermore, Khan et al. have conducted a study on the efficiency of Multinomial Naive Bayes with Empath. In contrast, my research delves into assessing the efficiency of Multinomial and Bernoulli Bayes classifiers employing not only Empath but also TF-IDF and the hybrid approach.

Despite acknowledging the promise of advanced pre-trained language models like BERT, ELECTRA, and ELMo, Khan et al. did not employ a specific framework or application for comparing the models. In contrast, my study, building upon the previous student's effort, utilizes an extended version of an existing framework to facilitate such comparisons, thereby providing a more systematic and replicable approach.

Albahr et al. also conducted a comparative analysis of four machine learning algorithms using NLP techniques on a public dataset. Their study, while insightful, does not leverage a dedicated framework for comparing the performance of the models, something both my work and the previous student's project focus on. This difference in approaches underscores the unique contribution of my research in offering a more structured and comprehensive assessment of machine learning models for fake news detection.

In this paper proposed by Barua et al.[10], they tackle the urgent and pervasive problem of misinformation disseminated through the Internet and social media platforms by employing an ensemble of advanced recurrent neural networks - specifically, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) - to discern the veracity of news articles.

To enhance accessibility and real-world applicability, the researchers have implemented an Android application that can be used to authenticate news articles from various sources on the internet. The application provides a simple user interface

with a splash screen displaying the app's logo. After establishing a connection with the Java Web Server, the app redirects to the main activity page.

To test the efficacy of the proposed model, they have prepared a large dataset that includes news from various sources, both real and fake. This dataset provides a diverse and comprehensive testing ground for the model. Furthermore, they validate the model against several standard datasets from the existing literature, which allows us to benchmark the performance against previous efforts in this field. Their evaluations indicate that the proposed model performs commendably, suggesting its potential for real-world applications in fake news detection.

While this paper introduces a robust ensemble model using LSTM and GRU for fake news detection and demonstrates its application in an Android app, my research differs in several key aspects. Unlike this project's focus on recurrent neural networks, the preceding student and me adopted a broader range of machine learning models, including some innovative techniques not explored in this study. Also, while their research relied on a single custom dataset and some standard ones, my work engaged with multiple datasets to mitigate potential biases and enhance the generalizability of my findings.

Furthermore, the application developed in the referenced paper is primarily designed for end-users to assess the credibility of news articles. In contrast, our application serves a different purpose. Its primary function is to enable comprehensive benchmarking, allowing users to assess and compare the performance of different machine learning algorithms for fake news detection.

None of the aforementioned research papers have implemented a user-friendly framework for comparing or benchmarking various Machine Learning models.

2.1.2 The Use of Naive Bayes Classifiers for Fake News Detection

The paper proposed by Hassan et al.[11] categorizes fake news detection strategies into Knowledge Based and Machine Learning based approaches. The latter is further split into Conventional and Neural Network methods. The paper elaborates on Support Vector Machines, Bernoulli Naive Bayes, and Multinomial Naive Bayes as Conventional methods, while discussing Convolutional Neural Networks and Recurrent Neural Networks under Neural Network approaches.

Since this paper is solely a survey, it does not include any experiments or results. Therefore, it does not extensively explore the comparison of performance among different machine learning models, nor does it discuss the impact of various feature extraction methods or the potential of transfer and incremental learning.

Mandical et al.[12] propose a system for reliable fake news classification using machine learning algorithms, including Multinomial Naive Bayes, Passive Aggressive Classifier, and Deep Neural Networks. The system's performance is assessed across eight datasets from diverse sources. The paper details the analysis and results of each model, highlighting the potential to simplify fake news detection with appropriately selected models and tools.

Comparing the research of Mandical et al., the previous student, and my own work, I note several distinctions. While Mandical et al. constructed a system or pipeline for fake news detection and performed comparisons among various ML and Deep learning models, they did not develop a user-friendly application as we did. Moreover, our study encompasses a wider range of ML models than their research. Additionally, they solely focused on investigating the impact of ML and Deep learning models with TF-IDF, whereas we explore the effects of TF-IDF, Empath, and the hybrid approach. Furthermore, Mandical et al. did not delve into the analysis of transfer learning

or incremental learning with any of these models, whereas I explicitly examine their influences in my study.

2.1.3 The Use of Transfer and Incremental Learning with NB Classifiers

Do et al.[13] introduce a novel approach for automating the process of text classification, specifically focusing on multiclass scenarios. Drawing on traditional methodologies such as the multinomial Naive Bayes and TF-IDF (Term Frequency-Inverse Document Frequency) classifiers, which employ a parameter function derived from training set statistics, Do et al. propose an algorithm that learns this parameter function from related classification tasks. Leveraging the concept of transfer learning, the learned parameter function forms the basis of a new text classification algorithm. The proposed technique automates the usually manual process of parameter function identification, making text classification more efficient and effective across various tasks.

Werner et al.[14] present DeepMoVIPS, a cutting-edge visual indoor positioning system designed for the identification of user location within buildings, where GPS is ineffective. Utilizing the image classification capabilities of deep convolutional neural networks, this method delivers a unique approach for symbolic indoor geolocation. They leverage concepts from multinomial Naive Bayes and transfer learning to adapt visual features from deep learning networks to the application domain. The proposed system showcases remarkable classification accuracy exceeding 95% in datasets representing work environments consisting of 16 rooms, evaluated over a four-week period.

Vergara et al.[15] present a novel approach to face recognition systems, specifically focusing on gender estimation. Existing systems often operate under controlled environments and on limited data, with high computational costs that prohibit real-time incremental learning. The proposed method seeks to overcome these limitations, leveraging Multinomial Naive Bayes and Local Binary Patterns. This method has been rigorously tested using a contemporary dataset for age and gender recognition, which includes realistic examples. For optimized results, the application of Adaboost, a machine learning algorithm, is also introduced and evaluated, paving the way for more effective and efficient face recognition systems.

This research of Metsis et al.[16] addresses the practical application of Naive Bayes in anti-spam email filtering, a popular usage in both commercial and open-source systems. Despite its wide use, the various forms of Naive Bayes aren't always fully acknowledged or utilized in the anti-spam literature. In this paper, they explore five distinct versions of Naive Bayes (including MNB and BNB) and evaluate their performance on six new, non-encoded datasets. These datasets, composed of 'ham' (non-spam) messages from specific Enron users and recently collected spam messages, offer a more realistic benchmark compared to previous studies. They preserve the temporal order of messages and mirror the fluctuating ratio of spam to ham emails that users encounter over time. Through an experimental procedure that simulates the incremental training of personalized spam filters, they provide a comprehensive comparison of the different Naive Bayes versions. This examination, illustrated through ROC curves, provides insights into the trade-off between true positives and true negatives in spam filtering.

None of the previously mentioned papers have utilized Transfer Learning or Incremental Learning with BNB (Bernoulli Naive Bayes) or MNB (Multinomial Naive Bayes) for fake news detection.

2.2 Summary and Research Gaps

The investigation and analysis of the existing literature related to fake news detection using machine learning (ML) models reveal several areas of uncharted research territories.

The first major gap in the current research corpus pertains to comparative analysis studies between different ML approaches, specifically those using Term Frequency-Inverse Document Frequency (TF-IDF), Empath, and a hybrid model that combines the two. Despite the vast number of papers on building applications or conducting experiments to compare different ML models in fake news detection, there is a dearth of studies that focus specifically on these three methodologies. Thus, there is a pressing need for research that explores and contrasts the efficacy of TF-IDF, Empath, and their hybrid approach in fake news detection.

In addition to the comparison of different ML models, there is a gap in the exploration of transfer learning application with the Bernoulli Naive Bayes (BNB) classifier in the context of fake news detection. Although there are documented studies on implementing transfer learning with the Multinomial Naive Bayes (MNB) model, the application of the same with the BNB model remains largely unexplored. Moreover, the comparison of these two Naive Bayes classifiers in the specific context of transfer learning in fake news detection is also conspicuously missing. This presents an opportunity for further investigation.

Lastly, the application of incremental learning with the MNB and BNB classifiers for fake news detection has not received adequate scholarly attention. While there is a body of research centered on the utilization of these classifiers with incremental learning for spam email detection, the application of this approach to the domain of fake news detection has been largely overlooked. The direct comparison between these two classifiers when used with incremental learning in fake news detection is also notably absent. Therefore, there is a significant need for studies that focus on the application and comparison of incremental learning with the MNB and BNB classifiers in the field of fake news detection.

3 Methodology

In this section, I elucidate the methodology underpinning the research. The exploration begins with a thorough discussion of the datasets employed, shedding light on their selection and significance in this study. This is succeeded by a comprehensive outline of the preprocessing techniques employed to cleanse and format the data, making it suitable for subsequent analysis. Then I will delve into a detailed exposition of the feature extraction methods applied, outlining their function in transforming the raw data into a form that can be effectively utilized by machine learning models. Then I will move on to discuss the machine learning models previously employed in the application, their characteristics, and their function in this context. Alongside this, I introduce new machine learning models that have been added to augment the existing system. The possibility of employing transfer and incremental learning is also explored, delineating the potential benefits of these advanced learning paradigms. Finally, I provide an in-depth description of the evaluation metrics employed to gauge the performance of our models, offering insights into their selection and interpretation. This holistic methodology aims to offer a balanced and robust approach to the analysis, building upon past techniques while integrating innovative practices.

3.1 Datasets and Preprocessing Techniques

3.1.1 FA-KES: A Fake News Dataset around the Syrian War

The FA-KES dataset[17], focusing on the Syrian war, represents a significant departure from the usual political, entertainment, or satirical fake news datasets. This collection comprises of articles from a wide spectrum of media outlets, covering multiple aspects of the Syrian conflict. It was generated using a semi-supervised fact-checking approach, leveraging human contributors and crowd-sourcing techniques to match articles against a 'ground truth' reference from the Syrian Violations Documentation Center. The dataset, containing 804 articles labeled as true or fake, provides a valuable resource for training machine learning models in credibility assessment. The construction method for the FA-KES dataset offers a generalized framework for creating fake news datasets in other conflict-related domains, given the presence of a suitable 'ground truth'.

3.1.2 Simplified Version of the Fake and Real Dataset

A simplified yet labelled version of the Fake and Real Dataset[18] is used[19]. The original Fake and Real dataset is much larger, but it splits fake news and real news into two separate csv files, neither of these two files is labelled. This dataset merged those two csv files, labelled each news article accordingly, and then extracted around 10k articles from the merged dataset. This dataset will be referred to as the "Merged" dataset in the Experiment section.

3.1.3 The WELFake Dataset

The WELFake Dataset[20], is the largest dataset used in this project, consisting of 72,134 news articles with 35,028 real and 37,106 fake news. For this, authors merged four popular news datasets (i.e. Kaggle, McIntire, Reuters, BuzzFeed Political) to prevent over-fitting of classifiers and to provide more text data for better ML training.

3.1.4 3 Datasets from Kaggle

The first dataset[21], comprising approximately 50,000 unique news articles, has been meticulously scraped from various online sources. It is organized into two distinct columns: 'text' and 'label'. Due to its substantial volume, this dataset is particularly well-suited for use in the Incremental Learning experiments. This dataset will

be referred to as the ”Scraped” dataset in the Experiment section.

The second dataset taken from a Kaggle competition[22], offers more complexity. It contains 20387 unique news articles. Its attributes include ’id’, ’title’, ’author’, ’text’, and ’label’, with the label categorizing the news as ’1’ for unreliable and ’0’ for reliable. This structure not only serves as a base for supervised learning but also allows exploring potential authorship-based patterns in misinformation spread. This dataset will be referred to as the ”Kaggle 1” dataset in the Experiment section.

To simulate a real-world scenario of incremental learning, a trimmed version of these datasets - WELFake, Scraped and Kaggle 1, containing the first 3000 rows (the entire dataset undergoes shuffling prior to the extraction of these rows), was prepared using a Python script detailed in the appendix section. The script prints the numbers of 1s and 0s after splitting the dataset, if the user is unsatisfied with the resulting dataset, they have the option to rerun the script until they achieve a balanced dataset that meets their requirements. The script also retains the original columns while creating a manageable subset. The trimmed datasets are named `trimmed-WELFake`, `trimmed-Scraped` and `trimmed-Kaggle1` accordingly.

These processed datasets will be used for initial model training, while the original, the remaining, larger datasets served to test the model’s incremental learning capacity with a Naive Bayes classifier. This approach facilitated testing scalability and adaptability of the model in handling larger data volumes, emulating real-world scenarios.

The last dataset[23] containing 3352 unique news articles, diverges from traditional datasets with its structure. It splits news articles into ’URLs’, ’Headline’, ’Body’, and ’Label’, the ’Body’ section, which contains a substantial amount of text, will be utilized for feature extraction. This dataset will be referred to as the ”Kaggle 2” dataset in the Experiment section.

The unique structures and characteristics of these datasets contribute significantly to understanding the multifaceted nature of fake news and to developing an effective detection system.

3.1.5 Preprocessing Techniques

The preprocessing stage of this project encompasses several integral tasks including data loading, preprocessing, feature extraction, statistical tests, and text vectorization, each critical to refining raw datasets for model training and testing.

Data loading and decoding are initial steps to ingest datasets of various formats, ensuring compatibility and ease of use. This is followed by feature extraction, which in this study utilizes lexical features. The Empath tool is employed for this purpose, offering a comprehensive analysis of text for discernible linguistic patterns.

In order to effectively reduce the dimensionality of the dataset and focus on the most relevant attributes, a feature selection process is employed. This process involves ANOVA F-value calculation for the provided features, and selection based on statistical significance.

Data cleaning and preprocessing steps ensure the quality of the dataset. These steps involve the removal of instances with missing values, exclusion of non-English text when necessary, and randomization of DataFrame rows. While the exclusion of non-English text was not a necessary procedure in this study, the framework supports this feature for datasets with multilingual content.

Text preprocessing, a crucial step for natural language processing tasks, includes the removal of punctuation and stopwords, and word stemming. These steps standardize the text data, facilitating more effective machine learning.

Additional operations performed on the dataset

include label assignment and selective column retention to streamline the dataset for machine learning algorithms. The dataset and labels are subsequently merged to form a unified structure.

The last stage of preprocessing is text vectorization. This step transforms the processed text into a numerical format suitable for machine learning algorithms. For this purpose, TF-IDF vectorization is used, which assigns scores to words in the text based on their frequency of occurrence, effectively transforming the textual data into a mathematical representation.

This process generates one file output, namely the `vectorizer.sav` as part of its operations, which saves the vectorization results to a pickle file, that can be reused for transfer and incremental learning, to facilitate consistency and reproducibility in data preprocessing and feature extraction.

3.2 The Novel Hybrid Approach

The hybrid approach proposed in Bedir Chaushi's thesis, that leverages the benefits of both TF-IDF and Empath, with the goal of creating a more robust set of features for text analysis. It suggests that this approach could be particularly beneficial for fake news detection.[4]

3.2.1 TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency. It's a text vectorization technique that assigns weights to words in a document based on their frequency in that document and their inverse frequency in the entire document corpus. It helps to understand the importance of a word in a document.[5]

The term frequency (TF) is calculated as the number of times a word appears in a document divided by the total number of words in the document. The inverse document frequency (IDF) is calculated as the logarithm of the total number of documents divided by the number of documents containing the term. The TF-IDF value is the product of these two quantities:

$$\text{tfidf}(t,d,D) = \text{tf}(t,d) \cdot \text{idf}(t,D)$$

where:

$$\text{tf}(t,d) = \frac{\text{Frequency of term t in document d}}{\text{Total number of terms in document d}}$$

and

$$\text{idf}(t,D) = \log\left(\frac{\text{Total number of documents in corpus D}}{\text{Number of documents D containing t}}\right)$$

3.2.2 Empath

Empath is a text analysis library that extracts a variety of features from text. It can recognize over 200 categories of human-centric lexicon, including emotions, attitudes, values, demographics, physical states, and behaviors. It also recognizes morphological and grammatical features, which can capture syntactic and semantic patterns in text.[6]

3.2.3 The Hybrid Approach

The idea in Chaushi's thesis is to combine TF-IDF and Empath to create a hybrid set of features that can capture both the content (TF-IDF) and style (Empath) of the text.[4]

TF-IDF vectorizes the content, emphasizing the importance of less frequent, potentially more meaningful words. It can effectively capture the "what" of the text. On the other hand, Empath analyzes lexical categories and the sentiment behind the text, capturing the "how" of the text.

By combining these two methods, we will get a comprehensive set of features, including both the factual and emotional aspects of the text, which could be very useful for detecting fake news.[4] This is because fake news often uses specific emotional cues and manipulative language styles, along with misleading or untrue content.

Here's a step-by-step outline of how the process works:

1. Compute TF-IDF vectors: Given a corpus of documents, calculate the TF-IDF score for each word in each document. This results in a vector representation of each document where the elements are TF-IDF scores.

2. Generate Empath features: For the same set of documents, extract the relevant features using Empath. This will also result in a vector representation of each document where the elements are Empath feature values.

3. Combine the vectors: For each document, concatenate the TF-IDF vector and the Empath feature vector. This results in a hybrid vector that includes both sets of features.

3.3 Overview of the Machine Learning Algorithms and Libraries Used

This section offers an overview of the various machine learning algorithms and libraries used in the existing application. The source code was primarily written in Python, making use of the versatile capabilities of several well-established machine learning libraries.

3.3.1 Logistic Regression

Logistic Regression (LR) is a simple yet effective classification algorithm that is often used as a benchmark for binary classification problems.[24] The implementation in this project uses the `LogisticRegression` module from the sklearn library. It's configured with various parameters such as `solver`, `C`, `penalty`, and `max_iter`, which control the type of solver to be used in the optimization problem, the inverse of regularization strength, the type of regularization to be used, and the maximum number of iterations, respectively.[4]

3.3.2 Decision Trees

Decision Trees (DT) are a non-parametric supervised learning method that can be used for both classification and regression.[25] They're simple to understand and interpret, which adds to their appeal. The project uses the `DecisionTreeClassifier` from the sklearn library. Parameters like `criterion`, `min_samples_split`, `splitter`, and `max_depth` control the function to measure the quality of a split, the minimum number of samples required to split an internal node, the strategy used to choose the split at each node, and the maximum depth of the tree, respectively.[4]

3.3.3 Nearest Neighbour

The k-nearest neighbors (KNN) algorithm is a type of instance-based learning where the function is approximated locally, and all computation is deferred until classification. It's a type of lazy learning where the computation is deferred until function evaluation.[26] The `KNeighborsClassifier` module from sklearn library is used in this project. Parameters include `algorithm`, `n_neighbors`, `weights`, and `p` that control the algorithm used to compute the nearest neighbors, number of neighbors to use, weight function used in prediction, and the power parameter for the Minkowski metric.[4]

3.3.4 Gradient Boosting Classifier

Gradient Boosting Classifiers (GBC) are a powerful ensemble machine learning algorithm that constructs a set of weak learners (typically decision trees), and then combines them to create a final strong learner. The project makes use of the `GradientBoostingClassifier` from the sklearn library.[27] Parameters include `criterion`, `loss`, `learning_rate`, `max_depth`, and `min_samples_split` which control the function to measure the quality of a split, the loss function to be optimized, learning rate which shrinks the contribution of each tree, maximum depth of the individual regression estimators and the minimum number of samples required to split an internal node, respectively.[4]

3.4 Addition of a New Machine Learning Algorithm

3.4.1 Analysis of NB Classifiers for Fake News Detection

Naive Bayes is a family of probabilistic algorithms that take advantage of probability theory and Bayes' Theorem to predict the category of a sample (like a piece of news or a customer review). They are probabilistic, which means that they calculate the probability of each category for a given sample, and then output the category with the highest one. The way they get these probabilities is by using Bayes' Theorem, which describes the probability of a feature, based on prior knowledge of conditions that might be related to that feature.[28]

The Naive Bayes formula is derived from Bayes' Theorem, a fundamental principle in probability theory and statistics which describes the relationship of conditional probabilities of statistical quantities.

The Bayes' Theorem is expressed as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

In the context of Naive Bayes, this formula says that the probability of an event A (like a document being of a certain class) given that another event B has occurred (like the document contains certain words) is proportional to the probability of event B given A times the probability of event A.[28][29]

When we apply Bayes' theorem to a classification problem with features $x_1, x_2, ..x_n$ and classes represented by y, the formula becomes:

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(x_1, x_2, \ldots, x_n|y)P(y)}{P(x_1, x_2, \ldots, x_n)}$$

This formula is interpreted as the probability of class y given a set of features $x_1, x_2, ..x_n$ is equal to the probability of those features given class y times the probability of class y, all divided by the probability of the features.[29] The goal of Naive Bayes is to maximize $P(y|x_1, x_2, ..x_n)$, or the probability of a class given a set of features.

The "naive" assumption in Naive Bayes is that all of the features $x_1, x_2, ..x_n$ are independent from each other. This simplifies the term $P(x_1, x_2, ..x_n|y)$ in the formula to $P(x_1|y)P(x_2|y)...P(x_n|y)$, which is the product of the individual probabilities of each feature given the class.[30]

So, the final form of the Naive Bayes formula is:

$$y = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

This formula says that the class of a set of features is chosen to be the class y that maximizes the product of the prior probability of y (i.e., $P(y)$) and the individual feature probabilities $P(x_i|y)$.[30]

In the context of text classification, each feature x_i would be a word, and $P(x_i|y)$ would be the probability of seeing word x_i in a document of class y. These probabilities can be estimated by counting the frequency of each word in documents of each class.[31]

Naive Bayes classifiers are highly scalable and well-suited for high-dimensional datasets,[32] making them a popular choice for text classification problems, such as spam filtering, sentiment analysis, and, of course, fake news detection.[16] The algorithm's efficiency and effectiveness in these domains can be attributed to its underlying principle: it calculates the probability of each attribute belonging to each class, and then takes the product of these probabilities to compute the probability of a data point belonging to a particular class. This makes it particularly effective for tasks where the input variables are conditionally independent given the output variable.

In the context of fake news detection, the Naive Bayes algorithm can be trained to learn the like-

lihood of certain words and phrases appearing in fake news articles as opposed to real ones.[33] For instance, it might learn that fake news articles are more likely to contain sensationalist or alarmist language, misleading statistics, or references to unreliable sources. By computing the product of these likelihoods, the algorithm can make a probabilistic prediction about whether a given article is more likely to be real or fake.

It is also worth noting that there are several variants of the Naive Bayes algorithm, each with its own strengths and limitations. These include the Multinomial Naive Bayes algorithm, which is commonly used for document classification problems, and the Bernoulli Naive Bayes algorithm, which is useful for binary/boolean features.[34]

For this project, I have chosen to use the Naive Bayes classifiers available in the scikit-learn library. The scikit-learn library provides several variants of Naive Bayes classifiers, and after careful consideration, I have decided to focus on Multinomial Naive Bayes (MNB) and Bernoulli Naive Bayes (BNB), and I am going to examine each variant of Naive Bayes classifiers provided by scikit-learn to assess their suitability for fake news detection:

3.4.2 Gaussian Naive Bayes

Gaussian Naive Bayes(GNB) is not suitable for fake news detection due to the nature of its assumptions and the characteristics of the data involved in fake news detection tasks.[35]

Gaussian Naive Bayes is commonly used for continuous and numeric data, it assumes that the features follow a Gaussian (normal) distribution. However, in the context of fake news detection, the data typically consists of discrete and categorical features, such as word presence or word frequencies, rather than continuous numerical variables. Gaussian Naive Bayes is not designed to handle this type of data.[35]

3.4.3 Categorical Naive Bayes

Categorical Naive Bayes (CNB) is a variant of the Naive Bayes algorithm that is designed for data with discrete categorical features.[36] Unlike the more commonly used MultinomialNB, which is designed for data with countable discrete features, CategoricalNB works with features that have a fixed number of categories.

In theory, CategoricalNB can be used with textual data that has been converted into categorical features. This can be done using techniques such as one-hot encoding, where each word is represented as a binary feature, indicating whether it is present or absent in a given document.[36]

However, in practice, CategoricalNB may not work well with textual data, because of the high dimensionality of the feature space.[36] Textual data typically has a very large number of unique words, which can lead to a large number of features, making it difficult for the algorithm to learn a good model. Additionally, CategoricalNB assumes that the categories are mutually exclusive, which may not be true for textual data, where words can have multiple meanings and belong to multiple categories.

3.4.4 Multinomial and Bernoulli Naive Bayes

I have chosen the Multinomial Naive Bayes (MNB) and Bernoulli Naive Bayes (BNB) for the following reasons:

1. Nature of the Data: The dataset for this project consists of categorical and binary features. MNB and BNB are specifically designed for handling such types of data. MNB is suitable when the features represent counts or frequencies, while BNB is appropriate for binary features.

2. Model Assumptions: MNB assumes that the features follow a multinomial distribution,[37] and BNB assumes a Bernoulli distribution.[38] These assumptions align well with the nature of the data

and can help capture the underlying probabilistic relationships between the features and the target variable.

3. Feature Independence: MNB and BNB also assume feature independence, but their application to textual data differs from Gaussian Naive Bayes. In the case of MNB and BNB for textual data, the assumption of independence is typically applied at the level of individual words or features, rather than considering complex dependencies and interactions between all features. While this assumption may not hold true for all textual data, MNB and BNB have been observed to perform well in practice for tasks like text classification and spam detection.[37][38]

4. Efficient and Scalable: MNB and BNB are computationally efficient and can handle large-scale datasets with high-dimensional feature spaces. They require minimal memory and training time, making them suitable choices for projects where efficiency is a consideration.

Based on these considerations, MNB and BNB are well-suited for this project, given the nature of the data, the assumptions made by the models, and their efficiency. By utilizing these classifiers from the scikit-learn library, I can leverage their capabilities to develop an effective and interpretable solution for the task at hand.

3.4.5 MNB vs BNB

Multinomial Naive Bayes (MNB):

The MNB model is a variation of Naive Bayes that is used when the feature vectors represent the frequencies with which certain events have been generated by a multinomial distribution. This is the distribution that arises in document classification where the occurrence (and count) of words within a document constitutes our features.[37] Each document is represented as a vector where each element corresponds to a word in the vocabulary, and the value represents the number of times the word appears in the document.

In the context of fake news detection, MNB is quite useful. It's capable of modeling the frequency of words, so if there are certain words or phrases that are significantly more common in fake news articles, the MNB model could potentially capture and utilize this information.

Bernoulli Naive Bayes (BNB):

The BNB model, on the other hand, is used for binary/boolean features. This model is used when the feature vectors are binary (0s and 1s), indicating whether or not a particular event has occurred.

For fake news detection, BNB could be used if the text data is transformed into a binary "bag of words" representation. Instead of keeping track of the count of each word (as in MNB).[38] While this does lose some information (i.e., the word count), it might be less sensitive to outliers and could potentially work better if the mere presence of certain words is more important than their frequency.

When working with the hybrid approach, the preferred algorithm would typically be Multinomial Naive Bayes over Bernoulli Naive Bayes.

Multinomial Naive Bayes (MNB):

As discussed in the previous paragraphs, MNB is typically used when the features are counts or relative frequencies. For the hybrid approach outlined, it would seem a better fit, because both TF-IDF and Empath generate continuous feature values. TF-IDF results in term frequencies, which are naturally aligned with MNB's expectation of counts. Empath also provides a form of counts, such as the count of terms associated with a particular emotional tone or topic.

Bernoulli Naive Bayes (BNB):

BNB, on the other hand, is used when the features are binary. It might be less appropriate for

this hybrid approach, given that TF-IDF and Empath features aren't binary. However, if I binarize these feature vectors, indicating the presence or absence of a term (for TF-IDF) or a certain characteristic (for Empath), then BNB might be a reasonable choice.

In conclusion, when working with count-based representations like TF-IDF and Empath, Multinomial Naive Bayes generally performs better. Bernoulli Naive Bayes, on the other hand, is better suited for binary/boolean features. Nonetheless, the best approach can vary depending on the specific characteristics of the dataset and problem, and it's often a good idea to experiment with different approaches to find the best solution.

3.5 Customization of the new ML Algorithm

In this section, I delve into the customization of the Naive Bayes classifier for the critical task of fake news detection. As a part of that application that aims to compare the performance of different machine learning algorithms, I've facilitated user-level customization for this algorithm. I understand that the ability to fine-tune parameters can dramatically influence the classifier's efficacy and thus, the comparative analysis. I have allowed users to adjust three key parameters of both Naive Bayes classifiers: 'alpha', 'fit_prior', and 'class_prior'.

By providing this flexibility, my application empowers users to experiment with these parameters, observe their impacts, and identify the optimal settings for their specific requirements. The customized Naive Bayes model thus serves as a valuable addition to the broader comparison and evaluation of different machine learning algorithms' effectiveness in fake news detection.

3.5.1 The Value of Alpha

In Naive Bayes classifiers, 'alpha' refers to a smoothing parameter, applied to avoid zero probabilities by adding a value to each feature's count. This technique, called Laplace smoothing, prevents erroneous classifications when the model encounters unseen words.[39]

Laplace smoothing addresses the issue of zero probability in Naive Bayes classifiers. When classifying text data, such as for fake news detection, the model utilizes probabilities derived from the training data. However, when an unseen word in a new document is encountered, it is assigned a zero probability, which can significantly impair the classification process. Laplace smoothing, by adding a pseudo-count (usually one) to each word token's frequency, ensures non-zero probabilities for all words, making the Naive Bayes classifier a more robust tool for fake news detection.[40]

In the context of fake news detection, where articles often introduce unique terminologies or language expressions, Laplace smoothing's capability to handle unseen words increases the model's flexibility and generalization ability. This technique assists in mitigating overfitting, where models perform well on the training data but fail on unseen data.[39]

Also, Laplace smoothing regularizes the Naive Bayes model, reducing sensitivity to variations in training data—a valuable feature for fake news detection due to potential wide variations in word distribution across articles.[40]

The incorporation of Laplace smoothing significantly enhances the capability of the Naive Bayes classifier in fake news detection. By assuring non-zero probabilities for all words, it resolves the zero-probability problem and improves the robustness of the model. This leads to better model performance, increased generalizability, and a higher detection rate of fake news articles, thus playing a critical role in the fight against misinformation.

3.5.2 The Impact of 'fit_prior'

The parameter 'fit_prior' plays a considerable role in the application of the Naive Bayes algorithm

for fake news detection. By default, 'fit_prior' is set to True, implying that the model will learn class prior probabilities from the training data. This is especially crucial in domains where the class distribution is imbalanced, such as in the case of fake news detection where the ratio of real to fake news might significantly differ.[37][38]

When 'fit_prior' is set to False, a uniform prior is used, meaning each class is considered equally probable a priori. In situations where the training dataset has an equal number of real and fake news articles, setting 'fit_prior' to False might not significantly affect the classifier's performance. However, in cases where this balance does not exist, not fitting the prior based on the data might hamper the modelś predictive ability. Therefore, the setting of 'fit_prior' must be in alignment with the specific distribution of the dataset for optimal fake news detection.[37][38]

3.5.3 The Influence of 'class_prior'

The 'class_prior' parameter allows for the specification of prior probabilities of the classes. This flexibility can be particularly useful in scenarios where the user has certain domain knowledge about the likely distribution of classes, which is not reflected in the training set. For example, if a user knows that in a particular context, fake news articles are more prevalent than real news, they can set the class prior probabilities accordingly.[37][38]

By default, 'class_prior' is set to None, indicating that the prior probabilities are adjusted according to the data. However, if specified, the priors are not adapted based on the data, offering the user more control over the model. This can be an advantageous feature, especially in contexts where the training data may not be representative of the real-world distribution of classes, such as in evolving landscapes of fake news.[37][38]

In summary, 'alpha', 'fit_prior', and 'class_prior' offer significant opportunities for tailoring the Naive Bayes classifier to match the unique conditions of fake news detection. The 'alpha' parameter in Laplace smoothing effectively handles unseen features in the data, while 'fit_prior' and 'class_prior' provide control over the model's prior probabilities, depending on the dataset and user knowledge. By enabling users to adjust these parameters, I strive to develop a more adaptable and potent tool for combating the spread of misinformation. This flexibility allows the Naive Bayes model to perform more efficiently, thereby facilitating a more reliable comparative analysis between different machine learning algorithms in the critical task of fake news detection.

3.6 The Potential of Transfer Learning and Incremental Learning

Exploiting the inherent strengths of both transfer learning and incremental learning presents an intriguing approach to enhancing the efficacy of Naive Bayes models for fake news detection. These advanced learning strategies have a unique potential to address challenges in the evolving landscape of information dissemination, where the ability to quickly adapt and respond to new patterns is critical.

Transfer learning leverages the pre-existing knowledge acquired from one domain to accelerate and refine the learning process in a different yet related domain. In the context of fake news detection, it could mean applying insights gained from general text classification to discern real from counterfeit news, thereby reducing the requirement for extensive labelled training data specific to fake news.[41]

Conversely, incremental learning focuses on continuously adapting the model with the influx of new data, thus fostering real-time learning and adaptation. This approach is particularly potent for large-scale data or in scenarios exhibiting changing data distributions, characteristics synonymous with the realm of news and social media.[42] Incremental learning ensures that the Naive Bayes model stays current and effective against the ever-evolving strategies employed in

fake news generation.

Unpacking the potential of these learning paradigms in the context of Naive Bayes-based fake news detection, I shall delve into the advantages and limitations of each, and explore their unique contributions to managing the contemporary challenge of fake news.

3.6.1 Overview of Transfer Learning

In the realm of machine learning, Transfer Learning is a methodology that leverages knowledge acquired from one task or domain to solve another task or problem in a different domain.[41] It offers a robust and efficient way to train models, particularly when the target task lacks abundant labeled data or when computational resources and time are limited.

The primary purpose and significance of transfer learning can be understood from the following perspectives:

1. Efficiency in Learning: Transfer learning allows the use of pre-trained models, thus mitigating the need to train models from scratch. This significantly enhances the speed and efficiency of model training.[43]

2. Addressing Data Scarcity: Many real-world problems may lack sufficient labeled data to train a complex model. Transfer learning exploits pre-trained models on large-scale datasets, enabling model fine-tuning with a smaller amount of labeled data from the target task.[44]

3. Enhancing Model Performance: Transfer learning can improve model performance, especially when the target task's dataset is small or noisy.[45]

To illustrate the concept of transfer learning, let's use an analogy. Suppose we want to develop a deep learning model to recognize a specific breed of cat, say the Maine Coon. If we were to start from scratch, we would need to collect and label a large number of Maine Coon images and then train a deep learning model, which could be time-consuming and challenging.

However, with transfer learning, we can utilize a pre-trained model, which has already been trained on a large-scale dataset (such as ImageNet) and has learned a lot of generic knowledge about object recognition, including cats. We then only need to fine-tune this pre-trained model with a smaller dataset of Maine Coon images. This way, we leverage the rich knowledge encapsulated in the pre-trained model, and we can develop a high-performing Maine Coon recognition model with less labeled data and less training time. This demonstrates the power of transfer learning.

Now, let's connect this to the field of fake news detection. Much like identifying a specific breed of cat, detecting fake news is a complex problem that can benefit from the knowledge learned in related tasks. For instance, a pre-trained model on a large-scale text corpus would have learned the nuances of the language, grammar, and context. This knowledge can be transferred to a fake news detection model, where the model can be fine-tuned using a smaller labeled dataset of fake and real news. This can improve the model's ability to detect subtle cues and patterns indicative of fake news, leading to improved performance.

In the context of this study, both Naive Bayes models employed for identifying fake news embrace transfer learning capabilities via the utilization of the `partial_fit` function. This functionality empowers the models to adapt and refine their parameters in real-time, accommodating the dynamic nature of news distribution.

3.6.2 Overview of Incremental Learning

Incremental learning, also known as online learning or out-of-core learning, is a machine learning paradigm that learns from data progressively over time, updating the model with each new piece of data. This is particularly effective when working with large datasets that cannot be loaded into

memory all at once.[46] It's also advantageous when dealing with data streams, where data is continuously generated.[47]

Incremental learning stands in contrast to batch learning, where the entire dataset is used to train the model at once.[48] With incremental learning, a model can be trained and then updated as new data comes in, making it more adaptable to changes in data distribution over time – a phenomenon known as concept drift.

Both Naive Bayes models used for fake news detection in this research supports incremental learning through the `partial_fit` function. It allows the model to update its parameters on the fly as new labeled data becomes available, thus providing a real-time response to the ever-evolving landscape of news dissemination.[49]

Advantages of Incremental Learning:

1. Efficient Large-Data Handling: Incremental learning is adept at processing vast datasets that surpass memory capacity constraints.[50] It accomplishes this by learning iteratively from data in discrete segments, making it a valuable strategy for addressing large-scale data challenges.[46]

2. Dynamic Adaptability: One of the primary advantages of incremental learning is its capability to adapt to changing data patterns over time.[42] This dynamic adaptability becomes particularly crucial when data distribution exhibits temporal variations, with incremental learning ensuring sustained model performance through continuous updates.[51]

3. Real-time Learning Capability: Incremental learning embodies the capacity to learn instantaneously from newly arriving data.[42] This real-time learning attribute is indispensable when models are required to rapidly assimilate and respond to fresh information.[47]

Limitations of Incremental Learning:

1. Risk of Forgetting: A notable challenge in incremental learning is the risk of obliterating previously learned information.[50] This phenomenon can occur, particularly when newly introduced data is markedly disparate or contradictory to past data.

2. The Stability-Plasticity Dilemma: Striking a balance between integrating new information (plasticity) and preserving prior knowledge (stability) presents a critical challenge in incremental learning.[51] Overemphasizing the adaptation to new data could instigate overfitting to recent trends, while underemphasis may render the model inert to novel trends or changes.

3. Sensitivity to Noise: Due to its very nature of updating the model parameters with every incoming data point, incremental learning might exhibit heightened sensitivity to noisy or aberrant data points.[52] This sensitivity could potentially undermine model performance by over-adjusting to the noise.

In essence, the process of incremental learning is a delicate balancing act between accommodating new data while maintaining the integrity of previously acquired knowledge. Each new data point could be seen as a potential opportunity for learning, but also a risk of disruption. Therefore, successful incremental learning requires careful management of these conflicting pressures.[47]

In this project, transfer learning and incremental learning will be achieved by saving a pre-trained model into a pickle file, alongside the vectorization outputs, and then reload these files and then fine-tuning the model with the `partial_fit` function provided by scikit-learn.

3.6.3 Transfer Learning vs Incremental Learning

While both incremental learning and transfer learning aim to leverage prior knowledge, they differ in their approaches and applications. Transfer learning involves applying knowledge learned from one task to enhance learning in a related but

distinct task. It is a strategy to improve learning efficiency or performance when we have some similarities between the source and target tasks or domains.

On the other hand, incremental learning is about adapting to new data over time within the same task. It is a method to handle large-scale or streaming data, and deal with changes in data distribution. Incremental learning doesn't necessarily imply learning from a different task or domain, but instead focuses on continual learning from an ever-growing dataset.

In the context of fake news detection, transfer learning might involve applying knowledge from a model trained on general text classification tasks to the specific task of classifying news as fake or real. In contrast, incremental learning for this task would involve updating the fake news detection model as new news articles become available, helping the model to stay current with the latest trends and tactics used in fake news creation.[53]

3.7 Evaluation Metrics

In order to evaluate the performance of our classifiers, it is essential to establish some measurement criteria. In this project, I have used four common evaluation metrics: Accuracy, Precision, Recall, and F1 Score. Before discussing the four metrics, it is essential to first understand the terms True Positive (TP), True Negative (TN), False Positive (FN), and False Negative (FP). This can be effectively explained through the use of a confusion matrix.

	Predicted: True	Predicted: False
Actual: True	True Positive (TP)	False Negative (FN)
Actual: False	False Positive (FP)	True Negative (TN)

Table 1: Confusion Matrix

Below are the definition and formulae for each metrics:

- Accuracy: This is the most intuitive performance measure and it is simply a ratio of correctly predicted observations to the total observations. It determines the overall correctness of the classifier.

$$Accuracy = (TP+TN)/(TP+TN+FP+FN)$$

- Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. It evaluates the exactness or quality of the classifie.

$$Precision = TP/(TP+FP)$$

- Recall (Sensitivity): Recall is the ratio of correctly predicted positive observations to the all observations in actual class. It measures the completeness, or the ability of the classifier to correctly find all positive instances.

$$Recall = TP/(TP+FN)$$

- F1 Score: F1 Score is the weighted average of Precision and Recall. It tries to find the balance between precision and recall. It is especially useful in cases of uneven class distribution.

$$F1score = 2*(Precision*Recall)/(Precision+Recall)$$

4 Implementation

This research project extends an existing web application built using Python's Dash framework, incorporating new machine learning models and features for enhanced user customization.

4.1 Machine Learning Models

Two variations of Naive Bayes classifiers from the scikit-learn library, namely Multinomial Naive Bayes (MNB) and Bernoulli Naive Bayes (BNB), were integrated into the web application. The user interface was modified to include a radio button selection, allowing users to choose their preferred Naive Bayes classifier for their specific needs.

4.2 Parameter Customization

To enhance flexibility and allow users to tailor the machine learning models to their specific requirements, I added a feature allowing users to customize all the parameters that the original MNB and BNB classes from scikit-learn offer. This customization ensures that the system can adapt to a range of different datasets and problem contexts.

4.3 Transfer Learning & Incremental Learning

An additional feature was added to support transfer and incremental learning. A radio button option was included to give users the choice to apply these learning techniques. If selected, a text field prompts users to input the location of their pre-trained model. The pre-trained model and TF-IDF vectorizer are saved separately as pickle files. When transfer or incremental learning is applied, the pickle files are loaded and used to fine-tune the pre-trained model with the new dataset.

In this implementation, the focus was not only on integrating machine learning models into a web application but also on giving users the flexibility to tailor the application to meet their specific needs and also allow the user to take advantage of transfer and incremental learning. The added features enable users to select and customize models, apply transfer or incremental learning, and load pre-trained models, allowing them to build on previous work and fine-tune models with new datasets.

5 Experiments and Results

In total 24 experiments will be conducted to evaluate the performance of various machine learning (ML) models and explore different techniques on specific datasets.

The first 12 experiments aim to compare the performance of all six ML models on three different datasets. The rest 12 experiments focus on testing the potential of Transfer and Incremental Learning using Multinomial and Bernoulli Naive Bayes classifiers on the various datasets.

5.1 Experiments Setup

In order to assess and compare the performance of the various models under consideration, each experiment will have a consistent setup. The primary goal of this setup is to provide a fair and consistent basis for comparison across different models and feature extraction methods.

Datasets: The Merged, Kaggle 1, Kaggle 2 and FA-KES datasets will be used for the first 12 Experiments. Kaggle 2 and Trimmed-Kaggle 1 will be used to train the pre-trained model for transfer learning, and all the `trimed` datasets will be used to train the pre-trained models for incremental learning, and all of these pre-trained models will be tested on larger datasets - WELFake, Kaggle 1 and Scraped. The Kaggle 2 dataset and all the `trimed` datasets will be also referred to as the ”pre-training” datasets in the following paragraphs.

Train-Test Split: The data in each dataset will be divided into a training set and a test set, using a 90-10 split for the first 12 experiments, 10-fold cross-validation will be performed, and the mean value for each measurement will be taken. For the rest 12 experiments, 95% of the pre-training datasets will be used for the pre-trained models, and 15% of the testing datasets will be used to fine-tune those pre-trained models.

Feature Extraction: We will be using three different feature extraction methods: TF-IDF, Empath, and the novel Hybrid approach for the first 12 experiments. For the rest experiments, TF-IDF will be used.

Timing: For the first 12 experiments, the value of `Runtime` indicates the time for vectorization and training the models, as I am also comparing the performance between different feature extraction techniques, for the rest experiments, the value of `Runtime` indicates the time to train (or pre-train and fine-tune) the models.

5.2 Comparing ML Models with TF-IDF, Empath and the Hybrid Approach

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	Logistic Regression	0.95	0.94	0.95	0.95	12.39
tf-idf	Decision Tree	0.88	0.87	0.88	0.88	42.26
tf-idf	KNN	0.81	0.76	0.84	0.8	34.11
tf-idf	Gradient Boosting	0.94	0.93	0.94	0.93	196.45
tf-idf	BNB	0.81	0.74	0.93	0.82	10.96
tf-idf	MNB	0.86	0.99	0.70	0.82	10.89
Empath	Logistic Regression	0.69	0.75	0.49	0.59	126.95
Empath	Decision Tree	0.67	0.64	0.67	0.65	128.1
Empath	KNN	0.73	0.74	0.62	0.67	127.6
Empath	Gradient Boosting	0.77	0.77	0.72	0.74	150.76
Empath	BNB	0.62	0.58	0.59	0.59	126.79
Empath	MNB	0.55	0.95	0.03	0.05	126.75
Hybrid	Logistic Regression	0.96	0.94	0.95	0.95	172.28
Hybrid	Decision Tree	0.91	0.9	0.91	0.9	173.89
Hybrid	KNN	0.84	0.8	0.79	0.8	173.28
Hybrid	Gradient Boosting	0.95	0.93	0.95	0.94	256.1
Hybrid	BNB	0.85	0.76	0.95	0.84	171.7
Hybrid	MNB	0.89	0.97	0.75	0.84	169.09

Table 2: Results of Kaggle 1 Dataset

Three different feature extraction methods, namely tf-idf, Empath, and the Hybrid approach, are each paired with these six models. In these results, models using tf-idf and the Hybrid method generally perform better. Logistic Regression performs the best among all models with an accuracy of 0.96.

In terms of runtime, the Gradient Boosting model using Hybrid approach takes the longest, reaching 256.1 seconds, while the Multinomial Naive Bayes (MNB) and Bernoulli Naive Bayes (BNB) models using TF-IDF have the shortest runtime, both around 11 seconds.

MNB outperformed BNB with tf-idf and the hybrid approach, with not only a higher accuracy but also a shorter runtime.

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	Logistic Regression	0.96	0.94	0.99	0.96	1.96
tf-idf	Decision Tree	0.95	0.95	0.96	0.95	2.9
tf-idf	KNN	0.88	0.86	0.94	0.89	2.25
tf-idf	Gradient Boosting	0.97	0.96	0.98	0.97	30.1
tf-idf	BNB	0.92	0.93	0.92	0.93	1.72
tf-idf	MNB	0.89	0.84	0.98	0.89	1.71
Empath	Logistic Regression	0.66	0.64	0.87	0.74	18.98
Empath	Decision Tree	0.77	0.79	0.78	0.79	19.1
Empath	KNN	0.76	0.75	0.83	0.79	19.26
Empath	Gradient Boosting	0.86	0.85	0.9	0.87	22.19
Empath	BNB	0.62	0.65	0.65	0.65	18.96
Empath	MNB	0.56	0.55	0.99	0.71	18.95
Hybrid	Logistic Regression	0.98	0.97	0.99	0.98	51.71
Hybrid	Decision Tree	0.96	0.97	0.95	0.96	51.1
Hybrid	KNN	0.89	0.88	0.93	0.9	52
Hybrid	Gradient Boosting	0.99	0.99	0.98	0.99	144.82
Hybrid	BNB	0.92	0.94	0.91	0.93	51.7
Hybrid	MNB	0.9	0.87	0.98	0.92	48.36

Table 3: Results of Kaggle 2 Dataset

Like the results from Kaggle 1 dataset, all models tend to perform better with tf-idf and the Hybrid approach, especially the novel Hybrid approach, which yield 99% and 98% accuracy for Gradient Boosting and Logistic Regression.

Talking about runtime, MNB and BNB are still the fastest among all six models. However this time, BNB outperformed MNB with a higher value in all four measurements, meanwhile MNB is still faster than BNB.

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	Logistic Regression	0.97	0.98	0.96	0.97	3.95
tf-idf	Decision Tree	0.99	0.99	0.99	0.99	5.95
tf-idf	KNN	0.83	0.88	0.77	0.83	5.28
tf-idf	Gradient Boosting	0.99	1	0.99	0.99	71.7
tf-idf	BNB	0.94	0.95	0.92	0.94	2.9
tf-idf	MNB	0.93	0.93	0.93	0.93	2.88
Empath	Logistic Regression	0.76	0.76	0.75	0.76	60.63
Empath	Decision Tree	0.75	0.75	0.76	0.75	61.33
Empath	KNN	0.76	0.78	0.73	0.75	60.93
Empath	Gradient Boosting	0.82	0.84	0.8	0.82	73.98
Empath	BNB	0.72	0.73	0.69	0.71	60.6
Empath	MNB	0.76	0.8	0.69	0.75	60.57
Hybrid	Logistic Regression	0.98	0.98	0.99	0.98	127.78
Hybrid	Decision Tree	0.99	1	1	0.99	127.6
Hybrid	KNN	0.89	0.88	0.83	0.87	128.59
Hybrid	Gradient Boosting	0.99	1	0.99	0.99	394.79
Hybrid	BNB	0.97	0.97	0.94	0.95	126.57
Hybrid	MNB	0.94	0.95	0.92	0.94	126.34

Table 4: Results of Merged Dataset

All models perform exceptionally well with tf-idf, particularly Decision Trees and Gradient Boosting, both achieving near-perfect accuracy and F1 scores. This indicates that almost all predictions are correct. However, KNN lags slightly behind, with both accuracy and F1 scores around 0.83. This could be attributed to KNN's less effective handling of high-dimensional features. BNB and MNB also exhibit strong performance with accuracy and F1 scores in the 0.93-0.94 range.

Except for KNN, all models perform exceedingly well with the hybrid method. Decision Trees and Gradient Boosting models achieve near-perfect accuracy and F1 scores. Logistic Regression, BNB, and MNB also show strong performance, with accuracy and F1 scores in the 0.94-0.98 range. Yet, as with the TF-IDF results, KNN is slightly weaker, with accuracy and F1 scores both around 0.87.

Meanwhile, Empath is outperformed by the other two feature extraction techniques as in the previous six experiments.

In terms of runtime, Gradient Boosting takes the longest in all scenarios, particularly with the Hybrid feature extraction method, requiring nearly 400 seconds. This is likely due to its iterative nature, requiring more time for gradual improvements in its predictions. Conversely, BNB and MNB consistently take the least amount of time across all scenarios, possibly due to their simplicity as models based on Naive Bayes theory, which have lower computational demands.

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	Logistic Regression	0.47	0.47	0.76	0.58	0.17
tf-idf	Decision Tree	0.54	0.52	0.57	0.54	0.46
tf-idf	KNN	0.49	0.48	0.7	0.57	0.19
tf-idf	Gradient Boosting	0.46	0.45	0.55	0.5	4.6
tf-idf	BNB	0.51	0.48	0.73	0.59	0.15
tf-idf	MNB	0.49	0.48	1	0.65	0.15
Empath	Logistic Regression	0.53	0.53	1	0.7	5.18
Empath	Decision Tree	0.53	0.56	0.56	0.56	5.19
Empath	KNN	0.51	0.54	0.58	0.55	5.35
Empath	Gradient Boosting	0.55	0.57	0.65	0.6	5.86
Empath	BNB	0.51	0.53	0.63	0.58	5.17
Empath	MNB	0.53	0.53	1	0.7	5.16
Hybrid	Logistic Regression	0.52	0.57	0.6	0.59	12.41
Hybrid	Decision Tree	0.48	0.54	0.51	0.53	12.1
Hybrid	KNN	0.53	0.58	0.63	0.6	12.41
Hybrid	Gradient Boosting	0.45	0.51	0.5	0.5	22
Hybrid	BNB	0.54	0.56	0.93	0.7	12.39
Hybrid	MNB	0.57	0.57	0.97	0.72	11.79

Table 5: Results of FA-KES Dataset

The performance of the models generally declines on the FA-KES dataset, which is smaller in size compared to the first two datasets. Regardless of the feature extraction method used, all models' accuracy lies between 0.45 to 0.57, significantly lower than those on the first two datasets. This might be due to the smaller dataset size, making it harder for the models to learn sufficient patterns for accurate prediction. And that is one major reason for me to consider these results as unreliable.

Another point worth noting here is that the MNB model achieved 100% recall with tf-idf and Empath, meaning that the model simply recognizes all the news articles as fake, the same happened with LR using Empath. This can also be attributed to the lack of data.

These results underscore the crucial impact of data volume on model performance. A smaller dataset might hinder the model from learning effective patterns, thus affecting its accuracy and other performance metrics.[5] Therefore, when dealing with smaller datasets, more complex models or more effective feature engineering might be necessary to enhance model performance.

5.2.1 Overall Analysis

Overall, it is evident that Logistic Regression consistently achieves the highest accuracy across all three datasets, positioning it as the most reliable model for prediction. When time efficiency is a priority, the Multinomial Naive Bayes (MNB) and Bernoulli Naive Bayes (BNB) models stand out, clocking in the shortest runtime.

In terms of feature extraction methods, the Hybrid approach invariably enhances all models' performance. However, this comes with the trade-off of a considerably longer runtime. The TF-IDF method, on the other hand, provides a favorable balance between performance and speed, yielding respectable accuracy in a relatively short time.

An interesting dynamic is seen between MNB and BNB models. MNB shows superior performance when dealing with larger datasets, while BNB takes the lead in the context of smaller data volumes,[34] demonstrating the importance of choosing the right model based on the specific characteristics of the dataset.

5.3 Results and Discussion on Transfer Learning

5.3.1 Pre-training on Kaggle 2 for Testing on Kaggle 1

Time taken for training the pre-trained model: 0.0083 seconds(BNB), 0.0059(MNB).

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.81	0.72	0.92	0.81	0.07
tf-idf	MNB	0.86	0.98	0.7	0.82	0.03

Table 6: Results of Kaggle 1 Dataset without Transfer Learning

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.78	0.71	0.89	0.79	0.0183
tf-idf	MNB	0.78	0.74	0.82	0.78	0.01

Table 7: Results of Kaggle 1 Dataset with Transfer Learning

While both models exhibited a minor dip in accuracy, the process of fine-tuning the pre-trained models proved to be significantly more time-efficient than training a new model from the ground up. Specifically, the Bernoulli Naive Bayes (BNB) model demonstrated a speed increase by a factor of seven, while the Multinomial Naive Bayes (MNB) model was six times faster. This represents a considerable advantage, underscoring the potential benefits of leveraging pre-trained models in terms of computational efficiency.[44]

In order to demonstrate the significance of transfer learning, I conducted a one-time experiment wherein I trained a model and saved it to a pickle file. I then directly applied this pre-trained model to a new dataset without utilizing transfer learning:

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.35	0.36	0.53	0.43	0.0162
tf-idf	MNB	0.44	0.45	0.92	0.6	0.008

Table 8: Results of applying Kaggle 2 model on Kaggle 1 without Transfer Learning

From these results, it can be noted that the performance of the models was notably diminished when applied to a new dataset without transfer learning. These results underpin the importance of transfer learning when applying pre-trained models to new datasets, highlighting its role in maintaining and potentially improving model performance. It's important to note that this process was carried out only once for illustrative purposes and will not be repeated for each individual experiment in the project. The key takeaway from this experiment is the crucial role of transfer learning in ensuring the generalization capability of machine learning models.

5.3.2 Pre-training on Kaggle 2 for Testing on FA-KES

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.49	0.48	0.72	0.58	0.12
tf-idf	MNB	0.48	0.48	1	0.6	0.12

Table 9: Results of FA-KES Dataset without Transfer Learning

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.5	0.53	0.61	0.57	0.0133
tf-idf	MNB	0.53	0.53	1	0.69	0.0079

Table 10: Results of FA-KES Dataset with Transfer Learning

From the results tables, it's clear that the application of transfer learning to the FA-KES Dataset did not lead to any significant improvements. Notably, the recall for the MNB model remains at 1 in both scenarios, indicating that the model has classified every single news article in the dataset as fake. This phenomenon was also observed in the previous experiment, leading to concerns about the model's validity. As such, these results have been deemed unreliable, prompting us to disregard them for our analysis.

5.3.3 Pre-training on Trimmed-Kaggle 1 for Testing on Kaggle 2

Time taken for training the pre-trained model: 0.014 seconds(BNB), 0.006(MNB).

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.87	0.82	0.93	0.86	0.0076
tf-idf	MNB	0.84	0.79	0.96	0.82	0.0074

Table 11: Results of Kaggle 2 Dataset without Transfer Learning

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.51	0.55	0.54	0.54	0.019
tf-idf	MNB	0.53	0.75	0.18	0.29	0.009

Table 12: Results of Kaggle 2 Dataset with Transfer Learning

Based on the results I have obtained, the application of transfer learning on the Kaggle 2 Dataset has actually decreased model performance rather than enhancing it.

This significant reduction in performance indicates that the transfer learning experiment was not successful in this context. It's possible that the pre-trained model didn't align well with the Kaggle 2 Dataset.

Therefore, this experiment can be deemed unsuccessful, highlighting the importance of ensuring compatibility between the pretraining and target tasks in a transfer learning scenario.[45]

5.3.4 Overall Analysis

The results from these experiments demonstrated that while transfer learning significantly enhances computational efficiency, its effectiveness in improving model performance varies based on the dataset used. In our case, an improvement in model performance was observed when testing on the Kaggle 1 dataset, but the application of transfer learning led to a reduction in performance for the Kaggle 2 dataset and yielded unreliable results on the FA-KES dataset.

These findings highlight the importance of dataset compatibility in transfer learning applications. It also underscores the need to consider the specific characteristics of the datasets when designing machine learning systems, as the success of transfer learning appears to be highly dependent on the similarity between the pretraining and target tasks.

5.4 Results and Discussion on Incremental Learning

5.4.1 Pre-training on Trimmed-Kaggle 1 for Testing on Kaggle 1

Time taken for training the pre-trained model: 0.014 seconds(BNB), 0.006(MNB).

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.81	0.72	0.92	0.81	0.07
tf-idf	MNB	0.86	0.98	0.7	0.83	0.03

Table 13: Results of Kaggle 1 Dataset without Incremental Learning

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.88	0.81	0.95	0.87	0.023
tf-idf	MNB	0.86	0.98	0.73	0.83	0.01

Table 14: Results of Kaggle 1 Dataset with Incremental Learning

Both models that employed incremental learning techniques demonstrated superior performance compared to their counterparts trained from scratch. The time investment required to fine-tune these two models was comparable to the time needed for transfer learning. However, they showcased considerably higher accuracy than those using transfer learning. This can be primarily attributed to the similarities between the pre-training dataset and the final testing dataset. By leveraging the inherent alignment between the two datasets, the models could more effectively learn and generalize, leading to enhanced accuracy.[52]

5.4.2 Pre-training on Trimmed-WELFake for Testing on WELFake

Time taken for training the pre-trained model: 0.009 seconds(BNB), 0.004(MNB).

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.9	0.89	0.9	0.89	0.14
tf-idf	MNB	0.86	0.9	0.82	0.85	0.06

Table 15: Results of WELFake Dataset without Incremental Learning

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.86	0.83	0.89	0.86	0.037
tf-idf	MNB	0.85	0.87	0.83	0.85	0.014

Table 16: Results of WELFake Dataset with Incremental Learning

The results indicate that applying incremental learning to the WELFake Dataset with both the BNB and MNB models results in a decrease in runtime without a substantial sacrifice in performance. Without incremental learning, the BNB and MNB models achieved accuracy scores of 0.9 and 0.86 respectively. However, the runtime was fairly higher, recorded at 0.14 and 0.06. After applying incremental learning, there's a slight reduction in accuracy, but the runtime improved significantly to 0.028 and 0.01 respectively. This demonstrates that incremental learning has the advantage of improving computational efficiency, which can be particularly beneficial in cases where computational resources or time are constraints.

5.4.3 Pre-training on Trimmed-Scraped for Testing on Scraped

Time taken for training the pre-trained model: 0.008 seconds(BNB), 0.003(MNB).

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.77	0.75	0.58	0.65	0.035
tf-idf	MNB	0.76	0.98	0.34	0.51	0.018

Table 17: Results of Scraped Dataset without Incremental Learning

Feature Extraction	Model	Accuracy	Precision	Recall	F1 Score	Runtime
tf-idf	BNB	0.82	0.8	0.67	0.73	0.0204
tf-idf	MNB	0.58	0.95	0.42	0.58	0.008

Table 18: Results of Scraped Dataset with Incremental Learning

The experiment with the Scraped Dataset shows mixed results when applying incremental learning. Although the runtime improved significantly for both the BNB and MNB models, the performance measures exhibited considerable fluctuations.

Firstly, considering the Bernoulli Naive Bayes (BNB) model, we see an improvement in accuracy, from 0.77 to 0.82, and a decrease in runtime, from 0.035 to 0.0114. Although these changes are relatively small, they suggest that incremental learning may offer some benefits for this model.

However, the MNB model's performance decreased significantly, with accuracy dropping from 0.76 to 0.58. This highlights that the benefits of incremental learning may not be uniform across all models or datasets.
The high precision but low recall score for MNB after applying incremental learning suggests that the model is not identifying all the positive instances correctly (low recall), but the instances it does identify as positive are indeed positive (high precision). In this context, "positive" refers to correctly identifying fake news. This could indicate that the model is very conservative and only labels news as false when it is highly certain.

The reason behind this result could be the nature of incremental learning. Incremental learning is designed to work on the assumption that the model is regularly updated with new data.[42] However, if the incoming data is not representative of the overall distribution of the dataset or if it introduces a new concept that the model has not encountered before, the model might struggle to correctly classify the instances.

This observation aligns with the nature of the Scraped Dataset, which encompasses news articles gathered via web crawlers from a myriad of sources. Given this diversity, it's plausible that the incoming data may not adhere closely to the distribution or characteristics of the initial data used for pretraining the model.

5.4.4 Overall Analysis

The results from the experiments with incremental learning underscore its potential to improve computational efficiency, with reduced runtimes observed in all cases. However, the impact on model performance varied depending on the dataset and the model used.

Incremental learning improved the accuracy of both models when tested on the Kaggle 1 dataset, attributed to the strong alignment between the pre-training and final testing datasets. However, when applied to the WELFake dataset, it resulted in a slight decrease in performance despite the improvement in runtime.

Interestingly, the impact of incremental learning was markedly different on the two models when applied to the Scraped dataset. While the Bernoulli Naive Bayes (BNB) model demonstrated slight improvements in accuracy, the Multinomial Naive Bayes (MNB) model's performance significantly declined. And it can be attributed to the inconsistency between the data for pre-training the model and the date for fine-tuning the model.

These findings suggest that the benefits of incremental learning are not universal, and its application needs to be evaluated based on the specific models and datasets in question. Particularly for diverse datasets like Scraped, the benefits of incremental learning may be limited due to the heterogeneous nature of incoming data.

6 Conclusion

This thesis aimed to improve an existing application for fake news detection by integrating additional machine learning libraries, enabling user parameter adjustments, and investigating the application of transfer learning across diverse datasets. Through numerous experiments, we have explored the complex landscape of fake news detection, emphasizing the importance of choosing the appropriate machine learning models, fine-tuning parameters, and effectively utilizing strategies such as transfer learning and incremental learning.

6.1 Summary of Main Contributions

1. **Integration of Additional Machine Learning Libraries**: The thesis successfully integrated an additional machine learning library into the existing application, thereby expanding the repertoire of available tools for fake news detection. This enhancement has diversified the application's methodological arsenal and widened the range of its capabilities.

2. **Enhanced User Control**: This project developed and incorporated a new feature that allows users to adjust key parameters of the application. This innovation boosts the application's versatility and adaptability, enabling it to cater to diverse needs and specific contexts.

3. **Investigation of Transfer Learning**: The work undertook a comprehensive exploration of transfer learning's potential in fake news detection. This investigation, while revealing that transfer learning does not always improve performance, has offered valuable insights that contribute to a nuanced understanding of the method's applicability in this domain.

4. **Exploration of Incremental Learning**: The project conducted a detailed examination of incremental learning's effectiveness, demonstrating its ability to reduce runtime, albeit occasionally at the cost of slight performance reductions. This trade-off is a significant finding that could inform decision-making in contexts where processing efficiency is paramount.

5. **Empirical Evaluation Across Datasets**: The work conducted rigorous empirical evaluations across various datasets, providing a comparative analysis of different machine learning models and feature extraction methods. These results contribute to a deeper understanding of fake news detection's complexities and offer guidance for selecting suitable methods based on specific dataset characteristics.

6.2 Discussion of the Ethical Considerations

The development and application of machine learning models for detecting fake news poses several substantial ethical dilemmas. One of the central issues is the inherently subjective nature of judging the authenticity or credibility of a piece of news. In many instances, news can contain a blend of true and false information, making it a challenge to categorize as wholly 'fake' or 'real'. The question of where the line is drawn, and who decides the degree of accuracy or error acceptable in the label of 'fake news', raises critical concerns about objectivity and potential for nuanced misinterpretations by the machine learning models.

Moreover, the digital transformation of news consumption has led to a decentralization of information dissemination, with less reliance on traditional mainstream media. Many people now receive their information from diverse sources, including social media platforms, YouTube, blogs, among others. This diversified landscape has significantly complicated the problem of fake news, making the control or tracking of misinformation spread a daunting task.

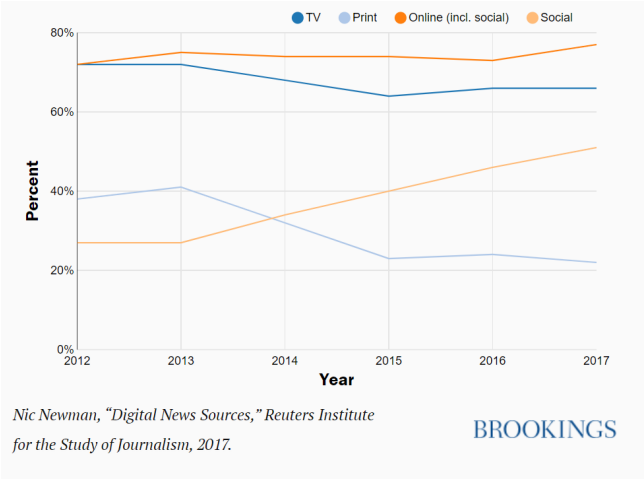

Figure 1: Change in overall news sources, 2012-2017

Another important consideration is the delicate balance between combating fake news and preserving press freedom.[7] While it is imperative to counteract misinformation, there's a risk that overregulation or excessive reliance on these machine learning models could inadvertently stifle legitimate, albeit unpopular or controversial, voices.

One further point of contention is the double-edged nature of AI. If we can harness the power of AI to detect fake news, we must also recognize the potential for using AI to generate fake news. Advances in technology, such as Stable Diffusion and GPT-3 like models, have demonstrated that AI can be used to create highly convincing fake news, with the help of AI, the pipeline of generating fake news can be automated, resulting in a higher throughput of fake news production. This underscores the importance of developing sophisticated detection models, but also highlights the need for stringent regulations and ethical guidelines to prevent misuse.

Lastly, there exists the potential for misuse of such tools. If not adequately regulated, these tools could be exploited by malicious actors for nefarious purposes, like disseminating more refined and harder-to-detect fake news, leading to increased societal polarization and discord.[7]

6.3 Limitations and Future Directions

Despite the contributions this project made to the existing application and our understanding of fake news detection, there are several limitations to note:

1. **Language Limitations**: Currently, the application primarily supports datasets in the English language. Its functionality and performance may not translate well to datasets in other languages due to differences in linguistic structures, syntax, semantics, and cultural contexts.

2. **Binary Classification**: At present, the application operates on a binary classification model, differentiating only between 'fake' and 'not-fake' news articles. This approach oversimplifies the multifaceted nature of misinformation, which exists on a spectrum and includes elements like satire, misinformation, disinformation, and propaganda.

3. **Lack of Dataset Diversity**: The datasets used in the project may not represent the full diversity and complexity of fake news. Fake news varies by source, topic, and stylistic features. If the training data doesn't reflect this variety, the models' generalizability could be impacted.

4. **Runtime and Computational Resources**: Despite improvements, the computational resources required for the novel hybrid approach when applied to large datasets may still be significant. This can limit the accessibility and scalability of the application, particularly for users with less powerful computational resources.

Moving forward, there are several promising directions for enhancing the application's capabilities and addressing the current limitations:

1. **Refining Transfer Learning**: The potential of transfer learning demonstrated in this thesis, despite some setbacks, suggests that it remains a promising avenue for further investigation. Future research could involve implementing transfer learning with other machine learning models, and integrate transfer learning with the novel hybrid approach.

2. **Refining Incremental Learning**: The results of the thesis suggest that incremental learning can reduce runtime with a slight tradeoff in performance. Future work could involve combining this approach with the novel hybrid feature extraction method, potentially enhancing performance while maintaining efficiency.

3. **Addition of New ML Models and Deep Learning Models**: The application's functionality could be broadened further by incorporating a wider array of machine learning models. The addition of deep learning models, in particular, could offer enhanced capabilities, given their well-documented efficacy in identifying intricate patterns within large datasets.

Appendices

This section contains supplementary materials that provide additional insights into the research and methods used in this thesis.

A Python Scripts

```python
import csv
import sys
import random
import collections

maxInt = sys.maxsize

while True:
    # decrease the maxInt value by factor 10
    # as long as the OverflowError occurs.
    try:
        csv.field_size_limit(maxInt)
        break
    except OverflowError:
        maxInt = int(maxInt/10)


def extract_rows(input_file, output_file_1, output_file_2, num_rows,
    target_column_index):
    with open(input_file, 'r', newline='', encoding='utf-8') as file:
        reader = csv.reader(file)
        header = next(reader) # Read and store the header

        data = list(reader) # Read all rows into a list
        random.shuffle(data) # Shuffle the list

        data_1 = data[:num_rows] # First num_rows for training
        data_2 = data[num_rows:] # Remaining rows

        # Count the number of 1s and 0s in the target column of the training data
        counter = collections.Counter(row[target_column_index] for row in data_1)
        print(f"In the trimmed dataset: {counter}")

    with open(output_file_1, 'w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerow(header)
        writer.writerows(data_1)

    with open(output_file_2, 'w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerow(header)
        writer.writerows(data_2)
```

```
42
43
44 # Usage example
45 input_file = 'WELFake_Dataset.csv'
46 output_file_1 = 'trimmed-WEL.csv'
47 output_file_2 = 'remaining-WEL.csv'
48 num_rows = 3000
49 target_column_index = -1 # change it to the number of column where the labels are
       located
50
51 extract_rows(input_file, output_file_1, output_file_2, num_rows, target_column_index)
```

Listing 1: Python Script to Trim Large Datasets

```
import time

from sklearn.feature_extraction.text import TfidfVectorizer

# import necessarry modules and methods

from helper import word_stemming, \
    remove_punctuation_stopwords, shuffle_csv, keep_columns, drop_na

import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
import pickle

results = []
final_div = []

data = pd.read_csv('./datasets/SCRAPED.csv')

# Create a DataFrame
df = pd.DataFrame(data)

feature_to_vectorize = 'text'
label = 'label'
slider_val = 15
alg = ['Naive Bayes']

df = drop_na(df, [feature_to_vectorize])

df = keep_columns(df, [feature_to_vectorize, label])

df = shuffle_csv(df)

df = remove_punctuation_stopwords(df, [feature_to_vectorize])

df = word_stemming(df, [feature_to_vectorize])

x = df[feature_to_vectorize]
y = df[label]

x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=(100 - slider_val) / 100)

# ------------- Code for pre-training the model ----------------
vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)

```

```
pickle.dump(vectorization, open("vectorizer.sav", 'wb'))

MNB = MultinomialNB()
start_time = time.time()
MNB.fit(xv_train, y_train)

filename = 'pickles/NaiveBayes-data.sav'

pickle.dump(MNB, open(filename, "wb"))

# ------------- Code for Transfer / Incremental Learning ----------------
#MNB = pickle.load(open("pickles/NaiveBayes-data.sav", "rb"))
#vectorization = pickle.load(open("vectorizer.sav", 'rb'))

# transform new data
#xv_train = vectorization.transform(x_train)
#xv_test = vectorization.transform(x_test)
#start_time = time.time()
#MNB.partial_fit(xv_train, y_train, classes=np.unique(y_train))
# -----------------------------------------------------------------------
end_time = time.time()
BNB_time = end_time - start_time

pred_bnb = MNB.predict(xv_test)
score = MNB.score(xv_test, y_test)

results.append(classification_report(
    y_test, pred_bnb, output_dict=True))
```

Listing 2: Python Script for Transfer / Incremental Learning with MNB

B Dependencies

Library	Version
dash	2.7.0
dash-bootstrap-components	1.2.1
dash-core-components	2.0.0
dash-daq	0.5.0
dash-extensions	0.1.8
dash-html-components	2.0.0
dash-renderer	0.9.0
dash-table	5.0.0
dateparser	1.1.1
empath	0.89
Flask	2.2.2
Flask-Caching	2.0.1
fpdf	1.7.2
h11	0.9.0
h2	3.2.0
h5py	3.8.0
hpack	3.0.0
Jinja2	3.1.2
matplotlib	3.6.2
nltk	3.7
ntlm-auth	1.5.0
numpy	1.24.3
oauthlib	3.2.2
openpyxl	3.0.10
pandas	1.4.2
plotly	5.11.0
pluggy	0.12.0
scikit-learn	1.1.2
scipy	1.9.2
six	1.16.0
sklearn	0.0
langdetect	1.0.9

Table 19: Python Libraries and Their Corresponding Versions Used in the Project

References

[1] X. Zhang and A. A. Ghorbani, "An overview of online fake news: Characterization, detection, and discussion," *Information Processing & Management*, vol. 57, no. 2, p. 102025, 2020.

[2] S. I. Manzoor, J. Singla, *et al.*, "Fake news detection using machine learning approaches: A systematic review," in *2019 3rd international conference on trends in electronics and informatics (ICOEI)*, pp. 230–234, IEEE, 2019.

[3] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *science*, vol. 359, no. 6380, pp. 1146–1151, 2018.

[4] B. Chaushi, "A benchmark framework for mainstream fake news detection algorithms and a novel hybrid word embedding approach," 2023.

[5] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization.," tech. rep., Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[6] E. Fast, B. Chen, and M. S. Bernstein, "Empath: Understanding topic signals in large-scale text," in *Proceedings of the 2016 CHI conference on human factors in computing systems*, pp. 4647–4657, 2016.

[7] D. P, *Ethical Considerations in Data-Driven Fake News Detection*, pp. 205–232. Cham: Springer International Publishing, 2021.

[8] J. Y. Khan, M. T. I. Khondaker, S. Afroz, G. Uddin, and A. Iqbal, "A benchmark study of machine learning models for online fake news detection," *Machine Learning with Applications*, vol. 4, p. 100032, 2021.

[9] A. Albahr and M. Albahar, "An empirical comparison of fake news detection using different machine learning algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 9, 2020.

[10] R. Barua, R. Maity, D. Minj, T. Barua, and A. K. Layek, "F-nad: An application for fake news article detection using machine learning techniques," in *2019 IEEE Bombay Section Signature Conference (IBSSC)*, pp. 1–6, 2019.

[11] E. A. Hassan and F. Meziane, "A survey on automatic fake news identification techniques for online and socially produced data," in *2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pp. 1–6, 2019.

[12] R. R. Mandical, N. Mamatha, N. Shivakumar, R. Monica, and A. N. Krishna, "Identification of fake news using machine learning," in *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1–6, 2020.

[13] C. B. Do and A. Y. Ng, "Transfer learning for text classification," in *Advances in Neural Information Processing Systems* (Y. Weiss, B. Schölkopf, and J. Platt, eds.), vol. 18, MIT Press, 2005.

[14] M. Werner, C. Hahn, and L. Schauer, "Deepmovips: Visual indoor positioning using transfer learning," in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–7, IEEE, 2016.

[15] D. Vergara, S. Hernández, and F. Jorquera, "Multinomial naive bayes for real-time gender recognition," in *2016 XXI Symposium on Signal Processing, Images and Artificial Vision (STSIVA)*, pp. 1–6, 2016.

[16] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam filtering with naive bayes-which naive bayes?," in *CEAS*, vol. 17, pp. 28–69, Mountain View, CA, 2006.

[17] R. E. S. J. M. F. M. Abu Salem, Fatima K; Al Feel, "Fa-kes: A fake news dataset around the syrian war." https://doi.org/10.5281/zenodo.2607278.

[18] C. Bisaillon, "Fake and real news." https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset.

[19] AMI, "Simplified fake news dataset." https://www.kaggle.com/datasets/fanbyprinciple/simplified-fake-news-dataset?select=test.csv.

[20] S. SHAHANE, "Welfake." https://www.kaggle.com/datasets/saurabhshahane/fake-news-classification.

[21] S. V. SINGH, "Fake news scraped." https://www.kaggle.com/datasets/shashankvikramsingh/fake-news-scraped.

[22] K. C. P. Competition, "Fake news." https://www.kaggle.com/competitions/fake-news/data?select=train.csv.

[23] JRUVIKA, "Fake news detection." https://www.kaggle.com/datasets/jruvika/fake-news-detection.

[24] J. S. Cramer, "The origins of logistic regression," 2002.

[25] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 18, no. 6, pp. 275–285, 2004.

[26] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, pp. 986–996, Springer, 2003.

[27] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[28] G. I. Webb, *Naïve Bayes*, pp. 713–714. Boston, MA: Springer US, 2010.

[29] Vikramkumar, V. B, and Trilochan, "Bayes and naive bayes classifier," *CoRR*, vol. abs/1404.0933, 2014.

[30] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, 2001.

[31] A. McCallum, K. Nigam, *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41–48, Madison, WI, 1998.

[32] B. Liu, E. Blasch, Y. Chen, D. Shen, and G. Chen, "Scalable sentiment classification for big data analysis using naive bayes classifier," in *2013 IEEE international conference on big data*, pp. 99–104, IEEE, 2013.

[33] M. Granik and V. Mesyura, "Fake news detection using naive bayes classifier," in *2017 IEEE first Ukraine conference on electrical and computer engineering (UKRCON)*, pp. 900–903, IEEE, 2017.

[34] G. Singh, B. Kumar, L. Gaur, and A. Tyagi, "Comparison between multinomial and bernoulli naïve bayes for text classification," in *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pp. 593–596, IEEE, 2019.

[35] scikit-learn, "Gaussiannb - scikit-learn 0.24.2 documentation." `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html`. Accessed: May 20, 2023.

[36] scikit-learn, "Categoricalnb - scikit-learn 0.24.2 documentation." `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.CategoricalNB.html`. Accessed: May 20, 2023.

[37] scikit-learn, "Multinomialnb - scikit-learn 0.24.2 documentation." `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html`. Accessed: May 20, 2023.

[38] scikit-learn, "Bernoullinb - scikit-learn 0.24.2 documentation." `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html`. Accessed: May 20, 2023.

[39] A. P. Noto and D. R. S. Saputro, "Classification data mining with Laplacian Smoothing on Naïve Bayes method," *AIP Conference Proceedings*, vol. 2566, 11 2022. 030004.

[40] E. R. Setyaningsih and I. Listiowarni, "Categorization of exam questions based on bloom taxonomy using naïve bayes and laplace smoothing," in *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, pp. 330–333, 2021.

[41] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[42] P. Joshi and P. Kulkarni, "Incremental learning: Areas and methods-a survey," *International Journal of Data Mining & Knowledge Management Process*, vol. 2, no. 5, p. 43, 2012.

[43] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[44] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, "To transfer or not to transfer," in *NIPS 2005 workshop on transfer learning*, vol. 898, 2005.

[45] N. Agarwal, A. Sondhi, K. Chopra, and G. Singh, "Transfer learning: Survey and classification," *Smart Innovations in Communication and Computational Sciences: Proceedings of ICSICCS 2020*, pp. 145–155, 2021.

[46] R. Ade and P. Deshmukh, "Methods for incremental learning: a survey," *International Journal of Data Mining & Knowledge Management Process*, vol. 3, no. 4, p. 119, 2013.

[47] C. Giraud-Carrier, "A note on the utility of incremental learning," *AI Communications*, vol. 13, no. 4, pp. 215–223, 2000.

[48] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias, "Three types of incremental learning," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1185–1197, 2022.

[49] J. Roure, "Incremental learning of tree augmented naive bayes classifiers," vol. 2527, pp. 32–41, 11 2002.

[50] Q. Yang, Y. Gu, and D. Wu, "Survey of incremental learning," in *2019 Chinese Control And Decision Conference (CCDC)*, pp. 399–404, 2019.

[51] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S.-T. Xia, "Maintaining discrimination and fairness in class incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[52] J. C. Schlimmer and R. H. Granger, "Incremental learning from noisy data," *Machine learning*, vol. 1, pp. 317–354, 1986.

[53] C. You, J. Xiang, K. Su, X. Zhang, S. Dong, J. Onofrey, L. Staib, and J. S. Duncan, "Incremental learning meets transfer learning: Application to multi-site prostate mri segmentation," in *Distributed, Collaborative, and Federated Learning, and Affordable AI and Healthcare for Resource Diverse Global Health* (S. Albarqouni, S. Bakas, S. Bano, M. J. Cardoso, B. Khanal, B. Landman, X. Li, C. Qin, I. Rekik, N. Rieke, H. Roth, D. Sheet, and D. Xu, eds.), (Cham), pp. 3–16, Springer Nature Switzerland, 2022.