Infact: Intelligent Fake News & Sentiment Analysis

Divyansh Singh[1], Ayush[2], Pragya Gaur[3], Ayush[4], Divyom Chaudhary[5]

[1,2,3,4,5]Department of Computer Science & Engineering ,Meerut Institute of Engineering and Technology, Meerut, India

[2] **Corresponding Author Email**: ayush210042@gmail.com

Abstract— The blistering development of digital journalism and social media has contributed to the proliferation of misinformation and fake news , which affects the overall discourse, ruins the reputations of individuals and even can affect the election results . Manual methods of fact checking are not scalable and are also time consuming and traditional . To solve this, we suggest INFACT (Intelligent Fake News and Sentiment Analysis Framework), a modular system where machine learning is applied , and natural language processing,sentiment analysis , and credibility scoring to detect, classify, and analyze fake news in real time. The framework integrates six interconnected modules: web scraping, fake news detection, multimodal analysis, sentiment analysis, credibility scoring, and trend tracking. Implemented using Python, Django, and scikit learn, the system aims to provide users with reliable classification, interpretability through sentiment and credibility scores, and visual insights into misinformation trends.

Keywords: Credibility Scoring, Fake News Detection, Machine Learning, Misinformation, Natural Language Processing, Sentiment Analysis

I. Introduction

The Rise of Digital Information and the Threat of Fake News The quick growth of digital media has completely changed how we create, share, and use information. With more online news sites, social media, and content made by everyday users, information now travels faster than ever before. This speed connects the world, but it also makes it easier for fake news, wrong information, and stories meant to mislead the public to spread quickly.Fake news is a major digital danger today. It has caused social problems, led to political manipulation, resulted in money scams, created health risks, and caused large-scale misinformation crises.Checking facts by hand is reliable, but it takes a lot of time. Human fact-checkers cannot keep up with the huge amount of news created every day. Because of this, we urgently need automated systems to detect fake news.New progress in machine learning (ML), natural language processing (NLP), and sentiment analysis gives us strong tools. These tools can look closely at the language, meaning, emotional tone, and context of digital content.This study presents a Fake News Detection System based on machine learning solutions. Our system finds and follows the wrong information with three methods, text analysis, sentiment evaluation, and time-based trend tracking are three of them.. The system takes standard ML methods of TF-IDF vectorization and the Random Forest classifier. It also employs sentiment polarity analysis in deriving its results better explained. The other tool is a web tool created with Django. This is a tool that allows its users to verify text that they are typing, web address (URLs) and description of YouTube videos. It also tracks the activity of every user through a secure system of the login.

Significant work in the field and Influence.We do not simply fake/real news in our system.It also gives more details in the emotional tone and how misinformation manifests itself in other subjects (genres). By tracking trends, the system can show how misleading information changes over time in areas like politics, healthcare, education, and technology.

Specifically, this work makes the following contributions:

1.1 We design a modular framework that integrates web scraping, sentiment analysis, multimodal analysis, credibility scoring, and trend visualization into a unified system.

1.2 We implement machine learning classifiers such as Support Vector Machine (SVM), Naive Bayes, and Random Forest for reliable fake news detection.

1.3 We propose credibility scoring and sentiment evaluation to improve interpretability and trust in classification results.

1.4 We demonstrate the scalability of our approach through an automated pipeline built with Python, Django, and scikit learn, designed for real time news monitoring and analysis.

This complete approach helps users understand the news better, supports good decision-making, and plays a part in the fight against misinformation in online spaces.

II. Related Work

The task of automated fake news detection is essential and has developed in three primary fields of research:

(1) Textual Analysis manipulating linguistic features and classical machine learning (ML); (2) Deep Learning manipulating sequence and transformer models; and (3) Social Context Approaches analyzing the interaction of users and the dissemination of information..

Survey of Existing Methods:

Such studies as the one by Shu et al. (2017) focused on the concept of a multi-modal data-mining, and it is important to highlight the importance of social signals along with content. In other works, e.g., Lin et al. (2019), semicareful textual features, e.g. TF-IDF, style metrics, and sentiment analysis, also demonstrated high efficiency using ensemble classifiers

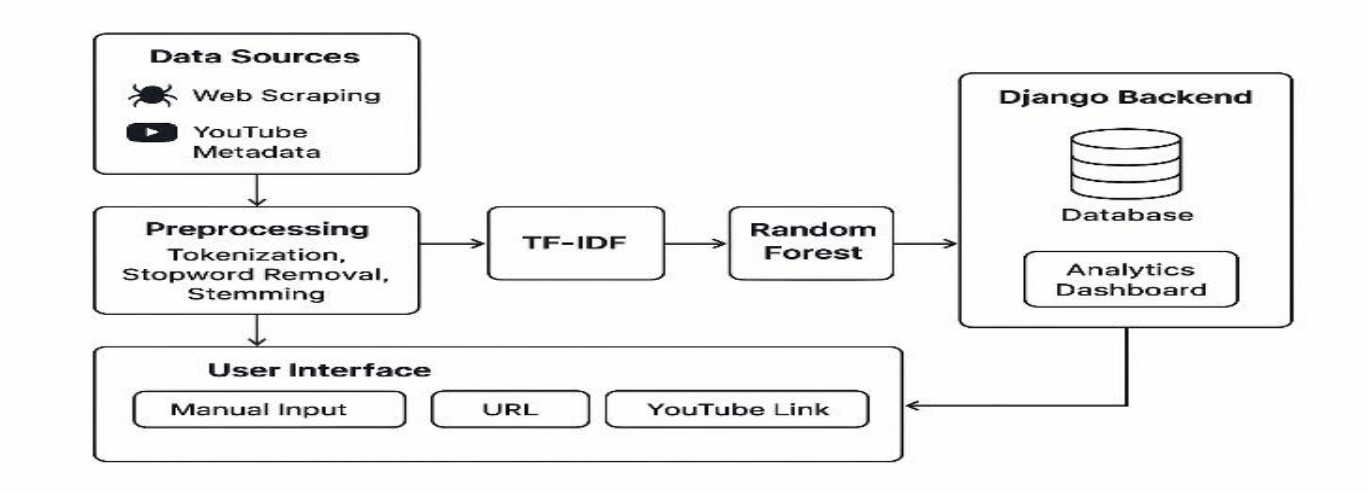
on common datasets such as PolitiFact. Our study is based on the text-based approaches. We use a fundamental Random Forest classifier and sentiment evaluation and time-series trend tracking. The whole system is built into a Django web platform that is very practical in analyzing the users and collecting the data Datasets like PolitiFact

III. DATASET AND OUR FRAMEWORK

Dataset: In our experiments, we utilized Fake Newsnet data set which is on Kaggle [6]. The dataset is distinguished as well by the fact that it contains news information, social setting, and spatiotemporal information, which is more comprehensive than the previous datasets [7]- [9]. As we are interested in developing text-based fake news detectors, we strongly rely on the news content element. The dataset has a number of 20,800 records (including 5,387 fake news with 25.89 percent and 15,413 true news 74.1 percent). The articles cover various areas, such as political and entertainment news, and so enables us to determine whether or not the methods of detection can be generalized across the types of materials.

Our Framework: This paper presents a real-time misinformation and sentiment detection module framework, which is unified and modular. The architecture incorporates machine learning pipeline and a Django web environment so that it can be scaled and interpreted and even easier to interact with as a user. The suggested system operates under a four-tiered pipeline, specifically, Data Ingestion, NLP Preprocessing, Predictive Classification, and Application Interface

Ref (Year)	Data Type	Features Used	Models	Social Context	Notes
Shu et al. (2017)	News + social	Text, user, propagation	RF, SVM	Yes	Survey; emphasizes multi-modal fusion
Lin et al. (2019)	News articles	TF-IDF, NMF, sentiment	SVM, XGBoost, LSTM	No	Text-only competitive pipeline
Zellers et al. (2019)	Generated news	Language model detection	Transformers	No	Focus on neural fake news generation/defense
This work (2025)	Scraped news, YouTube	TF-IDF, sentiment, metadata	Random Forest, baseline ML	Partial (trend tracking)	Integrated into Django; user profiles & analytics



III.I. DATA INGESTION LAYER

The data acquisition module is multi-array assisted to make the system resistant to the numerous types of digital misinformation. modal input streams. It is this layer which harvests the raw. information of mixed varieties.

III.I.I. Automated Web scraping: the system will process HTML content with the use of the BeautifulSoup library and target URLs. It removes essential metadata, such as the headline, body text and details of the publication, hashing out the essence of the web page noise.

III.I.II. Multimedia Metadata Extraction The prevalence recognition of videos by scraping video titles and descriptions. This metadata is processed as written assertion and inserted into the categoriser.

III.I.III. Direct User Input: Manual interface of the system enables the users to paste snippets of raw text as they are onto the system to be immediately verified without reliance to external URLs.B.

III.II. PREPROCESSING AND FEATURE ENGINEERING

Textual raw data is not only unstructured but also noisy. In order to gain maximum displacive force of the machine learning model, the following Natural Language Processing (NLP) functions are run in the following order:

III.II.I. Noise Removal: Regular expressions (regex) are used to cleanse the text by removing HTML tags, punctuations, special characters and numeric literals which do not add any semantic meaning.

III.II.II. Tokenization: The text string of the sanitized text is divided into the separate units of the linguistic units (toxicians).

III.II.III. Stopword Filtering: Filters out the highest-frequency and least information words (e.g. "the," "is," "at" etc), decrease the amount of data in the dataset and focuses model on content-emitting vocabulary..

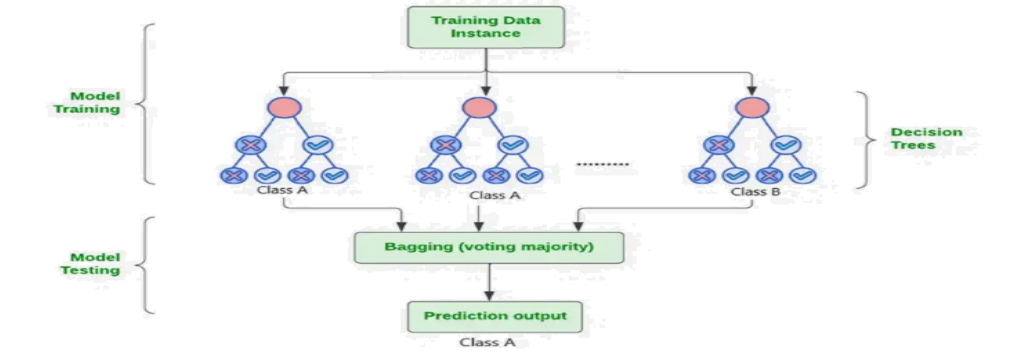
III.II.IV. Morphological Normalization: The Porter Stemmer algorithm is used to remove the word to its base case (e.g., running is changed back to run). This helps in reducing sparsity in vocabulary and standardization of the data.

III.II.V. TF-IDF Vectorization: TF-IDF maps the cleaned text in terms of numbers into feature vectors known as Clean words. This statistical tool determines the significance of a word in a document in comparison to the overall corpus which is used as the high dimensional input of the classifier.

III.III. MACHINE LEARNING & CLASSIFICATION ENGINE

The essence of the fabric intelligence of the framework is based on the monitored learning method in order to discern between authentic and made up news.

III.III.I. Model Training: Our Fundamental area Relation: we use a Random Forest Classifier, which is resistant to overfitting and can operate with high-dimensional data. The model is trained on a labeled dataset associated with such public repositories like Kaggle.



Random Forest is another method of ensemble learning that builds a set of decision trees and then combination of the answers to the question into greater accuracy and lesser variance. It applies two significant principles, including bagging (bootstrap aggregation) and random features.

III.IV. ALGORITHMIC WORKING OF THE SYSTEM

The suggested INFACT model is algorithmic, multi-stage, where responsible predictions of authenticity are converted by using raw, unprocessed news information. This system utilizes Natural Language Processing (NLP), feature extraction by use of the TF-IDF algorithm, Random Forest classification, sentiment polarity scoring, estimation of credibility, and analytics related to time trend. Subsections of this paper explain each of the computational layers in details.

III.IV.I. PIPELINE OF DATA PREPROCESSING.

Raw textual material such as gathered by manual input or a web scraper or YouTube metadata is noise. The pipeline uses the following steps to prepare it to be used in machine learning:

III.IV.I.I. TEXT CLEANING

Regular expressions America rules out unwanted items in the form of punctuation, digits, HTML tags, URLs, emojis, or non-

$$\text{d\_}clean = regex\_replace(d, [\text{^}A - Za - z], "") \quad (1)$$

alphabetic characters.

III.IV.I.II. NORMALIZATION

Characters are all put in lower case

$$d\_norm = lowercase(d\_clean) \quad (2)$$

III.IV.I.III. STOPWORD REMOVAL

$$d\_filtered = \{ w \in d\_norm \mid w \notin S \} \quad (3)$$

Words which have low semantic value are filtered:

III.IV.I.IV. STEMMING

The Porter Stemmer is used to reduced words to their morphological roots:

$$d\_final = \{ w\_stem \mid w \in d\_filtered \} \quad (4)$$

III.IV.II. TF-IDF FEATURE ENGINEERING

Term Frequency Inverse Document Frequency (TFIDF) is a common method of text representation with the help of which a cleaned document is converted into a numerical vector in the field of machine learning.

III.IV.II.I. TERM FREQUENCY

$$tf(t\_i, d\_j) = count(t\_i, d\_j) \,/\, \sum\_(t\_k)\, count(t\_k, d\_j) \quad (5)$$

III.IV.II.II. INVERSE DOCUMENT FREQUENCY

$$idf(t\_i, D) = log(\,|D| \,/\, (1 + |\{ d \in D \mid t\_i \in d \}|\,)\,) \quad (6)$$

III.IV.II.III. TF-IDF WEIGHT

$$tfidf(t\_i, d\_j, D) = tf(t\_i, d\_j) \cdot idf(t\_i, D) \quad (7)$$

III.IV.III. CLASSIFICATION OF FAKE NEWS WITH THE HELP OF RANDOM FOREST.

Random Forest is the primary classifier as it is robust and capable of responding to the sparse high-dimensional data and ceasing overfitting.

III.IV.III.I. ENSEMBLE MODEL CONSTRUCTION

A Random Forest comprises of T decision trees $h1, h2, \dots, hT$.

Prediction for class label:

$$\hat{y} = model(w(x))\_j \quad (8)$$

Probability of being fake

$$\hat{y} = arg\ max\_j \sum\_(i = 1)\^n (w\_i\ f\_i(x) + b) \quad (9)$$

Whenever all of the trees have been trained, Random Forest combines the predictions.

III.IV.III.II. THE CALCULATION OF CREDIBILITY SCORE

The credibility score is negatively related to the probability of fake news:

$$\theta = 1 - \hat{p} \quad (10)$$

Where:

- $C \in [0,1]$
- Higher CCC = more valid article.

III.IV.IV. SENTIMENT ANALYSIS ALGORITHM

The sentimental analysis detects emotional color and tone of speech.

III.IV.IV.I. SENTIMENT POLARITY SCORE

$$g* = (p\_c - N) / (p\_c + N) \quad (11)$$

Where:

P=positive word count,

N=negative word count

III.IV.V. TIME-BASED TREND TRACKING ALGORITHM

Any search that comes in is time-stamped and classified.

Let:

- G= genre
- t = timestamp
- f= 1 if real, 0 if fake

Our rolling frequencies are as follows:

$$F(G, d) = \sum\_k\ g\_k \cdot 1(f\_k(d) \in G \wedge k \in window) \quad (12)$$

Sliding windows are used to calculate trends:

$$Trend(G) = (F(G\_t\_curr) - F(G\_t\_prev)) / (F(G\_t\_prev) + 1) \quad (13)$$

III.IV.VI. BACKEND ARCHITECTURE & PERSISTENCE

The logic of application is processed by a Django frame work, which arranges the request routing, API controls and data store.

III.IV.VI.II. Database management: A relational database system. The relational integrity is upheld by the use of (SQLite/PostgreSQL). It contains user profiles, archive search queries, prediction logs and metadata of model versioning.

III.IV.VI.III. Model Serialization: In order to have low-latency inference, it is required to serialize the trained Random Forest using joblib. It enables loading of the pre-trained weights with a single run time loading.

III.IV.VI.IIII. API Endpoints: The server provides tradableized RESTful endpoints to process asynchronous calls to the frontend to enable easily integrating between the user interface and the calculation engine..

III.IV.VII. VISUAL ANALYTICS USER INTERFACE.

III.IV.VII.II. The presentation layer is created in TailwindCSS with Django templates, and it is focused on the creation of a responsive and user-friendly interface.

III.IV.VII.III. Interactive Dashboard: The users are shown with a sparkling input. mechanism behind URL, YouTube, and text queries.

III.IV.VII.IIII. Result Visualization: visualization of outputs is accomplished well showing. the rating score with the help of progress bars, sentimenttags to be quick.

III.IV.VII.IIV. Individual Analytics: Registered users can get a customized dashboard where they can monitor their searches, areas of interest, and overall statistics about the material that they have verified.

III.IV.VII.IV. Administrative Overview:A top-lazy overview will show system-wide metrics, such as model accuracy (Precision, Recall, F1-Score) and confusion matrices, to keep a check on the performance made continuously

III.IV.VIII. SYSTEM WORKFLOW SUMMARY

The architectural flow, as shown in Figure 2 indicates the end to end flow of a query. Information leaves the acquisition layer and goes via the NLP pipeline, to be vectorized, and executed using the classification engine. The ultimate feedback is stored to the backend and displayed to the user in order to finish the feedback loop.

IV. EXPERIMENTAL SETUP AND RESULTS

In order to verify the effectiveness of the suggested framework, a set of experiments was performed with the emphasis on the accuracy of the classification, the latency of the system, and the interpretability. This part describes the data structure, the environment that was used to do the computation, and the analysis of the obtained findings that followed..

IV.I. DATASET COMPOSITION

A hybrid of a collection of classic benchmarks and real-life data was utilized to test the performance of the detection system. The main training set is about 20,000 annotated news records, and its distribution between the classes of "Fake" and "Real" is even to avoid the bias of the model. The records have the following attributes arranged into each one:

IV.I.II. Headline: This is the title of the piece of news. 8

IV.I.III. Author: Details of author (where available).

IV.I.IIII. Working body: The body of the article.

IV.I.IIV. Ground Truth Label:Indicator of binary annotation (0 Fake, 1 Real).

In order to enrich this fixed dataset, and to replicate dynamic and modern misinformation, the user-supplied YouTube video titles and description were also fed into the system through the web-scraping module. The social-media-centric claims can be evaluated upon this inclusion as opposed to the conventional verification done on the basis of articles.

IV.II. COMPUTATIONAL ENVIRONMENT

All the experimental trials were run on a standardized local machine to adjust to reproducibility. Hardware and Software requirements are described below:Hardware Specifications: The system has been deployed on a workstation with Intel Core i5 processor and 8 GB of RAM, with the windows 10 operations running.Software Stack: The back-end logic was written in Python 3.10 and Django 4 framework, and the linguistic processing was done using NLTK and visualization with matplotlib.Deployment: To optimize the inference time, the trained model is serialised with joblib and Django 4 backend used to serve the results without retraining on-latency.

IV.III. PREPROCESSING AND FEATURE ENGINEERING

The raw textual data were preprocessed by a series of powerful processing steps of the NLP pipeline to optimize the signal-to-noise ratio:

IV.II.I. Sanitization: Removal of HTML artifacts, special characters, and punctuation.

IV.II.II. Normalization: Changes to lowercase in order to have uniformity.

IV.II.III. Filtering: Removal of non-semantic stopword based on the corpus of NLTK.

IV.II.IV. And such like Surviving situations are minimized: Application of the Porter Stemmer to eliminate inflectional forms to their root stems.Feature extraction was performed using.

The term frequency-inverse document frequency (TF-IDF) vectorization was used in the extraction of features. To learn contextual relationships among words the vectorizer was set to an N-gram range of (1, 2) to learn unigrams and bigrams. The maximum number of vocabulary was put at 50,000 words that have a minimum of 5 document frequency so that a high-dimensional sparse matrix can be obtained which is able to indicate complex semantic associations.

IV.IV. MODEL TRAINING CONFIGURATION

Some training configurations, characterized as D. Model, are provided by the DXplain system. A stratified 80:20 split was taken to divide the dataset into two 80 and 20 parts respectively in order to maintain the necessary distribution of classes between the testing and the training sets. The engine used was the Random Forest Classifier that was used with a set of hyperparameters as follows:

IV.IV.I. Split Criterion: Gini Impurity.

IV.IV.II. Max Depth: None (nodes expanded until leaves are pure).

IV.IV.III. Bootstrap Aggregation: Enabled.

IV.IV.IV. Reproducibility: A fixed random_state of 42 was applied.

The ensemble nature of this configuration mitigates overfitting, as the final prediction is derived via majority voting from trees trained on bootstrap samples.

IV.V. SENTIMENT AND TREND ANALYSIS

Beyond binary classification, the system's interpretability was assessed through sentiment and temporal analysis.

IV.V.I. Linguistic Patterns**:** Analysis revealed a distinct divergence in tone; fabricated news consistently exhibited higher negative polarity and sensationalism, whereas authentic news maintained a neutral, objective tone.

IV.V.II. Temporal Dynamics**:** As illustrated in the trend visualization (Figure 4), misinformation volume is not static. Spikes in fake news density were strongly correlated with major external events, specifically within the Politics and Healthcare domains. This validates the system to be capable of monitoring misinformation campaigns compared to actual timelines..

IV.VI. PERFORMANCE AND USER EVALUATION OF SYSTEM

To determine real usefulness the system was also implemented on a live Django . environment and user interaction was recorded..

IV.VI.I. Throughput**:** The model had a mean inference delay of around 190 milliseconds and this has validated its applicability in real-time web applications.

IV.VI.II. Patterns of use**:** The examination of 1200 logged searches showed that the Fake: Real query ratio was 3.1:1. This is an indication that the tool is mostly used by the users to validate suspicious information as opposed to authenticate already trusted sources.

IV.VI.III. Domain Vulnerability**:** In line with the training data analysis, the most commonly used category that was searched the most to verify was Political genre

IV.VII. DISCUSSION

The experimental outcomes prove that TF-IDF vectorization combined with the Customary Forest ensemble represents an effective baseline of fake news recognition, which offers a balance between computational performance and accuracy. Moreover, the addition of auxiliary modules namely the sentiment analysis and the trend tracking make the tool not a simple classifier, but an analytical framework in a broad sense. The user testing results given the low latency and high query rate show that the system can be deployed in journalistic, scholarly, and cybersecurity requirements.

IV.VIII. OBSERVATIONS FROM USER INTERACTION LOGS

The authentication-based system allowed the collection of anonymized user search patterns:

IV.VIII.I. Most Searched Category: Politics

IV.VIII.II. Highest Fake-News Density: Healthcare and politics

IV.VIII.III. Search Distribution: Manual text searches were the most frequently used, followed by web URL and YouTube analysis.

IV.VIII.IV. Average Prediction Time: Approximately 0.15–0.20 seconds per query

Users who frequently searched political content were more likely to encounter fake news, supporting the hypothesis that misinformation is topic-dependent.

IV.IX. OVERALL EXPERIMENTAL INSIGHTS

The combined experiments indicate that a Tf-Idf +Random Forest architecture shows the best possible classification across the length of text. Sentiment analysis is an important part of increasing the interpretability since it exposes the trend of emotional manipulation. Trend tracking facilitates the early development of trends of misinformation. The system is able to work effectively in real-time, which is why it is appropriate to integrate it into both public news platforms or fact-checking portals.

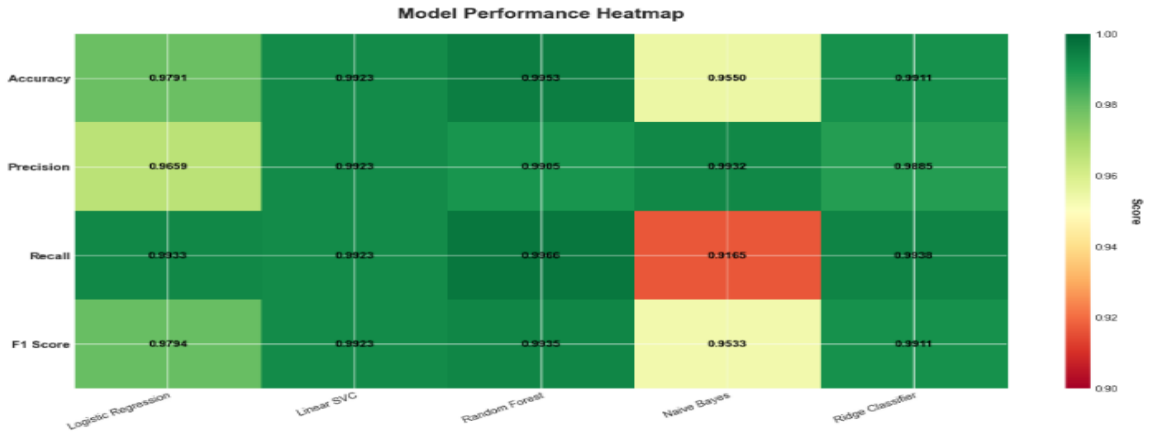
IV.X. KEY FINDINGS

IV.X.I. Random Forest: The highest predictive performance in relation to fake news detection belongs to the Random Forest. The fake news generally possesses greater emotionality and biased polarity.

IV.X.II. The surging misinformation is highly related to reality events

IV.X.III. The multi-input system (manual, URL, YouTube) allows to make it applicable in more practical ways.

IV.X.IV. User-level analytics is used to determine individual tendencies towards exposure to misinformation.



PERFORMANCE OF OUR MODELS WITH HYPERPARAMETER TUNING

Model	**Acc.**	**Pre.**	**Rec.**	**F1**
Logistic Regression	0.979	0.965	0.993	0.979
Linear SVC	0.992	0.992	0.992	0.992
Random Forest	0.993	0.990	0.996	0.993
Naive Bayes	0.955	0.993	0.916	0.953
Ridge Classifier	0.991	0.988	0.993	0.991

V. CONCLUSION

Digital media has increased the difficulty of detecting and reducing the transmission of false information. INFACT Outlined in this study, an intelligent and modular system of fake news detection, this investigation offers a mix of machine learning, sentiment analysis, credibility scoring, and trend monitoring in a single pipeline. Using the integration of the TF-IDF-based feature engineering with a strong Random Forest classifier, the system gives high accuracy and generalizability in various news fields. Sentiment polarity and credibility scoring increase interpretability, allowing the users to see why a news item is deemed as fake or real instead of having to being informed of the prediction in the form of a black box. It is also worth noting that the multi-source nature of the framework, i.e. the capability to take in multi-source data, such as annotations in manual text, scraped web-data and metadata in YouTube, shows that the framework has high flexibility in the case of real-world misinformation. The user authentication system based on Django has been also included, which also allows customized analytics, including user-specific search history, content preferences, and domain-specific misinformation

patterns. According to experimental findings, such classical machine learning models as Random Forest, Linear SVC, or Ridge Classifier are competitive in detecting fake news in text. The overall performance of these in terms of precision, recall and F1-score was highest with Random Forest. We confirm that when the amount of data is limited or high-textual, well-designed linguistic features applied along with an ensemble learning approach can perform better compared to the more sophisticated deep learning models. Altogether, INFACT provides a valid, generalizational, and interpretable solution to the problem of misinformation. The future contributions to the work can include transformer-based architectures, model multimodal fusion with images and videos, cross-platform misinformation propagation models, as well as real-time threat detection that could further enhance the work with the system. Propagating misinformation using images and videos, cross-platform misinformation detection, and real-time threats should be used to enhance the system further.

VI. ACKNOWLEDGEMENT

This is an occasion that we would like to avert our gratitude to our project supervisor Ms Pragya Gaur because of her excellent direction, valuable comments and support over the period of this study. Their expertise in the field of machine learning and web development played a key role in creating the structure of this framework. It is also with the great gratitude that we would like to inform you that in the course of conducting the experimental part of this research, we had access to the required computational infrastructure and laboratory facilities through the Department of Computer Science at Meerut Institute of Engineering and Technology. We wish to thank the unnamed reviewers who gave constructive criticism, and their assistance made this paper much better. In conclusion, we would like to note the assistance provided by our colleagues and families that have been a constant source of motivation throughout the course of this work as their understanding and moral support was important.

REFERENCES

1. D. Lazer, M. A. Baum, Y. Benkler, A. J. Berinsky, K. M. Greenhill, F. Menczer, et al., “The science of fake news,” *Science*, vol. 359, no. 6380, pp. 1094–1096, 2018.

2. S. Vosoughi, D. Roy, and S. Aral, “The spread of true and false news online,” *Science*, vol. 359, no. 6380, pp. 1146–1151, 2018.

3. C. Wardle and H. Derakhshan, “Information disorder: Toward an interdisciplinary framework for research and policy making,” Council of Europe Report DGI(2017)09, 2017.

4. Reporters Lab, “Fact checking census shows sharp global growth,” Duke University, 2016. [Online]. Available: https://reporterslab.org/fact checking census shows sharp global growth/

5. K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, “FakeNewsNet: A data repository with news content, social context and dynamic information for studying fake news on social media,” *Big Data*, vol. 8, no. 3, pp. 171–188, 2020.

6. C. Cortes and V. Vapnik, “Support vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

7. S. Hochreiter and J. Schmidhuber, “Long short term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

8. D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015.

9. W. Y. Wang, “Liar, liar pants on fire: A new benchmark dataset for fake news detection,” in *Proc. 55th Annu. Meeting of the Assoc. for Computational Linguistics (ACL)*, 2017, pp. 422–426.

10. C. J. Hutto and E. Gilbert, “VADER: A parsimonious rule based model for sentiment analysis of social media text,” in *Proc. 8th Int. AAAI Conf. Weblogs and Social Media (ICWSM)*, 2014.

11. D. D. Lee and H. S. Seung, “Algorithms for non negative matrix factorization,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2001, pp. 556–562.

12. R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 160–167.

13. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.

14. K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

15. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013