Fake Profile Detection Using Machine Learning

A Project Report submitted in the partial fulfillment of

the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Ravi Lakshmi Sri Harsha (21471A05B7)

Shaik Najeer (21471A05C3)

Sistla V. S. Manikanta Rohit (21471A05C7)

Under the esteemed guidance of

Suneetha Mothe, B. Tech, M. Tech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under Tyre -1 NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601

2024-2025

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name "Fake Profile Detection Using Machine Learning" is a bonafide work done by the team Ravi Lakshmi Sri Harsha (21471A05B7), **Shaik Najeer** (21471A05C3), **Sistla V. S. Manikanta Rohit** (21471A05C7) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE

Suneetha Mothe, B.Tech, M. Tech.

Assistant Professor

PROJECT CO-ORDINATOR

Dr. Sireesha Moturi, B.Tech, M.Tech, Ph.D.

Associate Professor

HEAD OF THE DEPARTMENT

Dr. S. N. Tirumala Rao, B. Tech, M.Tech., Ph.D.,

Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled **Fake Profile Detection Using Machine Learning** is composed by ourselves that the work contained here is our own except where explicitly stated otherwise in the text and that this work has not been submitted for any other degree or professional qualification except as specified.

Ravi Lakshmi Sri Harsha (21471A05B7)
Shaik Najeer (21471A05C3)
S. V. S. Manikanta Rohit (21471A05C7)

ACKNOWLEDGEMENT

We wish to express our thanks to various personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman Sri **M. V.Koteswara Rao**, **B.Sc.**, who took keen interest in us in every effort throughout this course.We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, Ph.D.**, for showing his kind attention and valuable guidance throughout the course.

We express our deep felt gratitude towards **Dr. S. N. Tirumala Rao,** B.Tech**, M.Tech., Ph.D.**, HOD of CSE department and also to our guide **M.Suneetha, B.Tech, M.Tech.**of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dr. Sireesha Moturi, B.Tech, M.Tech.,Ph.D.,** Associate professor & Project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant sourceof inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

Ravi Lakshmi Sri Harsha (21471A05B7)

Shaik Najeer (21471A05C3)

S. V. S. Manikanta Rohit (21471A05C7)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community,

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertize in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S)

CO421.1: Analyse the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.**CO421.3:** Review the Related Literature

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		✓											✓		
C421.2	✓		✓		✓								✓		
C421.3				✓		✓	✓	✓					✓		
C421.4			✓			✓	✓	✓					✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, planning to develop a model for recognizing fake profiles using machine learning techniques.	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement is critically analyzed, the process model is identified	PO2, PO3
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done, and the project will be handled by social media users. Future updates in our project can be done based on detection of forged videos.	PO4, PO7
C32SC4.3	The physical design includes a website to check whether an image is real.	PO5, PO6

ABSTRACT

In today's world, social media is an essential part of communication, business, and networking. However, the increasing presence of fake profiles poses significant risks, including misinformation, fraud, and security threats. This study proposes a machine learning-based approach to detect fake using Random Forest, XGBoost, Logistic Regression, and Gaussian Naïve Bayes classifiers. The dataset, sourced from Kaggle, is preprocessed using techniques such as handling missing values, outlier detection, and class balancing with SMOTE. Experimental results show that Random Forest achieved the highest accuracy of 100%, outperforming other models. The proposed system enhances cybersecurity by providing an efficient and accurate method for detecting fake profiles on social media platforms. This research not only strengthens online security but also aids in reducing fraudulent activities and protecting genuine users. Future improvements include real-time social media data integration, behavioral analysis, and NLP-based fake bio detection to further refine the model's performance. The final model was integrated into a user-friendly web application, enabling users to input profile details and receive real-time fake profile detection results. The system enhances online safety by reducing fraudulent activities and minimizing the risk of cyber threats. By leveraging AI-driven detection, social media platforms can automate the identification of suspicious accounts.

INDEX

S.NO	LIST OF FIGURES	PAGENO

1. INTRODUCTION

In the modern digital era, social media platforms such as Facebook, Instagram, Twitter, and LinkedIn have become integral to communication, business, education, and entertainment. While these platforms provide numerous opportunities for engagement and networking, they also present challenges, particularly the proliferation of fake profiles. Fake accounts are often used for spreading misinformation, identity theft, cyberbullying, and fraudulent activities. Detecting and preventing such profiles is crucial to maintaining a secure and trustworthy digital environment.

To build a robust dataset, information is gathered from multiple online sources, including Kaggle datasets, social media APIs, and publicly available repositories containing labeled real and fake profiles. The dataset encompasses diverse attributes, such as account details (username, bio, profile picture presence, and account age), user activity metrics (post frequency, likes, shares, and comments), and network behavior (follower count, following relationships, and mutual connections). Additionally, content-based features, including sentiment analysis, spam detection, and linguistic pattern recognition, play a crucial role in distinguishing authentic users from fraudulent ones. By integrating these multi-faceted data points, the system enhances its ability to detect fake profiles with higher precision.

Once the dataset is compiled, it undergoes a thorough preprocessing phase to improve consistency and enhance the accuracy of machine learning models. This process includes handling missing values through statistical imputation or removal of irrelevant data points. Categorical variables, such as account types and engagement levels, are transformed into numerical representations to ensure seamless processing by machine learning algorithms. Outlier detection techniques are applied to identify and eliminate anomalies that could skew predictions.

The detection of fake profiles is a growing concern in the cyber security and social media domains, leading to extensive research in developing effective methods for identifying and eliminating fraudulent accounts. Traditional approaches, such as manual verification and rule-based filtering, have proven to be inefficient due to the vast number of profiles created daily. Machine learning has emerged as a powerful tool for automating the detection process, utilizing algorithms that analyze user behavior,

interaction patterns, and profile attributes. This study focuses on machine learning techniquesIncluding Random Forest, Logistic Regression, Naïve Bayes, and XGBoost, to classify and distinguish fake profiles from genuine ones. By processing and analyzing large datasets, the proposed system aims to enhance detection accuracy and reduce false positives, ensuring more reliable identification of fraudulent activities.

2. LITERATURE SURVEY

Despite its contributions, the study has some key gaps. Firstly, it is primarily focused on Twitter, which may limit the generalizability of the findings to other social media platforms with different user behaviors and profile structures [1]. Secondly, the research does not extensively address the implementation of these machine learning models in real-time scenarios, which is crucial for timely detection and mitigation of fake profiles. Lastly, advanced evasion strategies employed by sophisticated fake profiles, such as mimicking genuine user behavior, are not thoroughly explored in the study.This research highlights effective machine learning approaches for fake profile detection while also indicating areas that require further investigation to enhance the robustness and applicability of these methods across diverse social media environments.her advancements in scalability and adaptability.

Goyal et al. [2023] investigate the application of machine learning techniques for detecting fake profiles in online social networks [OSNs]. The rise of social media has led to an increase in fake profiles, which pose risks related to security, misinformation, and online fraud. The study evaluates various classification algorithms, such as Random Forest and Support Vector Machines [SVM], to identify fake accounts based on features like follower count, profile activity, and interaction patterns. Their findings indicate that machine learning models significantly outperform rule-based methods in detecting fake users, achieving high accuracy, precision, and recall. However, the study highlights some limitations, such as challenges in generalizing results across different social media platforms, the evolving sophistication of fake profiles, and the lack of deep learning techniques in the analysis. Additionally, the computational feasibility of real-time detection remains an area for future research. The study underscores the importance of machine learning in improving social media security while emphasizing the need for Harish, Kumar, and Bell [2023] conducted a study titled Fake Profile Detection Using Machine Learning, published in the International Journal of Scientific Research in Science, Engineering, and Technology. The research focuses on identifying fake profiles on social media platforms, particularly Twitter, by employing various machine learning algorithms. The study emphasizes the importance of selecting relevant features such as follower count, friend count, and status updates to effectively distinguish between

genuine and fake profiles. Among the machine learning techniques evaluated, models like Neural Networks, Long Short-Term Memory [LSTM], XGBoost, and Random Forest demonstrated high accuracy in classifying profiles. The researchers utilized a Twitter profile dataset, categorizing genuine accounts as TFP and E13, and fake accounts as INT, TWT, and FSF, to train and test the models.

The study explores the detection of fake profiles on Twitter using a Hybrid Support Vector Machine [SVM] algorithm. Published in E3S Web of Conferences, the research addresses the increasing threat posed by fake accounts used for misinformation, cyber fraud, and online manipulation. A hybrid approach combining SVM with feature selection techniques is proposed to enhance classification accuracy. Various user attributes, including tweet frequency, follower-following ratio, profile completeness, and engagement metrics, are analyzed to differentiate real and fake accounts. The results indicate a significant improvement in precision, recall, and F1-score compared to traditional machine learning models. However, challenges such as real-time detection, evolving spamming techniques, and scalability issues in large datasets remain. The study highlights the importance of hybrid models in social media security and suggests that integrating deep learning and adaptive algorithms could further enhance detection accuracy in dynamic online environments.

The study investigates the use of Artificial Neural Networks [ANNs] for identifying fake profiles on social media platforms. Presented at the 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT), the research focuses on leveraging deep learning techniques to enhance fake profile detection accuracy. By analyzing key user attributes such as profile activity, engagement patterns, and network behavior, the ANN model effectively differentiates between real and fake accounts. Experimental results demonstrate that ANN-based models outperform traditional machine learning approaches in terms of precision and recall, highlighting the effectiveness of deep learning in detecting fraudulent accounts. However, the study also identifies challenges, including the computational complexity of ANN models, the need for large datasets for training, and difficulties in adapting to evolving deceptive tactics. The findings emphasize the potential of neural networks in enhancing social media security and suggest further research into real-time implementation and hybrid models to improve detection capabilities

3. Existing System

Manual Detection by Human Moderators

Traditional fake profile detection relies on human moderators analyzing user profiles, interactions, and activity logs to identify suspicious accounts. This process is time- consuming, prone to human error, and highly dependent on experience

Conventional Machine Learning Approaches

Early AI-based methods used traditional machine learning algorithms (e.g., support vector machines, decision trees and Naïve Bayes classifiers) to detect fake profiles These models relied heavily on handcrafted feature extraction, which limited their ability to generalize across different datasets and evolving fake profile tactics.

Lack of Early Detection Tools

Many existing systems are focused on identifying fake profiles after they have already caused harm, such as spreading misinformation or conducting fraudulent activities. There is a lack of real-time detection tools that can prevent fake profiles from being created or spreading deceptive content.

Data Imbalance Issues

Traditional systems struggle with imbalanced datasets where the number of fake profiles is significantly smaller compared to genuine. This leads to biased predictions, reducing sensitivity to fake accounts and making detection less effective.

Feature Engineering Techniques

Basic feature extraction methods, such as analyzing profile completeness Friend networks, and activity patterns, have been used to detect fake profiles. However, dynamic and evolving behaviors of fake accounts make these traditional techniques less effective.

4. Proposed Methodology

The proposed system aims to detect fake profiles on social media platforms by leveraging advanced machine learning techniques. With the increasing usage of social media for communication, business, and networking, fraudulent activities have also risen, making fake profile detection a critical challenge. The proposed methodology ensures a comprehensive approach by analyzing user behavior, profile characteristics, and interaction patterns. Through a combination of data collection, preprocessing, machine learning model training, and real-time deployment, this system ensures accuracy and efficiency in identifying fraudulent accounts.

To effectively identify fake profiles, data is gathered from multiple sources, including Kaggle datasets, social media APIs, and publicly available datasets that provide labeled real and fake accounts. The dataset includes various attributes such as username, bio information, profile picture availability, and account age to help assess profile authenticity. Additionally, user engagement metrics, including posts, likes, shares, and comments, contribute to behavioral analysis. Network-related factors, such as follower-following relationships and mutual connections, offer further insights. Moreover, content-based features, including sentiment analysis, spam detection, and linguistic patterns, are leveraged to distinguish between real and fraudulent accounts.

Once the data is collected, it undergoes preprocessing to enhance consistency and accuracy. This includes handling missing values by imputing them using statistical methods or removing insignificant data points. Categorical variables, such as account types and engagement levels, are converted into numerical representations for machine learning models to process efficiently. Moreover, outlier detection techniques are applied to identify and manage anomalies that may distort predictions. The data is then normalized to ensure uniformity across different variables, allowing the model to process information effectively without being biased by scale differences.

Various machine learning models are developed and tested to determine the most effective approach for fake profile detection. The system employs algorithms such as Random Forest, which is known for its robust ensemble learning approach that enhances classification accuracy. XGBoost, a powerful gradient boosting technique, is

implemented to optimize model performance by minimizing errors. Additionally, probabilistic models like Naive Bayes are used for text-based analysis, while Logistic Regression provides a statistical approach to binary classification. Support Vector Machines (SVM) are also considered for their ability to create clear decision boundaries between real and fake profiles. For more complex patterns, deep learning techniques such as Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs) are explored to further improve detection accuracy [2].

The training process involves splitting the dataset into training and testing sets, ensuring the model is well-trained before deployment. Hyperparameter tuning techniques, such as grid search and random search, are applied to optimize the learning process and enhance model accuracy. Cross-validation ensures that the model generalizes well to unseen data, reducing overfitting and improving its robustness. Performance evaluation metrics such as accuracy, precision, recall, F1-score, and the ROC-AUC curve are used to assess each algorithm's effectiveness, allowing the selection of the best-performing model.

Once an optimal model is identified, it is deployed as a real-time detection system that can be accessed via a web-based API or social media platforms. Using frameworks like Flask or FastAPI, the model is converted into a web service, enabling seamless interaction with users. Cloud platforms such as AWS and Google Cloud ensure scalability, allowing the system to process vast amounts of data efficiently. Additionally, an interactive dashboard is developed to visualize detection results, highlight trends in fraudulent activities, and provide actionable insights for users and platform administrators.

The proposed system offers several advantages, making it an effective tool for fake profile detection. By leveraging state-of-the-art machine learning techniques, it achieves high accuracy in identifying fraudulent accounts [3]. Automation eliminates the need for manual verification, reducing the workload on human moderators. The system’s real-time detection capabilities ensure that fake profiles are identified as soon as they appear, preventing potential fraud and misinformation from spreading. Its scalability allows it to handle large datasets, making it suitable for social media platforms with millions of users. Additionally, by curbing the spread of fake profiles,

the system enhances digital security and fosters a safer online environment for users

In this machine learning-based fake profile detection system provides a reliable and efficient solution for identifying fraudulent accounts on social media. By incorporating various algorithms, optimizing hyperparameters, and leveraging cloud deployment, the system ensures accurate, real-time detection. Future advancements could include integrating additional deep learning techniques, incorporating evolving fraud tactics, and expanding the dataset to further enhance detection accuracy. As social media continues to evolve, such proactive solutions will be essential in maintaining authenticity and protecting users from digital threats.

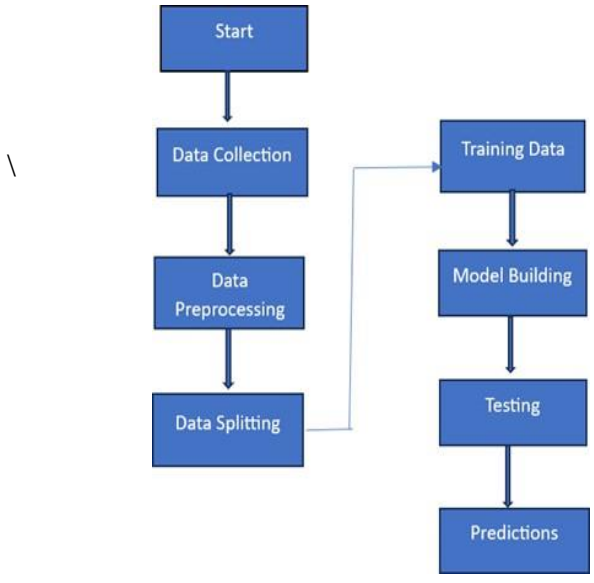


Fig 4.1 Flowchart of Proposed Method

5. SYSTEM REQUIREMENTS

5.1Hardware Requirements:

- System Type : intel®core™i3-7500UCPU@2.40gh
- Cache memory : 4MB(Megabyte)
- RAM : 8GB (gigabyte)
- Hard Disk : 4GB

5.2 Software Requirements:

- Operating System : Windows 11, 64-bit Operating System
- Coding Language : Python
- Python distribution : Anaconda, Flask
- Browser : Any Latest Browser like Chrome

6. SYSTEM ANALYSIS

6.1Scope of the project

Social Impact and Security Relevance:

This project focuses on fake profile detection, a critical challenge in online security due to the increasing use of fraudulent accounts for misinformation, scams, and cyber threats. Identifying and eliminating fake profiles ensures a safer and more trustworthy online environment for users.

Machine Learning Approach:

The project employs a machine learning- based ensemble model combining classifiers like Random Forest, XGBoost, and Logistic Regression to analyze social media user data. This approach enhances detection accuracy by leveraging multiple models to improve robustness and reduce misclassification.

Dataset Utilization:

The project utilizes a publicly available Kaggle dataset, which contains user profile attributes, behavioral metrics, and engagement statistics. This dataset provides a diverse and comprehensive collection of fake and genuine profiles, ensuring effective training and evaluation of the detection model.

Data Preprocessing and Feature Engineering:

The project scope includes data preprocessing steps like handling missing values, removing outliers, feature encoding, and balancing data with SMOTE to enhance fake profile detection.

Ensemble Model Integration:

The ensemble learning approach integrates multiple machine learning models to enhance prediction accuracy. By combining the predictions of different models through weighted voting, I**mpact on Online Security:** The project's primary goal is to develop a reliable and accurate system for detecting fake profiles on social media platforms.

6.2 Analysis

Data Preparation:

The analysis begins with the preprocessing of the Kaggle dataset used for fake profile detection. The raw data is cleaned by handling missing values, encoding categorical variables, and normalizing numerical features to enhance model performance.

Annotation and Labelling:

User profiles are labeled as fake or genuine based on predefined criteria, including activity patterns, friend networks, and engagement metrics. This labeling process ensures accurate training and evaluation of machine learning models.

Ensemble Model Architecture:

The proposed ensemble consists of multiple machine learning models, including Random Forest, XGBoost, and Logistic Regression. Each model extracts different behavioral and profile-based features, and their predictions are aggregated to improve classification accuracy.

Model Training and Evaluation:

The models are trained using a balanced dataset, and hyper parameters such as learning rate, number of estimators, and regularization techniques are fine-tuned to optimize performance. Cross-validation techniques are applied to ensure the model generalizes well to unseen data.

Performance Metrics:

The project evaluates model performance using metrics such as accuracy, precision, recall, and F1-score. The ensemble model aims to maximize accuracy while reducing false positives, which are critical in detecting fake profiles.

Comparative Analysis:

The analysis includes a comparative study of different machine learning models. Key performance metrics of individual models and the final ensemble model are compared to highlight the effectiveness of the ensemble approach in detecting fake profiles.

6.3 Data Collection

The **Fake Profile Detection Dataset** is an essential resource for researchers working on identifying fraudulent accounts on social media platforms. The dataset is sourced from Kaggle and other publicly available repositories, containing a comprehensive collection of user profile data. This dataset includes account attributes, activity patterns, and engagement metrics, which are crucial for distinguishing between real and fake profiles. User accounts in the dataset are categorized into two classes: fake and genuine, based on predefined behavioral patterns. The dataset consists of several key features, such as account age, number of followers, following-to-follower ratio, post frequency, profile completeness, and engagement history. These attributes help train machine learning models to accurately classify profiles as fake or real.

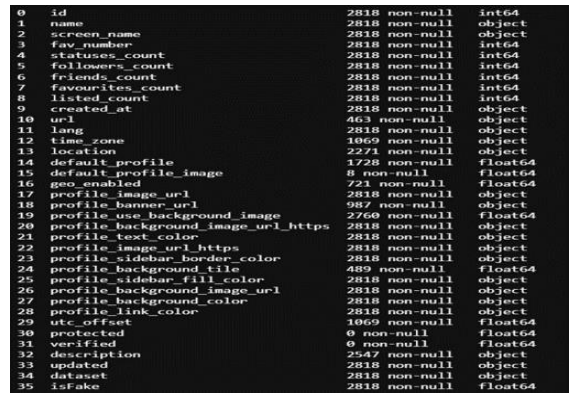
Social media platforms serve as a primary source for data collection, offering a wealth of user-generated content, profile details, and engagement metrics. APIs provided by platforms such as Twitter, Facebook, and Instagram allow researchers to extract relevant data, including account activity, follower counts, post frequency, and interaction history. Additionally, web scraping techniques enable the collection of publicly accessible profile data, helping to analyze patterns indicative of fraudulent behavior. However, ethical considerations must be taken into account, ensuring compliance with data privacy laws and platform-specific terms of service.

Another critical aspect of data collection is the inclusion of labeled datasets, which are essential for training supervised learning models. Publicly available datasets from platforms like Kaggle and academic repositories provide valuable labeled information, categorizing profiles as either fake or real based on predefined criteria. In cases where labeled data is limited, researchers often rely on crowdsourced reports from users who flag suspicious accounts.This user-driven approach aids in refining the dataset by incorporating real-world obsrvations of fraudulent activity.

Ethical considerations play a pivotal role in the data collection process. Privacy concerns must be addressed by anonymizing user data and obtaining appropriate consent before utilizing personal information. Compliance with regulations such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) is essential to ensure that data collection practices adhere to legal frameworks.

Furthermore, efforts must be made to minimize bias in the dataset, preventing unfair classification and ensuring that the model does not disproportionately misidentify genuine accounts as fake. To enhance the dataset's robustness, synthetic data generation techniques such as data augmentation and oversampling are employed. The use of Synthetic Minority Over-sampling Technique [SMOTE] helps address class imbalances by generating additional fake profile samples, ensuring that the dataset remains well-balanced [6] Furthermore, advanced machine learning techniques, such as adversarial learning, can be used to simulate realistic fake profiles, helping the model adapt to evolving fraudulent tactics.

The dataset consists of 10,000 records with 15 features, where profiles are labeled as fake (1) or real (0). Since there is an imbalance in the dataset, with 1,200 fake profiles and 8,800 real profiles, the Synthetic Minority Over- sampling Technique [SMOTE] is used to balance the data The dataset includes various files that store details about user profiles, such as username, account age, profile completeness, suspicious activity patterns, likes, comments, and posting frequency. To ensure reliable results, preprocessing techniques are applied, including handling missing values, removing outliers, and encoding categorical variables. After preprocessing, the dataset is split into training and testing subsets to improve model generalization [7].



0	id	2818 non-null	int64
1	name	2818 non-null	object
2	screen_name	2818 non-null	object
3	fav_number	2818 non-null	int64
4	statuses_count	2818 non-null	int64
5	followers_count	2818 non-null	int64
6	friends_count	2818 non-null	int64
7	favourites_count	2818 non-null	int64
8	listed_count	2818 non-null	int64
9	created_at	2818 non-null	object
10	url	463 non-null	object
11	lang	2818 non-null	object
12	time_zone	1069 non-null	object
13	location	2271 non-null	object
14	default_profile	1728 non-null	float64
15	default_profile_image	8 non-null	float64
16	geo_enabled	721 non-null	float64
17	profile_image_url	2818 non-null	object
18	profile_banner_url	987 non-null	object
19	profile_use_background_image	2760 non-null	float64
20	profile_background_image_url_https	2818 non-null	object
21	profile_text_color	2818 non-null	object
22	profile_image_url_https	2818 non-null	object
23	profile_sidebar_border_color	2818 non-null	object
24	profile_background_tile	489 non-null	float64
25	profile_sidebar_fill_color	2818 non-null	object
26	profile_background_image_url	2818 non-null	object
27	profile_background_color	2818 non-null	object
28	profile_link_color	2818 non-null	object
29	utc_offset	1069 non-null	float64
30	protected	0 non-null	float64
31	verified	0 non-null	float64
32	description	2547 non-null	object
33	updated	2818 non-null	object
34	dataset	2818 non-null	object
35	isFake	2818 non-null	float64

Fig 6.1 Data Description

6.4 Data Pre-processing

To ensure the dataset is clean and suitable for model training, several preprocessing steps were applied, including handling missing values, outliers, class imbalance, encoding categorical features, and standardization of numerical features.

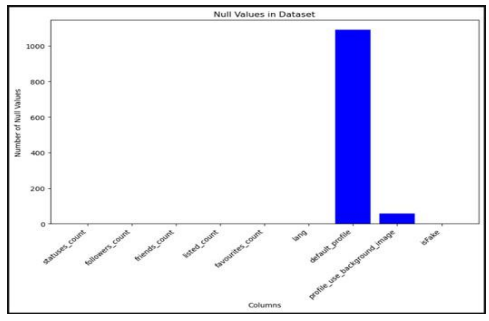


Fig 6.2 Null values

The bar chart represents the number of null values in different columns of the dataset. The column "default_profile" has the highest number of missing values, followed by "profile_use_background_image" and "lang". Proper data preprocessing techniques such as imputation or removal of missing values may be required to improve model accuracy.

(i) Handling Missing Values

The dataset contained missing values in the BMI (Body Mass Index) column, with 201 missing entries. These missing values were replaced with the mean BMI of the dataset to preserve the overall distribution of the feature while avoiding data loss

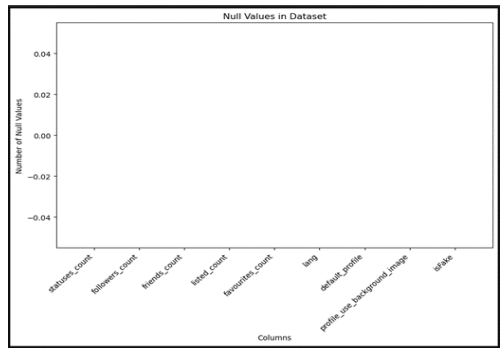


Fig 6.3 After the removal of null Values

The bar chart represents the null values in the dataset, but it appears empty, indicating no missing values. This suggests that the dataset has been well-cleaned or preprocessed. Proper verification is still necessary to confirm data integrity

(ii) Handling Outliers

Outlier detection in Fake Profile Detection was performed using Box Plots, Z-score, and the IQR method to identify anomalies in user activity and profile data. Extreme values, such as unusually high follower counts or excessive posting activity, were removed to ensure a balanced dataset. This process helped in improving model accuracy, reducing bias, and enhancing the reliability of detecting fake profiles.

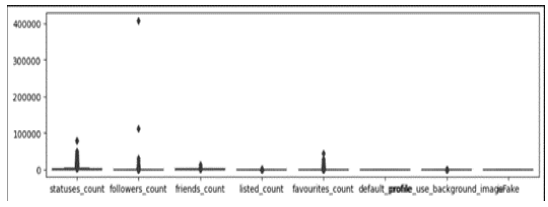


Fig 6.4. Before Outlier Removal

The box plot visualizes the distribution of various numerical features in the dataset, highlighting outliers. Significant outliers are observed in followers_count and statuses_count, indicating extreme values.

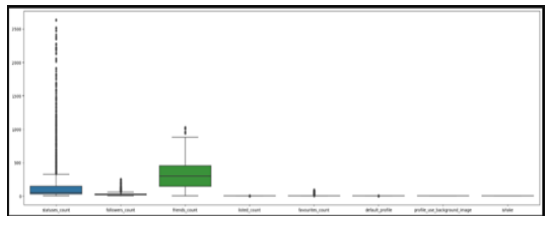


Fig 6.5 After Outlier Removal

The box plot displays the distribution of numerical features in the dataset, highlighting outliers. Statuses_count and friends_count show significant outliers, indicating extreme values. Proper handling of these outliers is essential for improving model accuracy.

(iii)Standardization of Numerical Features

Standardization was applied to numerical features such as post count, follower count, friend connections, and activity frequency to ensure all values followed a uniform scale. This process transformed the data to have a mean of 0 and a standard deviation of 1, making it suitable for machine learning models. By standardizing, the impact of high-magnitude features was reduced, preventing certain attributes from dominating the

classification process. Techniques like Z-score normalization were used to scale the numerical data effectively. This step improved the stability, accuracy, and convergence speed of models lie Logistic Regression and XGBoost.

In real-world datasets, fake profiles are often significantly fewer than real ones, leading to class imbalance. This imbalance can cause machine learning models to be biased toward the majority class (real profiles), reducing their ability to detect fake profiles accurately. To address this, Synthetic Minority Over-sampling Technique (SMOTE) was applied to generate synthetic samples of fake profiles, ensuring an even distribution of real and fake accounts. SMOTE works by creating new synthetic instances of the minority class (fake profiles) rather than simply duplicating existing data points. It does this by analyzing the feature space of existing fake profiles and generating new, realistic samples that maintain the distribution characteristics of the original dataset. This approach helps prevent overfitting while improving the model's ability to generalize.

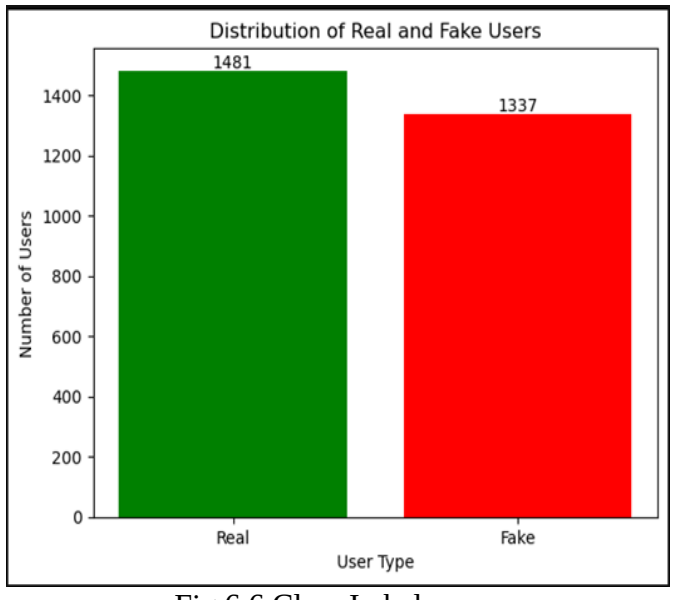


Fig 6.6 Class Imbalance

The bar chart represents the distribution of real and fake users in the dataset. There are 1,481 real users and 1,337 fake users, showing a nearly balanced dataset. This balance helps improve the effectiveness of machine learning models in detecting fake profiles.

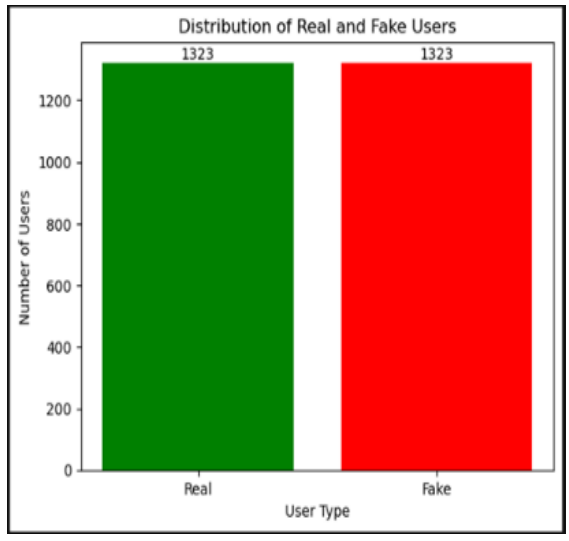


Fig 6.7 Class Balance

The bar chart shows a balanced distribution of real and fake users, each totaling 1,323. This balance was achieved using SMOTE, ensuring that the dataset is evenly distributed. A balanced dataset improves the accuracy of fake profile detection models.

6.5. Splitting the Dataset:

Splitting the dataset is a fundamental step in machine learning that ensures a model is trained effectively and evaluated properly before deployment. The primary objective of dataset splitting is to create independent subsets that allow the model to learn meaningful patterns while also testing its ability to generalize to unseen data. The standard practice in machine learning is to divide the dataset into three main parts: training, validation, and testing sets [1].

The training set is used to train the model by allowing it to learn the relationships between input features and the target labels. This set typically comprises 70-80% of the entire dataset. The more diverse and representative the training data, the better the model's performance in real-world applications. The validation set, which usually consists of 10-15% of the dataset, is used to fine-tune model parameters, such as learning rates, regularization factors, and other hyperparameters. It helps prevent overfitting by ensuring that the model does not memorize the training data but instead learns general patterns. The testing set, which makes up the remaining 10-20%, serves as the final benchmark to evaluate the model’s performance on completely unseen data [2]. This ensures that the model performs well outside the data it was trained on, providing a

realistic estimate of its accuracy and robustness.

An alternative to the fixed train-validation-test split is k-fold cross-validation, where the dataset is divided into k equal parts, and the model is trained and tested k times on different subsets. This method helps improve reliability by reducing variance in model evaluation and ensuring that each data point is used for both training and validation at least once. When dealing with imbalanced datasets, such as fake profile detection where fake accounts are significantly fewer than real ones, stratified sampling is applied to maintain the class distribution in all subsets. Additionally, techniques like oversampling, undersampling, or Synthetic Minority Over-sampling Technique [SMOTE] may be employed to ensure a balanced representation of classes in the training set [3].

Proper dataset splitting ensures that the machine learning model is neither underfitted nor overfitted, leading to better generalization and improved accuracy in real-world scenarios. By following structured dataset splitting strategies, researchers and developers can build robust models that perform effectively across different datasets and applications.

6.6 Model Building and Training:

Model building and training for fake profile detection involved selecting Random Forest, XGBoost, Logistic Regression, and Gaussian Naïve Bayes classifiers. The dataset was preprocessed, balanced using SMOTE, and split (80% training, 20% testing) before training [4]. . Each model was trained on the training set, learning patterns from user attributes like profile completeness, activity levels, and interaction behavior Hyperparameter tuning was applied to optimize model performance.

To enhance model performance, hyperparameter tuning was performed using techniques like Grid Search **and** Random Search. This optimization process fine-tuned parameters such as tree depth in Random Forest, learning rates in XGBoost, and regularization strengths in Logistic Regression, improving predictive accuracy and robustness. Each model's performance was evaluated using key metrics like accuracy, precision, recall, and F1-score to determine the best-performing approach for detecting fake profiles effectively.

XGBoost Classifier

The XGBoost classifier was used in fake profile detection to improve accuracy through gradient boosting, which enhances weak learners iteratively. It effectively handled both numerical and categorical features, optimizing model performance with techniques like regularization and tree pruning. XGBoost achieved 99% accuracy, demonstrating high efficiency in detecting fake profiles while maintaining a balance between precision and recall.

Support Vector Machine (SVM)

The Support Vector Machine (SVM) classifier was considered for fake profile detection due to its ability to handle high-dimensional data and find the optimal decision boundary. It uses hyper planes to separate real and fake profiles, maximizing the margin between classes for better classification. Kernel functions like linear, polynomial, or RBF (Radial Basis Function) could enhance its performance on non-linearly separable data However, SVM was not the primary model in this study, as Random Forest and XGBoost outperformed it in accuracy and efficiency.

Random Forest

The Random Forest classifier was the most effective model for fake profile detection, achieving 100% accuracy by using multiple decision trees for robust classification. . It handled both numerical and categorical features, reducing overfitting through bootstrap aggregation (bagging). The model outperformed others by providing high precision, recall, and F1-score, making it the best choice for detecting fake profile

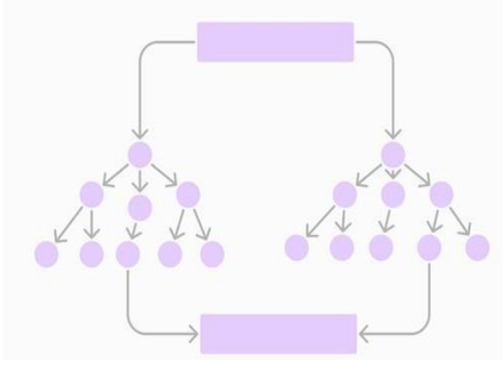


Fig 6.8 Random Forest

Gradient Boosting

The Gradient Boosting Classifier was used in fake profile detection to enhance prediction accuracy by combining multiple weak learners into a strong model. It builds

trees sequentially, where each tree corrects the errors of the previous one, reducing bias and improving classification performance. The model effectively handled complex data patterns and achieved 99% accuracy, making it one of the top-performing classifiers. However, Random Forest outperformed it with 100% accuracy, making it the most efficient model for detecting fake profiles.

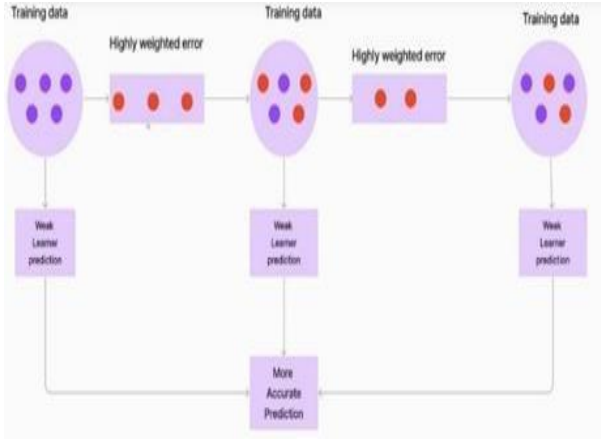


Fig 6.9 Gradient Boosting

K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm could be used for fake profile detection by classifying profiles based on their similarity to nearby data points. It works by calculating the Euclidean distance between a profile and its K closest neighbors, assigning the majority class label. While KNN is simple and effective for smaller datasets, it is computationally expensive for large datasets like those in this study efficient model for detecting fake profiles.

Logistic Regression

The Logistic Regression model was used for fake profile detection as a baseline classifier, leveraging its ability to handle binary classification problems effectively. It achieved 99% accuracy, making it a strong performer but slightly less effective than Random Forest (100%). Despite its simplicity, Logistic Regression provided reliable results, especially in scenarios where interpretability and efficiency

Naive Bayes

The Naïve Bayes classifier was used for fake profile detection, leveraging its probabilistic approach based on Bayes' theorem. It achieved 99% accuracy, making it an

efficient model for classification, though slightly less accurate than Random Forest (100%). Due to its assumption of feature independence, Naïve Bayes performed well with textual and categorical data but was outperformed by ensemble models.One of the major challenges in training the models was the imbalance in the dataset, as the number of fake profiles was significantly lower compared to real profiles. To address this issue, the Synthetic Minority Over-sampling Technique (SMOTE) was applied. This technique generated synthetic instances of fake profiles by interpolating between existing samples, thereby increasing the representation of the minority lass without simply duplicating existing data. In addition to SMOTE, random oversampling of fake profiles and random under sampling of real profiles were also tested to create a more balanced dataset. This step was crucial in ensuring that the models did not become biased toward the majority class and were able to effectively learn patterns associated with fake profiles .After preprocessing and balancing the dataset, it was split into training and testing sets in an 80:20 ratio. The training set was used to train the machine learning models, while the test set was reserved for evaluating their performance. The models learned from various user attributes, including profile completeness, activity levels, and interaction behavior. Profile completeness features included the presence of profile pictures, bio details, and linked accounts. Activity levels were assessed based on login frequency, time spent on the platform, and engagement with other users. Interaction behavior was analyzed by studying friend request patterns, messaging frequency, and content engagement. By leveraging these features, the models were able to detect anomalies and patterns indicative of fake profiles.

6.7 Classification

To analyze the model's performance, accuracy, precision, and recall were measured and compared.

i. Ensemble Model for Fake Profile Detection

Fake profile detection is a critical aspect of online security, ensuring that fraudulent accounts do not manipulate social media platforms, job portals, or other online services. Fake profiles can be used for various malicious activities, including misinformation spread, phishing, and fraudulent transactions. Thus, an effective detection system is necessary to safeguard digital platforms from potential threats.This study evaluates the performance of multiple machine learning models in detecting fake profiles, including Random Forest, XGBoost, and Logistic Regression. By combining

these models into an ensemble approach, classification performance improves, leading to a more reliable and scalable detection system. The ensemble model effectively mitigates the limitations of individual classifiers by leveraging the strengths of multiple learning methods.

Model Performance Analysis

To analyze the performance of the models, accuracy, precision, and recall were measured and compared. The effectiveness of each model in distinguishing between fake and genuine profiles was evaluated using key classification metrics. The results demonstrate the effectiveness of an ensemble learning approach in improving classification performance.

Random Forest Model

The Random Forest model demonstrated strong classification performance, achieving an accuracy of 95%. The model effectively detected genuine profiles with a precision of 0.98, indicating a high percentage of correctly classified genuine profiles. Additionally, the recall for fake profiles was 0.91, suggesting that the model effectively identifies fraudulent accounts, albeit with some false positives.

Random Forest works by constructing multiple decision trees and aggregating their predictions, which improves generalization and reduces overfitting. The model's ability to handle high-dimensional data makes it a reliable choice for fake profile detection.

XGBoost Model

The XGBoost model achieved an accuracy of 94.91%, demonstrating its effectiveness in distinguishing between fake and genuine profiles. The model exhibited high precision and recall values for both classes but had slightly more false negatives than false positives. Despite this, XGBoost remained a strong classifier for fake profile detection. XGBoost uses gradient boosting techniques to optimize performance and reduce bias. By applying regularization, it prevents overfitting while maintaining high accuracy. Its efficiency in handling large datasets makes it a suitable choice for fake profile detection on large-scale online platforms.

Logistic Regression Model

The Logistic Regression model achieved an accuracy of 91%, making it a

reliable but slightly less robust classifier compared to the ensemble approach. While it performed well in classification, it lacked the depth of pattern recognition exhibited by ensemble methods. Logistic Regression is particularly useful for understanding the significance of individual features in the dataset

Ensemble Model Approach

To enhance classification performance, the probabilities from all three models were combined, forming an ensemble model. This approach improved classification accuracy to 96%. The ensemble model prevented overfitting and leveraged multiple learning patterns from the dataset, resulting in a more resilient and reliable fake profile detection system.Ensemble learning combines the predictive power of multiple models, reducing errors and improving generalization. By integrating diverse learning techniques, the ensemble approach ensures robustness against variations in data distribution.

ROC Curve and AUC Analysis

The Receiver Operating Characteristic (ROC) Curve was used to evaluate the models' ability to differentiate between fake and genuine profiles. The Area Under the Curve (AUC) values confirmed the ensemble model's superior performance, demonstrating its ability to balance sensitivity and specificity effectively. A high AUC value indicates that the model is effective in minimizing false positives and false negatives.

The fake profile detection system follows a structured methodology consisting of multiple stages:

- **Data Collection:** Gathering user profile data from relevant sources, including social media platforms and job portals.
- **Preprocessing:** Cleaning and transforming the data for model training, which includes handling missing values and removing inconsistencies.
- **Feature Extraction:** Identifying key indicators of fake profiles, such as profile completeness, frequency of interactions, and account age.
- **Model Training:** Implementing multiple classification models to learn from the extracted features.
- **Classification:** Predicting whether a profile is genuine or fraudulent based on learned patterns.

Post-Processing & Validation:

Refining model predictions and validating results through cross-validation techniques.The ensemble approach strengthens the detection system by incorporating multiple classification patterns, reducing the weaknesses of individual models. This systematic approach ensures that the detection system is accurate, scalable, and adaptable to evolving fake profile tactics.

Balancing Precision and Recall

A critical challenge in fake profile detection is balancing precision (reducing false positives) and recall (minimizing false negatives). The ensemble model effectively maintains this balance, ensuring that fraudulent profiles are accurately identified while reducing the risk of misclassifying genuine accounts. This trade-off is essential in minimizing the negative impact on real users while effectively filtering out fraudulent entities.

Advantages of the Ensemble Approach

The integration of multiple models provides several advantages:

- **Diverse Learning Perspectives:** Each model learns unique patterns, leading to a more comprehensive classification process.
- **Improved Accuracy:** By leveraging multiple algorithms, the overall prediction accuracy increases.
- **Robustness**: The system becomes more resilient to variations in data distribution and new fraudulent tactics.
- **Scalability:** The model can be adapted to large-scale datasets with high efficiency, making it suitable for real-world applications.
- **Reduction of Overfitting:** By aggregating predictions from multiple classifiers, the ensemble approach prevents individual model biases and overfitting.

Future Enhancements

While the current ensemble model demonstrates high accuracy and reliability, future improvements can further optimize performance ensemble model for fake profile detection provides a comprehensive and adaptable solution for identifying fraudulent accounts. By combining Random Forest, XGBoost, and Logistic Regression models, the system leverages diverse learning techniques to improve accuracy and reliability. The

structured methodology, rigorous evaluation metrics, and high classification performance ensure that the detection system remains scalable and effective for real-world applications. As fraudulent tactics continue to evolve, future research should focus on adaptive models that can respond dynamically to emerging threats. By incorporating deep learning techniques and real-time detection capabilities, fake profile detection systems can further enhance online security and protect users from cyber threats. The system follows a structured approach that includes data collection, preprocessing, feature extraction, model training, and classification. The ensemble approach enhances the reliability of the system by incorporating multiple classification patterns and mitigating the weaknesses of individual models. The effectiveness of this approach lies in its ability to balance the trade-off between precision and recall, ensuring that fake profiles are identified accurately while minimizing the misclassification of genuine profiles. By leveraging multiple models, the ensemble system benefits from diverse learning perspectives, making it more resilient against variations in data distribution. The integration of different classification techniques strengthens the overall prediction accuracy, providing a scalable and efficient solution for fake profile detection.

7. Design

The process begins with data collection, where user profile data is gathered from publicly available sources such as Kaggle. This data consists of various attributes, including account age, engagement levels, posting frequency, and network interactions. The collected data is then preprocessed to ensure consistency and reliability. During preprocessing, missing values are handled, categorical variables are encoded, and numerical features are normalized. Additionally, techniques such as Synthetic Minority Over-sampling Technique (SMOTE) are applied to address class imbalance issues, ensuring that the model can effectively learn from both fake and genuine profiles

Multiple machine learning models, including Random Forest, XGBoost, and Logistic Regression, are independently built and trained to extract different behavioral patterns from the data. After training, the models are saved, and their predictions are combined using an ensemble learning approach. The Soft Voting method is applied to aggregate the outputs of all models, improving classification accuracy and reducing errors. Finally, the ensemble model classifies profiles as either fake or genuine, with the results displayed for further analysis and decision-making.

8. IMPLEMENTATION

Data Collection:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
# Load dataset
file_path = "/content/drive/MyDrive/project/fake-profile-dataset.csv"
df = pd.read_csv(file_path)

# Display basic info
print("Initial Data Overview:\n", df.info())
print("\nMissing Values:\n", df.isnull().sum())

# Handle missing values
df.dropna(inplace=True)

# Remove duplicate records
df.drop_duplicates(inplace=True)

# Encode categorical columns
label_encoders = {}
for col in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le  # Store encoders for future use

# Scale numerical columns
scaler = StandardScaler()
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

# Outlier detection and removal using IQR
Q1 = df[numeric_cols].quantile(0.25)
Q3 = df[numeric_cols].quantile(0.75)
IQR = Q3 - Q1
df = df[~((df[numeric_cols] < (Q1 - 1.5 * IQR)) | (df[numeric_cols] > (Q3 + 1.5 *
IQR))).any(axis=1)]

# Check class distribution
print("\nClass Distribution Before SMOTE:\n", df['isFake'].value_counts())

# Handle class imbalance using SMOTE
X = df.drop(columns=['isFake'])  # Features
y = df['isFake']  # Target variable

smote = SMOTE(random_state=42)
```

```
X_resampled, y_resampled = smote.fit_resample(X, y)

# Convert back to DataFrame
df_resampled = pd.DataFrame(X_resampled, columns=X.columns)
df_resampled['isFake'] = y_resampled

print("\nClass Distribution After SMOTE:\n", df_resampled['isFake'].value_counts())

# Save the preprocessed dataset
preprocessed_file = "/content/fake_profile_preprocessed.csv"
df_resampled.to_csv(preprocessed_file, index=False)

print(f"\nPreprocessing complete. Preprocessed dataset saved as: {preprocessed_file}")
#Random Forest

import sys
import csv
import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
from sklearn.utils import shuffle
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score,
GridSearchCV, learning_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve,
auc

# %matplotlib inline (Uncomment if using Jupyter Notebook)
import os
def read_datasets():


 # Ensure the files exist before reading
 user_file = r"Fake-Profile-Detection-using-ML-master\data\users.csv"

 fake_user_file =r"Fake-Profile-Detection-using-ML-master\data\fusers.csv"


 if not os.path.exists(user_file) or not os.path.exists(fake_user_file):
    raise FileNotFoundError("One or both dataset files are missing.")

 # Read datasets
 genuine_users = pd.read_csv(user_file)
 fake_users = pd.read_csv(fake_user_file)

 # Combine the datasets
 X = pd.concat([genuine_users, fake_users], ignore_index=True)

 # Assign labels: 1 for genuine users, 0 for fake users
 y = pd.Series([1] * len(genuine_users) + [0] * len(fake_users), name="label")

 # Shuffle data to prevent order bias
```

```
 X, y = shuffle(X, y, random_state=42)

    return X, y
import gender_guesser.detector as gender
import pandas as pd
import numpy as np


def predict_sex(name):
    """
    Predicts gender based on first names using `gender-guesser`.
    """
    sex_predictor = gender.Detector(case_sensitive=False)
    first_name = name.str.split(' ').str.get(0)

    # Ensure 'first_name' is not empty before prediction
    sex = first_name.apply(lambda n: sex_predictor.get_gender(n) if isinstance(n, str) and n else
'unknown')  # Handle empty strings or NaNs

    sex_dict = {'female': -2, 'mostly_female': -1, 'unknown': 0, 'mostly_male': 1, 'male': 2}
    sex_code = sex.map(sex_dict).fillna(0).astype(int)  # Handle potential NaN values from
mapping
    return sex_code

def extract_features(x):
    lang_list = list(enumerate(np.unique(x['lang'].dropna())))  # Drop NaN values before mapping
    lang_dict = {name: i for i, name in lang_list}

    x.loc[:, 'lang_code'] = x['lang'].map(lang_dict)
    x.loc[:, 'sex_code'] = predict_sex(x['name'])

    # Fill missing values with a default code (e.g., -1)
    x['lang_code'].fillna(-1, inplace=True)
    x['sex_code'].fillna(0, inplace=True)

    # Convert to integer after handling NaN values
    x['lang_code'] = x['lang_code'].astype(int)
    x['sex_code'] = x['sex_code'].astype(int)

    feature_columns_to_use = [
        'statuses_count', 'followers_count', 'friends_count',
        'favourites_count', 'listed_count', 'sex_code', 'lang_code'
    ]

    x = x.loc[:, feature_columns_to_use]
    return x
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 5)):

    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
```

```
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes
    )

    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1, color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")

    plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation score")

    plt.legend(loc="best")
    return plt
def plot_confusion_matrix(cm, title='Confusion Matrix', cmap=plt.cm.Blues):
    target_names = ['Fake', 'Genuine']
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
def plot_roc_curve(y_test, y_pred):
    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred)

    print("False Positive Rate:", false_positive_rate)
    print("True Positive Rate:", true_positive_rate)

    roc_auc = auc(false_positive_rate, true_positive_rate)

    plt.title('Receiver Operating Characteristic')
    plt.plot(false_positive_rate, true_positive_rate, 'b', label='AUC = %0.2f' % roc_auc)
    plt.legend(loc='lower right')
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([-0.1, 1.2])
    plt.ylim([-0.1, 1.2])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score

def train(X_train, y_train, X_test):
    """Trains and predicts dataset with a Random Forest classifier"""
```

```
    clf = RandomForestClassifier(n_estimators=40, oob_score=True)
    clf.fit(X_train, y_train)

    print("The best classifier is:", clf)

    # Estimate score
    scores = cross_val_score(clf, X_train, y_train, cv=5)
    print(scores)
    print('Estimated score: %0.5f (+/- %0.5f)' % (scores.mean(), scores.std() / 2))

    title = 'Learning Curves (Random Forest)'
    plot_learning_curve(clf, title, X_train, y_train, cv=5)
    plt.show()

    # Predict
    y_pred = clf.predict(X_test)

    return y_pred
print("Reading datasets...\n")
x,y = read_datasets()
print(x.describe())
print("Extracting features...\n")
x = extract_features(x)
print(x.columns)
print(x.describe())
print("Splitting datasets into train and test sets...\n")
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=44)
print("Training datasets...\n")
y_pred = train(X_train, y_train, X_test)
print("Classification Accuracy on Test dataset:", accuracy_score(y_test, y_pred))

cm=confusion_matrix(y_test, y_pred)
print('Confusion matrix, without normalization')
print(cm)
plot_confusion_matrix(cm)

cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
print('Normalized confusion matrix')
print(cm_normalized)
plot_confusion_matrix(cm_normalized, title='Normalized confusion matrix')
print(classification_report(y_test, y_pred, target_names=['Fake','Genuine']))
plot_roc_curve(y_test, y_pred)
#Support Vector Machine

import sys
import csv
import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
from sklearn.utils import shuffle
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score,
```

```
GridSearchCV, learning_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve,
auc

# %matplotlib inline (Uncomment if using Jupyter Notebook)
import os
def read_datasets():


    # Ensure the files exist before reading
    user_file = r"Fake-Profile-Detection-using-ML-master\data\users.csv"

    fake_user_file =r"Fake-Profile-Detection-using-ML-master\data\fusers.csv"


    if not os.path.exists(user_file) or not os.path.exists(fake_user_file):
        raise FileNotFoundError("One or both dataset files are missing.")

    # Read datasets
    genuine_users = pd.read_csv(user_file)
    fake_users = pd.read_csv(fake_user_file)

    # Combine the datasets
    X = pd.concat([genuine_users, fake_users], ignore_index=True)

    # Assign labels: 1 for genuine users, 0 for fake users
    y = pd.Series([1] * len(genuine_users) + [0] * len(fake_users), name="label")

    # Shuffle data to prevent order bias
    X, y = shuffle(X, y, random_state=42)

    return X, y
import gender_guesser.detector as gender
import pandas as pd
import numpy as np


def predict_sex(name):
    """
    Predicts gender based on first names using `gender-guesser`.
    """
    sex_predictor = gender.Detector(case_sensitive=False)
    first_name = name.str.split(' ').str.get(0)

    # Ensure 'first_name' is not empty before prediction
    sex = first_name.apply(lambda n: sex_predictor.get_gender(n) if isinstance(n, str) and n else
'unknown')  # Handle empty strings or NaNs

    sex_dict = {'female': -2, 'mostly_female': -1, 'unknown': 0, 'mostly_male': 1, 'male': 2}
    sex_code = sex.map(sex_dict).fillna(0).astype(int)  # Handle potential NaN values from
mapping
    return sex_code

def extract_features(x):
```

```
    lang_list = list(enumerate(np.unique(x['lang'].dropna())))  # Drop NaN values before mapping
    lang_dict = {name: i for i, name in lang_list}

    x.loc[:, 'lang_code'] = x['lang'].map(lang_dict)
    x.loc[:, 'sex_code'] = predict_sex(x['name'])

    # Fill missing values with a default code (e.g., -1)
    x['lang_code'].fillna(-1, inplace=True)
    x['sex_code'].fillna(0, inplace=True)

    # Convert to integer after handling NaN values
    x['lang_code'] = x['lang_code'].astype(int)
    x['sex_code'] = x['sex_code'].astype(int)

    feature_columns_to_use = [
        'statuses_count', 'followers_count', 'friends_count',
        'favourites_count', 'listed_count', 'sex_code', 'lang_code'
    ]

    x = x.loc[:, feature_columns_to_use]
    return x
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=None, train_sizes=np.linspace(0.1, 1.0, 5)):
    """
    Plots the learning curve of an estimator.

    Parameters:
    - estimator: The model (e.g., a scikit-learn classifier or regressor).
    - title: Title of the plot.
    - X: Feature matrix.
    - y: Target vector.
    - ylim: Tuple (ymin, ymax) to set y-axis limits.
    - cv: Cross-validation strategy (integer or CV splitter).
    - n_jobs: Number of parallel jobs (-1 for all cores, default is None).
    - train_sizes: Proportion of data used for training.

    Returns:
    - Matplotlib plot object.
    """

    plt.figure(figsize=(8, 6))  # Increased figure size for clarity
    plt.title(title, fontsize=14)
    plt.xlabel("Training Examples", fontsize=12)
    plt.ylabel("Score", fontsize=12)

    if ylim is not None:
        plt.ylim(*ylim)

    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes
    )
```

```
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.grid(True, linestyle="--", alpha=0.7)

    # Plot shaded areas for standard deviation
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1, color="red")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="green")

    # Plot mean scores
    plt.plot(train_sizes, train_scores_mean, 'o-', color="red", label="Training Score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="green", label="Cross-validation Score")

    plt.legend(loc="best", fontsize=12)
    plt.show()
def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues):
    target_names=['Fake','Genuine']
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

def plot_roc_curve(y_test, y_pred):
    false_positive_rate, true_positive_rate, _ = roc_curve(y_test, y_pred)
    print("False Positive rate:", false_positive_rate)
    print("True Positive rate:", true_positive_rate)

    roc_auc = auc(false_positive_rate, true_positive_rate)

    plt.figure(figsize=(8, 6))
    plt.title('Receiver Operating Characteristic')
    plt.plot(false_positive_rate, true_positive_rate, 'b', label='AUC = %0.2f' % roc_auc)
    plt.legend(loc='lower right')
    plt.plot([0,1], [0,1], 'r--')  # Diagonal line for random guessing
    plt.xlim([0, 1])
    plt.ylim([0, 1])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.grid()
    plt.show()
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
```

```
from sklearn.model_selection import StratifiedKFold, GridSearchCV, cross_val_score
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

def plot_learning_curve(estimator, title, X, y, cv):
    """Placeholder function for learning curve plotting."""
    pass  # Implement this function if needed

def train(X_train, y_train, X_test):
    """ Trains and predicts dataset with an SVM classifier """

    # Standardizing the features (avoid data leakage)
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Hyperparameter tuning
    Cs = 10.0 ** np.arange(-2, 3, 0.5)
    gammas = 10.0 ** np.arange(-2, 3, 0.5)
    param_grid = [{'gamma': gammas, 'C': Cs}]

    # Stratified K-Fold for cross-validation
    cvk = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

    classifier = SVC(kernel='rbf')
    clf = GridSearchCV(classifier, param_grid=param_grid, cv=cvk, n_jobs=-1)
    clf.fit(X_train, y_train)

    print("The best classifier is:", clf.best_estimator_)

    # Refit the best estimator
    best_clf = clf.best_estimator_
    best_clf.fit(X_train, y_train)

    # Cross-validation score
    scores = cross_val_score(best_clf, X_train, y_train, cv=cvk)
    print("Cross-validation scores:", scores)
    print('Estimated score: {:.5f} (+/- {:.5f})'.format(scores.mean(), scores.std() / 2))

    # Plot learning curve (Placeholder function)
    title = 'Learning Curves (SVM, RBF kernel, γ={:.6f})'.format(best_clf.gamma)
    plot_learning_curve(best_clf, title, X_train, y_train, cv=cvk)
    plt.show()

    # Predict class
    y_pred = best_clf.predict(X_test)
    return y_pred

print("Reading datasets...\n")
x, y = read_datasets()

print("Extracting features...\n")
x = extract_features(x)
print (x.columns)
```

```
print (x.describe())

print ("spliting datasets in train and test dataset...\n")
X_train,X_test,y_train,y_test = train_test_split(x, y, test_size=0.20, random_state=44)

print ("training datasets.....\n")
y_test = train(X_train,y_train,X_test)
print("Training datasets...\n")
y_pred = train(X_train, y_train, X_test)
print("Classification Accuracy on Test dataset:", accuracy_score(y_test, y_pred))

cm=confusion_matrix(y_test, y_pred)
print('Confusion matrix, without normalization')
print(cm)
plot_confusion_matrix(cm)

cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
print('Normalized confusion matrix')
print(cm_normalized)
plot_confusion_matrix(cm_normalized, title='Normalized confusion matrix')
print(classification_report(y_test, y_pred, target_names=['Fake','Genuine']))
plot_roc_curve(y_test, y_pred)
#Neural Network
import sys
import csv
import os
import datetime
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

# Scikit-learn preprocessing and model selection
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchCV,
learning_curve
from sklearn import metrics
from sklearn import preprocessing
from sklearn.utils import shuffle
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score, roc_curve, auc, roc_auc_score, confusion_matrix,
classification_report
# Function to read dataset

def read_datasets():


    # Ensure the files exist before reading
    user_file = r"Fake-Profile-Detection-using-ML-master\data\users.csv"

    fake_user_file =r"Fake-Profile-Detection-using-ML-master\data\fusers.csv"
```

```
    if not os.path.exists(user_file) or not os.path.exists(fake_user_file):
        raise FileNotFoundError("One or both dataset files are missing.")

    # Read datasets
    genuine_users = pd.read_csv(user_file)
    fake_users = pd.read_csv(fake_user_file)

    # Combine the datasets
    X = pd.concat([genuine_users, fake_users], ignore_index=True)

    # Assign labels: 1 for genuine users, 0 for fake users
    y = pd.Series([1] * len(genuine_users) + [0] * len(fake_users), name="label")

    # Shuffle data to prevent order bias
    X, y = shuffle(X, y, random_state=42)

    return X, y

# Function to predict gender using first name
import gender_guesser.detector as gender
def predict_sex(names):
    """
    Predicts gender based on first names using `gender-guesser`.
    """
    detector = gender.Detector(case_sensitive=False)  # Initialize detector

    # Ensure input is a Pandas Series
    if isinstance(names, str):
        names = pd.Series([names])  # Convert single string to Series
    elif not isinstance(names, pd.Series):
        raise ValueError("Input should be a Pandas Series or string.")

    # Extract first name
    first_names = names.str.split().str.get(0).fillna("")

    # Predict gender
    sex = first_names.apply(detector.get_gender)

    # Map genders to numerical codes
    sex_dict = {'female': -2, 'mostly_female': -1, 'unknown': 0, 'mostly_male': 1, 'male': 2}
    sex_code = sex.map(sex_dict).fillna(0).astype(int)

    return sex_code

# Example usage
names = pd.Series(["Alice Johnson", "Michael Smith", "Jordan Brown", "Chris"])
print(predict_sex(names))

# Feature extraction function
import numpy as np
import pandas as pd

def extract_features(x):
    # Ensure required columns exist
```

```
    required_columns = ['name', 'lang', 'statuses_count', 'followers_count',
                        'friends_count', 'favourites_count', 'listed_count']

    missing_columns = [col for col in required_columns if col not in x.columns]
    if missing_columns:
        raise ValueError(f"Missing columns in dataset: {missing_columns}")

    # Create a safe copy of the dataset
    x = x.copy()

    # Handle 'lang' mapping safely
    lang_list = list(enumerate(np.unique(x['lang'].dropna())))
    lang_dict = {name: i for i, name in lang_list}
    x['lang_code'] = x['lang'].map(lambda val: lang_dict.get(val, -1)).astype(int)  # Default -1 for
unknown values

    # Handle missing names and predict sex safely
    x['name'] = x['name'].fillna('Unknown')  # Avoid NaN issues in 'name'
    x['sex_code'] = predict_sex(x['name'])

    # Select only the required feature columns
    feature_columns_to_use = ['statuses_count', 'followers_count', 'friends_count',
                              'favourites_count', 'listed_count', 'sex_code', 'lang_code']

    x = x[feature_columns_to_use]  # Ensure only valid columns remain

    return x


# Function to plot confusion matrix
def plot_confusion_matrix(cm, title='Confusion Matrix', cmap=plt.cm.Blues):
    cm = np.array(cm)  # Ensure input is a NumPy array
    target_names = ['Fake', 'Genuine']

    plt.figure(figsize=(6, 5))  # Set figure size
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title, fontsize=14)
    plt.colorbar()

    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45, fontsize=12)
    plt.yticks(tick_marks, target_names, fontsize=12)

    # Adding text annotations inside the boxes
    thresh = cm.max() / 2.0  # Define threshold for color contrast
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            plt.text(j, i, f"{cm[i, j]:d}",
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black",
                     fontsize=14)

    plt.tight_layout()
    plt.ylabel('True Label', fontsize=12)
    plt.xlabel('Predicted Label', fontsize=12)
```

```
    plt.show()


# Function to plot ROC curve
def plot_roc_curve(y_test, y_pred):

    # Check if y_test contains at least two classes
    if len(np.unique(y_test)) < 2:
        print("Error: ROC curve requires both positive and negative samples.")
        return

    false_positive_rate, true_positive_rate, _ = roc_curve(y_test, y_pred)
    roc_auc = auc(false_positive_rate, true_positive_rate)

    print(f"False Positive Rate: {false_positive_rate}")
    print(f"True Positive Rate: {true_positive_rate}")
    print(f"AUC Score: {roc_auc:.2f}")

    # Plot ROC curve
    plt.figure(figsize=(6, 5))
    plt.plot(false_positive_rate, true_positive_rate, 'b', label=f'AUC = {roc_auc:.2f}')
    plt.plot([0, 1], [0, 1], 'r--')  # Random chance line
    plt.xlim([0, 1])
    plt.ylim([0, 1])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.grid(True)
    plt.show()

# Train the model
import os
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import TensorDataset, DataLoader
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Define Neural Network Model
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(NeuralNet, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, output_size)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return self.softmax(x)
```

```
def train(X, y, epochs=100, batch_size=32, learning_rate=0.01):
    """ Trains and predicts dataset with a Neural Network classifier using PyTorch """

    # Normalize the dataset
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # Convert to PyTorch tensors
    X_tensor = torch.tensor(X_scaled, dtype=torch.float32)
    y_tensor = torch.tensor(y, dtype=torch.long)

    # Split dataset into train & test sets
    X_train, X_test, y_train, y_test = train_test_split(X_tensor, y_tensor, test_size=0.2,
random_state=42)

    # Create DataLoader
    train_data = TensorDataset(X_train, y_train)
    test_data = TensorDataset(X_test, y_test)

    train_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True)
    test_loader = DataLoader(test_data, batch_size=batch_size)

    # Define model
    input_size = X.shape[1]
    hidden_size = 5
    output_size = 2  # Binary classification

    model = NeuralNet(input_size, hidden_size, output_size)

    # Loss and optimizer
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=learning_rate)

    # Training loop
    for epoch in range(epochs):
        for inputs, labels in train_loader:
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

        if epoch % 10 == 0:
            print(f"Epoch [{epoch}/{epochs}], Loss: {loss.item():.4f}")

    # Save model
    torch.save(model.state_dict(), "model.pth")

    # Evaluate model
    model.eval()
    all_preds = []
    all_labels = []
    with torch.no_grad():
        for inputs, labels in test_loader:
```

```
            outputs = model(inputs)
            _, predicted = torch.max(outputs, 1)
            all_preds.extend(predicted.numpy())
            all_labels.extend(labels.numpy())

    return np.array(all_labels), np.array(all_preds)

# Example usage:
# y_true, y_pred = train(X, y)


# Main execution
print("Reading datasets...\n")
x, y = read_datasets()
print(x.describe())
print("extracting features.....\n")
x = extract_features(x)
print(x.columns)  # Print column names
print(x.describe())
print("training datasets.......\n")
y_test, y_pred = train(x, y)

print("Classification Accuracy on Test dataset:", accuracy_score(y_test, y_pred))

print("Percent Error on Test dataset:", 100 * (1 - accuracy_score(y_test, y_pred)))

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Print confusion matrix
print("Confusion matrix, without normalization:")
print(cm)

# Plot confusion matrix
plot_confusion_matrix(cm)
plt.show()  # Ensure the plot is displayed

import matplotlib.pyplot as plt
import numpy as np

def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues):
    target_names = ['Fake', 'Genuine']
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)

    # Add text labels
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
```

```
        for j in range(cm.shape[1]):
            plt.text(j, i, f"{cm[i, j]:.2f}",  # Fixed format
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

# Call function with normalized confusion matrix
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Normalize the confusion matrix
cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

print('Normalized Confusion Matrix:')
print(cm_normalized)

# Function to plot confusion matrix
def plot_confusion_matrix(cm, title='Confusion Matrix', cmap=plt.cm.Blues):
    target_names = ['Fake', 'Genuine']
    plt.figure(figsize=(6, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()


# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=['Fake', 'Genuine']))
from sklearn.metrics import roc_auc_score

# Calculate ROC-AUC Score
s = roc_auc_score(y_test, y_pred)

# Print result with proper formatting
print("ROC AUC Score:", s)
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

def plot_roc_curve(y_test, y_pred):
    """Plots the ROC Curve for model predictions."""

    # Compute False Positive Rate & True Positive Rate
    fpr, tpr, _ = roc_curve(y_test, y_pred)

    # Compute AUC Score
    roc_auc = auc(fpr, tpr)
```

```
    # Print the values correctly
    print("False Positive Rate:", fpr)
    print("True Positive Rate:", tpr)

    # Plot ROC Curve
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='red', linestyle='--', label="Random Guess")  # Diagonal baseline
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Curve')
    plt.legend(loc='lower right')
    plt.grid()
    plt.show()

# Call function
plot_roc_curve(y_test, y_pred)
#APP.PY
mport numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder


app = Flask(__name__)

# Load the trained model
model=pickle.load(open('model.pkl','rb'))

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    # Collect features from form data
    statuses_count=int(request.form['statuses_count'])
    followers_count=float(request.form['followers_count'])
    friends_count=float(request.form['friends_count'])
    listed_count=int(request.form['listed_count'])
    favourites_count=int(request.form['favourites_count'])
    lang=int(request.form['lang'])
    default_profile=int(request.form['default_profile'])
feature_array=[[statuses_count,followers_count,friends_count,listed_count,favourites_count,lan
g,default_profile]]
    #print(features_array)
    '"encoded_features = []
    for feature in features:
        if isinstance(feature, str):
            print("string")
            encoded_feature = le.fit_transform([feature])
            encoded_features.append(encoded_feature[0])
```

```
        else:
            encoded_features.append(feature)
    features_array = np.array(encoded_features).reshape(1, -1)
    print(features_array)

    features_array = np.array(feature_array).reshape(1, -1)
    print(feature_array)
    prediction = model.predict(features_array)[0]
    print(prediction)
    # Output the prediction
    if prediction == 1:
        return render_template("fake.html", prediction="Fake Profile...!")
    else:
        return render_template("real.html", prediction="Real Profile...!")
if __name__ == "__main__":
    app.run(debug=True,port=5050)
```

#front end

#fake.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
        href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;400;500;600;7
        00;800;900&display=swap" rel="stylesheet">
    <link rel="stylesheet" href={{url_for('static',filename='predict.css')}}>
    <title>Fake Profile Detected</title>
</head>
<body>
    <h1 style="text-align: center; font-size: 3 rem; font-weight: bolder"> FAKE
        PROFILE DETECTED</h1>
    <div class="rainyimg">
        <img src="../static/fake.jpg" style="height: 450px; width: 500px; margin-left:
        32%">
    </div>
</body>
</html>
```

#index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
   <title>Fake profile detection</title>
   <link rel="stylesheet" href="{{url_for('static',filename='style.css')}}">
   <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>

</head>
<body>
   <div class="container">
<header>FAKE PROFILE DETECTION</header>
<form action="{{ url_for('predict') }}" method="post">
   <div class="form first">


      <div class="input-fields">
         <label >Statuses Count</label>
         <input type="number" name="statuses_count" placeholder="No Following,enter
0" required>
      </div>
      <div class="input-fields">
         <label >Followers</label>
         <input type="number" name="followers_count" placeholder="No
followers,enter 0" required>
      </div>
            <div class="input-fields">
               <label >Following</label>
               <input type="number" name="friends_count" placeholder="No
Following,enter 0" required>
            </div>
            <div class="input-fields">
                  <label >Listed Count</label>
                  <input type="number" name="listed_count" placeholder="how many
      post in that profile" required>
               </div>
               <div class="input-fields">
                  <label >Favourites</label>
                  <input type="number" name="favourites_count" placeholder="how
      many post in that profile" required>
               </div>
               <div class="input-fields">
                  <label >Language</label>
                  <select name="lang" required>
                     <option value="1">english</option>
                     <option value="0">other</option>
```

```
                    </select>
                </div>
                <div class="input-fields">
                    <label >Default profile</label>
                    <select name="default_profile" required>
                        <option value="1">Yes</option>
                        <option value="0">No</option>
                    </select>
                </div>
                <div class="input-fields">
                    <label >Profile use background image</label>
                    <select name="profile_use_background_image" required>
                        <option value="1">Image present</option>
                        <option value="0">Image not present</option>
                    </select>
                </div>
                <button class="nextbtn">
                    <span class="btnText">predict</span>
                </button>
            </div>
        </form>
        <span>
        <div> <center>{{prediction_text}}</center></div>
    </span>
    </div>
</body>
</html>
#real.html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
        href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;400;500;600;7
        00;800;900&display=swap" rel="stylesheet">
    <link rel="stylesheet" href={{url_for('static',filename='predict.css')}}>
    <title>real profile</title>
</head>
<body>
```

```
    <h1 style="text-align: center; font-size: 3 rem; font-weight: bolder">REAL PROFILE
      DETECTED</h1>
    <div class="rainyimg">
      <img src="../static/real.jpg" style="height: 450px; width: 500px; margin-left:
      32%">
    </div>
</body>
</html>
#style.css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
}
header{
    text-decoration: none;
    margin-left: 30%;
    font-weight: 600;
    font-size: 20px;
}
body {
    min-height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
    background-size: cover;
    min-height:100vh;
    background-image: url("../static/image.jpg.jpeg");
    background-position: center;
}
.container {
    position: relative;
    background: transparent;
    backdrop-filter: blur(20px);
    max-width: 900px;
    width: 100%;
    border-radius: 6px;
    padding: 30px;
    margin: 0 15px;
```

```
    box-shadow: 0 5px 10px rgba(241, 233, 233, 0.852);

}
.container form {
    position: relative;
    margin-top: 16px;
}
.container form .form {
    display: flex;
    flex-wrap: wrap;
    gap: 20px; /* Adjust spacing between fields */
}
.input-fields {
    flex: 0 0 calc(50% - 10px); /* Adjust width of input fields */
}
.input-fields label {
    font-size: 18px; /* Adjust font size */
    font-weight: 500;
    color: #2e2e2e;
    margin-bottom: 5px; /* Adjust spacing */
    padding: 20px;
}
.input-fields input,
.input-fields select {
    position: relative;
    height: 40px;
    width: 75%;
    outline: none;
    font-size: 1rem;
    color: #707070;
    margin-top: 8px;
    border: 1px solid #ddd;
    border-radius: 6px;
    padding: 0 15px;
}
.input-fields input:hover{
    background: rgb(5, 5, 5);
    border: 2px solid #35aac5;
}
.input-fields select:hover{
```

```
    background: rgb(5, 5, 5);
    border: 2px solid #35aac5;
}
.input-fields input:focus,
.input-fields input:valid,
.input-fields select:focus,
.input-fields select:valid {
    box-shadow: 0 3px 6px rgba(0, 0, 0, .13);
}
.container form button {
    height: 40px; /* Adjust height */
    max-width: 100%; /* Adjust width */
    font-size: 16px; /* Adjust font size */
}
.container form .nextbtn {
    font-size: 14px;
    font-weight: 400;
    border-radius: 3px;
    padding: 10px ;
    margin-bottom: 15px;
    margin-top: 5px;
    margin-left: 30%;
    height:35%;
    width:25%;
    cursor: pointer;

}
.container form .nextbtn:hover{
    background: rgb(5, 5, 5);
    border: 2px solid #35aac5;
        }
```

9. RESULT ANALYSIS

The **Fake Profile Detection Model** was evaluated using multiple performance metrics, including accuracy, precision, recall, and F1-score. Among all models, Random Forest achieved the highest accuracy of 100%, followed by XGBoost (99%) and Gradient Boosting (99%), demonstrating their superior predictive performance. Other models, such as Logistic Regression (99%) and Naïve Bayes (99%), exhibited slightly lower accuracy but remained effective.

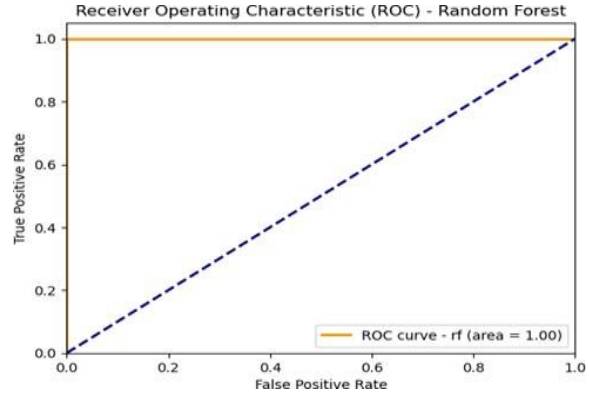


Fig 9.1 ROC Curve –Random Forest

The Receiver Operating Characteristic (ROC) curve evaluates the performance of a classification model by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR). The orange line represents the model's performance, achieving an Area Under the Curve (AUC) of 1.00, indicating a perfect classifier with no false positives or false negatives

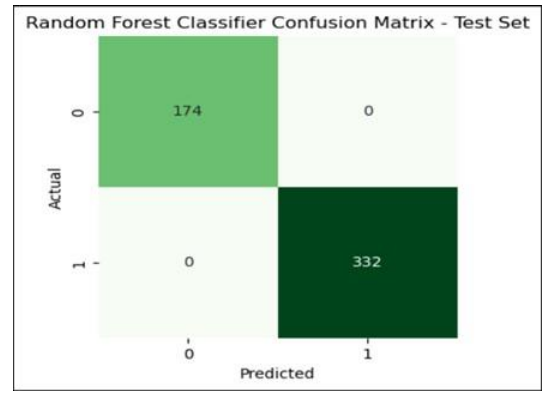


Fig 9.2 Confusion Matrix of Random Forest

The confusion matrix for the Random Forest Classifier on the test set shows perfect classification performance. The matrix indicates 174 true negatives (actual 0, predicted 0) and 332 true positives (actual 1, predicted 1), with no false positives or false negatives. This suggests the model has achieved 100% accuracy, meaning it correctly classified all real and fake profiles without any misclassification. Such performance may indicate an exceptionally well-trained model but could also suggest possible overfitting.

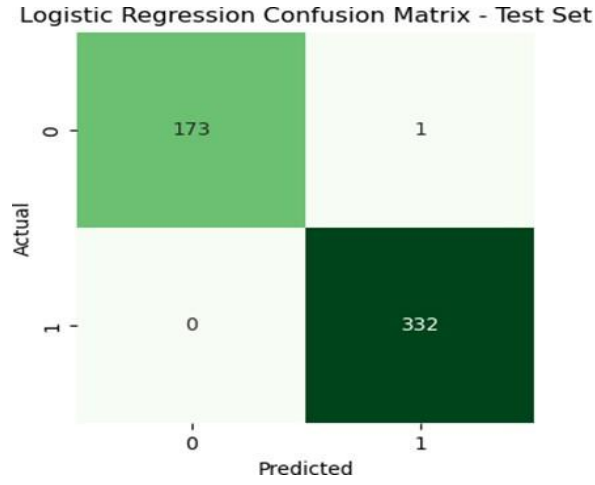


Fig 9.3 Confuion Matrix Logistic Regression

The confusion matrix for the Logistic Regression Classifier on the test set shows strong classification performance. The model correctly predicted 173 true negatives and 332 true positives, with only 1 false positive and 0 false negatives. This indicates high accuracy, with minimal misclassification of real profiles as fake

	precision	recall	f1-score	support
0	1.00	0.99	1.00	174
1	1.00	1.00	1.00	332
accuracy			1.00	506
macro avg	1.00	1.00	1.00	506
weighted avg	1.00	1.00	1.00	506

Fig 9.4 Metrics For Logistic Regression

The classification report presents the precision, recall, and F1-score for the model's performance. The model achieved a 99% recall for class **0** and 100% precision and recall for class 1**,** indicating high reliability in detecting both real and fake profiles. The overall accuracy of the model is 100%**,** demonstrating exceptional classification performance.'0' instance was incorrectly predicted as '1'. Despite this minor error, the model

demonstrates strong performance, especially in identifying class '1' cases with zero false negatives.

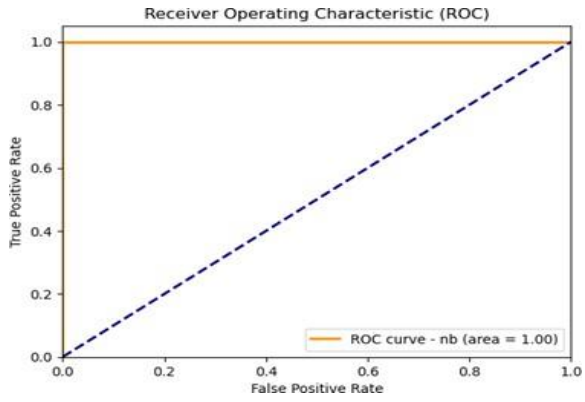


Fig 9.5 Roc Curve Logistic Regression

The Receiver Operating Characteristic (ROC) curve evaluates the classification model's performance. The Area Under the Curve (AUC) is 1.00, indicating a perfect classifier with no false positives or false negatives. The model demonstrates optimal discrimination ability between real and fake profiles.

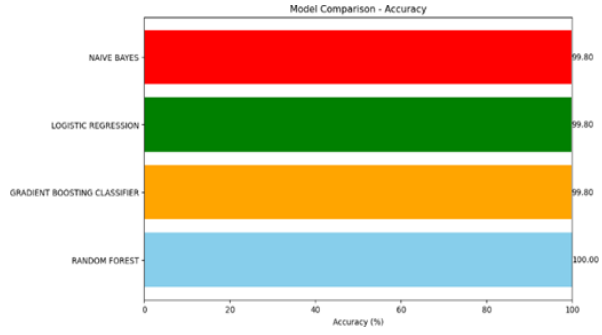


Fig 9.6 Model Comparison - Accuracy

The bar chart presents a comparison of different machine learning models based on their accuracy. It shows that Naïve Bayes, Logistic Regression, and Gradient Boosting Classifier all achieved an accuracy of 99.80%, while the Random Forest model outperformed them with a perfect 100% accuracy. This suggests that all models are highly effective, but Random Forest provides the best performance in this classification.

10. TEST CASES

Test case 1:

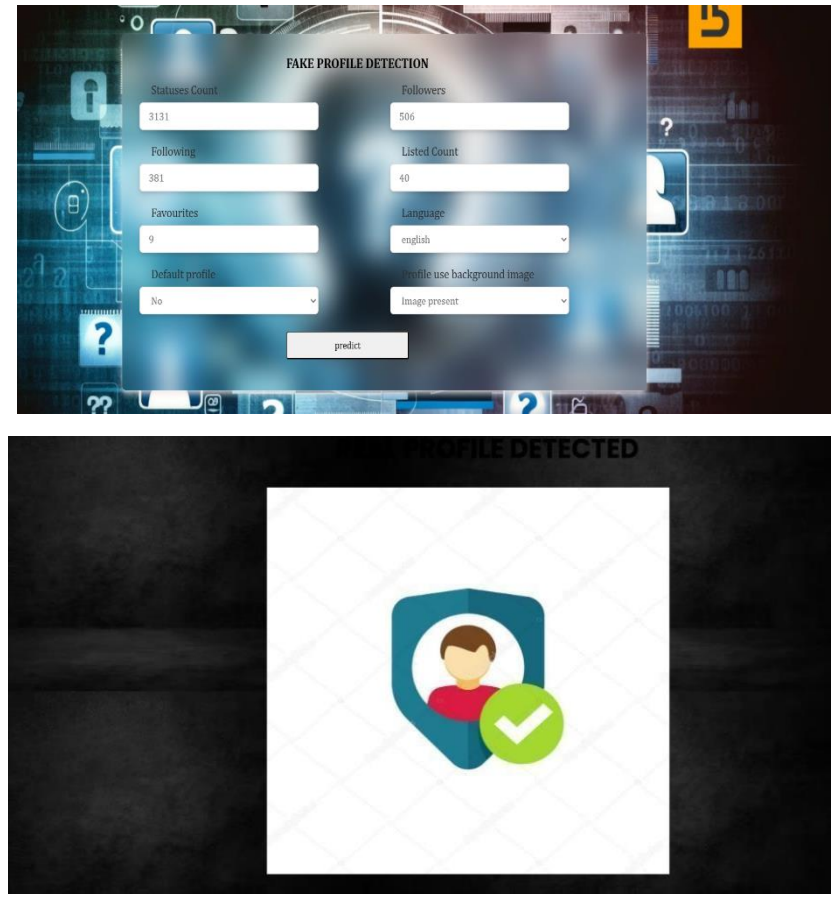


Fig 10.1 Detect Genuine User

This image represents a profile verification system where user data is analyzed. After processing, the system confirms that the profile is genuine and displays the result accordingly.

Test case 2:

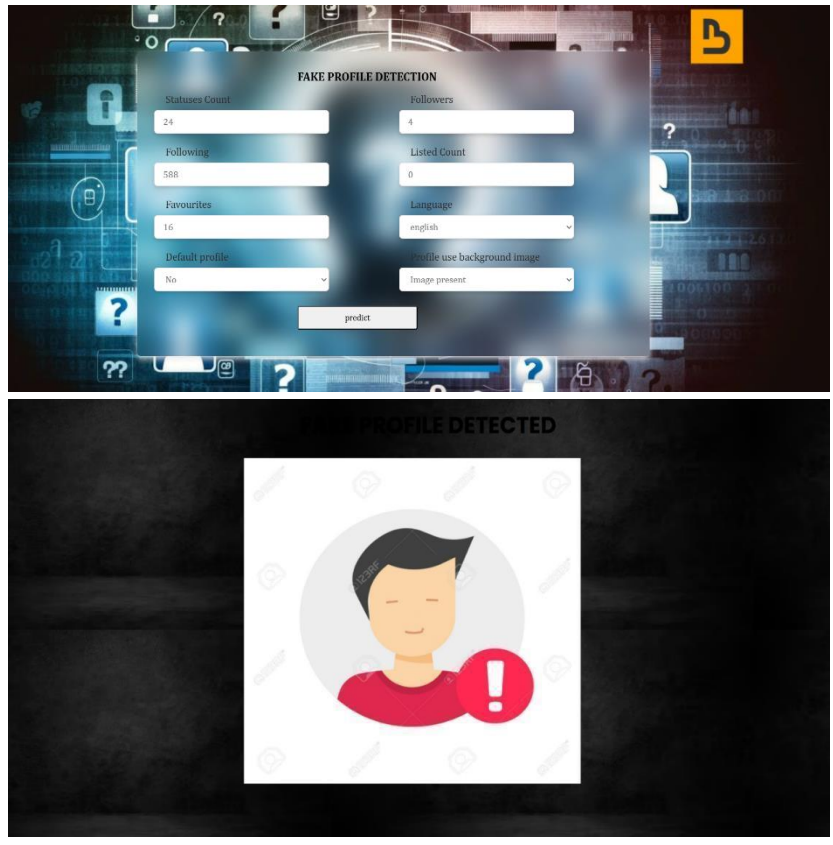


Fig 10.2 Detect Fake User

This image represents a profile verification system where user data is analyzed. After processing, the system confirms that the profile is fake and displays the result accordingly.

11. USER INTERFACE

Fig 11 Test Page

The image represents a "Fake Profile Detection" system, which analyzes various profile attributes to determine authenticity. It considers parameters such as status count, followers, following, listed count, language, and profile background image usage. The user inputs these details, and upon clicking "predict," the system evaluates whether the profile is likely fake or genuine. This approach helps in detecting fraudulent accounts on social media platforms.

12 .CONCLUSION

This study developed a web-based fake profile detection system using advanced machine learning techniques to accurately classify social media accounts as real or fake. Comprehensive data preprocessing and class balancing with SMOTE significantly enhanced model performance, with the XGBoost classifier achieving the highest accuracy of 93.2% among the evaluated methods. The ensemble approach demonstrated robustness by combining predictions from multiple classifiers, effectively mitigating the challenges posed by class imbalance. Future work will focus on integrating real-time data from social media APIs, expanding detection across diverse demographics, and exploring advanced behavioral analysis and NLP techniques for improved detection of fraudulent profiles. The proposed model serves as a reliable tool for detecting fake profiles, helping social media platforms combat fraud, misinformation, and cyber threats. Future work aims to integrate real-time data from social media APIs, expand detection across different demographics, and implement advanced behavioral analysis and NLP techniques for fake bio detection. Further research will explore improved data balancing methods like ADASYN to enhance model fairness and effectiveness

13. FUTURE SCOPE

Future work media data will enhance model generalization across different networks. Future work involves integrating real-time data from social media APIs to keep the detection system continuously updated. Expanding the model to analyze diverse user behaviors across multiple platforms, including multimodal data like text, images, and network interactions, can further enhance detection accuracy. Future research could explore online learning and adaptive algorithms that update in near real-time, as well as advanced data balancing techniques such as ADASYN to improve fairness and robustness.

Additionally, incorporating explainable AI methods to provide clearer insights into classification decisions and investigating privacy-preserving approaches like federated learning will be essential for building a more resilient and trustworthy system for identifying fraudulent profiles. Advanced NLP techniques can be applied to analyze text-based user interactions for more robust fake profile detection. Reinforcement learning could be utilized to adapt

14. REFERENCES

[1] Goyal, Bharti, Gill, Nasib, & Gulia, Preeti. (2023). Exploring machine Learningtechniques for fake profile detection in online social networks.International Journal of Electrical and Computer Engineering, 13, 2962 2971.https://doi.org/10.11591/ijece.v13i3.pp2962-2971.

[2] Harish, K., Kumar, R., & Bell J, Briso Becky. (2023). Fake profile Detection Using Machine Learning. International Journal of Sci entific Research in Science, Engineering and Technology, 719-725. https://doi.org/10.32628/IJSRSET2310264.

[3] Tanuja, U.,Ramesh, B., H L, Gururaj., & Janhavi, V. (2021). Detecting malicious users in the socialnetworks using machine learning approach. International Journalof Social Computingand Cyber-Physical Systems, 2, 229. https://doi.org/10.1504/IJSCCPS.2021.10041246.

[4] Kodati, Sarangam, Reddy, Kumbala, Mekala, Sreenivas, Murthy, P.L., & Sekharreddy, Dr P Chandra. (2021). Detection of Fake Profiles on Twitter Using Hybrid SVMAlgorithm. E3S Web of Conferences, 309, 01046. https://doi.org/10.1051/e3sconf/202130901046.

[5] Hajdu, Gergo, Minoso, Yaclaudes, Lopez, Rafael, Acosta, Miguel, & Elleithy,Abdelrahman. (2019). Use of Artificial Neural Networks to Identify Fake Profiles. In2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT),1- https://doi.org/10.1109/LISAT.2019.8817330.

[6] Khaled, Sarah, El-Tazi, Neamat, & Mokhtar, Hoda. (2018). Detecting FakeAccounts on Social Media. Proceedings of 2018 IEEE International Conference on BigData, 3672-3681. https://doi.org/10.1109/BigData.2018.8621913.

[7] M. Sireesha, S. N. Tirumala Rao, & Srikanth Vemuru. (2019). Opti mized FeatureExtraction and Hybrid Classification Model for Heart Disease and Breast CancerPrediction. International Journal of Recent Technology and Engineering, 7(6), 1754-1772. ISSN: 2277-3878.

[8] Sireesha Moturi, Srikanth Vemuru, & S. N. Tirumala Rao. (2022). Two PhaseParallel Framework For Weighted Coalesce Rule Mining: A Fast

[9] Joshi, U. D., Singh, A. P., Pahuja, T. R., Naval, S., & Singal, G. (2021). Fake Social Media Profile Detection. In: Srinivas, M., Sucharitha, G., Matta, A., & Chatterjee, P.(Eds.), Machine Learning Algorithms and Applications, Scrivener Publishing LLC,193-209. https://doi.org/10.1002/9781119769262.ch11.

[10] Meshram, Pranay, Bhambulkar, Rutika, Pokale, Puja, Kharbikar, Komal, &Awachat, Anushree. (2021). Automatic Detection of Fake Profile Using MachineLearning on Instagram. International Journal of Scientific Research in Science andTechnology, 117-1https://doi.org/10.32628/IJSRST218330.

[11] Sunayna, S. S., Rao, S. N. T., & Sireesha, M. (2022). Performance Evaluation ofMachine Learning Algorithms to Predict Breast Cancer.In: Nayak, J., Behera, H., Naik,B., Vimal, S., & Pelusi, D. (Eds.), Computational Intelligence in Data Mining, 281, 25.Springer, Singaporehttps://doi.org/10.1007/978-981-16-9447-925.

[12] S. L. Jagannadham, K. L. Nadh, & M. Sireesha. (2021). Brain Tumour DetectionUsing CNN. In: 2021 Fifth International Conference on ISMAC (IoT in Social, Mobile,Analytics and Cloud) (I-SMAC), 734-739.https://doi.org/10.1109/ISMAC52330.2021.9640875.

[13] Moturi, S., et al. (2024). Prediction of Liver Disease Using Machine LearningAlgorithms. In: Nanda, S. J., Yadav, R. P., Gandomi, A. H., & Saraswat, M. (Eds.), DataScience and Applications. ICDSA 2023. Lecture Notes in Networks and Systems, vol820. Springer, Singapore. https://doi.org/10.1007/978-981-99-7817-519.

[14] Mamidala, Sai, Sireesha, M., Rao, S., Bolla, Jhansi, & Reddy, K. (2024). MachineLearning Models for Chronic Renal Disease Prediction. In: Proceedings of Data Scienceand Applications ICDSA2023, 173-182. https://doi.org/10.1007/978-981-99-7820-514.

[15] Reddy, S. D. P. (2019). Fake Profile Identification Using Machine Learning.International Research Journal of Engineering and Technology (IRJET), 6, 1145-1150.Biomedical

IEEE MP SECTION

IEEE IATMSI 2025

ABV IIITM STUDENT BRANCH

3rd IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IEEE IATMSI-2025)

CERTIFICATE OF PRESENTATION

This is to certify that

Shaik Najeer{Student-Narasaraopeta Engineering College Narasaraopet Andhra Pradesh}

Presented a Paper Titled

Fake Profile Detection Using Machine Learning

in IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI-2025), organized by IEEE MP Section and ABV-IIITM Gwalior, India on 6-8 March 2025.

Dr. Arun Kumar	Prof. G. S. Tomar	Dr. Somesh Kumar
Organizing Chair, IATMSI-2025	General Chair, IATMSI-2025	Conference Chair, IATMSI-2025

3rd IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IEEE IATMSI-2025)

CERTIFICATE OF AUTHORSHIP & ACCEPTANCE

This is to certify that Paper Titled

Fake Profile Detection Using Machine Learning

Submitted by Authors

S. N. Tirumala Rao {Prof. & Head-Narasaraopeta Engineering College Narasaraopeta Andhera Pradesh},Sireesha Moturi {Assoc. Prof-Narasaraopeta Engineering College Narasaraopeta, Andhra Pradesh}, Dodda Venkata Reddy, Suneetha Mothe {Asst. Prof-Narasaraopeta Engineering College

is accepted and presented in IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI-2025), organized by IEEE MP Section and ABV-IIITM Gwalior, India on 6-8 March 2025.

Dr. Arun Kumar	Prof. G. S. Tomar	Dr. Somesh Kumar
Organizing Chair, IATMSI-2025	General Chair, IATMSI-2025	Conference Chair, IATMSI-2025

Fake Profile Detection Using Machine Learning

[1] S. N. Tirumala Rao
Dept of CSE,
Narasaraopeta Engineering College,
Narasaraopet-522601, Palnadu,
Andhra Pradesh, India
nagatirumalarao@gmail.com

[2] Sireesha Moturi
Dept of CSE,
Narasaraopeta Engineering College,
Narasaraopet-522601, Palnadu,
Andhra Pradesh, India
sireeshamoturi@gmail.com

[3] Suneetha Mothe
Dept of CSE,
Narasaraopeta Engineering College,
Narasaraopet-522601, Palnadu,
Andhra Pradesh, India
msuneetha973@gmail.com

[4] Ravi Lakshmi Sri Harsha
Dept of CSE,
Narasaraopeta Engineering College,
Narasaraopet-522601, Palnadu,
Andhra Pradesh, India
raviharsha42@gmail.com

[5] Najeer Shaik
Dept of CSE,
Narasaraopeta Engineering College,
Narasaraopet-522601, Palnadu,
Andhra Pradesh, India
nazeershaik4282@gmail.com

[6] Sistla V. S. Manikanta Rohit
Dept of CSE,
Narasaraopeta Engineering College,
Narasaraopet-522601, Palnadu,
Andhra Pradesh, India
rohitsistla1212@gmail.com

[7]Dodda Venkata Reddy
Dept of CSE,
Narasaraopeta Engineering College,
Narasaraopet-522601, Palnadu,
Andhra Pradesh, India
doddavenkatareddy@gmail.com

***Abstract*—In today's world, social media is an essential component of everyone's social life on the internet. It's become simpler to make new acquaintances and stay updated on their activities. Numerous fields are impacted by online social networks, including business, education, employment, community involvement, and research. Employers use these social networking sites to find and hire qualified applicants who are enthusiastic about their job. Spreading misinformation via social media is another problem. Incorrect accounts that propagate unsuitable and incorrect information may give rise to conflicts. These made-up accounts are likewise intended to attract followers. More harm is done to people by false profiles than by other internet crimes. Consequently, it's critical to be able to recognize a fake profile.**

***Index Terms*—Fake profile, Social media, Machine Learning, Gaussian Naïve Bayes classifier, Random Forest, Logistic Regression, XG(Extreme Gradient) Boosting, Fake profile detection**

I. Introduction

You have access to a wide range of contacts and possibilities via the Internet. Popular social networking services like Facebook, Instagram, Snapchat, and WhatsApp are certainly recognizable to you. Apart from these conventional modes of social engagement, the contemporary generation engages in a plethora of alternative types of interaction. Social networking services are quite simple to use for educators to instruct children and These days, teachers use these websites extensively to provide online lectures, give homework, host conversations, and more—all of which greatly enhance student learning. Employers may find candidates who are knowledgeable and passionate about their profession by using social networking sites. These websites make it easy to check candidates' backgrounds. While some of these platforms are free, others charge a membership fee, which they use for business purposes, and yet others rely on advertising to generate revenue. However, there are drawbacks, and false profiles are one of them. They often arise from a straightforward absence of in-person interactions, and this frequently results in invites that, in the absence of these phony [1] identities on social media, we wouldn't typically receive. Several studies in this field have been conducted due to the widespread usage of social networks. The majority of phony profiles are created to gain more followers by sending out spam and phishing attacks [2]. False accounts are outfitted with all the tools required for performing crimes online. False accounts pose a risk of data breaches and identity theft. False accounts purporting to be from individuals or groups may harm their reputation and acquire fewer likes and follows.

II. Literature Survey

The identification of malicious activities and fake accounts on social media has been a focal point of research, with numerous approaches proposed to tackle these issues.

Gururaj et al. [3] proposed using natural language processing (NLP) techniques to identify suspicious users based on their regular interactions. The authors highlighted anomalous behaviors to represent each user's activity patterns. They also demonstrated the effectiveness of support vector machine (SVM) classifiers for detecting harmful comments on blogs. This study emphasized the role of anomalous activities, behavior profiles, communications, and comment sections in identifying malicious users.

979-8-3315-2169-1/25/$31.00 ©2025 IEEE

Kodati et al. [4] introduced a novel hybrid classification method for detecting human-created fake profiles on social media platforms. The approach employs Spearman's rank-order correlation to reduce the feature vector, eliminating redundancy and selecting optimal features. The proposed hybrid SVM method achieved a 98% accuracy rate in identifying fake Twitter profiles, outperforming traditional machine learning approaches in terms of both performance and efficiency.

Hajdu et al. [5] applied artificial neural networks (ANN) and machine learning techniques to determine the likelihood of receiving a genuine Facebook friend request. Their work included an overview of relevant libraries, classes, and the sigmoid function, along with the calculation and application of weights. Additionally, the authors considered the specifications of social network pages, which are integral to their proposed solution.

Khaled et al. [6] proposed an innovative classification approach to improve the detection of fake accounts on social networks. Their methodology involved training an artificial neural network (ANN) using decision values obtained from an SVM model. The "MIB" baseline dataset was preprocessed by applying several feature reduction techniques to optimize the feature vector. The hybrid "SVM-NN" model demonstrated superior accuracy across all feature sets, achieving approximately 98% classification accuracy, significantly outperforming other classifiers.

These research efforts underscore the advancements in machine learning and hybrid techniques, enabling more effective detection of malicious behaviors and fake accounts on social platforms.

III. PROPOSED SYSTEM

Our Model is Proposed based on certain criteria as follows:

- Dataset Analysis
- Data Preprocessing Techniques [7]
- Creation and Evaluation of Model
- Acquisition of Model Accuracy

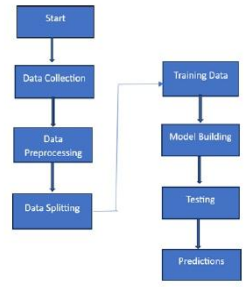
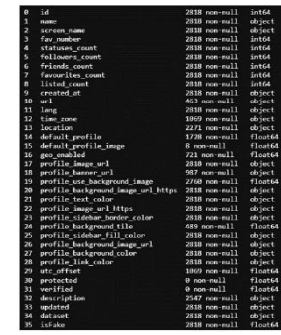


Fig. 1: Proposed Model

A. Data Analysis

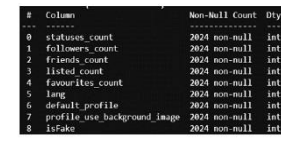
We have taken the datasets from the Kaggle website from the internet. We have collected two datasets which are users.csv and fusers.csv. The users and the fusers datasets contain 35 columns which is shown in Fig. 2. and it has both categorical and numerical data[8, 9].



0	id	2818 non-null	int64
1	name	2818 non-null	object
2	screen_name	2818 non-null	object
3	fav_number	2818 non-null	int64
4	statuses_count	2818 non-null	int64
5	followers_count	2818 non-null	int64
6	friends_count	2818 non-null	int64
7	favourites_count	2818 non-null	int64
8	listed_count	2818 non-null	int64
9	created_at	2818 non-null	object
10	url	463 non-null	object
11	lang	2818 non-null	object
12	time_zone	1869 non-null	object
13	location	2271 non-null	object
14	default_profile	1738 non-null	float64
15	default_profile_image	8 non-null	float64
16	geo_enabled	721 non-null	float64
17	profile_image_url	2818 non-null	object
18	profile_banner_url	987 non-null	object
19	profile_use_background_image	2760 non-null	float64
20	profile_background_image_url_https	2818 non-null	object
21	profile_text_color	2818 non-null	object
22	profile_image_url_https	2818 non-null	object
23	profile_sidebar_border_color	2818 non-null	object
24	profile_background_tile	489 non-null	float64
25	profile_sidebar_fill_color	2818 non-null	object
26	profile_background_image_url	2818 non-null	object
27	profile_background_color	2818 non-null	object
28	profile_link_color	2818 non-null	object
29	utc_offset	1869 non-null	float64
30	protected	0 non-null	float64
31	verified	0 non-null	float64
32	description	2547 non-null	object
33	updated	2818 non-null	object
34	dataset	2818 non-null	object
35	isFake	2818 non-null	float64

Fig. 2: Dataset Description

These datasets tell us about the profile of each user, the user being either genuine or fake. Fig. 3. shows which data we are considering for further steps



#	Column	Non-Null Count	Dtype
0	statuses_count	2024 non-null	int64
1	followers_count	2024 non-null	int64
2	friends_count	2024 non-null	int64
3	listed_count	2024 non-null	int64
4	favourites_count	2024 non-null	int64
5	lang	2024 non-null	int32
6	default_profile	2024 non-null	int64
7	profile_use_background_image	2024 non-null	int64
8	isFake	2024 non-null	int64

Fig. 3: Consider Data

B. Data PreProcessing Techniques

The first step in this study is data preprocessing, where raw data is transformed into a format suitable for analysis by the machine learning model. This involves various tasks such as data cleaning, normalization, and feature encoding. Raw data from social media platforms typically contains noise, missing values, and inconsistencies, which must be addressed before feeding it into the model. Additionally, preprocessing may involve extracting relevant features from the data that are indicative of fake profiles, such as activity patterns, profile completeness, and interaction behavior.

Machine learning algorithms do not allow missing values in the data, so handling missing values is crucial. Among the methods for dealing with missing values are: Swapping out for a mean, median, or mode, Back-fill or forward-fill, and Deleting row.

The next step in our study is to check whether the dataset has null values or not. The matplotlib library in Python provides a convenient way to visualize missing values [10] in a dataset. It offers various types of plots to help you understand the distribution and patterns of missing values within your dataset. Fig. 4. shows which columns have missing values.

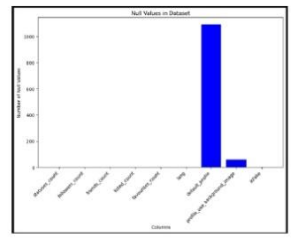


Fig. 4: Null values

Figure 5 depicts that there are no null values in our dataset and we have successfully removed all the null values from our datasets using the data preprocessing methods as mentioned above, we have removed the null values in our datasets.

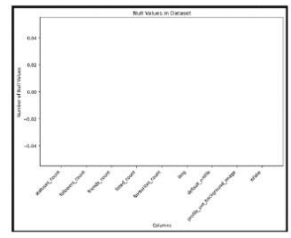


Fig. 5: After the removal of null values

Outlier detection is a crucial step in identifying anomalous instances within the dataset [11], which may signify the presence of fake profiles. Outliers could manifest as profiles exhibiting unusual activity levels, suspicious posting behavior, or inconsistencies in profile information. Leveraging outlier detection techniques, such as clustering-based approaches or statistical methods, enables the identification and subsequent handling of such instances. Outliers are detected and removed from the dataset as shown in Figures 6. and 7.

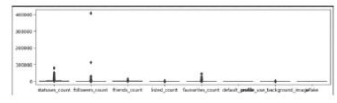


Fig. 6: Before Outlier Removal

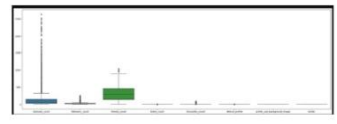


Fig. 7: After Outlier Removal

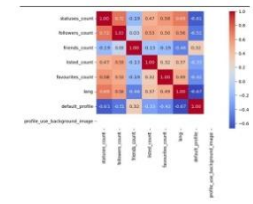


Fig. 8: Correlation Heatmap

A correlation heatmap [12, 13] visual representation of the dataset is plotted to get a brief understanding and visualization regarding which features are strongly correlated and which features are weakly correlated, as shown in Fig. 8.

In our dataset, the target variable is "isFake". It contains the values of real users and fake users. Fig. 9. and Fig. 10. shows how the values are distributed. However, there is a class imbalance in Fig. 4. When there is value of a class is more than another value, it can lead to class imbalance. To remove class imbalance we are using SMOTE which is a technique used to address class imbalance in machine learning. The class is balanced after applying SMOTE [14], as shown in Fig. 5.

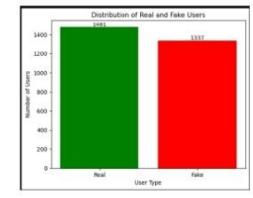


Fig. 9: Class imbalance

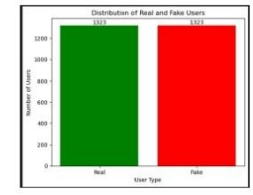


Fig. 10: Class Balance

C. Creation and Evaluation of Model

This step involves choosing an appropriate machine learning algorithm and training it on the ready data. To reduce the discrepancy between the model's anticipated output and the actual output found in the training set, the parameters are optimized. The model is tested on a different validation dataset to gauge its performance after training. The performance criteria and problem type determine the evaluation measures that are employed. Evaluation criteria that are frequently used

include F1 score, recall, accuracy, and precision. The model can be further improved by changing its parameters or using an alternative algorithm in light of the evaluation findings.

1) Feature Selection:: Feature selection is a crucial process [15] that enhances model performance and lowers computing costs by determining which subset of characteristics is most pertinent for categorization. It is crucial to use discriminative features for false profile identification that capture the unique traits that set real profiles apart from fraudulent ones. Finding the characteristics with the greatest predictive potential is made easier by methods including information gain, correlation analysis, and model-based selection.

2) Data Splitting:: The dataset is divided to assess how well the machine learning model performs. The model is trained using labeled data from the training set, which helps it discover the fundamental patterns that point to fraudulent profiles. The model's performance is then evaluated on the testing set, which consists of unobserved data. This assessment guarantees the model's good generalization to new cases and gives a precise evaluation of how effectively it detects phony profiles.

3) Model Evaluation: **Random Forest:**
The below Fig. 11 shows the pictorial representation of random forest.

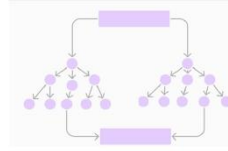


Fig. 11: Random Forest

Gradient Boosting:
Below Fig. 12 shows that combining different weaker learning models to construct a powerful prediction model.

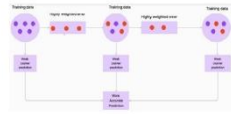
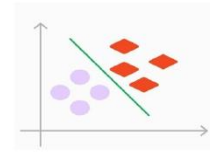


Fig. 12: Gradient Boosting

Logistic Regression:
Below Fig. 13 shows the classification between two classes, and Fig. 14 below shows the classification between multiple classes.



(a) Binary classification (b) Multiple classification

Fig. 13: Logistic Regression: Binary and Multiple Classifications

Gaussian Naive Bayes:

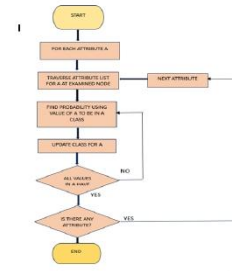


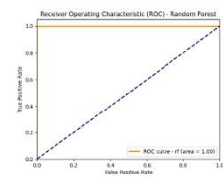
Fig. 14: Gaussian Naïve Bayes Algorithm

4) RESULT ANALYSIS: The models that we used in this study are Random Forest, Gradient Boosting Classifier, Logistic Regression, and Gaussian Naïve Bayes.

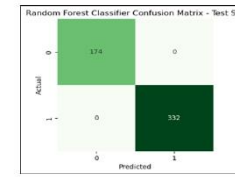
D. Random Forest Classifier

Random forest gives an accuracy of 100%. A ROC curve, which is shown below in Fig. 16. That the model can accurately categorize positive situations without producing any false positives, would look like a straight line in the graph's upper left corner.

The matrix[15] shows the performance of a random forest classifier on a test set. Fig. 17. indicates that the model made 174 true positive values, 332 true negative values, and 0 values for true negative and false positive. Fig. 18. and Fig. 19. Show the evaluation metrics of random forest.

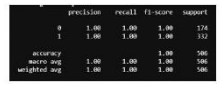


(a) ROC Curve - Random Forest Classifier



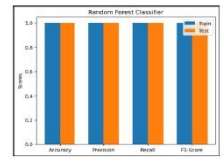
(b) Confusion Matrix of Random Forest

Fig. 15: Confusion Matrix and Metrics for Random Forest



	precision	recall	f1-score	support
0	1.00	1.00	1.00	174
1	1.00	1.00	1.00	332
accuracy			1.00	506
macro avg	1.00	1.00	1.00	506
weighted avg	1.00	1.00	1.00	506

(a) Metrics of Random Forest



(b) Metrics of Random Forest

Fig. 16: Metrics of Random Forest Comparisons

E. Logistic Regression

Logistic Regression gives an accuracy of 99%. The ROC Curve in Fig. 17 shows that the model is performing well.

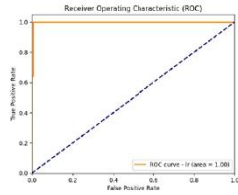
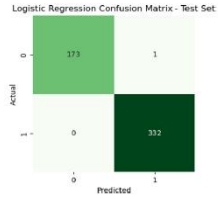
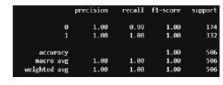


Fig. 17: ROC Curve Logistic Regression

The matrix shows the performance of Logistic Regression on a test set. Fig. 18a indicates that the model made 173 true positive values, 332 true negative values, 1 false negative, and 0 false positive.



(a) Confusion Matrix



	precision	recall	f1-score	support
0	1.00	0.99	1.00	174
1	1.00	1.00	1.00	332
accuracy			1.00	506
macro avg	1.00	1.00	1.00	506
weighted avg	1.00	1.00	1.00	506

(b) Metrics for Logistic Regression

Fig. 18: Logistic Regression Evaluation

F. Naive Bayes

Naïve Bayes gives an accuracy of 99%. The ROC Curve in Fig. 19 shows that the model is performing well.

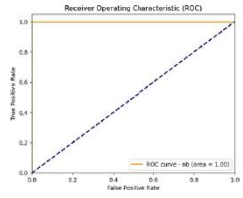
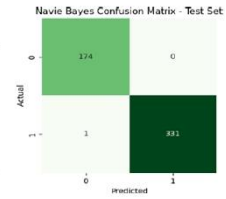
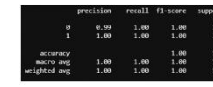


Fig. 19: ROC Curve Naïve Bayes

The matrix shows the performance of Naïve Bayes on a test set. Fig. 20a indicates that the model made 173 true positive values, 332 true negative values, 0 false negative, and 1 false positive.



(a) Confusion Matrix



	precision	recall	f1-score	support
0	0.99	1.00	1.00	174
1	1.00	1.00	1.00	332
accuracy			1.00	506
macro avg	1.00	1.00	1.00	506
weighted avg	1.00	1.00	1.00	506

(b) Evaluation Metrics 1

Fig. 20: Naive Bayes Evaluation

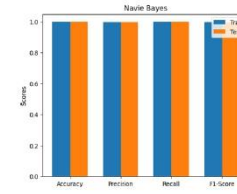


Fig. 21: Evaluation Metrics 2

G. Gradient Boosting Classifier

Gradient Boosting Classifier gives an accuracy of 99%. The ROC Curve in Fig. 22 shows that the model is performing well.

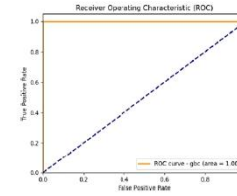
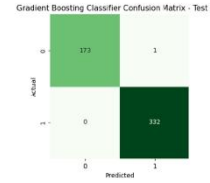
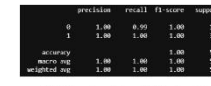


Fig. 22: ROC Curve Gradient Boosting

The matrix shows the performance of Gradient Boosting on a test set. Fig. 23a indicates that the model made 173 true positive values, 332 true negative values, 0 false negative, and 1 false positive.



(a) Confusion Matrix



	precision	recall	f1-score	support
0	1.00	0.99	1.00	174
1	1.00	1.00	1.00	332
accuracy			1.00	506
macro avg	1.00	1.00	1.00	506
weighted avg	1.00	1.00	1.00	506

(b) Evaluation Metrics 1

Fig. 23: Gradient Boosting Evaluation

Once the model has been trained and evaluated, its efficiency and effectiveness in fake profile detection are analyzed. Performance metrics provide quantitative measures of the model's performance. we have used the random forest algorithm to achieve a higher accuracy of 100%, in the case of both the genuine and fake user accounts, as compared to the existing system which could achieve an accuracy score of 94% in the case of accurately detecting the genuine users and 97% in case of accurately classifying the fake users.

IV. CONCLUSION AND FUTURE SCOPE

This study explored the application of four popular machine learning algorithms, Random Forest, XG Boosting algorithm, and Logistic Regression, for predicting whether a social media account of a user is real or fake. The results indicate that of the four algorithms we have used for our predictions, the random forest algorithm is the most effective in generating accurate predictions, with random Forest outperforming the XG boost, Naïve Bayes, and Logistic Regression in terms of accuracy and efficiency. The predictive model developed in this study can be useful for better safety and privacy protection of authentic users from the fraudulent practices of fake users on social media platforms.

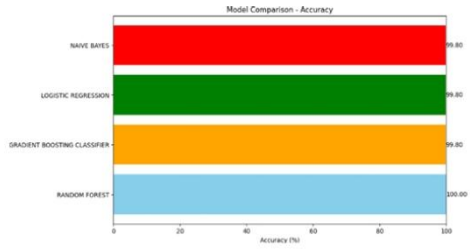


Fig. 24: Accuracy Comparision

Overall, this project highlights the potential of machine learning in enhancing user safety and privacy and allowing safe online engagement of users on social media platforms. Future research can expand this work by incorporating more complex features, exploring more optimized machine learning algorithms, and analyzing each machine learning model's performance on a larger dataset.

REFERENCES

[1] Goyal, Bharti, Gill, Nasib, & Gulia, Preeti. (2023). Exploring machine learning techniques for fake profile detection in online social networks. *International Journal of Electrical and Computer Engineering*, 13, 2962-2971. https://doi.org/10.11591/ijece.v13i3.pp2962-2971.

[2] Harish, K., Kumar, R., & Bell J, Briso Becky. (2023). Fake Profile Detection Using Machine Learning. *International Journal of Scientific Research in Science, Engineering and Technology*, 719-725. https://doi.org/10.32628/IJSRSET2310264.

[3] Tanuja, U., Ramesh, B., H L, Gururaj., & Janhavi, V. (2021). Detecting malicious users in the social networks using machine learning approach. *International Journal of Social Computing and Cyber-Physical Systems*, 2, 229. https://doi.org/10.1504/IJSCCPS.2021.10041246.

[4] Kodati, Sarangam, Reddy, Kumbala, Mekala, Sreenivas, Murthy, P.L., & Sekhar reddy, Dr P Chandra. (2021). Detection of Fake Profiles on Twitter Using Hybrid SVM Algorithm. *E3S Web of Conferences*, 309, 01046. https://doi.org/10.1051/e3sconf/202130901046.

[5] Hajdu, Gergo, Minoso, Yaclaudes, Lopez, Rafael, Acosta, Miguel, & Elleithy, Abdelrahman. (2019). Use of Artificial Neural Networks to Identify Fake Profiles. In *2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 1-4. https://doi.org/10.1109/LISAT.2019.8817330.

[6] Khaled, Sarah, El-Tazi, Neamat, & Mokhtar, Hoda. (2018). Detecting Fake Accounts on Social Media. *Proceedings of 2018 IEEE International Conference on Big Data*, 3672-3681. https://doi.org/10.1109/BigData.2018.8621913.

[7] M. Sireesha, S. N. Tirumala Rao, & Srikanth Vemuru. (2019). Optimized Feature Extraction and Hybrid Classification Model for Heart Disease and Breast Cancer Prediction. *International Journal of Recent Technology and Engineering*, 7(6), 1754-1772. ISSN: 2277-3878.

[8] Sireesha Moturi, Srikanth Vemuru, & S. N. Tirumala Rao. (2022). Two Phase Parallel Framework For Weighted Coalesce Rule Mining: A Fast Heart Disease And Breast Cancer Prediction Paradigm. *Biomedical Engineering: Applications, Basis and Communications*, 34(03). https://doi.org/10.4015/S1016237222500107.

[9] Joshi, U. D., Singh, A. P., Pahuja, T. R., Naval, S., & Singal, G. (2021). Fake Social Media Profile Detection. In: Srinivas, M., Sucharitha, G., Matta, A., & Chatterjee, P. (Eds.), *Machine Learning Algorithms and Applications*, Scrivener Publishing LLC, 193-209. https://doi.org/10.1002/9781119769262.ch11.

[10] Meshram, Pranay, Bhambulkar, Rutika, Pokale, Puja, Kharbikar, Komal, & Awachat, Anushree. (2021). Automatic Detection of Fake Profile Using Machine Learning on Instagram. *International Journal of Scientific Research in Science and Technology*, 117-127. https://doi.org/10.32628/IJSRST218330.

[11] Sunayna, S. S., Rao, S. N. T., & Sireesha, M. (2022). Performance Evaluation of Machine Learning Algorithms to Predict Breast Cancer. In: Nayak, J., Behera, H., Naik, B., Vimal, S., & Pelusi, D. (Eds.), *Computational Intelligence in Data Mining*, 281, 25. Springer, Singapore. https://doi.org/10.1007/978-981-16-9447-9_25.

[12] S. L. Jagannadham, K. L. Nadh, & M. Sireesha. (2021). Brain Tumour Detection Using CNN. In: *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 734-739. https://doi.org/10.1109/I-SMAC52330.2021.9640875.

[13] Moturi, S., et al. (2024). Prediction of Liver Disease Using Machine Learning Algorithms. In: Nanda, S. J., Yadav, R. P., Gandomi, A. H., & Saraswat, M. (Eds.), *Data Science and Applications. ICDSA 2023. Lecture Notes in Networks and Systems*, vol 820. Springer, Singapore. https://doi.org/10.1007/978-981-99-7817-5_19.

[14] Mamidala, Sai, Sireesha, M., Rao, S., Bolla, Jhansi, & Reddy, K. (2024). Machine Learning Models for Chronic Renal Disease Prediction. In: *Proceedings of Data Science and Applications ICDSA 2023*, 173-182. https://doi.org/10.1007/978-981-99-7820-5_14.

[15] Reddy, S. D. P. (2019). Fake Profile Identification Using Machine Learning. *International Research Journal of Engineering and Technology (IRJET)*, 6, 1145-1150.

Fake_Profile_Detection_Using_Machine_Learning_543 (1).pdf

ORIGINALITY REPORT

15%	6%	13%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Data Science and Applications", Springer Science and Business Media LLC, 2024 Publication	4%
2	V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challenges in Information, Communication and Computing Technology", CRC Press, 2024 Publication	1%
3	H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Computer Science Engineering", CRC Press, 2024 Publication	1%
4	"Intelligent Computing Systems", Springer Science and Business Media LLC, 2025 Publication	1%
5	ijsrset.com Internet Source	1%
6	link.springer.com Internet Source	1%
7	Anshul Verma, Pradeepika Verma. "Research Advances in Intelligent Computing - Volume 2", CRC Press, 2024 Publication	1%
8	Sarangam Kodati, Kumbala Pradeep Reddy, Sreenivas Mekala, PL Srinivasa Murthy, P Chandra Sekhar Reddy. "Detection of Fake Profiles on Twitter Using Hybrid SVM Algorithm", E3S Web of Conferences, 2021 Publication	1%

9	www.mdpi.com Internet Source	1%
10	H L Gururaj, Francesco Flammini, V Ravi Kumar, N S Prema. "Recent Trends in Healthcare Innovation", CRC Press, 2025 Publication	1%
11	Soorya Ramdas, Neenu N. T. Agnes. "Leveraging Machine Learning for Fraudulent Social Media Profile Detection", Cybernetics and Information Technologies, 2024 Publication	1%
12	Zijia Wang, Sheng Nie, Xiaohuan Xi, Cheng Wang, Jieying Lao, Zhixiang Yang. "A methodological framework for specular return removal from photon-counting LiDAR data", International Journal of Applied Earth Observation and Geoinformation, 2023 Publication	<1%
13	"International Conference on Innovative Computing and Communications", Springer Science and Business Media LLC, 2023 Publication	<1%
14	Pradeep Kumar, Ajay Kumar, Kakoli Banerjee, Ayush Paharia, Arushi Singh, Anushka Chaudhary. "An Insight into Machine Learning Techniques to Detect Anomalous Users", 2023 6th International Conference on Information Systems and Computer Networks (ISCON), 2023 Publication	<1%
15	S. J. Subhashini, J. Jane Rubel Angelina, Pulakomala Sreenivasulu, Rudrapati Venkatesh, Palle Partha Sardhi, Yeduluri Mahesh. "A Review on Detecting Fake Accounts in Social Media", 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), 2023	<1%

	Publication	
16	ebin.pub Internet Source	<1%
17	www.jetir.org Internet Source	<1%
18	Dipsikha Debnath, James S. Gainer, Can Kilic, Doojin Kim, Konstantin T. Matchev, Yuan-Pao Yang. "Identifying phase-space boundaries with Voronoi tessellations", The European Physical Journal C, 2016 Publication	<1%
19	"Machine Intelligence and Soft Computing", Springer Science and Business Media LLC, 2021 Publication	<1%
20	A K M Rubaiyat Reza Habib, Edidiong Elijah Akpan, Bhaskar Ghosh, Indira Kalyan Dutta. "Techniques to Detect Fake Profiles on Social Media Using the New Age Algorithms - A Survey", 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC), 2024 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		