

An Industry-Oriented Mini Project Report
On
**“Detecting Fake Profiles on Social Networks using Machine
Learning and NLP”**

Submitted in Partial Fulfillment of the Academic Requirement for
the Award of Degree

BACHELOR OF TECHNOLOGY
in
Computer Science and Engineering (Artificial Intelligence and Machine learning)

Submitted By:

J.JASHNAVI	(22R01A6618)
A.SAHITHYA	(22R01A6667)
R.LIKITHA	(22R01A66J0)

Under the esteemed guidance of

Dr.S.Dhanalakshmi
Professor,CSE(AI&ML)



CMR INSTITUTE OF TECHNOLOGY
(UGCAUTONOMOUS)

Approved by AICTE, Affiliated to JNTUH, Accredited by NAAC with A+ Grade,

Kandlakoya (V), Medchal Dist - 501 401

www.cmrihyderabad.edu.in

2024-25

CMR INSTITUTE OF TECHNOLOGY

(UGCAUTONOMOUS)

Approved by AICTE, to JNTUH, accredited by NAAC with A+ Grade,

Kandlakoya(V), Medchal Dist-501 401

www.cmrihyderabad.edu.in



CERTIFICATE

This is to certify that an Industry oriented Mini Project entitled with “Project Title” is being submitted by:

J.JASHNAVI	(22R01A6618)
A.SAHITHYA	(22R01A6667)
R.LIKITHA	(22R01A66J0)

To JNTUH, Hyderabad, in partial fulfillment of the requirement for award of the degree of B. Tech in CSE (AI&ML) and is a record of a Bonafide work carried out under our guidance and supervision. The results in this project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other University forward of any other degree or diploma.

Signature of Guide

Dr.S.Dhanalakshmi
(Professor)
(CSE(AI&ML))

Signature of Project Coordinator

Dr.S.Dhanalakshmi
(Professor)
(CSE(AI&ML))

Signature of HOD

Prof.P.Pavan Kumar
(Head of Department)
(CSE(AI&ML))

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M Janga Reddy**, Director, **Dr. G. Madhusudhana Rao**, Principal and **Prof. P. Pavan Kumar**, Head of Department, Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to, **Dr. S. Dhanalakshmi**, Professor Industry Oriented Mini Project Coordinator and internal guide **Dr. S. Dhanalakshmi**, Professor Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), CMR Institute of Technology and our external team from the Industry for their constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for every step towards success.

J.JASHNAVI	(22R01A6618)
A.SAHITHYA	(22R01A6667)
R.LIKITHA	(22R01A66J0)

ABSTRACT

Social networking sites have become an integral part of modern life, with millions of users creating profiles and engaging in interactions daily, regardless of time or location. These platforms offer various benefits, including real-time communication, information sharing, and social connectivity. However, they also come with significant risks, particularly concerning user privacy and security. One of the most prominent issues is the proliferation of fake profiles.

These deceptive accounts are often created for malicious purposes, including phishing, identity theft, misinformation campaigns, cyberbullying, and online fraud. Such activities can severely impact genuine users and undermine the trust and safety of social media environments. To tackle this growing problem, it is essential to classify user profiles accurately, distinguishing between authentic and fake accounts. Traditional methods of detection have proven helpful to some extent, but their accuracy and effectiveness remain limited due to the evolving strategies of attackers. In this paper, we propose a method that combines machine learning techniques with natural language processing (NLP) to enhance the accuracy of fake profile detection.

We utilize algorithms such as Support Vector Machine (SVM) and Naïve Bayes, known for their efficiency in classification tasks. NLP techniques are used to preprocess and analyze textual content from user profiles, allowing the extraction of meaningful features. By integrating machine learning with NLP, our approach significantly improves the precision of identifying fake profiles, offering a more robust and scalable solution. This method contributes to strengthening user trust, protecting personal information, and maintaining a safer social networking environment.

INDEX

ACKNOWLEDGEMENT	iii
ABSTRACT	iv
INDEX	v
LIST OF FIGURES	vi
LIST OF SCREENSHOTS	vii
1.INTRODUCTION	8
1.1 ABOUT PROJECT	8
1.2 EXISTING SYSTEM	9-10
1.3 PROPOSED SYSTEM	10-12
2.REQUIREMENT SPECIFICATIONS	13
2.1 REQUIREMENT ANALYSIS	13-14
2.2 SPECIFICATION PRINCIPLES	14-15
3.SYSTEM DESIGN	16
3.1 ARCHITECTURE/BLOCKDIAGRAM	16
3.2 UML DIAGRAMS	17-19
4.IMPLEMENTATION	20
4.1 PROJECTMODULES	20
4.2 ALGORITHMS	21-22
4.3 SAMPLECODE	23-32
5.TESTING	33
5.1 TESTINGMETHODS	33-35
6.RESULTS	36
7.CONCLUSION	40
8.REFERENCES	41

LIST OF FIGURES

Figure No.	Figure Particulars	Page No.
3.1	Architecture	16
3.2.1	UseCase Diagram	17
3.2.2	Class Diagram	18
3.2.3	Sequence Diagram	19

LISTOFSCREENSHOTS

Screenshot No.	Particular	Page No.
6.1	User page	38
6.2	Login page	39
6.3	Register form	39
6.4	Users Remote data	40
6.5	Output form	40
6.6	Sample Output form	41

1. INTRODUCTION

1.1 ABOUT PROJECT

Social networking has end up a well-known recreation within the web at present, attracting hundreds of thousands of users, spending billions of minutes on such services. Online Social network (OSN) services variety from social interactions-based platforms similar to Face book or MySpace, to understanding dissemination-centric platforms reminiscent of twitter or Google Buzz, to Social interaction characteristic brought to present systems such as Flickr. The opposite hand, enhancing security concerns and protecting the OSN privateness still signify a most important bottleneck and viewed mission.

When making use of Social network's (SN's), one of a kind men and women share one-of-a-kind quantities of their private understanding. Having our individual know-how entirely or in part uncovered to the general public, makes us excellent targets for unique types of assaults, the worst of which could be identification theft. Identity theft happens when any individual uses character's expertise for a private attain or purpose. During the earlier years, online identification theft has been a primary problem considering it affected millions of people's worldwide. Victims of identification theft may suffer unique types of penalties; for illustration, they would lose time/cash, get dispatched to reformatory, get their public image ruined, or have their relationships with associates and loved ones damaged.

At present, the vast majority of SN's does no longer verifies ordinary users' debts and has very susceptible privateness and safety policies. In fact, most SN's applications default their settings to minimal privateness; and consequently, SN's became a best platform for fraud and abuse. Social Networking offerings have facilitated identity theft and Impersonation attacks for serious as good as naïve attackers. To make things worse, users are required to furnish correct understanding to set up an account in Social Networking web sites. Easy monitoring of what customers share on-line would lead to catastrophic losses, let alone, if such bills had been hacked.

Profile information in online networks will also be static or dynamic. The details which can be supplied with the aid of the person on the time of profile creation is known as static knowledge, the place as the small print that are recounted with the aid of the system within the network is called dynamic knowledge. Static knowledge includes demographic elements of a person and his/her interests and dynamic knowledge includes person runtime habits and locality in the network. Are many of the instances utilized by false profiles on social networking sites. False profiles are the profiles which are not specific i.e. They're the profiles of men and women with false credentials. The false Face book profiles more commonly are indulged in malicious and undesirable activities, causing problems to the social community customers.

1.2 EXISTING SYSTEM

Chai et al awarded on this paper is a proof-of inspiration gain knowledge of. Even though the prototype approach has employed most effective normal systems in normal language processing and human-pc interplay, the results realized from the user trying out are significant. By using comparing this simple prototype approach with a wholly deployed menu procedure, they've discovered that users, principally beginner users, strongly pick the common language dialog-based approach. They have additionally learned that in an ecommerce environment sophistication in dialog administration is most important than the potential to manage complex typical language sentences.

In addition, to provide effortless access to knowledge on ecommerce web sites, natural language dialog-based navigation and menu-pushed navigation should be intelligently combined to meet person's one-of-a-kind wants. Not too long ago, they have got accomplished development of a new iteration of the approach that includes enormous enhancements in language processing, dialog administration and information management. They believed that average language informal interfaces present powerful personalized alternatives to conventional menupushed or search-based interfaces to web sites.

LinkedIn is greatly preferred through the folks who're in the authentic occupations. With the speedy development of social networks, persons are likely to misuse them for unethical and illegal conducts. Creation of a false profile turns into such adversary outcomes which is intricate to identify without apt research. The current solutions which were virtually developed and theorized to resolve this contention, mainly viewed the traits and the social network ties of the person's social profile. The limited publicly available profile data of LinkedIn makes it ineligible in making use of the existing tactics in fake profile identification. For that reason, there is to conduct distinctive study on deciding on systems for fake profile identification in LinkedIn. Shalinda Adikari and Kaushik Dutta researched and identified the minimal set of profile data that are crucial for picking out false profiles in LinkedIn and labeled the appropriate knowledge mining procedure for such project.

Z.Halim et al. Proposed spatio-temporal mining on social network to determine circle of customers concerned in malicious events with the support of latent semantic analysis. Then compare the results comprised of spatio temporal co incidence with that of original organization/ties with in social network, which could be very encouraging as the organization generated by spatio-temporal co-prevalence and actual one are very nearly each other.Total, scan indicate that Latent Semantic Indexing participate in very good for picking out malicious contents, if the feature set is competently chosen. One obvious quandary of this technique is how users pick their function set and the way rich it's. If the characteristic set is very small then most of the malicious content material will not be traced. However, the bigger person function set, better the performance won.

3.2 Disadvantages

- ❖ The system is not implemented Learning Algorithms like SVM, Naive Bayes.
- ❖ The system is not implemented any the problems involving social networking like privacy, online bullying, misuse, and trolling and many others.

1.3 PROPOSED SYSTEM

In this paper, we present a system that leverages machine learning combined with natural language processing (NLP) techniques to effectively detect fake profiles on online social networks. The proliferation of fake accounts poses significant challenges to the security and integrity of these platforms, leading to identity theft, misinformation, and various malicious activities. To enhance the accuracy of fake profile detection, we integrate two widely used classifiers: Support Vector Machine (SVM) and Naïve Bayes. The Support Vector Machine (SVM) algorithm is a powerful supervised learning method that classifies data by finding the optimal hyperplane that separates data points belonging to different classes. The goal of SVM is to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class. These critical data points are called support vectors, as they directly influence the position and orientation of the separating hyperplane. By maximizing the margin, SVM achieves better generalization on unseen data, making it highly effective for distinguishing between genuine and fake profiles based on extracted features from user behavior and profile attributes.

Naive Bayes is a probabilistic classifier that assigns a probability to each profile belonging to a certain class, based on the observed features. It is termed “naïve” because it assumes feature independence, meaning it treats each feature as contributing separately to the classification decision, regardless of any potential dependencies among them. For instance, when analyzing fake profiles, features such as posting time, language used, date of publication, and geographic location might influence each other, but Naïve Bayes simplifies this relationship.

Despite this assumption, Naïve Bayes often performs well, especially when dealing with large datasets, due to its simplicity and computational efficiency. By combining these two algorithms with NLP preprocessing techniques—which clean and extract meaningful textual features from user-generated content—the proposed system improves detection accuracy and robustness. This hybrid approach allows the system to analyze both linguistic patterns and behavioral data, resulting in a more reliable identification of fake profiles across social networks, ultimately contributing to safer online environments.

In recent years, the widespread use of social networking platforms has transformed the way people connect, share information, and communicate globally. However, this growth has also led to a surge in fake profiles, which pose significant risks to users and the integrity of these networks. Fake profiles are often used for malicious purposes such as identity theft, spreading misinformation, phishing, and manipulating public opinion. Therefore, detecting and eliminating such accounts is critical for maintaining a secure and trustworthy social networking environment. To address this challenge, we propose a system that combines machine learning and natural language processing (NLP) techniques to detect fake profiles with high accuracy. Specifically, our approach integrates two popular classifiers: Support Vector Machine (SVM) and Naïve Bayes, both of which bring unique strengths to the problem.

Support Vector Machine is a robust classification algorithm that excels in handling high-dimensional data. It works by identifying the best decision boundary, known as the hyperplane, which separates different classes in the feature space. The main objective of SVM is to maximize the margin between the hyperplane and the nearest data points from each class, referred to as support vectors. These support vectors are essential because they define the boundary and directly affect the model's ability to generalize to new data. By focusing on maximizing the margin, SVM reduces classification errors and improves performance when distinguishing between real and fake user profiles.

On the other hand, the Naïve Bayes classifier is based on Bayesian probability theory and applies a straightforward assumption that all features are independent of one another. Despite this simplification, Naïve Bayes is often surprisingly effective, especially in text classification tasks, which makes it suitable for analyzing the textual content and metadata associated with social network profiles. For example, features such as the timing of posts, language patterns, geolocation tags, and user activity can all contribute independently to predicting whether a profile is fake. Naïve Bayes calculates the probability of a profile belonging to either the fake or genuine category based on these features, enabling quick and efficient classification.

The combination of SVM and Naïve Bayes classifiers provides a complementary approach that leverages both geometric separation of data points and probabilistic inference. Before classification, natural language processing techniques are applied to preprocess and extract relevant features from user profiles and their activity data. NLP helps in cleaning the data, removing noise, and identifying meaningful patterns in user-generated text such as posts, comments, and profile descriptions. Our proposed system benefits from this multi-faceted approach by improving detection accuracy and robustness against evolving tactics used by fake profile creators. In conclusion, the integration of machine learning algorithms like SVM and Naïve Bayes with NLP preprocessing offers a powerful tool for combating fake profiles on social networks. This system not only enhances security by accurately identifying fraudulent accounts but also helps preserve user trust and platform integrity. As social networks continue to grow, such intelligent detection mechanisms will become increasingly vital for maintaining safe and authentic online communities.

3.4 Advantages

- In the proposed system, Profile information in online networks will also be static or dynamic. The details which can be supplied with the aid of the person on the time of profile creation is known as static knowledge, the place as the small print that are recounted with the aid of the system within the network is called dynamic knowledge.
- In the proposed system, Social Networking offerings have facilitated identity theft and Impersonation attacks for serious as good as naïve attackers.
- By using NLP, the system automatically extracts meaningful textual features from user profiles and posts, which reduces the need for manual feature selection and speeds up the analysis process. The inclusion of dynamic profile data allows for real-time monitoring of user behavior, enabling the system to detect suspicious activities promptly.

2.REQUIREMENT SPECIFICATIONS

2.1 REQUIREMENT ANALYSIS

1.Functional Requirements

These define what the system should do:

- **Fake Profile Detection:** The system must identify fake profiles on social media platforms using machine learning (SVM and Naive Bayes) and NLP techniques.
- **Data Classification:** Classify social network profiles as genuine or fake based on static (user input) and dynamic (network behavior) data.
- **Profile Data Analysis:**
 - Use features like post timing, language, and location.
 - Treat static (profile info) and dynamic (behavioral activity) data differently.
- **Use of ML Algorithms:**
 - **Support Vector Machine (SVM)** to find the optimal hyperplane for classifying data points.
 - **Naive Bayes** as a probabilistic classifier assuming feature independence.

2.Non-Functional Requirements

These specify how the system should behave:

- **Accuracy:** Improved accuracy over existing systems in detecting fake profiles.
- **Performance:** Should work efficiently even with limited available profile data (e.g., in platforms like LinkedIn).
- **Scalability:** Capable of handling increasing numbers of user profiles.
- **Security:** Should handle privacy and impersonation threats properly.

3.System Requirements

a. Hardware Requirements

- **Processor:** Pentium-IV or higher
- **RAM:** Minimum 4 GB
- **Hard Disk:** Minimum 20 GB
- **Input Devices:** Standard Windows keyboard, 2/3 button mouse
- **Monitor:** SVGA display

b. Software Requirements

- **Operating System:** Windows 7 Ultimate
- **Coding Language:** Python
- **Front-End:** Python with HTML, CSS, JavaScript
- **Back-End:** Django ORM
- **Database:** MySQL (via WAMP Server)

2.2 SPECIFICATION PRINCIPLES

1. Correctness

- All listed hardware and software specifications must accurately support the project's goals, such as running a Python-based ML/NLP application.
- Example: Windows 7 Ultimate and Python are compatible; Django ORM and MySQL work together reliably.

2. Clarity

- Each requirement must be stated clearly and unambiguously.
- Example: "RAM: Minimum 4 GB" is better than "enough RAM".

3. Completeness

- The specification should cover all necessary components—hardware, software, front-end, back-end, and database.
- It includes both **static** (e.g., profile data) and **dynamic** (e.g., activity logs) aspects of the system.

4. Verifiability

- Requirements must be testable and measurable.
- Example: "Pentium IV or higher" and "MySQL via WAMP Server" are specific enough to verify during setup.

5. Feasibility

- The specified configuration must be achievable with current technology and available resources.
- Example: All components can be run on a mid-range machine, avoiding the need for high-end infrastructure.

6. Traceability

- Every requirement should be linked back to a project goal or feature (e.g., fake profile detection, ML classification).
- Example: Python and Django are chosen because the algorithms (SVM, Naive Bayes) will be implemented in Python.

7. Consistency

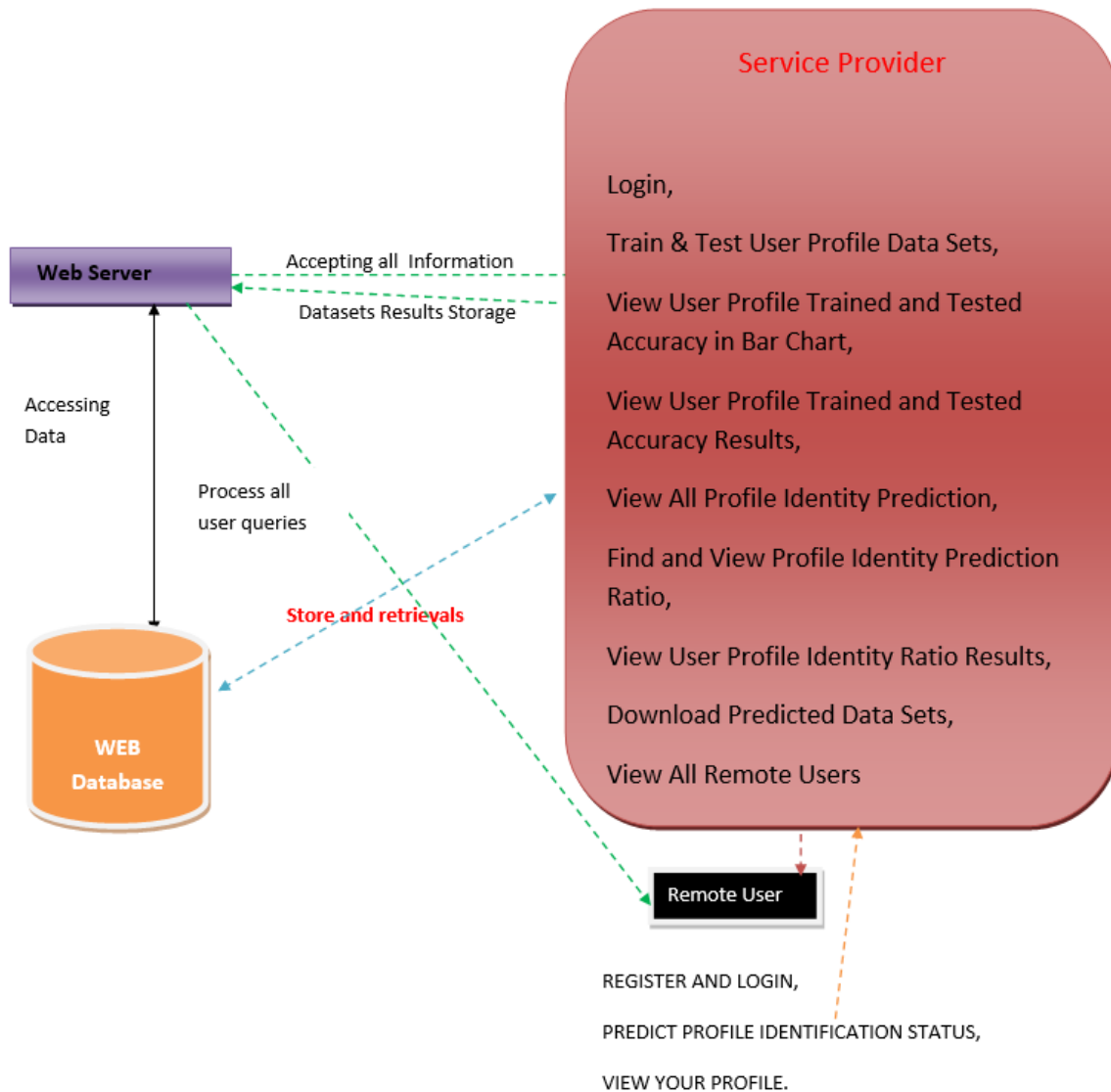
- There should be no conflicts between hardware and software requirements.
- Example: Ensuring Django version used is compatible with MySQL version provided by WAMP.

8. Modifiability

- Specifications should be structured in a way that allows for easy updates.
- Example: If the project needs to move from MySQL to PostgreSQL, changes should require minimal effort.

3.SYSTEM DESIGN

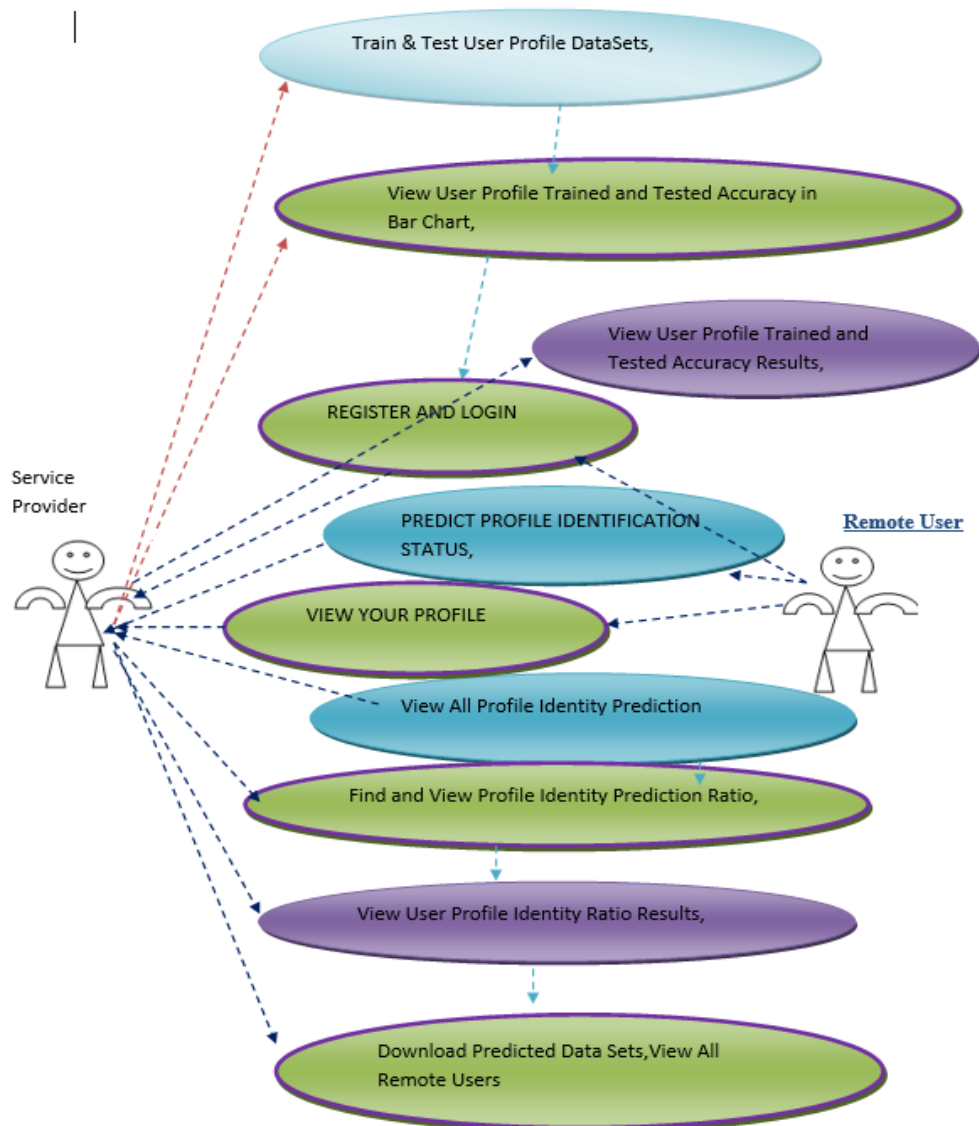
3.1 ARCHITECTURE/BLOCKDIAGRAM



3.2 UML DIAGRAMS

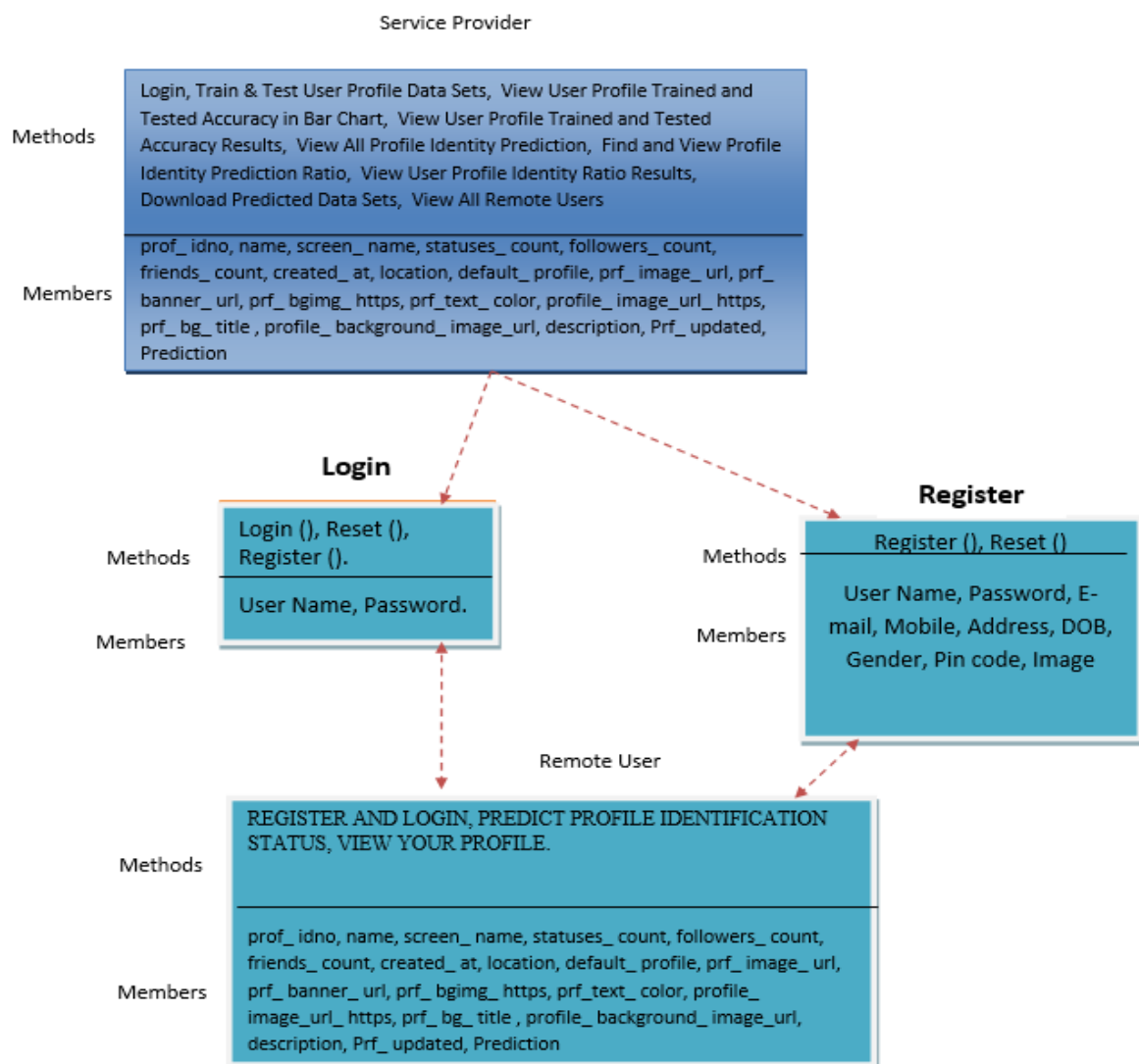
3.2.1. Use case diagram:

The use case diagram illustrates interactions between users and the fake profile detection system. Key actors include administrators and social network users. Use cases involve user profile submission, data analysis, fake profile classification, and result display. The system automates detection using machine learning models like SVM and Naïve Bayes.



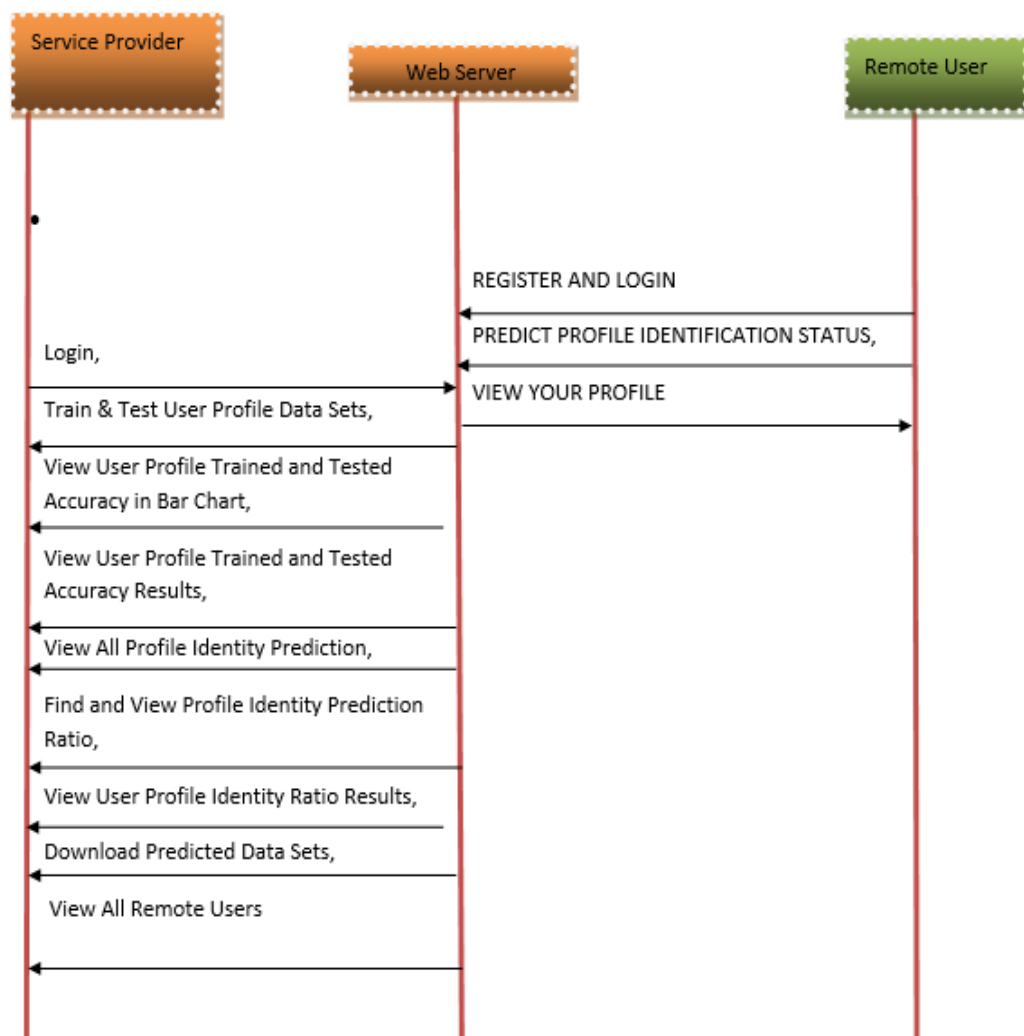
3.2.2 Class Diagram :

The class diagram represents the system's structure, showing classes such as UserProfile, FeatureExtractor, Classifier, and ResultManager. UserProfile holds profile data, while FeatureExtractor processes static and dynamic features. Classifier contains SVM and Naïve Bayes logic. ResultManager stores and displays classification outcomes. Relationships reflect inheritance, associations, and data dependencies.



3.2.3 Sequence Diagram :

The sequence diagram captures the flow of fake profile detection. A user submits profile data, triggering feature extraction. The system invokes classifiers (SVM/Naïve Bayes), processes results, and updates the status. Admin views classified profiles. The sequence shows timely interactions between user interface, processing modules, and the backend result storage system.



4. IMPLEMENTATION

4.1 PROJECT MODULES

Service Provider

In this module, the Service Provider has to login by using valid user name and password. After login successful he can do some operations such as Login, Train & Test User Profile Data Sets, View User Profile Trained and Tested Accuracy in Bar Chart, View User Profile Trained and Tested Accuracy Results, View All Profile Identity Prediction, Find and View Profile Identity Prediction Ratio, View User Profile Identity Ratio Results, Download Predicted Data Sets, View All Remote Users

View and Authorize Users

In this module, the admin can view the list of users who all registered. In this, the admin can view the user's details such as, user name, email, address and admin authorizes the users.

Remote User

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like Register and Login, Predict Profile Identification Status, View your Profile.

4.2 ALGORITHMS

Support Vector Machine(SVM):

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification tasks. It works by finding the optimal hyperplane that best separates data points into different categories. This hyperplane maximizes the margin between classes, which helps improve generalization to unseen data. SVM is especially effective in high-dimensional spaces and is robust against overfitting, particularly when the number of dimensions exceeds the number of samples. In fake profile detection, SVM classifies profiles by analyzing their features, such as user activity patterns or text data, to determine whether a profile is genuine or fake based on learned distinctions.

Naïve Bayes:

Naïve Bayes is a probabilistic classifier that applies Bayes' Theorem under the “naive” assumption that features are independent. Despite this simplification, it performs competitively with more complex models and is especially popular for text classification tasks. It calculates the posterior probability of each class given a profile's attributes, such as bio text, posting time, or language usage, and selects the class with the highest probability. Its efficiency and ease of implementation make it suitable for large datasets. In the context of fake profile detection, Naïve Bayes helps classify users based on patterns in their static and dynamic profile information.

Decision Tree Classifier:

A Decision Tree is a supervised learning algorithm that splits data into branches based on feature values, creating a tree-like structure where each internal node represents a test on a feature, and each leaf node represents a class label. It recursively partitions the dataset, choosing the feature that yields the highest information gain or lowest impurity. Decision Trees are easy to interpret and visualize, making them ideal for decision-making processes. In fake profile detection, a Decision Tree can use inputs like account age, activity level, and language use to classify profiles as genuine or fake by following a logical flow of decisions.

Random Forest:

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode (for classification) or mean (for regression) of the predictions. It combines the strengths of multiple weak learners to reduce overfitting and improve accuracy. Each tree is trained on a different subset of the data and features, introducing randomness that enhances generalization. In fake profile detection, Random Forest can analyze diverse profile features and their interactions to robustly predict whether a profile is fake. Its ability to handle large datasets and rank feature importance makes it a valuable tool in social network analysis.

Gradient Boosting:

Gradient Boosting is a powerful ensemble machine learning technique used for regression and classification tasks. It builds a strong model by sequentially adding weak learners, typically shallow decision trees, where each new model corrects the errors of the previous ones. The method optimizes a differentiable loss function by computing the residuals and fitting new models to them. A learning rate controls the contribution of each model to prevent overfitting. Gradient Boosting is known for high predictive accuracy and is used in libraries like XGBoost, LightGBM, and CatBoost. It performs well on structured data and is widely used in data science competitions.

K-Nearest Neighbors (KNN):

K-Nearest Neighbors (KNN) is a simple yet effective classification algorithm that makes predictions based on the closest training examples in the feature space. It's a lazy learner, meaning it doesn't build a model during training but instead stores the data. During prediction, it identifies the 'K' most similar instances and assigns the majority class among them to the new data point. KNN is highly intuitive and works well with a good distance metric. In fake profile detection, KNN can be used to classify a new profile based on its similarity to known genuine or fake profiles based on feature similarity.

Logistic Regression:

Logistic Regression is a statistical method used for binary and multiclass classification problems. It models the probability that a given input belongs to a certain class using a logistic (sigmoid) function. Unlike linear regression, it's suited for categorical outcomes like 'fake' or 'genuine' profiles. Logistic Regression does not assume normality of predictors, making it flexible for various datasets. In the context of fake profile identification, it analyzes features like posting frequency, profile completeness, and language patterns to estimate the probability of a profile being fake, offering interpretable coefficients that show the contribution of each feature to the prediction.

4.3 SAMPLE CODE

Remote User:

Manage.py

```
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'fake_profile_identification.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

Views.py

```
from django.db.models import Count
from django.db.models import Q
from django.shortcuts import render, redirect, get_object_or_404
import datetime
import openpyxl
import string
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from sklearn.ensemble import VotingClassifier
import warnings
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
import numpy as np
```

```

import pandas as pd
from sklearn.tree import DecisionTreeClassifier
# Create your views here.
from Remote_User.models import
ClientRegister_Model,profile_identification_type,detection_ratio,detection_accuracy

def login(request):

    if request.method == "POST" and 'submit1' in request.POST:
        username = request.POST.get('username')
        password = request.POST.get('password')
        try:
            enter = ClientRegister_Model.objects.get(username=username,password=password)
            request.session["userid"] = enter.id

            return redirect('ViewYourProfile')
        except:
            pass

    return render(request,'RUser/login.html')

def Register1(request):

    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        phoneno = request.POST.get('phoneno')
        country = request.POST.get('country')
        state = request.POST.get('state')
        city = request.POST.get('city')
        ClientRegister_Model.objects.create(username=username, email=email,
password=password, phoneno=phoneno,
country=country, state=state, city=city)

        return render(request, 'RUser/Register1.html')
    else:
        return render(request,'RUser/Register1.html')

def ViewYourProfile(request):
    userid = request.session['userid']
    obj = ClientRegister_Model.objects.get(id= userid)
    return render(request,'RUser/ViewYourProfile.html',{'object':obj})

def Predict_Profile_Identification_Status(request):

```



```

expense = 0
kg_price=0
if request.method == "POST":

    prof_idno= request.POST.get('prof_idno')
    name= request.POST.get('name')
    screen_name= request.POST.get('screen_name')
    statuses_count= request.POST.get('statuses_count')
    followers_count= request.POST.get('followers_count')
    friends_count= request.POST.get('friends_count')
    created_at= request.POST.get('created_at')
    location= request.POST.get('location')
    default_profile= request.POST.get('default_profile')
    prf_image_url= request.POST.get('prf_image_url')
    prf_banner_url= request.POST.get('prf_banner_url')
    prf_bgimg_https= request.POST.get('prf_bgimg_https')
    prf_text_color= request.POST.get('prf_text_color')
    profile_image_url_https= request.POST.get('profile_image_url_https')
    prf_bg_title= request.POST.get('prf_bg_title')
    profile_background_image_url= request.POST.get('profile_background_image_url')
    description= request.POST.get('description')
    Prf_updated = request.POST.get('Prf_updated')

    df = pd.read_csv('Profile_Datasets.csv')

    def clean_text(text):
        "Make text lowercase, remove text in square brackets,remove links,remove
punctuation
        and remove words containing numbers."
        text = text.lower()
        text = re.sub('[.*?]', "", text)
        text = re.sub('https?://\S+|www\.\S+', "", text)
        text = re.sub('<.*?>+', "", text)
        text = re.sub('[%s]' % re.escape(string.punctuation), "", text)
        text = re.sub('\n', "", text)
        text = re.sub('\w*\d\w*', "", text)
        text = re.sub("@", "", text)
        text = re.sub("'", "", text)
        text = re.sub('https: //', "", text)
        text = re.sub('Ã¢â¬ââ¬â¢', "", text)
        text = re.sub('\n\n', "", text)

        return text

    df['processed_content'] = df['name'].apply(lambda x: clean_text(x))
    def apply_results(label):

```

```

    if (label == 0):
        return 0 # Fake
    elif (label == 1):
        return 1 # Genuine

df['results'] = df['Label'].apply(apply_results)
cv = CountVectorizer(lowercase=False)
y = df['results']
X = df["id"].apply(str)
print("X Values")
print(X)
print("Labels")
print(y)
X = cv.fit_transform(X)
models = []
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
X_train.shape, X_test.shape, y_train.shape

# SVM Model
print("SVM")
from sklearn import svm
lin_clf = svm.LinearSVC()
lin_clf.fit(X_train, y_train)
predict_svm = lin_clf.predict(X_test)
svm_acc = accuracy_score(y_test, predict_svm) * 100
print(svm_acc)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, predict_svm))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, predict_svm))
models.append(('svm', lin_clf))

print("KNeighborsClassifier")
from sklearn.neighbors import KNeighborsClassifier
kn = KNeighborsClassifier()
kn.fit(X_train, y_train)
knpredict = kn.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, knpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, knpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, knpredict))
models.append(('KNeighborsClassifier', kn))

```

```

classifier = VotingClassifier(models)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

prof_idno1 = [prof_idno]
vector1 = cv.transform(prof_idno1).toarray()
predict_text = classifier.predict(vector1)

pred = str(predict_text).replace("[", "")
pred1 = pred.replace("]", "")

prediction = int(pred1)

if prediction == 0:
    val = 'Fake Profile'
elif prediction == 1:
    val = 'Genuine Profile'

print(val)
print(pred1)

profile_identification_type.objects.create(
    prof_idno=prof_idno,
    name=name,
    screen_name=screen_name,
    statuses_count=statuses_count,
    followers_count=followers_count,
    friends_count=friends_count,
    created_at=created_at,
    location=location,
    default_profile=default_profile,
    prf_image_url=prf_image_url,
    prf_banner_url=prf_banner_url,
    prf_bgimg_https=prf_bgimg_https,
    prf_text_color=prf_text_color,
    profile_image_url_https=profile_image_url_https,
    prf_bg_title=prf_bg_title,
    profile_background_image_url=profile_background_image_url,
    description=description,
    Prf_updated=Prf_updated,
    Prediction=val)

return render(request, 'RUser/Predict_Profile_Identification_Status.html',{ 'objs':val})
return render(request, 'RUser/Predict_Profile_Identification_Status.html')

```

Models.py

```
from django.db import models
```

```
# Create your models here.
```

```
from django.db.models import CASCADE
```

```
class ClientRegister_Model(models.Model):
```

```
    username = models.CharField(max_length=30)
```

```
    email = models.EmailField(max_length=30)
```

```
    password = models.CharField(max_length=10)
```

```
    phoneno = models.CharField(max_length=10)
```

```
    country = models.CharField(max_length=30)
```

```
    state = models.CharField(max_length=30)
```

```
    city = models.CharField(max_length=30)
```

```
class profile_identification_type(models.Model):
```

```
    prof_idno= models.CharField(max_length=3000)
```

```
    name= models.CharField(max_length=3000)
```

```
    screen_name= models.CharField(max_length=3000)
```

```
    statuses_count= models.CharField(max_length=3000)
```

```
    followers_count= models.CharField(max_length=3000)
```

```
    friends_count= models.CharField(max_length=3000)
```

```
    created_at= models.CharField(max_length=3000)
```

```
    location= models.CharField(max_length=3000)
```

```
    default_profile= models.CharField(max_length=3000)
```

```
    prf_image_url= models.CharField(max_length=3000)
```

```
    prf_banner_url= models.CharField(max_length=3000)
```

```
    prf_bgimg_https= models.CharField(max_length=3000)
```

```
    prf_text_color= models.CharField(max_length=3000)
```

```
    profile_image_url_https= models.CharField(max_length=3000)
```

```
    prf_bg_title= models.CharField(max_length=3000)
```

```
    profile_background_image_url= models.CharField(max_length=3000)
```

```
    description= models.CharField(max_length=3000)
```

```
    Prf_updated= models.CharField(max_length=3000)
```

```
    Prediction= models.CharField(max_length=3000)
```

```
class detection_ratio(models.Model):
```

```
    names = models.CharField(max_length=300)
```

```
    ratio = models.CharField(max_length=300)
```

```
class detection_accuracy(models.Model):
```

```
    names = models.CharField(max_length=300)
```

```
    ratio = models.CharField(max_length=300)
```

fake profile identification

urls.py

```
from django.conf.urls import url
from django.contrib import admin
from Remote_User import views as remoteuser
from fake_profile_identification import settings
from Service_Provider import views as serviceprovider
from django.conf.urls.static import static

urlpatterns = [
    url('admin/', admin.site.urls),
    url(r'^$', remoteuser.login, name="login"),
    url(r'^Register1/$', remoteuser.Register1, name="Register1"),
    url(r'^Predict_Profile_Identification_Status/$',
remoteuser.Predict_Profile_Identification_Status,
name="Predict_Profile_Identification_Status"),
    url(r'^ViewYourProfile/$', remoteuser.ViewYourProfile, name="ViewYourProfile"),
    url(r'^serviceproviderlogin/$',serviceprovider.serviceproviderlogin,
name="serviceproviderlogin"),
    url(r'^View_Remote_Users/$',serviceprovider.View_Remote_Users,name="View_Remote_Users"),
    url(r'^charts/(?P<chart_type>\w+)', serviceprovider.charts,name="charts"),
    url(r'^charts1/(?P<chart_type>\w+)', serviceprovider.charts1, name="charts1"),
    url(r'^likeschart/(?P<like_chart>\w+)', serviceprovider.likeschart, name="likeschart"),
    url(r'^View_Profile_Identity_Prediction_Ratio/$',
serviceprovider.View_Profile_Identity_Prediction_Ratio,name="View_Profile_Identity_Prediction_Ratio"),
    url(r'^likeschart1/(?P<like_chart1>\w+)', serviceprovider.likeschart1, name="likeschart1"),
    url(r'^Train_Test_DataSets/$', serviceprovider.Train_Test_DataSets,
name="Train_Test_DataSets"),
    url(r'^View_Profile_Identity_Prediction/$', serviceprovider.View_Profile_Identity_Prediction,
name="View_Profile_Identity_Prediction"),
    url(r'^Download_Trained_DataSets/$', serviceprovider.Download_Trained_DataSets,
name="Download_Trained_DataSets"),
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Settings.py

```
import os
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'm+1edl5m-5@u9u!b8=-4-4mq&o1%agco2xpl8c!7sn7!eowjk#'
```

```
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'Remote_User',
    'Service_Provider',
]
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'fake_profile_identification.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [(os.path.join(BASE_DIR, 'Template/htmls'))],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]
```

```

WSGI_APPLICATION = 'fake_profile_identification.wsgi.application'

# Database

# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'fake_profile_identification',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}

# Password validation

# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

```
# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'Template/images')]
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'Template/media')

STATIC_ROOT = '/static/'

STATIC_URL = '/static/'
```

\

5.TESTING

5.1 TESTING METHODS

The following are the Testing Methodologies:

- **Unit Testing.**
- **Integration Testing.**
- **User Acceptance Testing.**
- **Output Testing.**
- **Validation Testing.**

1. Unit Testing

Unit Testing focuses on the verification of individual components or modules of the system such as the NLP preprocessing module, feature extraction functions, and classifiers (SVM and Naïve Bayes). Each function is tested independently to ensure it performs as expected. For example, the text processing functions must clean and tokenize data correctly, while the classifier functions should accurately classify inputs based on the training data. Unit tests ensure internal logic correctness and help identify bugs early in development.

2. Integration Testing

Integration Testing ensures that various modules of the system interact correctly. In this project, different units such as the user input module, preprocessing unit, machine learning models, and the database must work together seamlessly. Integration testing checks if data flows correctly from the frontend to the backend and whether modules exchange data as intended. It validates whether, for instance, the processed profile data is accurately passed to the classifier and whether the result is stored and displayed properly.

3. System Testing

System Testing is conducted on the entire integrated system to evaluate the system's compliance with specified requirements. For this fake profile identification system, it ensures that the complete workflow—from profile input, data processing, classification, to result display—functions smoothly. It verifies that the system accurately distinguishes between fake and genuine profiles and that all components (frontend, backend, ML, and database) are operating as a cohesive unit.

4. User Acceptance Testing (UAT)

User Acceptance Testing involves validation by the intended users, such as social network administrators or moderators. It evaluates whether the system is user-friendly, the interface is intuitive, and the predictions made by the system are understandable and actionable. Feedback gathered during UAT is used to make final adjustments. It ensures the system meets the real-world expectations of its users.

5. Output Testing

Output Testing checks whether the system produces the correct results and whether these are presented in a clear and required format. For this project, it ensures that the prediction results (fake/genuine profile labels, confidence levels) are displayed properly on the user interface or as reports. It confirms that the results align with user expectations and data input.

6. Validation Testing

Validation Testing ensures that all input data is correct and within specified limits. In this system, inputs such as profile details, text posts, timestamps, and location data are validated for format and completeness. The system must reject invalid data with appropriate error messages. This testing safeguards against incorrect processing and helps maintain data integrity.

7. Test Data Preparation

Test data is essential for evaluating the system under different conditions. In this project, both artificial and live test data are used. Artificial test data is created to simulate edge cases and test all logic paths, while live test data, drawn from actual social media sources, simulates real usage. Together, they help identify defects and verify system robustness.

8. Using Live Test Data

Live test data, sourced from real social network profiles, offers realistic scenarios to test the system's practical performance. It shows how the system handles actual user behaviors and profile structures. However, it may not cover rare or extreme cases, making it important to use alongside artificial data.

9. Using Artificial Test Data

Artificial test data is designed to thoroughly test all control and logic paths. This data includes a wide range of scenarios including invalid, boundary, and extreme inputs. It is often generated by independent testers to ensure objectivity and completeness, helping identify hidden flaws in the system.

10. User Training

User training is vital to ensure that end users can effectively use the system. Training sessions or user guides help users understand system functionality, navigate the interface, and interpret the results accurately. As the system is designed for users with basic computer knowledge, the interface is kept simple and intuitive.

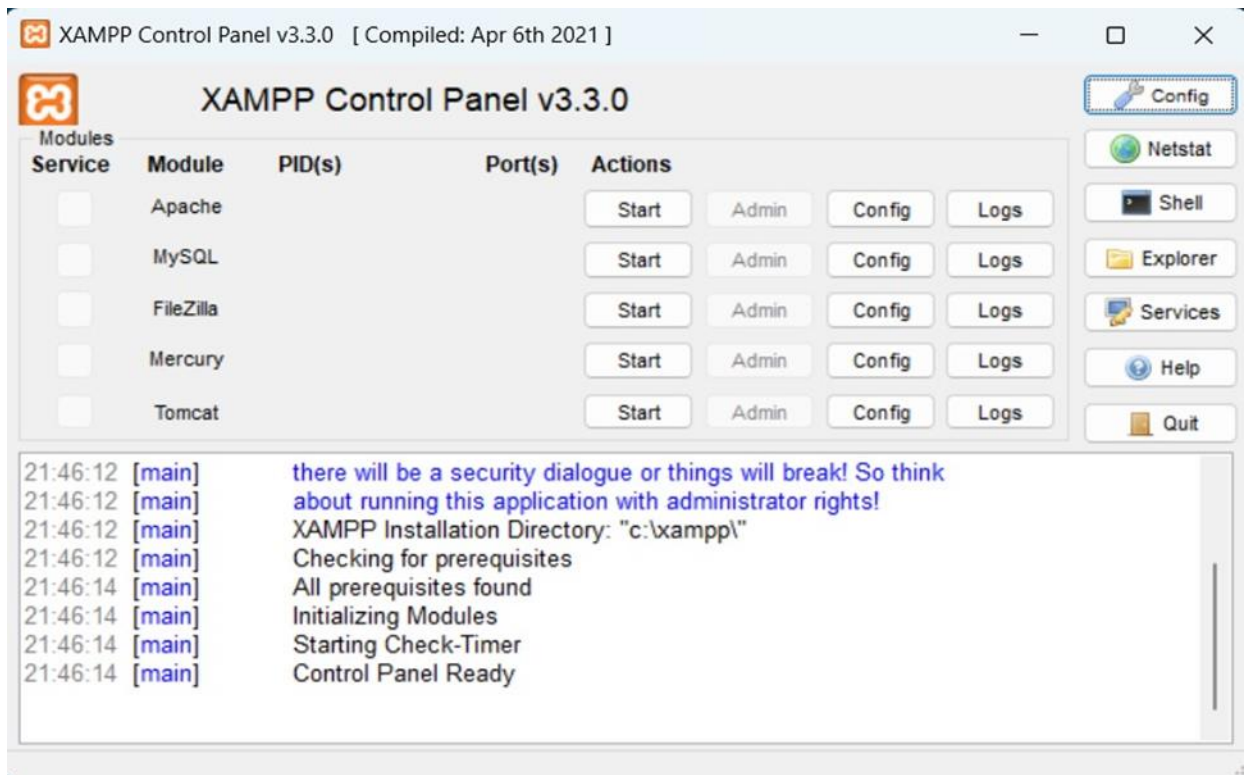
11. Maintenance

Maintenance involves making necessary updates, fixing bugs, and adapting the system to new requirements. For this project, modular design and clean coding practices facilitate easy updates and debugging. As social networks evolve, the system can be enhanced with new features or retrained with updated data to maintain its effectiveness.

6. RESULTS

The implementation of the proposed system using machine learning and natural language processing (NLP) techniques has resulted in a more efficient and accurate method for identifying fake profiles in social networks. By integrating Support Vector Machine (SVM) and Naïve Bayes algorithms, the system is able to distinguish genuine profiles from fake ones with improved precision. These classifiers analyze various attributes of user profiles, including static information provided during account creation and dynamic information generated through user activity. SVM effectively identifies optimal decision boundaries, while Naïve Bayes leverages probabilistic reasoning based on features such as post frequency, language usage, and geolocation data. Unlike traditional systems which did not utilize learning algorithms, this approach significantly enhances the detection rate of fake profiles, even in data-restricted environments like LinkedIn. Furthermore, the system is designed to detect broader social threats such as online impersonation, identity theft, and content misuse. It operates within a Python and Django framework, backed by a MySQL database, making it deployable on standard computing hardware. Overall, the results of this system demonstrate an intelligent and scalable solution to a critical problem in modern online communication platforms, ensuring a more secure and trustworthy user environment.

6.1



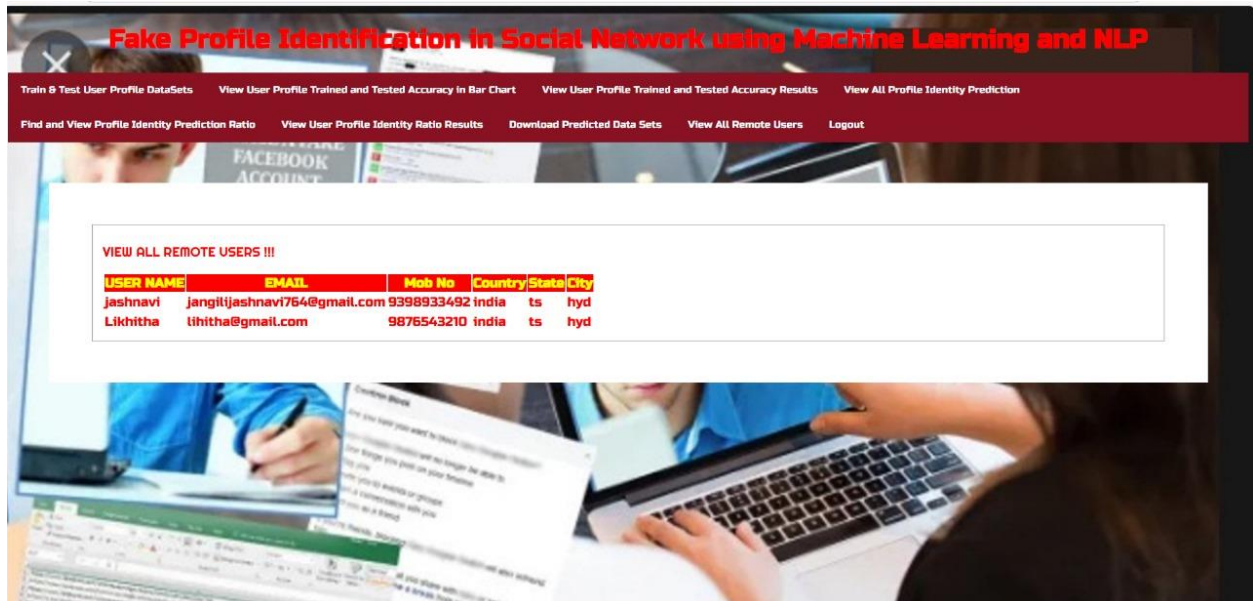
6.2 Login Page



6.3 Register form



6.4 Remote Users data



Fake Profile Identification in Social Network using Machine Learning and NLP

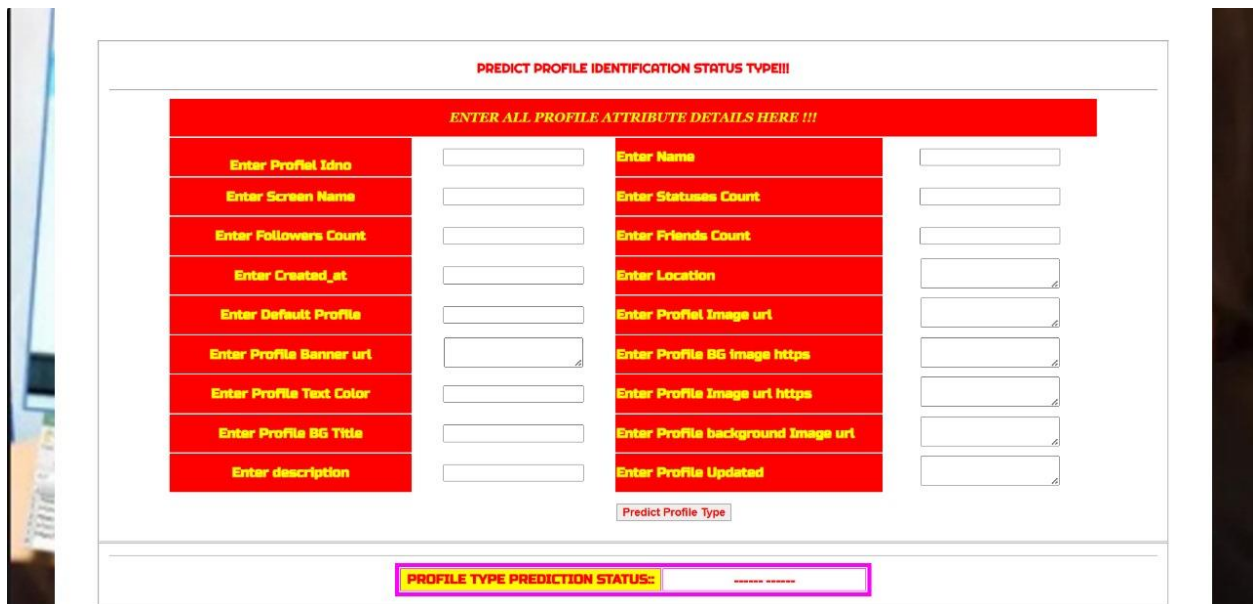
Train & Test User Profile DataSets View User Profile Trained and Tested Accuracy in Bar Chart View User Profile Trained and Tested Accuracy Results View All Profile Identity Prediction

Find and View Profile Identity Prediction Ratio View User Profile Identity Ratio Results Download Predicted Data Sets View All Remote Users Logout

VIEW ALL REMOTE USERS !!!

USER NAME	EMAIL	Mob No	Country	State	City
jashnavi	jangitijashnavi764@gmail.com	9398933492	india	ts	hyd
Likhitha	lihitha@gmail.com	9876543210	india	ts	hyd

6.5 Output label



PREDICT PROFILE IDENTIFICATION STATUS TYPE!!!

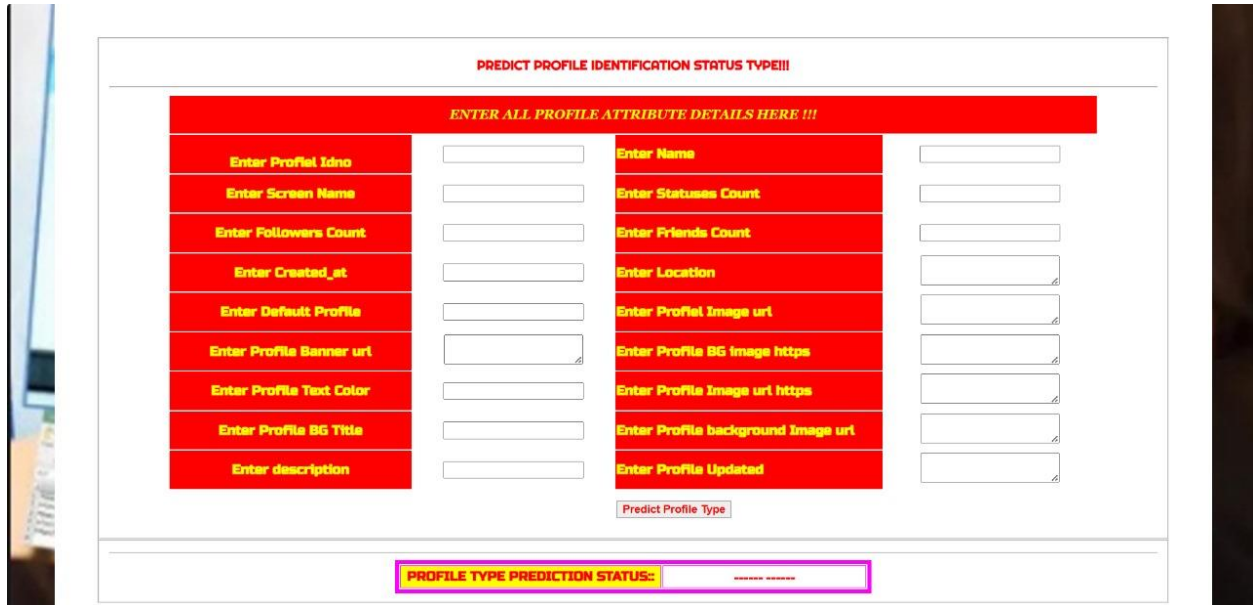
ENTER ALL PROFILE ATTRIBUTE DETAILS HERE !!!

Enter Profil Idno	<input type="text"/>	Enter Name	<input type="text"/>
Enter Screen Name	<input type="text"/>	Enter Statuses Count	<input type="text"/>
Enter Followers Count	<input type="text"/>	Enter Friends Count	<input type="text"/>
Enter Created_at	<input type="text"/>	Enter Location	<input type="text"/>
Enter Default Profile	<input type="text"/>	Enter Profil Image url	<input type="text"/>
Enter Profile Banner url	<input type="text"/>	Enter Profile BG image https	<input type="text"/>
Enter Profile Text Color	<input type="text"/>	Enter Profile Image url https	<input type="text"/>
Enter Profile BG Title	<input type="text"/>	Enter Profile background Image url	<input type="text"/>
Enter description	<input type="text"/>	Enter Profile Updated	<input type="text"/>

Predict Profile Type

PROFILE TYPE PREDICTION STATUS:

6.6 Sample Output form



The image shows a web browser displaying a form titled "PREDICT PROFILE IDENTIFICATION STATUS TYPE!!!". The form has a red header bar with the text "ENTER ALL PROFILE ATTRIBUTE DETAILS HERE !!!". Below this, there are two columns of input fields. The left column contains: "Enter Profile Idno", "Enter Screen Name", "Enter Followers Count", "Enter Created_at", "Enter Default Profile", "Enter Profile Banner url", "Enter Profile Text Color", "Enter Profile BG Title", and "Enter description". The right column contains: "Enter Name", "Enter Statuses Count", "Enter Friends Count", "Enter Location", "Enter Profile Image url", "Enter Profile BG image https", "Enter Profile Image url https", "Enter Profile background Image url", and "Enter Profile Updated". Each input field is a text box with a small icon on the right. Below the input fields is a button labeled "Predict Profile Type". At the bottom of the form, there is a yellow box with the text "PROFILE TYPE PREDICTION STATUS:" followed by a small, empty box.

PREDICT PROFILE IDENTIFICATION STATUS TYPE!!!	
ENTER ALL PROFILE ATTRIBUTE DETAILS HERE !!!	
Enter Profile Idno	<input type="text"/>
Enter Screen Name	<input type="text"/>
Enter Followers Count	<input type="text"/>
Enter Created_at	<input type="text"/>
Enter Default Profile	<input type="text"/>
Enter Profile Banner url	<input type="text"/>
Enter Profile Text Color	<input type="text"/>
Enter Profile BG Title	<input type="text"/>
Enter description	<input type="text"/>
Enter Name	<input type="text"/>
Enter Statuses Count	<input type="text"/>
Enter Friends Count	<input type="text"/>
Enter Location	<input type="text"/>
Enter Profile Image url	<input type="text"/>
Enter Profile BG image https	<input type="text"/>
Enter Profile Image url https	<input type="text"/>
Enter Profile background Image url	<input type="text"/>
Enter Profile Updated	<input type="text"/>

PROFILE TYPE PREDICTION STATUS:

7.CONCLUSION

In this paper, we have proposed an effective method for detecting fake profiles in social network environments by integrating machine learning algorithms with natural language processing (NLP) techniques. The increasing misuse of social networking platforms, such as the creation of fake profiles for malicious or deceptive purposes, poses a significant challenge to online safety and trust. To address this issue, our approach combines the strengths of NLP for text data analysis and machine learning for accurate classification. For our experimental setup, we utilized a Facebook dataset as a case study to test and validate the performance of the proposed system. The dataset underwent various NLP pre-processing steps, including tokenization, stop-word removal, and stemming, to extract meaningful information from the profile descriptions and user-generated content. This pre-processed data was then fed into two widely used classification algorithms: Support Vector Machine (SVM) and Naive Bayes. The SVM algorithm was used for its ability to construct an optimal hyperplane that separates fake profiles from genuine ones, while Naïve Bayes applied a probabilistic approach, evaluating the likelihood of a profile being fake based on independent features such as language patterns, post frequency, and other behavioral data. These algorithms proved highly effective in improving the accuracy rate of fake profile detection, outperforming traditional heuristic-based methods. Overall, the proposed system demonstrates that by combining machine learning and NLP, we can build a scalable, intelligent framework capable of identifying fake profiles with greater efficiency and accuracy. This contributes not only to improving user trust on platforms like Facebook but also to creating a safer digital environment. Future enhancements can involve real-time detection and extending the approach to other social media platforms such as Twitter, Instagram, and LinkedIn.

8.REFERENCES

- [1] Michael Fire et al. (2012). "Strangers intrusion detection-detecting spammers and fake profiles in social networks based on topology anomalies." Human Journal 1(1): 26-39.
- Günther, F. and S. Fritsch (2010). "neuralnet: Training of neural networks." The R Journal 2(1): 30-38
- [2] Dr. S. Kannan, Vairaprakash Gurusamy, "Preprocessing Techniques for Text Mining", 05 March 2015.
- [3] Shalinda Adikari and Kaushik Dutta, Identifying Fake Profiles in LinkedIn, PACIS 2014 Proceedings, AISeL
- [4] Z. Halim, M. Gul, N. ul Hassan, R. Baig, S. Rehman, and F. Naz, "Malicious users' circle detection in social network based on spatiotemporal co-occurrence," in Computer Networks and Information Technology (ICCNIT), 2011 International Conference on, July, pp. 35–390.
- [5] Liu Y, Gummadi K, Krishnamurthy B, Mislove A, "Analyzing Facebook privacy settings: User expectations vs. reality", in: Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, ACM, pp.61–70.
- [6] Mahmood S, Desmedt Y, "Poster: preliminary analysis of google?'s privacy. In: Proceedings of the 18th ACM conference on computer and communications security", ACM 2011, pp.809–812.
- [7] Stein T, Chen E, Mangla K, "Facebook immune system. In: Proceedings of the 4th workshop on social network systems", ACM 2011, pp
- [8] Saeed Abu-Nimeh, T. M. Chen, and O. Alzubi, "Malicious and Spam Posts in Online Social Networks," Computer, vol.44, no.9, IEEE2011, pp.23–28.
- [9] J. Jiang, C. Wilson, X. Wang, P. Huang, W. Sha, Y. Dai, B. Zhao, Understanding latent interactions in online social networks, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, ACM, 2010, pp. 369–382
- [10] Kazienko, P. and K. Musiał (2006). Social capital in online social networks. Knowledge-Based Intelligent Information and Engineering Systems, Springer.