A

Mini Project Report on

DETECTING FAKE NEWS USING ADVANCED MACHINE LEARNING ALGORITHMS

Submitted for partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted by

M VAISHNAVI	**21K81A12A0**
A ADHYA	**21K81A1266**
R ARJUN	**21K81A12B2**

Under the Guidance of

S VEERESH KUMAR

ASSISTANT. PROFESSOR

DEPARTMENT OF INFORMATION TECHNOLOGY

St. MARTIN'S ENGINEERING COLLEGE

UGC Autonomous Affiliated to JNTUH, Approved by AICTE,
Accredited by NBA & NAAC A+, ISO 9001-2008
Certified Dhulapally, Secunderabad - 500 100
www.smec.ac.in

NOVEMBER – 2024

St. MARTIN'S ENGINEERING COLLEGE

UGC Autonomous Affiliated to JNTUH,
Approved by AICTE NBA & NAAC A+
Accredited Dhulapally, Secunderabad - 500 100

www.smec.ac.in

Certificate

This is to certify that the project entitled "**DETECTING FAKE NEWS USING ADVANCED MACHINE LEARNING ALGORITHMS**" is being submitted by M.Vaishnavi(21K81A12A0),A.Adhya(21K81A1266),R.Arjun (21K81A12B2) in Fulfilment of the requirement for the award of degree of BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY is recorded of bonafide work carried out by them. The result embodied in this report have been verified and found satisfactory

Signature of Guide
Mr .S VEERESH KUMAR
Assistant Professor
Department of Information Technology

Signature of HOD
Dr. N. KRISHNAIAH
Professor and Head of Department
Department of Information Technology

Internal Examiner **External Examiner**

Place:
Date:

St. MARTIN'S ENGINEERING COLLEGE

UGC Autonomous

Affiliated to JNTUH, Approved by AICTE

NBA & NAAC A+ Accredited Dhulapally, Secunderabad - 500 100

www.smec.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

Declaration

We, the students of '**Bachelor of Technology in Department of Information Technology'**, session: 2021 - 2025**, St.Martin's Engineering College, Dhulapally, Kompally, Secunderabad,** hereby declare that the work presented in this Project Work entitled **"DETECTING FAKE NEWS USING ADVANCED MACHINE LEARNING ALGORITHMS"** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. This result embodied in this project report has not been submitted in any university for award of any degree.

M VAISHNAVI	**21K81A12A0**
A ADHYA	**21K81A1266**
RARJUN	**21K81A12B2**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowded our efforts with success.

First and foremost, we would like to express our deep sense of gratitude and indebtedness to our College Management for their kind support and permission to use the facilities available in the Institute.

We especially would like to express our deep sense of gratitude and indebtedness to **Dr. P. SANTOSH KUMAR PATRA**, Professor and Group Director, St. Martin's Engineering College Dhulapally, for permitting us to undertake this project. We wish to record our profound gratitude to **Dr. M. SREENIVAS RAO**, Principal, St. Martin's Engineering College, for has motivation and encouragement

We are also thankful to **Dr. N. KRISHNAIAH**, Head of the Department, Information Technology, St. Martin's Engineering College, Dhulapally, Secunderabad. for his support and guidance throughout our project as well as Project Coordinator **Mr.T.SURESH**, Assistant Professor, Information Technology department for his valuable support.

We would like to express our sincere gratitude and indebtedness to our Project supervisor **S VEERESH KUMAR** Assistant Professor, Information Technology, St. Martins Engineering College, Dhulapally, for his support and guidance throughout our project.

Finally, we express thanks to all those who have helped us successfully completing this project. Furthermore, we would like to thank our family and friends for their moral support and encouragement. We express thanks to all those who have helped us in successfully completing the project.

M VAISHNAVI **21K81A12A0**

AADHYA **21K81A1266**

RARJUN **21K81A12B2**

ABSTRACT

Online social networks have permeated our social lives in the current generation. These sites have allowed us to see our social lives differently than they did in the past. Nowadays we can connect with new friends and maintain relationships with them via social and personal activities become quite easy. Online Social Networks (OSN) are contributed in all areas such as Research in all domains, Job-related areas, Technology oriented areas, Health care, and business-oriented areas, Information gathering and data collection, and so on. One of the biggest problems on these social media platforms is fake profiles. Impersonating to be someone else and causing harm and defamation to the real person or advertising or popularizing removed propaganda on someone's name to get more benefit is the motto of such profile creators. There have been many studies regarding these fake accounts and how can they be mitigated. Many approaches such as graph-level activities or feature analysis have been taken into consideration to identify fake profiles. These methods are outdated when compared to arising issues of these days. In this paper, we proposed a technique using machine learning for fake profile detection which is efficient. . The benchmark data set is collected and mixed with manual data first furthermore; a data cleaning technique is used to present the data more feasibly. Then the preprocessed data is used for model building with sufficient information such as profile name, profile ID name, number of followers, and so on. We added Cross validation process where many training algorithms are implemented on the given data and are then tested on the same data. Based on the experiments the RF classifier performed better than the other classification methods. The Random Forest classifier is used to forecast the profile whether is fake or genuine in an efficient way.

LIST OF FIGURES

LIST OF TABLES

LIST OFACRONYMS AND DEFINITIONS

S. No:	Acronym	Definition
01.	PTSD	Post-traumatic Stress Disorder
02.	CNN	Convolutional Neural Network
03.	SVM	Support Vector Machine
04.	UML	Unified Modelling Language
05.	RF	Random Forest
06.	RNN	Recurrent Neural Networks
07.	NB	Naive Bayes
08.	PTSD	Post-traumatic Stress Disorder
09.	ANN	Artificial Neural Network

CONTENTS

CHAPTER 1
INTRODUCTION

1.1 OBJECTIVE

On the internet, you can access a variety of opportunities and connections. You're probably familiar with popular social networking sites like Facebook, WhatsApp, Instagram, and Snapchat. In addition to these forms of social interaction, our current generation also participates in many other forms of interaction [1-2]. Social networking sites are very easy.979-8-3503-1156-3/22/$31.00 ©2022 IEEE for teachers to train youngsters and Teachers in this era have become very familiar with these websites, giving online lectures, giving assignments, holding discussions, and so on, which improves education significantly. Using these social networking sites, employers can hire human beings who are skilled and enthusiastic about the work; their history can easily be looked into by utilizing these websites. These platforms are free however some cost the membership price and make use of this for commercial enterprise functions and the rest of them elevate cash with the aid of the usage of advertising [3-4]. However, it also has some downsides and one of those is fake profiles. They are usually a result of the simple lack of engagement with people face- to-faces and this can often lead to invitations that we wouldn't normally receive if these fake profiles weren't present on social networks [5]. Due to the wide use of social networks, there have been many studies done in this domain,

.

1.2 OVERVIEW

ML models are trained on large datasets containing labeled news articles as either "fake" or "real." Datasets like the Fake News Corpus, LIAR, and BuzzFeed News dataset are popular choices . Text preprocessing is crucial to clean and normalize data. Techniques such as tokenization, stemming , stop word removal, and transforming text to lowercase are commonly used. The content of articles is analyzed for linguistic patterns. Features include word count, frequency of specific words, grammar structures, sentiment, and readability score . Data such as user engagement metrics (likes, shares, comments) and profiles of sharers provide additional context. Metadata, like publication date, source, and author information, also helps in differentiating legitimate articles from fake news. Algorithms like Support Vector Machines (SVM), Decision Trees, and Naive Bayes were initially used for fake news detection, leveraging statistical properties of text. Deep learning models are more effective due to their ability to capture complex relationships within text . Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) these models are effective in understanding the sequential nature of text. CNNs can capture local patterns in the text, such as phrases indicative of fake news . Transformer-based models, especially BERT and GPT, have become popular due to their high accuracy in understanding nuanced text and contextual cues. Hybrid Models Combining models that analyze text with models that analyze metadata and social context features improves accuracy. For instance, a hybrid model might use BERT to analyze the article content and a separate model to analyze social context.

CHAPTER 2
LITERATURE SURVEY

Many issues, including fake profiles and online impersonation, have arisen in today's online social networks. No one has developed a potential solution for these issues as of yet. To safeguard human social lives, we want to provide a novel model for computerized fake profile early detection in this project. We can also make it simpler for websites to change the large diversity of profiles by using our automatic detecting technology which is hard to accomplish manually. To discover the traits or a mixture of them that help to differentiate between real and false records, several fake document focus approaches rely on the examination of a person's interpersonal organization profiles. Specifically, a classifier that can recognize bogus data is built using machine learning techniques after many attributes are obtained from the profiles and posts. Padmavati et al. approach the problem of fake accounts on social media by a method using Deterministic Finite Automata (DFA) [7]. The paper analyzes the features of the existing user and their friends by creating an accounting pattern. The pattern is made with regular expressions based on some attributes such as the working and living community and so on which are used for pattern matching with any friend requests. The drawback of this approach is that the generation of regular expression takes quite a long for a person who has friends in many communities. The authors contend that the approach could be even more efficient for the real In their article, Mohammad rezaetal. employed graph analysis and classification algorithms to examine the issue of phony accounts on social networks. Twitter was the social media platform of choice. They developed a strategy based on how similar the user's friends were. Before extracting new features using Principal Component Analysis (PCA), it first employs the buddy similarity criterion from the network graph [8]. Next, the data is balanced and delivered to the classifier using the synthetic minority oversampling method (SMOTE).

A medium Gaussian SVM classifier was selected after utilizing the cross-validation procedure since it has an AUC of 1. This method's flaw is that phony accounts must only function within the network to avoid being detected by looking at the accounts of their friends. The profile data from social media called Instagram was taken from the Kaggle website.

They employed classification algorithms such as SVM, KNN, RF, NB, and XG Boost to train the model. After computing the accuracy and confusion matrix, the RF classifier stood out as the suitable model for the data set with the best results of prediction.

Later the IDs of the Fake profiles are written into a data dictionary. In 2018, Abhishek Narayanan et al. discussed recognizing fake profiles in their paper and their data set was taken from the social media named Twitter. At first feature extraction on the data was done on which machine learning algorithms namely SVM, RF, and LR performed which gave a very much appreciated result for the random forest in the end. Later after performing some accuracy testing and confusion matrices random forest classifier stood out with 88% of precise prediction of fake profiles on Twitter. It was more efficient and comparatively took the shortest to achieve the results. Their future work is intended towards ensuring the security of the users while surfing other using social media[14].In 2012, Mauro Conti et al. discussed through the paper the ways the issue can tackle [15]. The first thing done was to check if a particular profile is similar to the population of real users. Then they used graph structures for fake profile detection. They observed the user's connection that is the friends list whether there is a greater number of random ones or whether there are certain numbers of mutual friends.

CHAPTER 3
SYSTEM ANALYSIS AND DESIGN

3.1 EXISTING SYSTEM

Sentiment analysis is a classification problem where the main focus is to predict the polarity of words and then classify them into positive or negative sentiment. Classifiers used are of mainly two types, namely lexicon-based and machine learning based. The former include Senti Word Net and Word Sense Disambiguation . Sentiment Analysis which means to analyze the underlying emotions of a given text using Natural Language Processing.ML algorithm using some labelled data and then use that model to predict a class for a new text. Fake News Detection is a natural language processing task that involves identifying and classifying news articles or other types of text as real or fake. The goal of fake news detection is to develop algorithms that can automatically identify and flag fake news articles, which can be used to combat misinformation and promote the dissemination of accurate information.

Why do we need a SVMAlgorithm?

The Support Vector Machine (SVM) algorithm is an essential tool in machine learning, particularly for classification and regression tasks, because of its effectiveness in handling complex datasets with higher dimensions. Here's why we need the SVM algorithm

Effective in High-Dimensional Spaces:

SVM performs well in spaces where there are many features, making it suitable for text classification, image recognition, and bioinformatics where data often has many variables.

Margin of Separation:

SVM maximizes the margin between data classes, which helps in achieving better generalization on new, unseen data. This margin maximization reduces the risk of overfitting.

Works Well with Non-linear Data:

SVM uses the "kernel trick," which allows it to create non-linear boundaries by transforming the data into a higher-dimensional space. This makes it powerful for data that isn't linearly separable.

How does SVMAlgorithm work?

The SVM working can be explained based on the below algorithm:

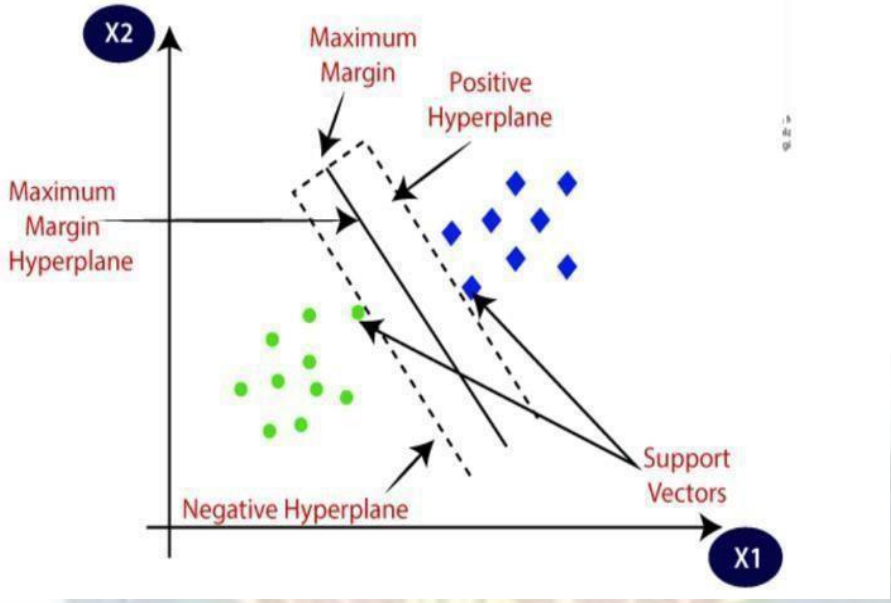


Figure 3.1 Working of SVM Algorithm

Initial Data: The algorithm receives labeled training data for classification

.Hyperplane Selection: SVM tries to identify the hyperplane that best separates classes by maximizing the margin between classes.

Maximization Problem: It translates this task into a mathematical optimization problem where the goal is to maximize the margin while minimizing classification errors. SVM uses optimization techniques to solve this.

Kernel Transformation (if necessary): For data that isn't linearly separable, SVM applies a kernel function to project the data into a higher-dimensional space.

Decision Boundary: Once trained, the SVM uses the hyperplane to classify new data points based on

which side of the hyperplane they fall on.

Kernels allow SVM to create complex decision boundaries, which enables it to handle non-linear data.

1. DISADVANTAGES OF EXISTING SYSTEM:

- Machine learning models rely heavily on the data they are trained on. If the training data is biased, outdated, or imbalanced, the model may learn and propagate those biases, leading to inaccurate or unfair results. High-quality labeled datasets for fake news detection are difficult to obtain and can be subjective.
- Fake news detection often requires understanding nuanced language, satire, sarcasm, and cultural context.
- Fake news creators continuously evolve their tactics, creating new forms of misleading content. Machine learning models trained on past data may not adapt quickly to novel strategies used to deceive readers, resulting in a "cat and mouse" problem.

3.2 PROPOSED SYSTEM

In our proposed work Greedy and Dynamic Blocking Algorithms suggests tweets by coordinating clients with different clients. It gathers client input as evaluations gave by client to explicit tweets and discovers coordinate in rating practices among clients to discover gathering of clients having comparative inclinations. One of the principle highlights on the landing page of Twitter shows a rundown of top terms purported moving themes consistently. It can join both substance based and synergistic separating approaches. The proposed framework utilizes changed cosine likeness technique.

3.2.1 ADVANTAGES

1. Scalability:

ML algorithms can analyze vast data and identify patterns to detect false news. They can handle large datasets in real time, which is essential for monitoring news feeds and social media platforms where new content is generated continuously. This scalability allows the algorithm to keep up with the pace of information production and identify false news as soon as it appears.

2. Speed

Speed is critical when it comes to detecting online fake news. ML algorithms can process a vast amount of data quickly, enabling the system to immediately detect false news. The quicker the automatic detection is, the less harm the false news can cause. The algorithm can also be trained to prioritize certain types of true and false news online, enabling it to focus on the most relevant information.

3. Accuracy

One of the critical advantages of machine learning is its ability to learn from past data and improve its accuracy over time. Training the algorithm with labeled data allows it to recognize patterns in the text and images that indicate false news. The algorithm can also be updated regularly, ensuring it stays up-to-date with the latest types of false news.

NAIVE BAYES (NB):-

The Naive Bayes algorithm is a supervised machine learning algorithm used for classification tasks, such as text classification. It models the distribution of inputs for a given class or category and assumes that features are conditionally independent of each other given the class label :

1. Simple and Efficient Computationally Efficient:

a. Naive Bayes is relatively fast and efficient, particularly with large datasets. It's based on simple probabilistic calculations .Minimal Training Data Requirement:

b. The model works well even with a small training dataset, as it relies on probabilistic modeling rather than complex parameter tuning.

2. Assumption of Feature Independence Naive Assumption:

c. Naive Bayes assumes that all features are independent of each other, given the target class. This is often an unrealistic assumption, especially when features are correlated, but it simplifies computation significant Conditional Independence:

d. Despite this strong assumption, Naive Bayes often performs well in practice, even when there's a moderate degree of feature correlation.

3. Handles Categorical and Continuous Data Different Variants:

e. Naive Bayes has different variations for different data types: Gaussian Naive Bayes: For continuous data with normal distribution assumption Multinomial Naive Bayes:

f. For discrete count data, often used in text classification .Bernouli Naive Bayes:

g. For binary data or binary features, also common in text classification with binary term frequency features.. :

4. Performs Well on Text Classification Problems Widely Used for NLP:

- Navy Bayes has frequently used in text classification, spam filtering, and sentiment analysis due to its efficiency and effectiveness in dealing with word frequency features in documents.

5. Robust to Irrelevant Feature Noise Tolerance:

- Naive Bayes can handle irrelevant features well. Since it calculates probabilities independently, features that don't contribute to the outcome don't impact the prediction much.

6. Classification and Prediction:

- Once trained, the NB model can classify new data points by calculating on which side of the hyperplane they lie.

  For example, when new physiological data is input into the model, NB predicts whether it falls within the "stressed" or "not stressed" region, based on the position.

7. Handling Multiclass ClassificaVtion:

- In cases where stress is categorized into multiple levels (e.g., low, moderate, high stress), SVM can be extended with approaches like **One-vs-One** or **One- vs-Rest** to classify among these levels.

IMPORTANCE OF NAIVE BAYES ALGORITHM:

SIMPLICITY:-

Effective with Limited Data: Collecting large datasets for stress detection can be challenging, especially with physiological data. SVM performs well on small to medium-sized datasets by focusing only on support vectors, which are the critical points closest to the classification boundary. This allows SVM to deliver accurate predictions even when training data is limited.

Handles High-Dimensional Data: Stress detection models often include a range of features, such as heart rate, skin conductance, body temperature, and facial expressions, resulting in high-dimensional data. SVM excels in these scenarios, effectively handling many features and maintaining performance.

Non-Linear Classification with Kernel Functions: Stress data often has non-linear patterns; for instance, changes in physiological signals can have complex relationships with stress levels. SVM's kernel trick (using kernels like radial basis function or polynomial) allows it to transform data into a higher-dimensional space, where non-linear relationships become linear and separable. This enhances its ability to detect complex stress patterns accurately.

ADVANTAGES:

Simple: It's considered a simpler classifier than others, with parameters that are easier to estimate. It's often one of the first algorithms taught in data science and machine learning courses.

Fast and scalable: It's fast and efficient, and scales linearly with the number of predictors and rows. It can also be used for parallel execution.

Handles high-dimensional data: It can handle high-dimensional data, which can be difficult for other classifiers.

Can handle both continuous and discrete data: It can handle both continuous and discrete data.

Doesn't require a lot of training data: It doesn't require as much training data as other models.

3.2 SYSTEM CONFIGURATION

REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

REQUIREMENT SPECIFICATION

FUNCTIONAL REQUIREMENTS:

- Graphical User interface with the User.

SOFTWARE REQUIREMENTS:

For developing the application, the following are the Software Requirements:

1. Python
2. Django

Operating Systems supported

1.Windows 10.

Technologies and Languages used to Develop

1. Python

Debugger and Emulator

Any Browser (Particularly Chrome)

CHAPTER 4
SYSTEM REQUIREMENTS AND SPECIFICATION
4.1 Database

Dataset	Description	Key	Application
LIAR Dataset	A dataset containing 12,800 labeled short statements from various news sources. Each statement is classified as either true, mostly true, half true, mostly false, or false.	Labels: True, Mostly True, Half True, Mostly False. Statements from politicians and public figures.Text-based features like sentiment, subjectivity	Fake news detection Based on the truthfulness of public statements. - Political discourse analysis.
BuzzFeed News Dataset	Contains 15,000 articles and headlines labeled as fake or real, focusing on online rumors and hoaxes. Often includes sensationalized headlines.	- Labels: Fake or Real. -Features include headline length, clickbait indicators, sentiment analysis. -Focus on sensationalized headlines.	- Detection of clickbait An and sensationalist headlines. -Identifying misleading or fake news articles.
Fake News Detection Dataset	A collection of news articles from various online sources, labeled as fake or real. The dataset includes articles with different political biases and real-world events.	- Labels: Fake or Real. -Textual features: keywords, n-grams, and sentiment. -Source and author information. -Contextual features like article engagement.	- Fake news classification. -Detecting biased news articles. -Identifying misinformation in news
FNSPID (Fake News Spoofing and Impact Dataset)	A dataset designed for the detection of spoofed News articles, containing text and metadata of news articles published	- Text-based features (headline, body). -Source credibility and metadata. -Impact metrics (shares, likes, etc.).	- Detecting fake news based on both content and impact (social media engagement). - News credibility analysis.

Kaggle Fake News Dataset	A popular dataset from Kaggle containing 20,000 labeled news articles with binary labels (Fake or Real). The dataset includes metadata like author and publisher.	- Labels: Fake or Real. -Features: Text content, author/publisher, keywords, n-grams. -Article engagement and social sharing data.	-Classifying news articles as fake or real. -Fake news detection for automated content moderation systems.

4.2 ALGORITHMS

1. SUPPORT VECTOR MACHINE

SVM is particularly useful for classifying large datasets where data points are difficult to separate, as it tries to find the optimal boundary (or "hyperplane") that best distinguishes different financial events, such as fraud detection or credit scoring.

SVM STEPS:

1.Data Collection & Preprocessing: Gather financial transaction data from various sources like customer records, bank statements, and ERP systems. Clean and preprocess this data to remove noise, missing values, and irrelevant information.

2.Feature Extraction: Extract important features from the transaction data. These features could include transaction amounts, dates, account types, and other relevant financial indicators.

3. SVM Model Training: Use the preprocessed and feature-extracted data to train the SVM model.

4.Prediction and Classification: Once trained, the SVM model is applied to new, unseen data. It classifies each transaction based on the hyperplane, predicting whether it belongs to a particular class. 5.Continuous Learning and Optimization: ChatFin leverages self-learning AI to improve its predictions over time. The model is continuously updated with new financial data to refine the hyperplane and make more accurate future predictions.

2. RANDOM FOREST

This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance. In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks).

RANDOM FOREST-STEPS:

1. Data Preparation & Feature Selection

Collect customer data like transaction history, demographics, and interactions with the banking platform.

2. Create Multiple Decision Trees (Bootstrap Sampling)

Randomly select subsets of the data (with replacement) to create multiple decision trees. Each tree is trained on a different subset, ensuring variety and reducing bias.

3.Train Decision Trees Independently Each decision tree learns from its subset of data and makes predictions independently.

4.Aggregate Predictions (Ensemble Approach)

For classification tasks (e.g., predicting fraud), each tree casts a "vote" for the outcome. The majority vote across all trees becomes the final prediction.

For regression tasks (e.g., predicting loan default probability), the average of the predictions from all trees is taken

5.Model Evaluation & Optimization Evaluate the model performance using accuracy, precision, recall, or other relevant metrics.

ADVANTAGES:

1. Resistance to Overfitting
2. Large Datasets Handling
3. High Predictive Accuracy

3. ARTIFICIAL NEURALNETWORK(ANN)

ANN is a machine learning algorithm inspired by the structure and functioning of the human brain. ANNs are composed of nodes, or neurons, organized in layers (input, hidden, and output layers) that process and transform information. They are used for tasks such as classification, regression, and pattern recognition due to their ability to model complex relationships.

Steps to Implement an ANN Algorithm

1. Data Collection and Preprocessing: Gather and preprocess the data (normalize, scale, or encode) to ensure it's suitable for training.
2. Initialize Network Architecture: Define the structure, including the number of layers, number of neurons per layer, and activation functions (e.g., ReLU, Sigmoid).
3. Forward Propagation: Pass inputs through the network layer by layer, applying weights, biases, and activation functions to calculate output values.
4. Calculate Loss: Use a loss function (like Mean Squared Error for regression or Cross-Entropy for classification) to determine the difference between the predicted output

and actual target values.

5. Backpropagation: Calculate gradients of the loss with respect to weights and biases, adjusting weights in each layer to minimize the error.
6. Update Weights: Use an optimization algorithm like Gradient Descent or Adam to update weights iteratively and minimize the loss.
7. Train and Test: Repeat forward propagation, loss calculation, and backpropagation across multiple epochs. Evaluate performance on test data to avoid overfitting.

4.2 DESIGN

4.2.1 SYSTEM ARCHITECTURE

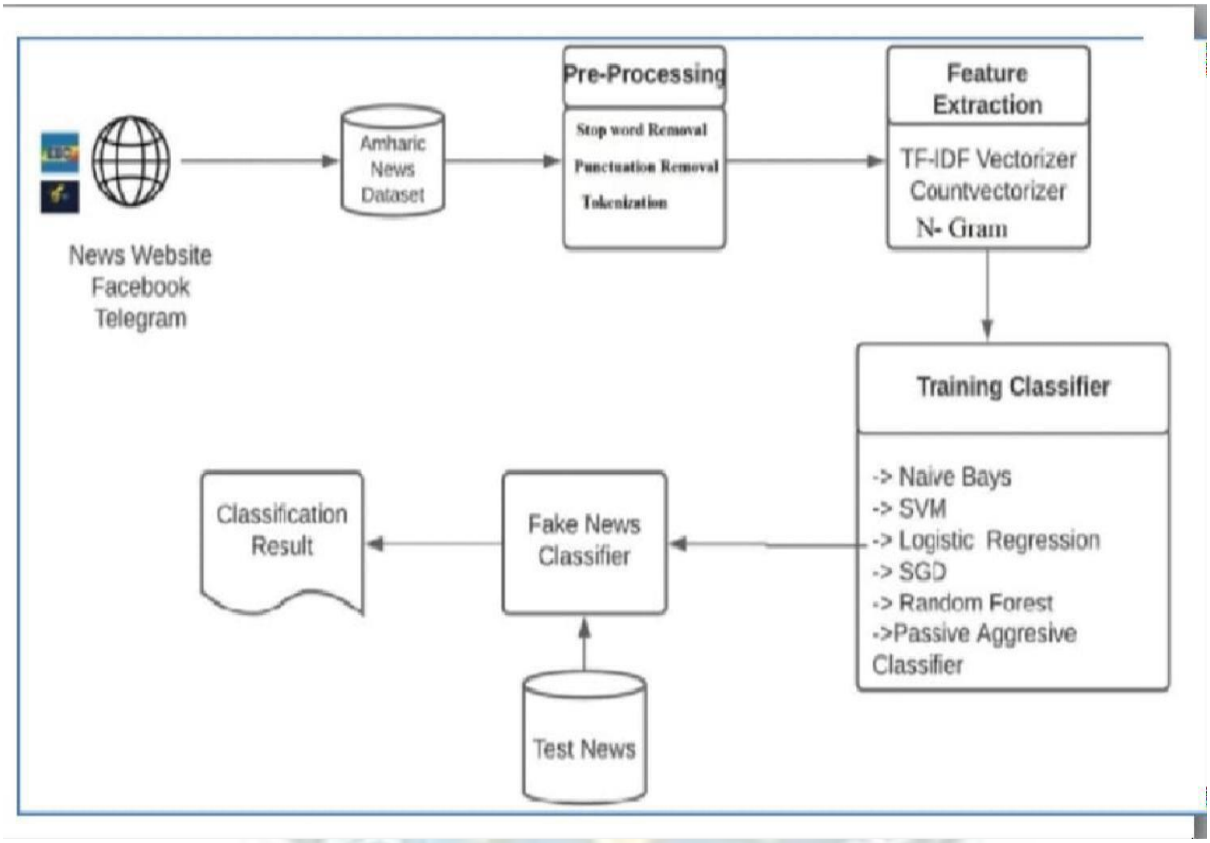


Figure 4.2.1 System Architecture

Fake news or information disorder is false or misleading information (misinformation, including disinformation, propaganda, and hoaxes) presented as news. Fake news often has the aim of damaging the reputation of a person or entity, or making money through advertising revenue. Although false news has always been spread throughout history, the term fake news was first used in the 1890s when sensational reports in newspapers were common. Nevertheless, the term does not have a fixed definition and has been applied broadly to any type of false information presented as news. It has also been used by high-profile people to apply to any news unfavorable to them.

4.2.2 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a visual representation of the flow of data within a system or process. It is a structured technique that focuses on how data moves through different processes and data stores within an organization or a system. DFDs are commonly used in system analysis and design to understand, document, and communicate data flow and processing.

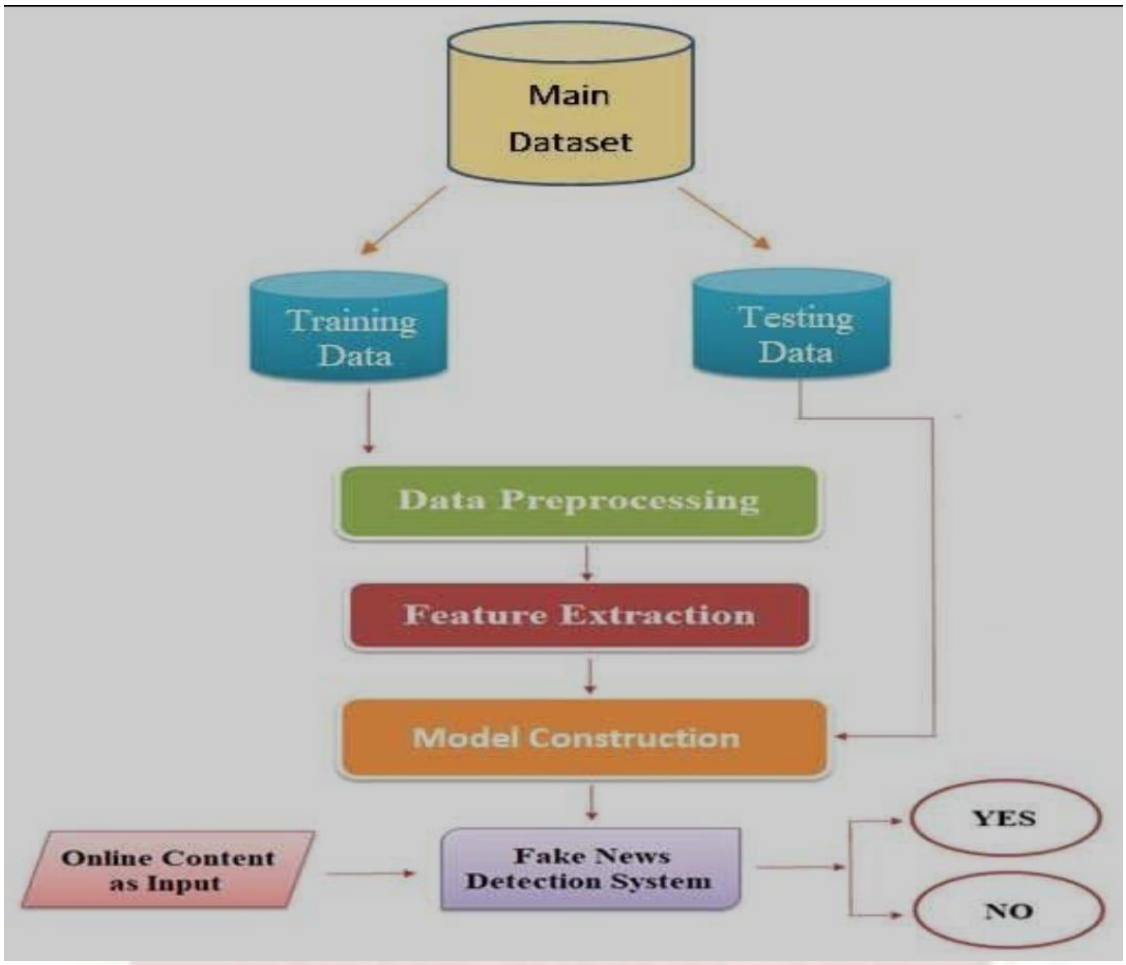


Figure 4.2.2 Data flow

4.2.3 UML DIAGRAM

UML stands for Unified Modeling Language. UML is a standardized general- purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. The Unified Modeling Language Is a standard language for Specifying, In its current form UML is comprised of two major components: a Meta- model Visualization and Documenting the artifacts of software system, as well as for tIn the future, some form of method or process may also be added to; or modeling and other non-software systems. The UML represents a collection associated with, UML of best engineering practices that have proven successful in the modeling of large and complex

GOALS: The Primary goals in the design of the UML are as systems. The UML is a very important part of developing objects-oriented follows:

the software development process. The UML uses mostly graphical notations toexprtehsast they can develop and exchange meaningful models. Provide extendibility and specialization mechanisms to extend the core the design of software projects.

concepts.

4.2.4 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

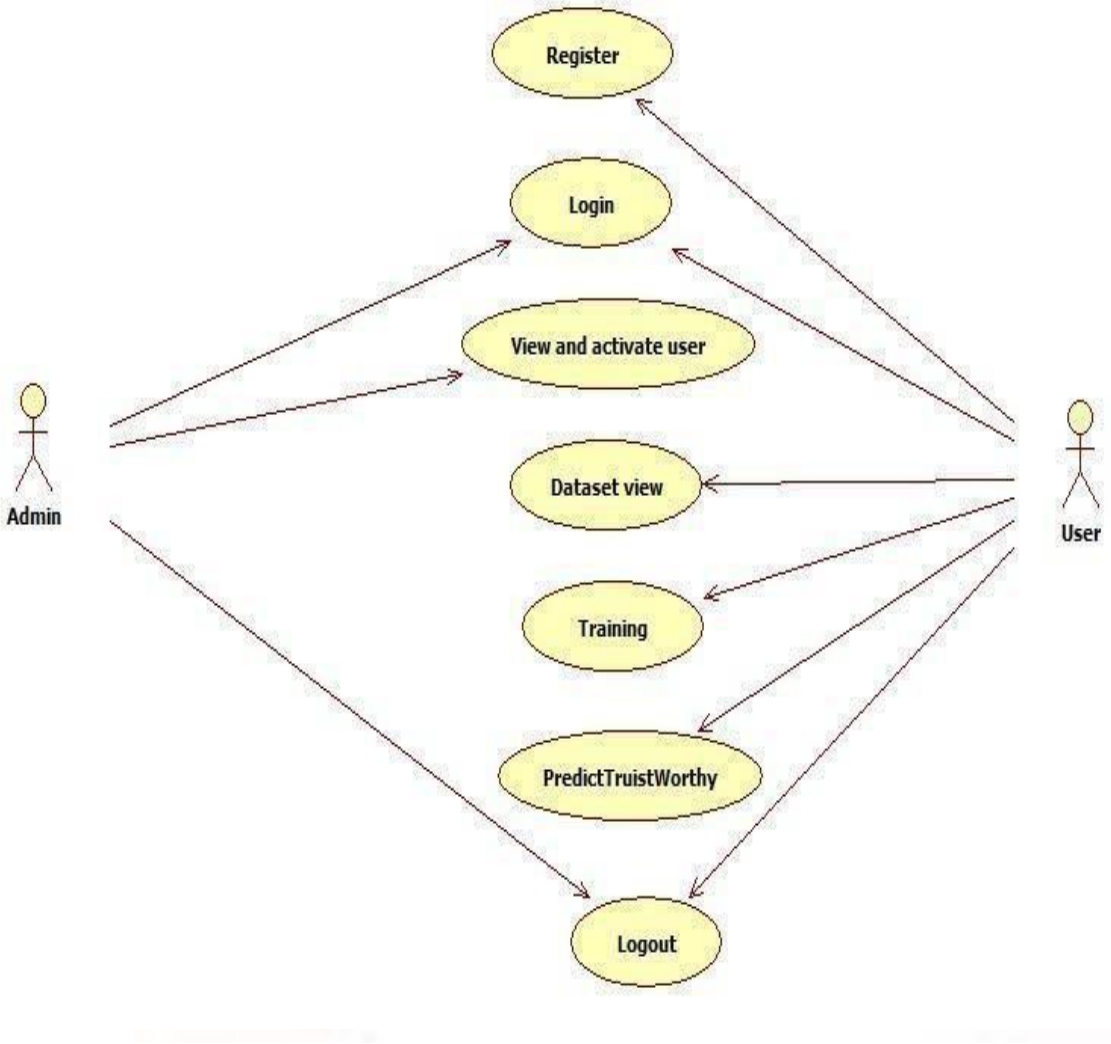


Figure 4.2.4 Use Case Diagram

4.2.5 CLASS DIAGRAM

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an “is-a” or “has-a” relationship. Each class in the class diagram may be capable of providing certain functionalities.

These functionalities provided by the class are termed “methods” of the class. Apart from this, each class may have certain “attributes” that uniquely identify the class.

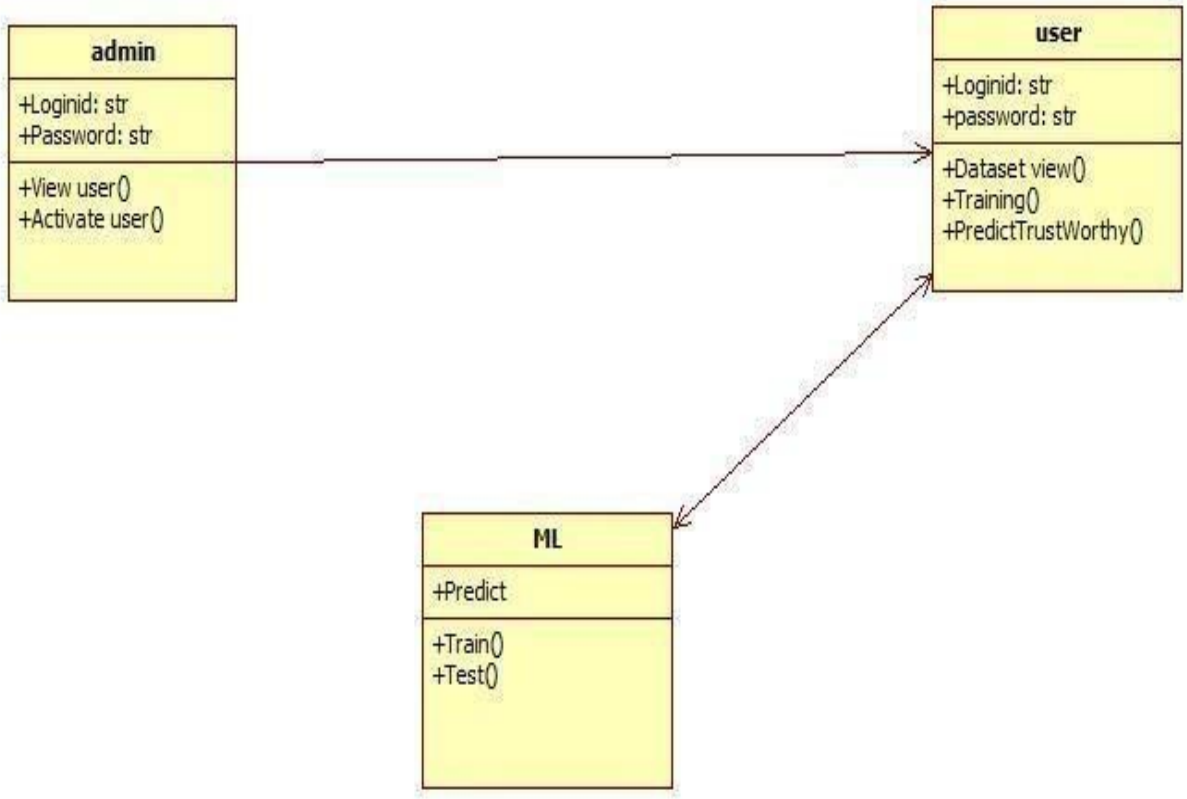


Figure 4.2.5 Class Diagram

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an “is-a” or “has-a” relationship. Each class in the class diagram may be capable of providing certain functionalities

A class diagram is a crucial part of the Unified Modeling Language (UML) that represents the static structure of a system by displaying its classes, attributes, operations, and the relationships among objects. This diagram is widely used in object-oriented modeling to illustrate how different classes interact within a system, providing a blueprint for constructing and organizing code. Each class in the diagram is represented by a rectangular box divided into three sections: the top section holds the class name, the middle section lists the attributes or properties of the class, and the bottom section displays the operations or methods the class can perform. Relationships between classes are shown through various types of lines and symbols, such as associations, dependencies, generalizations, and realizations, each denoting different forms of interaction or hierarchy. Associations indicate a relationship where instances of one class connect to instances of another, while inheritance is shown through generalizations, implying a superclass-subclass relationship. Aggregation and composition, special forms of association, are used to represent whole-part relationships between classes, where composition indicates a stronger dependency. Class diagrams offer a visual means of organizing complex software systems, providing developers with a clear and organized view of how classes and objects should interact, enabling more effective system design and implementation.

4.2.6 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

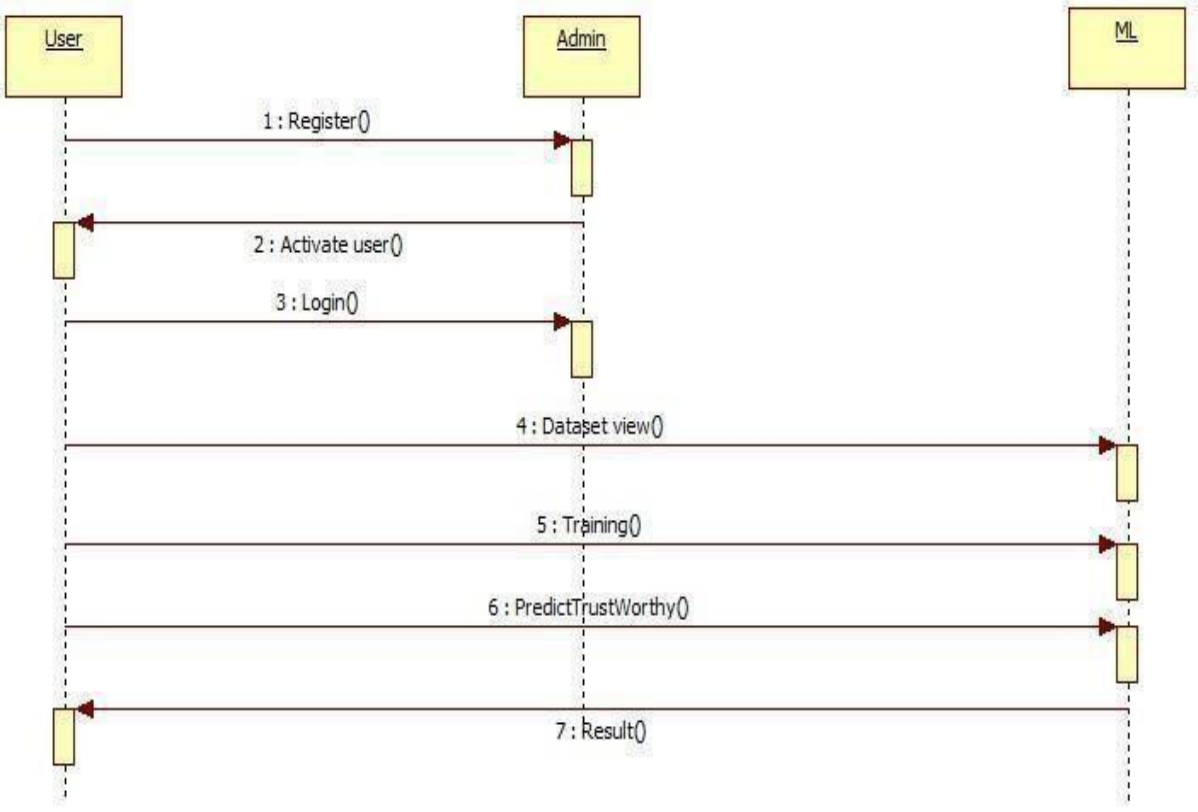


Figure 4.2.6 Sequence Diagram

4.2.7 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control

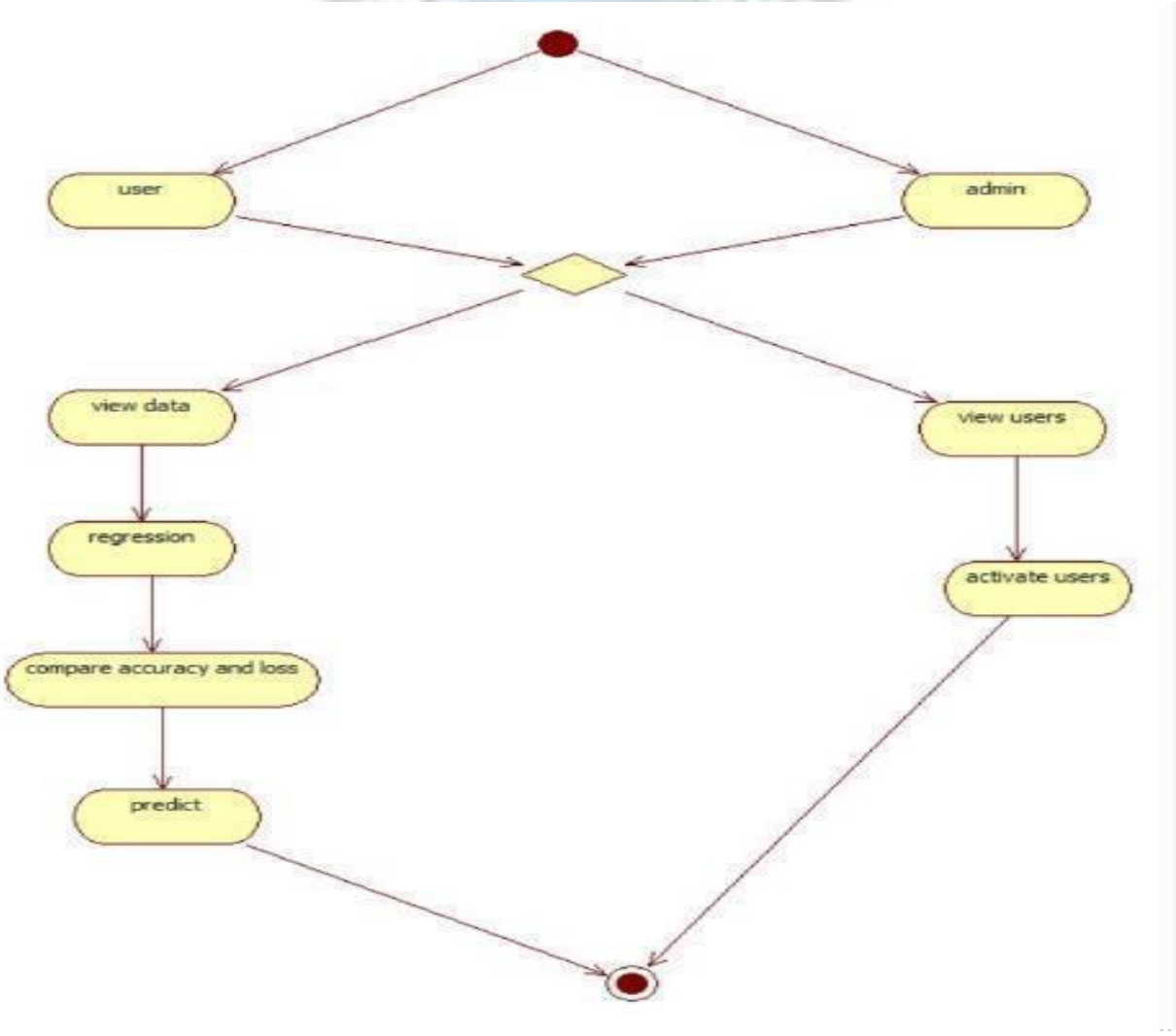


Figure 4.2.7 Activity Diagram

An activity diagram is a type of UML (Unified Modeling Language) diagram that models the flow of control or activities within a system, focusing on the sequence of actions and the flow of data. Activity diagrams are often used to represent the dynamic aspects of a system, highlighting how processes or operations occur step by step. They can be used to model both business workflows and system-level processes, making them versatile tools for both software engineering and business process modeling. An activity diagram essentially depicts the workflow of a system, showing how different activities or actions interrelate.The structure of an activity diagram is composed of several key elements: activities (depicted as rounded rectangles), transitions (arrows representing the flow of control), decision nodes (diamond shapes indicating conditional branches), merge nodes (to combine alternative paths), and start and end nodes (represented by filled circles and concentric circles respectively). Activity diagrams can also include forks and joins, which are used to represent parallel processes. For example, a fork splits the flow into multiple parallel paths, while a join merges them back into a single flow. These elements provide a comprehensive view of the system's processes and their dependencies.Activity diagrams are often used for both high-level and low-level modeling. High-level activity diagrams can depict broad workflows, such as the overall flow of a business process, while more detailed diagrams can describe individual system operations, like the interactions between user input and database queries in a software application. They are helpful in visualizing complex processes, understanding the dependencies between activities, and identifying potential bottlenecks or areas for optimization. In software development, activity diagrams are particularly useful during the design phase to specify workflows, use cases, and interactions between system components

One of the strengths of activity diagrams is their ability to model both sequential and parallel processes. This feature allows them to represent scenarios that involve concurrency, such as multiple users interacting with a system simultaneously or tasks that can proceed independently. Furthermore, activity diagrams are effective in illustrating the flow of information within a process, making them invaluable for system analysis and design, especially when dealing with complex workflows or systems with multiple states and interaction**S.**

4.3 MODULES

User Admin

Data Preprocessing Machine Learning Results

MODULES DESCRIPTION

User: The User can register the first. While registering he required a valid user email and mobile for further communications. Once the user register then admin can activate the user. Once admin activated the user then user can login into our system. User can upload the dataset based on our dataset column matched. For algorithm execution data must be in float format. Here we took human details dataset. User can also add the new data for existing dataset based on our Django application. User can click the Classification in the web page so that the data calculated Accuracy based on the algorithms.

Admin: Admin can login with his login details. Admin can activate the registered users. Once he activate then only the user can login into our system. Admin can view the overall data in the browser. Admin can click the Results in the web page so calculated Accuracy based on the algorithms is displayed. All algorithms execution complete then admin can see the overall accuracy in web page.

Data Pre-processing:

A dataset can be viewed as a collection of data objects, which are often also called as a records, points, vectors, patterns, events, cases, samples, observations, or entities .Data objects are described by a number of features that capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurred, etc. Features are often called as variables, characteristics, fields, attributes, or dimensions. The data preprocessing in this forecast uses techniques like removal of noise in the data, the expulsion of missing information, modifying default values if relevant and grouping of attributes for prediction at various levels.

4. SYSTEM REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that needto store large arrays/objects in memory will require more RAM, whereas Need to perform numerous calculations or tasks more quickly requirement is faster.

1. HARDWARE REQUIREMENTS:

- System : Intel Core i3.
- Hard disk : 1TB.
- Monitor : 15''LED.
- Input devices : Keyboard, Mouse.
- Ram : 8GB.

4.4.2 SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics The appropriation of requirements and implementation constraints gives the requirements, overview of the project.

- Operating system : Windows 10.
- Coding Language : Python.
- Tool : PyCharm, Visual Studio Code.
- Database : SQLite

5. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirements.

TYPES OF TESTS

1. UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2. INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

3. FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4. SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5. WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6. BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

CHAPTER 5
SOURCE CODE

```
from ast import alias
from concurrent.futures import process
from django.shortcuts import render


# Create your views here.
from django.shortcuts import render, HttpResponse
from django.contrib import messages


import Fake_Profile_Identification_using_ANN


from .forms import UserRegistrationForm
from .models import UserRegistrationModel
from django.conf import settings
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as plticker
import datetime as dt
from sklearn import preprocessing, metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn import metrics


from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
```

```
def UserRegisterActions(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            print('Data is Valid')
            form.save()
            messages.success(request, 'You have been successfully registered')
            form = UserRegistrationForm()
            return render(request, 'UserRegistrations.html', {'form': form})
        else:
            messages.success(request, 'Email or Mobile Already Existed')
            print("Invalid form")
    else:
        form = UserRegistrationForm()
    return render(request, 'UserRegistrations.html', {'form': form})


def UserLoginCheck(request):
    if request.method == "POST":
        loginid = request.POST.get('loginid')
        pswd = request.POST.get('pswd')
        print("Login ID = ", loginid, ' Password = ', pswd)
        try:
            check = UserRegistrationModel.objects.get(
                loginid=loginid, password=pswd)
            status = check.status
            print('Status is = ', status)
            if status == "activated":
                request.session['id'] = check.id
                request.session['loggeduser'] = check.name
                request.session['loginid'] = loginid
                request.session['email'] = check.email
                print("User id At", check.id, status)
                return render(request, 'users/UserHomePage.html', {})

else:
```

```
                messages.success(request, 'Your Account Not at activated')
                return render(request, 'UserLogin.html')
        except Exception as e:
            print('Exception is ', str(e))
            pass
        messages.success(request, 'Invalid Login id and password')
    return render(request, 'UserLogin.html', {})


def UserHome(request):

    return render(request, 'users/UserHomePage.html', {})


def DatasetView(request):
    path = settings.MEDIA_ROOT + "//" + 'train.csv'
    df = pd.read_csv(path, nrows=100)
    df = df.to_html
    return render(request, 'users/viewdataset.html', {'data': df})

def training(request):
    import warnings
    warnings.filterwarnings("ignore")


    import pandas as pd
    import matplotlib.pyplot as plt
    import numpy as np
    import seaborn as sns

    sns.set_style("darkgrid")
    sns.set_palette("pastel")
    plt.rcParams.update({'font.size': 20})

    import tensorflow as tf
    from tensorflow.keras.layers import Dense, Dropout

    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import classification_report,accuracy_score,roc_curve,confusion_matrix
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

# Загружаем набор данных для обучения
import os
path=os.path.join(settings.MEDIA_ROOT,'train.csv')
train = pd.read_csv(path)
train

# Загружаем данные для тестирования
path1=os.path.join(settings.MEDIA_ROOT,'test.csv')
test = pd.read_csv(path1)
test

train.info()
train.describe()
plt.figure(figsize=(20,10))
sns.countplot(train["fake"])
plt.show()

plt.figure(figsize=(20,10))
sns.countplot(train["private"])
plt.show()

plt.figure(figsize=(20,10))
sns.countplot(train["profile_pic"])
plt.show()

fake = train[train["fake"] == 1]
not_fake = train[train["fake"] == 0]
fig = plt.figure(figsize = (20, 10))
sns.distplot(fake["nums_length_username"], label = "fake")
sns.distplot(not_fake["nums_length_username"], label = "not fake")
fig.legend()
plt.show()

plt.figure(figsize=(20, 20))
cm = train.corr()
ax = plt.subplot()
sns.heatmap(cm, annot = True, ax = ax)
plt.show()

X_train = train.drop(columns = ['fake'])
X_train = train.drop(columns = ['fake'])
X_test = test.drop(columns = ['fake'])

# Набор данных для обучения и тестирования (выходные данные)
y_train = train['fake']
y_test = test['fake']

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
scaler_x = StandardScaler()
X_train = scaler_x.fit_transform(X_train)
X_test = scaler_x.transform(X_test)

y_train = tf.keras.utils.to_categorical(y_train, num_classes = 2)
y_test = tf.keras.utils.to_categorical(y_test, num_classes = 2)

model = Sequential()
model.add(Dense(50, input_dim = 11, activation = 'relu'))
model.add(Dense(150, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(25, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(2, activation = 'softmax'))
model.summary()

model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
epochs_hist = model.fit(X_train, y_train, epochs = 20, verbose = 1, validation_split = 0.1)
print(epochs_hist.history.keys())

model.save("model.h5")

accuracy = epochs_hist.history['accuracy']
val_accuracy = epochs_hist.history['val_accuracy']

loss  = epochs_hist.history['loss']
val_loss = epochs_hist.history['val_loss']

plt.figure(figsize=(20,10))
plt.plot(epochs_hist.history['accuracy'])
plt.plot(epochs_hist.history['val_accuracy'])

plt.plot(epochs_hist.history['loss'])
plt.plot(epochs_hist.history['val_loss'])

plt.title('Model Loss Progression During Training/Validation')
plt.ylabel('Training and Validation Losses')
plt.xlabel('Epoch  Number')
plt.legend(['Training Loss', 'Validation Loss'])
plt.show()

predicted = model.predict(X_test)

predicted_value = []
test = []
for i in predicted:
    predicted_value.append(np.argmax(i))

for i in y_test:
    test.append(np.argmax(i))
```

```
print(classification_report(test,   predicted_value))

    plt.figure(figsize=(6, 6))
    cm = confusion_matrix(test, predicted_value)
    sns.heatmap(cm, annot=True)
    plt.show()

    return render(request, "users/training.html", {"accuracy": accuracy, "loss": loss})

def predictTrustWorthy(request):
    if request.method == 'POST':
        # Extracting data from the POST request
        profile_pic = request.POST.get("profile_pic")
        nums_length_username = request.POST.get("nums_length_username")
        fullname_words = request.POST.get("fullname_words")
        nums_length_fullname = request.POST.get("nums_length_fullname")
        name_username = request.POST.get("name_username")
        description_length = request.POST.get("description_length")
        external_URL = request.POST.get("external_URL")
        private = request.POST.get("private")
        posts = request.POST.get("posts")
        followers = request.POST.get("followers")
        follows = request.POST.get("follows")

        # Loading the dataset
        path = settings.MEDIA_ROOT + '/' + 'train.csv'
        df = pd.read_csv(path)
        data = df.dropna()

        # Check and remove unexpected values in 'passed' column
        data['fake'] = data['fake'].apply(lambda x: 1 if x == 'yes' else 0)

        # Selecting relevant columns for training
        features = ['profile_pic','nums_length_username','fullname_words','nums_length_fullname',
'name_username','description_length','external_URL','private','posts','followers','follows']

        X = pd.get_dummies(data[features])


        # Creating the test set
        test_set = {'profile_pic': profile_pic, 'nums_length_username': nums_length_username,
'fullname_words': fullname_words,
                    'nums_length_fullname': nums_length_fullname, 'name_username': name_username,
'description_length': description_length,
                    'external_URL': external_URL, 'private': private,'posts': posts, 'followers': followers,
'follows': follows}

        # Creating a DataFrame for the test set
        test_df = pd.DataFrame([test_set])

        # One-hot encoding the test set
        test_df = pd.get_dummies(test_df)
```

```
        # Matching the columns between the training and test sets
        missing_cols = set(X.columns) - set(test_df.columns)
        for col in missing_cols:
            test_df[col] = 0

        # Reordering the columns to match the training set
        test_df = test_df[X.columns]

        # Preparing the data for training
        y = data['fake']
        X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_state=101)

        # Initializing and training the model
        OBJ = RandomForestClassifier(n_estimators=10, criterion='entropy')
        OBJ.fit(X_train, Y_train)

        # Making predictions
        print(test_df.values)
        y_pred = OBJ.predict(test_df.values)
        print(y_pred)

        if y_pred[0] == 0:
            msg = 'fake profile is 0'
        elif y_pred[0] == 1:
            msg = 'not fake profile is 1'
        else:
            msg = 'Invalid prediction category'

        return render(request, "users/predictForm.html", {"msg": msg})
    else:
        return  render(request,  'users/predictForm.html',  {})
```

Base.html :

```
{% load static %}
<!DOCTYPE  html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">

  <title>Arsha Bootstrap Template - Index</title>
  <meta content="" name="description">
  <meta content="" name="keywords">

  <!-- Favicons -->
  <link href="{% static 'assets/img/favicon.png' %}" rel="icon">
  <link href="{% static 'assets/img/apple-touch-icon.png' %}" rel="apple-touch-icon">

  <!-- Google Fonts -->
```

```
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i
|Jost:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600
i,700,700i" rel="stylesheet">

  <!-- Vendor CSS Files -->
  <link href="{% static 'assets/vendor/aos/aos.css' %}" rel="stylesheet">
  <link href="{% static 'assets/vendor/bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">
  <link href="{% static 'assets/vendor/bootstrap-icons/bootstrap-icons.css' %}" rel="stylesheet">
  <link href="{% static 'assets/vendor/boxicons/css/boxicons.min.css' %}" rel="stylesheet">
  <link href="{% static 'assets/vendor/glightbox/css/glightbox.min.css' %}" rel="stylesheet">
  <link href="{% static 'assets/vendor/remixicon/remixicon.css' %}" rel="stylesheet">
  <link href="{% static 'assets/vendor/swiper/swiper-bundle.min.css' %}" rel="stylesheet">

  <!-- Template Main CSS File -->
  <link href="{% static 'assets/css/style.css' %}" rel="stylesheet">

  <!-- =======================================================
  * Template Name: Arsha
  * Updated: Sep 18 2023 with Bootstrap v5.3.2
  * Template   URL:    https://bootstrapmade.com/arsha-free-bootstrap-html-template-corporate/
  * Author: BootstrapMade.com
  * License: https://bootstrapmade.com/license/
  ======================================================== -->
</head>

<body>
    <style>
        span{
            color:orangered;
        }
        header{
            background-color:GREEN;
        }
        body{
            background-image: url({% static 'assets/img/peddanna.jpeg' %});
        }
    </style>

  <!-- ======= Header ======= -->
  <header id="header" class="fixed-top ">
    <div  class="container  d-flex  align-items-center">

      <h1 class="logo me-auto"><a href="index.html"><span>Fake Profile Identification Using
ANN</span></a></h1>
      <!-- Uncomment below if you prefer to use an image logo -->
      <!-- <a href="index.html" class="logo me-auto"><img src="assets/img/logo.png" alt=""
class="img-fluid"></a>-->

      <nav id="navbar" class="navbar">
        <ul>
          <li><a class="nav-link scrollto active" href="{% url 'index' %}">Home</a></li>
```

```
          <li><a class="nav-link scrollto" href="{% url 'UserLogin' %}">User</a></li>
          <li><a class="nav-link   scrollto" href="{% url 'UserRegister' %}">Register</a></li>
        </ul>
        <i class="bi bi-list mobile-nav-toggle"></i>
      </nav><!-- .navbar -->

    </div>
  </header><!-- End Header -->

{% block contents %}

{% endblock %}

  <!-- ======= Footer ======= -->
  <footer id="footer">

  </footer><!-- End Footer -->

  <div id="preloader"></div>
  <a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi
bi-arrow-up-short"></i></a>

  <!-- Vendor JS Files -->
  <script src="{% static 'assets/vendor/aos/aos.js' %}"></script>
  <script src="{% static 'assets/vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
  <script src="{% static 'assets/vendor/glightbox/js/glightbox.min.js' %}"></script>
  <script src="{% static 'assets/vendor/isotope-layout/isotope.pkgd.min.js' %}"></script>
  <script src="{% static 'assets/vendor/swiper/swiper-bundle.min.js' %}"></script>
  <script src="{% static 'assets/vendor/waypoints/noframework.waypoints.js' %}"></script>
  <script src="{% static 'assets/vendor/php-email-form/validate.js' %}"></script>

  <!-- Template Main JS File -->
  <script src="{% static 'assets/js/main.js' %}"></script>

</body>

</html>
```

Admin views :

```
from django.shortcuts import render
from django.contrib import messages
from users.forms import UserRegistrationForm
from users.models import UserRegistrationModel

# Create your views here.
def AdminLoginCheck(request):
    if request.method == 'POST':
        usrid = request.POST.get('loginid')
        pswd = request.POST.get('pswd')
        print("User ID is = ", usrid)
```

```
        if usrid == 'admin' and pswd == 'admin':
            return render(request, 'admins/AdminHome.html')
        else:
            messages.success(request, 'Please Check Your Login Details')
    return render(request, 'AdminLogin.html', {})

def AdminHome(request):
    return  render(request,  'admins/AdminHome.html',{})

def RegisterUsersView(request):
    data = UserRegistrationModel.objects.all()
    return render(request,'admins/viewregisterusers.html',{'data':data})


def ActivaUsers(request):
    if request.method == 'GET':
        id = request.GET.get('uid')
        status = 'activated'
        print("PID = ", id, status)
        UserRegistrationModel.objects.filter(id=id).update(status=status)
        data = UserRegistrationModel.objects.all()
        return render(request,'admins/viewregisterusers.html',{'data':data})
```

CHAPTER 6
EXPERIMENTAL RESULTS

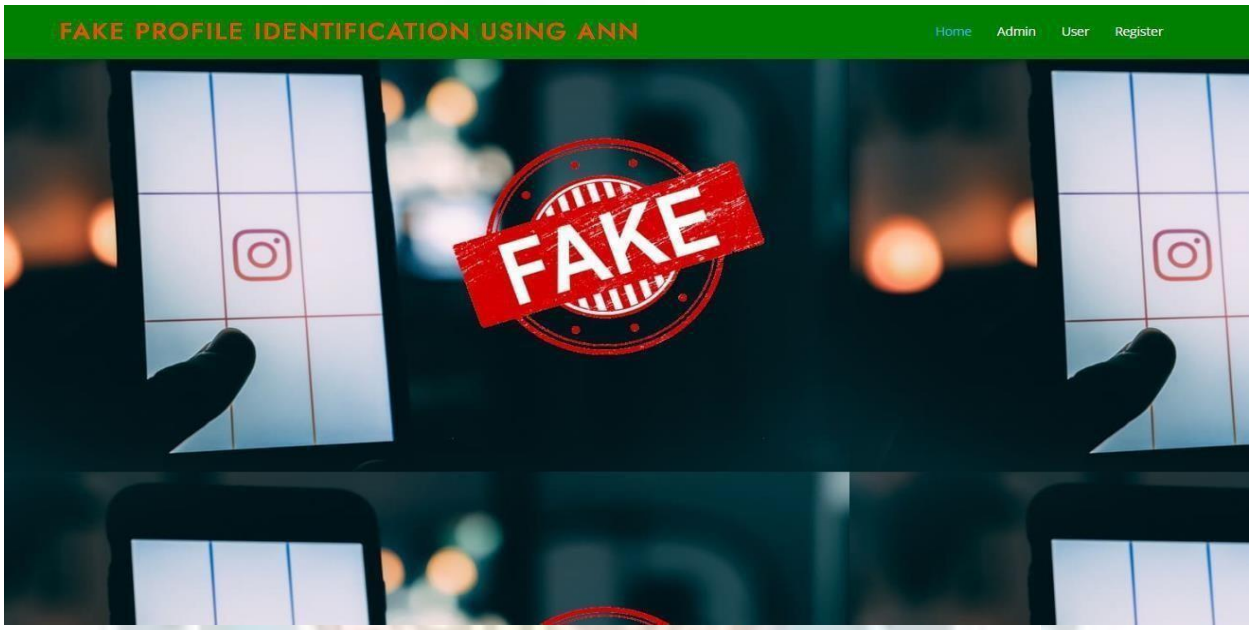


Figure 6.1 Home page

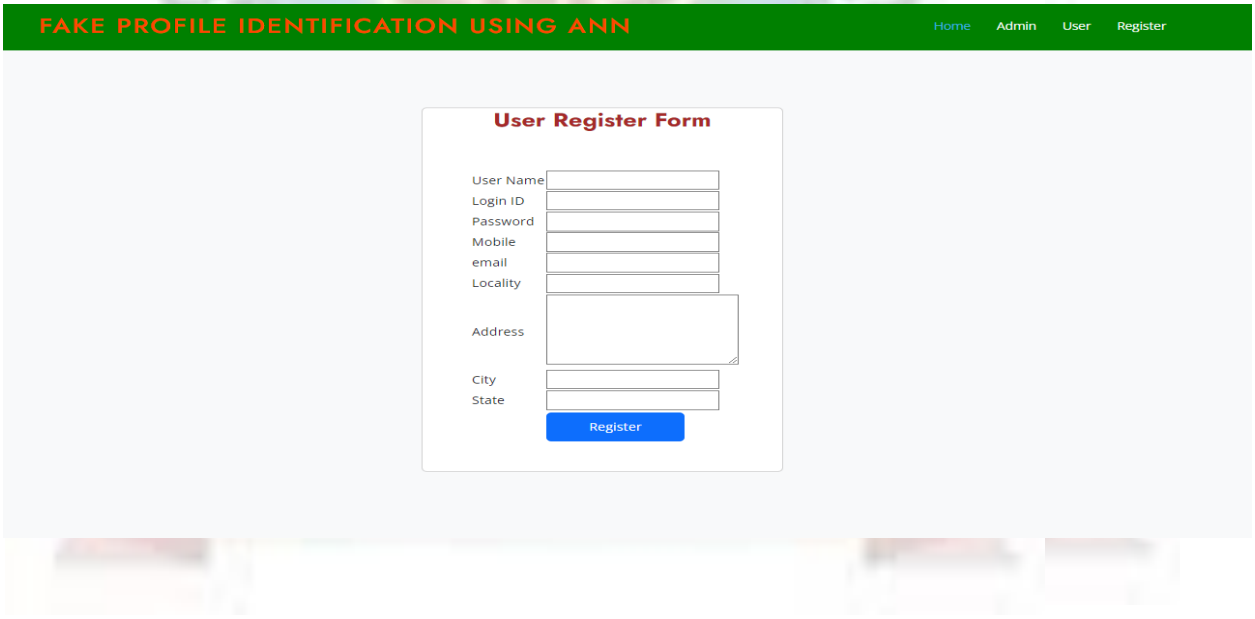


Figure 6.2 Register Form

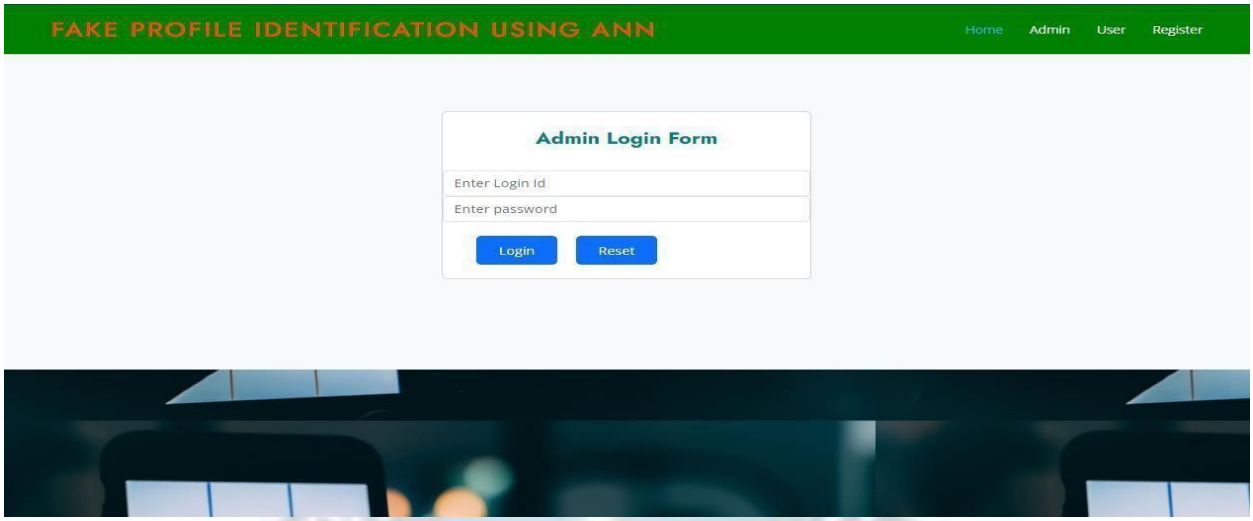
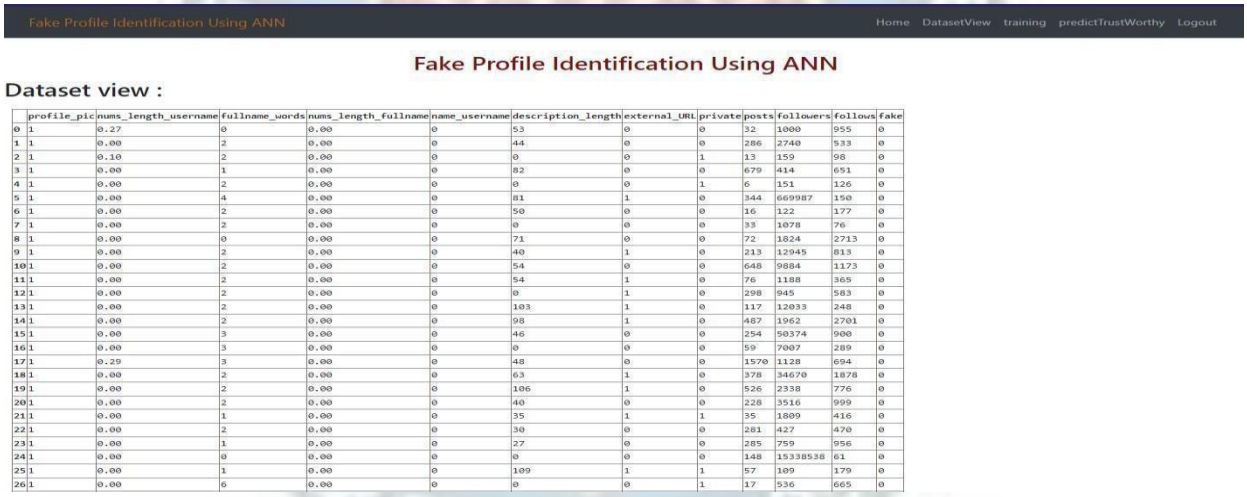


Figure 6.3 Admin Login Page



	profile_pic	nums_length_username	fullname_words	nums_length_fullname	name_username	description_length	external_URL	private	posts	followers	follows	fake
0	1	0.27	0	0.00	0	53	0	0	32	1000	955	0
1	1	0.00	2	0.00	0	44	0	0	286	2740	533	0
2	1	0.10	2	0.00	0	0	0	1	13	159	98	0
3	1	0.00	1	0.00	0	82	0	0	679	414	651	0
4	1	0.00	2	0.00	0	0	0	1	6	151	126	0
5	1	0.00	4	0.00	0	81	1	0	344	669987	150	0
6	1	0.00	2	0.00	0	50	0	0	16	122	177	0
7	1	0.00	2	0.00	0	0	0	0	33	1078	76	0
8	1	0.00	0	0.00	0	71	0	0	72	1824	2713	0
9	1	0.00	2	0.00	0	40	1	0	213	12945	813	0
10	1	0.00	2	0.00	0	54	0	0	648	9884	1173	0
11	1	0.00	2	0.00	0	54	1	0	76	1188	365	0
12	1	0.00	2	0.00	0	0	1	0	298	945	583	0
13	1	0.00	2	0.00	0	103	1	0	117	12033	248	0
14	1	0.00	2	0.00	0	98	1	0	487	1962	2701	0
15	1	0.00	3	0.00	0	46	0	0	254	50374	900	0
16	1	0.00	3	0.00	0	0	0	0	59	7007	289	0
17	1	0.29	3	0.00	0	48	0	0	1570	1128	694	0
18	1	0.00	2	0.00	0	63	1	0	378	34670	1878	0
19	1	0.00	2	0.00	0	106	1	0	526	2338	776	0
20	1	0.00	2	0.00	0	40	0	0	228	3516	999	0
21	1	0.00	1	0.00	0	35	1	1	35	1809	416	0
22	1	0.00	2	0.00	0	30	0	0	281	427	470	0
23	1	0.00	1	0.00	0	27	0	0	285	759	956	0
24	1	0.00	0	0.00	0	0	0	0	148	15338538	61	0
25	1	0.00	1	0.00	0	109	1	1	57	109	179	0
26	1	0.00	6	0.00	0	0	0	1	17	536	665	0

Figure 6.4 Admin Home Page

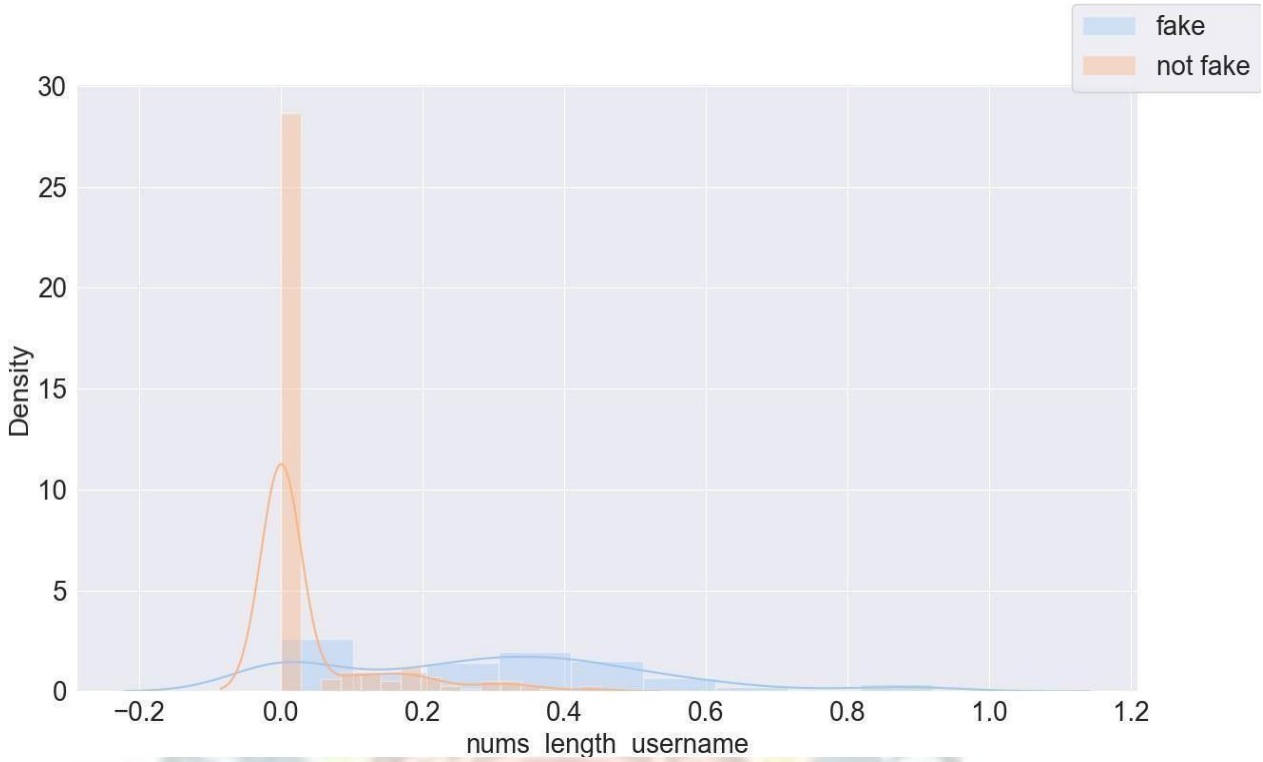


Figure 6.5 Activate User

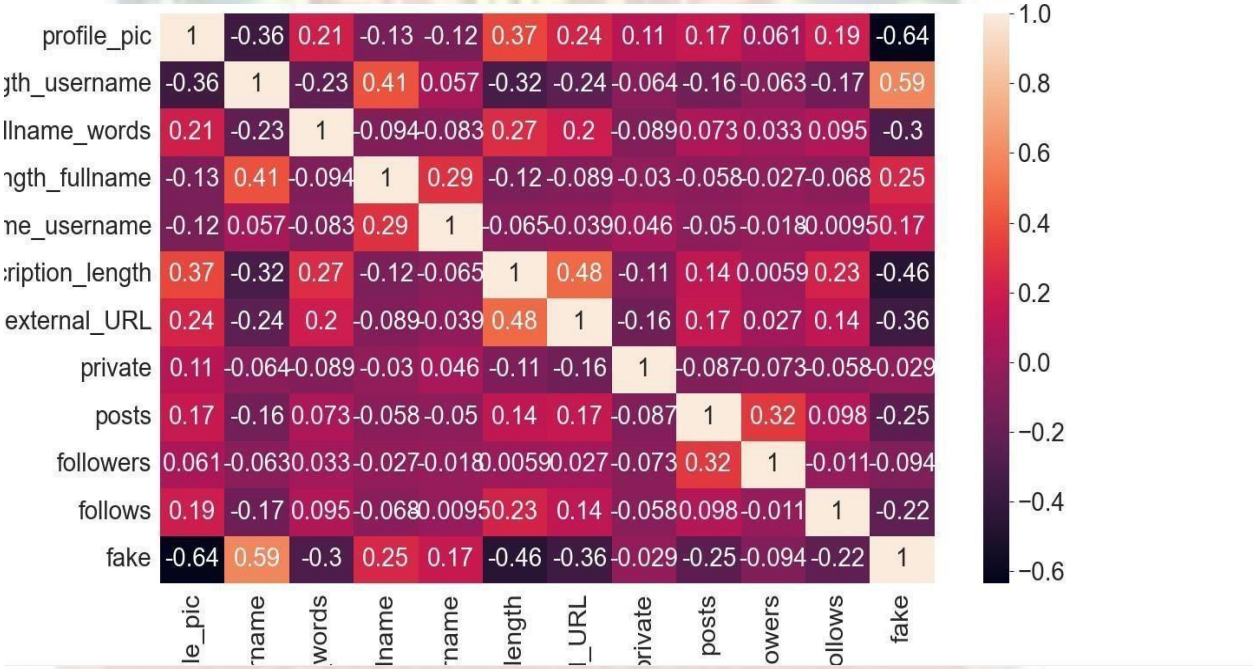


Figure 6.6 Admin side forecasting results

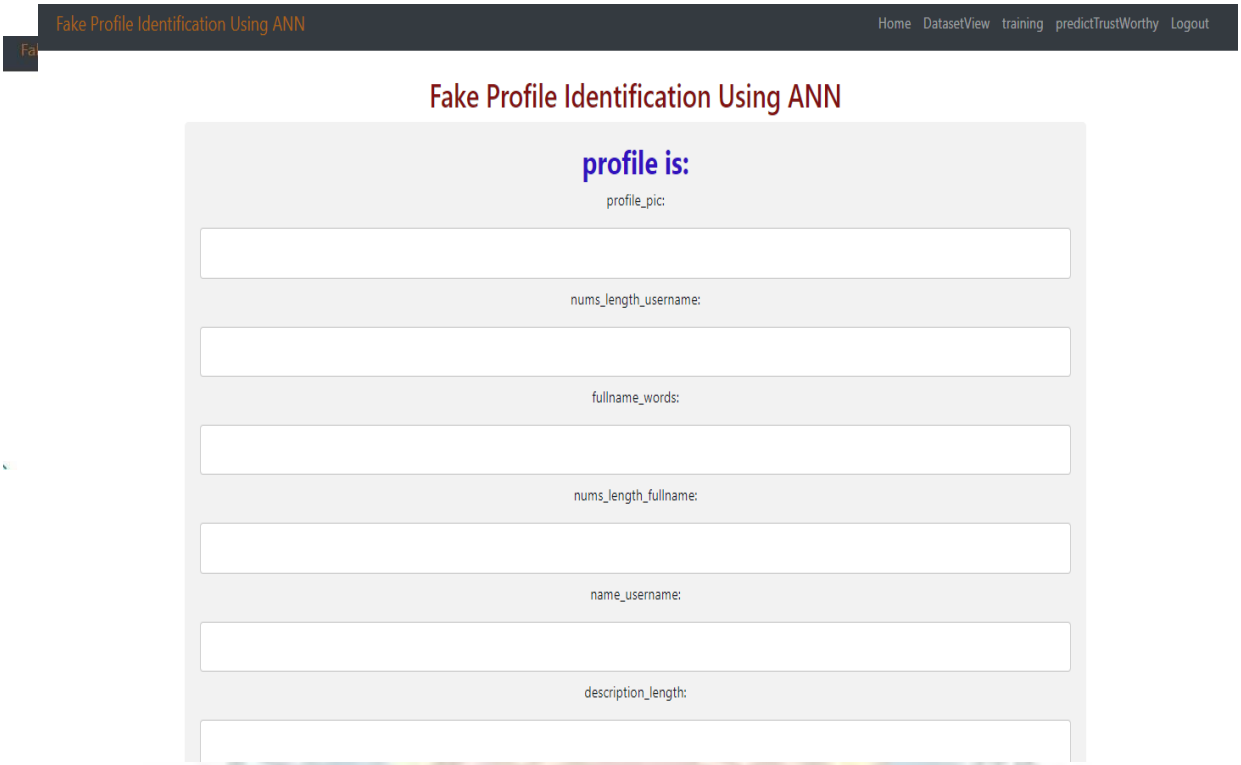


Figure 6.8 User

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

1. CONCLUSION

A comparison is done between 2 models trained with different data sets where one had more amount of data than the other. The results turned out better for the data set with more data in it. We have given a framework with the use of which we can become aware of fake profiles in any online social community via the usage of the Random Forest Classifier model and pipe-lining with a very excessive efficiency of as high as 95% on average.

2. FUTURE ENHANCEMENT

Beyond the aforementioned advancements, three additional frontiers promise thrilling developments in the fight against online personas. Imagine a system capable of recognizing deepfakes or AI-generated text in real-time, not just through brute force analysis but by comprehending the underlying manipulation techniques.

1. Multi-Modal Data Fusion

Integrate Diverse Data Sources: Combine data from wearables, voice analysis, facial expressions, and textual inputs to create a holistic model of stress. For example, wearable sensors could collect physiological data, while voice and text sentiment analysis provides emotional context.

Time-Series Analysis: Develop models that analyze time-series data from multiple sensors to detect patterns in physiological signals like heart rate, skin conductance, and respiratory rate, improving real-time stress detection

2. Self-Supervised and Semi-Supervised Learning

Reduce Labeling Effort: Use self-supervised learning methods to learn from unlabeled data and discover stress patterns independently, reducing the need for extensive labeled datasets.

Personalized Stress Models: Develop models that use self-supervised learning on individual user data, allowing for better adaptation to individual differences in stress response without extensive labeled training.

3. Emotion and Behavior Recognition

Voice and Facial Recognition Models: Advanced deep learning models like CNNs and RNNs could analyze subtle changes in facial expressions, voice pitch, or tone to detect signs of stress.
Natural Language Processing (NLP): In scenarios where text or voice input is available (e.g., chat or calls), NLP can analyze the choice of words, sentiment, and tone for stress indicators.

4. Context-Aware MLModels

Contextual Information: Incorporate contextual data (e.g., time of day, location, activity) into stress detection models to distinguish between situational and chronic stress, improving the relevance of predictions.
Transfer Learning for Environments: Use transfer learning to adapt models trained in one setting (e.g., an office environment) to other settings (e.g., home or outdoors), making them more versatile and effective across various environments.

5. Adaptive and Personalized Models

Reinforcement Learning for Real-Time Adaptation: Reinforcement learning could enable models to adapt to real-time feedback, optimizing the accuracy of stress detection over time for each user.
Meta-Learning for Rapid Personalization: Meta-learning models could quickly adapt to new users or changes in an individual's stress response patterns, providing tailored, accurate predictions with minimal retraining.

REFERENCES

[1]. B. Pang and L. Lee, “Opinion mining and sentiment analysis,” Found. Trends Inf. Retrieval, vol. 2, no. 1/2, pp. 1– 135, 2008.

[2]. J. Bollen, H. Mao, and A. Pepe, “Modeling public mood and emotion: Twitter sentiment and socio- economic phenomena,” in Proc. Int. AAAI Conf. Weblogs Social Media, 2011, pp. 17–21.

[3]. B. O’Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith, “From tweets to polls: Linking text sentiment to public opinion time series,” in Proc. Int. AAAI Conf. Weblogs Social Media, 2010, pp. 122– 129.

[4]. M. Hu and B. Liu, “Mining and summarizing customer reviews,” in Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2004, pp.168–177.

5]. Y. Wu, S. Liu, K. Yan, M. Liu, and F. Wu, “Opinion Flow: Visual analysis of opinion diffusion on social media,” IEEE Trans. Vis. Comput. Graph, vol. 20, no. 12, pp. 1763–1772, Dec. 2014.

[6]. B. Pang, L. Lee, and S.Vaithyanathan, “Thumbs up: Sentiment classification using machine learning techniques,” in Proc. ACL Conf. Empirical Methods Natural Language Process., 2002, pp. 79–86.)

[7]. A.Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” Stanford Univ., Stanford, CA, USA, Project Rep. CS224N, pp. 1–12, 2009.

[8]. F. Wu, Y. Song, and Y. Huang, “Microblog sentiment classification with contextual knowledge regularization,” in Proc. 29th AAAI Conf. Artif. Intell., 2015, pp. 2332–2338.

[[9].Kaliyar, R. K., Goswami, A., Narang, P., & Sinha, S. (2020). FNDNet–A deep convolutional neural network for
fake news detection. Cognitive Systems Research, 61, 32-44.

[10]. Kaur, S., Kumar, P. & Kumaraguru, P. (2020). Automating fake news detection system using multi-level voting model. Soft Computing, 24(12), 9049–9069.

[11]. Kesarwani, A., Chauhan, S. S., & Nair, A. R. (2020). Fake News Detection on Social Media using K-Nearest Neighbor Classifier. 2020 International Conference on Advances in Computing and Communication Engineering (ICACCE), Las Vegas, NV, USA, pp.1-4,

[12]. Khan, J. Y., Khondaker, M., Islam, T., Iqbal, A., & Afroz, S. (2019). A benchmark study on machine learning methods for fake news detection. Computation and Language.