

DETECTION OF MALICIOUS SOCIAL BOTS USING VARIATIONAL AUTOENCODER GAN

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE DEGREE OF
BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING



by

Batch - A1

G. Bhavya Yaraswini(18JG1A0525)

K. Niharika (18JG1A0549)

G. Ragini (18JG1A0530)

I. Sahithi (18JG1A0535)

Under the esteemed guidance of

Dr. P.V.S.L. Jagadamba

Professor

CSE Department

Department of Computer Science and Engineering

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

[Approved by AICTE NEW DELHI, Affiliated to JNTUK Kakinada]

[Accredited by National Board of Accreditation(NBA) for B.Tech. CSE, ECE & IT – Valid from 2019-20 to 2021-22]

Kommadi, Madhurawada, Visakhapatnam – 530048

2018 – 2022

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the project report titled “**DETECTION OF MALICIOUS SOCIAL BOTS USING VARIATIONAL AUTOENCODER GAN**” is a bonafide work of following IV/IV B. Tech students in the Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering for Women affiliated to JNT University, Kakinada during the academic year 2021-2022, in fulfilment of the requirement for the award of the degree of Bachelor of Technology of this university.

G. Bhavya Yaraswini (18JG1A0525)

K. Niharika (18JG1A0549)

G. Ragini (18JG1A0530)

I. Sahithi (18JG1A0535)

Dr. P.V.S.L. Jagadamba

Professor

(Internal Guide)

Dr. P.V.S.L. Jagadamba

Professor

(Head of department)

External Examiner

ACKNOWLEDGEMENT

We take the opportunity to thank one and all who have helped in making the project possible. We are thankful to **Gayatri Vidya Parishad College of Engineering for Women**, for giving us the opportunity to work on a project as part of the curriculum.

Our sincere thanks to our guide **Dr. P.V.S.L. Jagadamba**, Head of the Department of Computer Science and Engineering for her simulating guidance and assistance from the beginning of the project and also for providing the lab facility to complete the project work.

Our sincere thanks to our beloved Vice-Principal **Dr. G. Sudheer** for providing the best faculty and lab facility throughout these academic years.

Our sincere thanks to our beloved principal **Dr. R.K. Goswami** for providing the best faculty and lab facility throughout these academic years.

Finally we are thankful to our entire faculty and our lab technicians for their good wishes and constructive criticism, which led to the successful completion of the project.

TABLE OF CONTENTS

TOPICS	PAGE NO.
Abstract	i
List of Figures	ii
List of Tables	iii
List of Screens	iv
List of Acronyms	v
1. INTRODUCTION	1
1.1 Motivation of the Project	1
1.2 Problem definition	2
1.3 Objective of Project	2
1.4 Limitations of Project	2
1.5 Organization of documentation	3
2. LITERATURE SURVEY	4
2.1 Introduction	4
2.2 Existing System	5
2.3 Disadvantages of Existing System	5
2.4 Proposed System	6
2.5 Conclusion	8
3. ANALYSIS	9
3.1 Introduction	9
3.2 Requirements Specification	9
3.2.1 Software requirements	9
3.2.2 Hardware requirements	9
3.3 Content diagram	11
3.4 Algorithms and Flowchart	12
3.4.1 Algorithms	12
3.4.2 Flowchart	15

3.5 Conclusion	16
4. DESIGN	17
4.1 Introduction	17
4.2 UML diagrams	17
4.2.1 Use case diagram	17
4.2.2 Sequence diagram	19
4.2.3 Class diagram	20
4.3 Conclusion	21
5. IMPLEMENTATION AND RESULTS	22
5.1 Introduction	22
5.2 Explanation of Key functions	22
5.3 Implementation	22
5.3.1 Back-end code	23
5.3.2 Front-end code	30
5.4 Results	64
5.4.1 Result Analysis	64
5.4.2 Output Screens	68
5.5 Conclusion	73
6. TESTING AND VALIDATION	74
6.1 Introduction	74
6.1.1 Scope	74
6.1.2 Defects and Failures	74
6.2 Types of Testing	75
6.3 Design of Test Cases and Scenarios	76
6.4 Conclusion	76
7. CONCLUSION	77
8. REFERENCES	78

ABSTRACT

Online social networks (OSNs) have influenced users to share information related to social activities like news, links, opinion and promote products and services. It also provides an additional space for an attacker to steal a user's personal information and to perform malicious activities such as generating fake identities and performing phishing attacks in OSNs. Malicious social bots represents software programs that automatically creates multiple fake accounts, frequently interacts with legitimate user accounts. In many OSNs, the bots are much less in number when compared to legitimate users. Most of the conventional supervised learning models often suffer from a highly imbalanced dataset with many legitimate users belonging to one class and only a few malicious social bots belonging to another class. To solve the imbalanced distribution of malicious social bots, an array of models such as FNN-GAN, LSTM-GAN, BiLSTM-GAN, GRU-GAN with two types of preprocessing techniques Autoencoders and Variational Autoencoders with Random Forest (Adaboost) and Xgboost as classifier algorithms are proposed which balances the data from Twitter Network and further improves the accuracy of malicious social bot detection. The results produced an accuracy of 97.3% for Variational Auto-encoder GRU-GAN (VAE-GGAN) model.

Key words: Malicious social bots, Variational Auto-encoder (VAE), Generative Adversarial Network (GAN), Imbalanced distribution.

LIST OF FIGURES

S.No.	Figure No.	Figure Name	Page No.
1.	Fig. 1.	Existing System	6
2.	Fig. 2.	Proposed System	8
3.	Fig. 3.	Content Diagram	11
4.	Fig. 4.	GAN	12
5.	Fig. 5.	FNN-GAN	13
6.	Fig. 6.	LSTM-GAN	13
7.	Fig. 7.	Bi-LSTM GAN	13
8.	Fig. 8.	GRU-GAN	14
9.	Fig. 9.	AE and VAE	14
10.	Fig. 10.	Flowchart	15
11.	Fig. 11.	Usecase Diagram	18
12.	Fig. 12.	Sequence Diagram	20
13.	Fig. 13.	Class Diagram	21
14.	Fig. 14.	Execution Flow	23

LIST OF TABLES

S.No.	Table No.	Table Name	Page No.
1.	Table 1.	Dataset	7
2.	Table 2.	Hardware Requirements	9
3.	Table 3.	Back-end packages used	10
4.	Table 4.	Front-end packages used	11
5.	Table 5.	GAN Analysis	65
6.	Table 6.	Model Comparison in terms of Accuracy	67
7.	Table 7.	Performance of Xgboost with 5 fold cross validation	68
8.	Table 8.	Test cases and Scenarios	76

LIST OF SCREENS

S.No.	Screen No.	Screen Name	Page No.
1.	Screen 1.	Generator and Discriminator Loss	65
2.	Screen 2.	Discriminator Accuracy for real and fake samples	66
3.	Screen 3.	Noise before AE	66
4.	Screen 4.	Noise after AE	66
5.	Screen 5.	Noise after VAE	67
6.	Screen 6.	Home page	68
7.	Screen 7.	About page	69
8.	Screen 8.	About page (facts)	69
9.	Screen 9.	Details page (introduction)	70
10.	Screen 10.	Details page (proposed system)	70
11.	Screen 11.	Details page (architecture and methodology)	71
12.	Screen 12.	Details page (performance_	71
13.	Screen 13.	Detect page	72
14.	Screen 14.	Detect page (after predict)	72
15.	Screen 15.	Contact us page	73

LIST OF ACRONYMS

OSN	Online Social Network
GAN	Generative Adversarial Network
UML	Unified Modeling Language
FNN	Feed Forward Neural Network
AE	Auto Encoder
VAE	Variational Auto Encoder
LSTM	Long Short-Term Memory
Bi-LSTM	Bi-directional Long Short-Term Memory
GRU	Gated Recurrent Units

1. INTRODUCTION

1.1 MOTIVATION OF THE PROJECT

Online social networks (OSNs) have emerged as a widely used platform where people conveniently share information related to news, links, opinions and promotions. Online social networking sites contain huge amount of data such as data from online reviews, online ratings and discussions forums which are generated by users from various communities. The data can be accessed seamlessly due to proliferation of online social network technologies. This in turn provides an additional space for an attacker to steal user's personal information and to perform malicious activities such as generating fake identities, manipulating online ratings, spreading social spam content and performing phishing attacks in online social networks. In recent years, malicious social bots are the major threats in OSNs. Malicious social botnet is a group of malicious social bots, where each malicious social bot represents a software program which automatically creates multiple fake accounts, frequently interacts with legitimate user accounts. According to a study nearly 25% of Twitter users are bots and 1/3rd of tweets on any topic is generated by social bots. Further, malicious social botnet tries to disseminate social spam content (or fake information) with malicious intention and post malicious (or phishing) links in the tweet. For example, malicious social bots can be created using Twitter API. Moreover, such type of malicious social bots can affect OSN environment with several vulnerabilities.

Several approaches have been proposed to detect malicious social bots in Twitter network. In many OSNs, malicious social bots are much less in number when compared to the legitimate users. In order to detect bots more efficiently a self-adaptive Learning Automata model also has been proposed [1]. But most of the existing studies on social bot detection rely on supervised learning models [1], [3], [5]. However, most of the conventional supervised learning models often suffer from highly imbalanced dataset with many legitimate users belonging to one class and only a few malicious social bots belonging to another class. Therefore, the detection of malicious social bot with better accuracy is one of the challenging tasks. A Conditional Generative Adversarial Network (improved CGAN) has been proposed to overcome imbalances in the data [2].

In order to solve imbalanced distribution of malicious social bot, a Variational Auto-Encoder Generative Adversarial Network (VAE-GAN) has been proposed. GAN's are used to discover the patterns in input data so that they can generate new fake examples of the dataset which cannot be

differentiated from the real ones. Further Variational Auto Encoder is used to improve the quality of the data. The proposed VGAN algorithm helps to detect malicious social bots accurately (in terms of precision, recall, F-measure, and accuracy) in Twitter.

The major contributions are as follows.

1. Balance the minority distribution of data in Twitter Network using Generative Adversarial Network (GAN).
2. Reconstruct the features by removing noise from the data using Variational Auto-Encoders (VAE).
3. Design different Classifiers to analyse the behaviour of users in different environments.
4. Design of a VAE-GAN algorithm by integrating GAN model with Variational AE on balanced data.
5. Evaluate the performance of the proposed VAE-GAN model in terms of precision, recall, F-measure, and accuracy in the Twitter network.

1.2 PROBLEM DEFINITION

To develop a web application which can detect malicious social bots from humans based on the account information.

1.3 OBJECTIVE OF THE PROJECT

The objective of our project is to detect malicious social bots by distinguishing their behavior from legitimate users. The framework provides a helping-hand in recognizing malicious activities performed in twitter network like, spam content, fake URLs, illegitimate news, fake accounts, promotion of fake products etc. Thus, a layman can be protected from getting manipulated.

1.4 LIMITATIONS OF THE PROJECT

The limitation of our project is that the bot prediction is done by using static account information without considering the temporal data.

1.5 ORGANIZATION OF THE PROJECT

The documentation of our project has been divided into the following sections:

Chapter 1: Introduction describes the motivation of this project and objective of the developed project.

Chapter 2: Literature survey describes the primary terms involved in the development of this project and overview of the papers that were referred before starting the project.

Chapter 3: The Analysis chapter deals with detail analysis of the project. Software Requirement Specification further contain user requirements analysis, software requirement analysis and hardware requirement analysis.

Chapter 4: Design includes UML diagrams along with explanation of the system design and the organization.

Chapter 5: Contains step by step implementation of the project and screenshots of the outputs.

Chapter 6: Gives the testing and validation details with design of test cases and scenarios along with the screenshots of the validations.

Chapter 7: In this section we conclude the project by citing all the important aspects of the project along with the different ways in which the project can be developed further are given in the future enhancement section of the conclusion.

Chapter 8: This chapter contains the references to our project.

2. LITERATURE SURVEY

2.1 INTRODUCTION

Rout *et al.* [1] analyzed the social botnet behavior using URL features in Twitter network. The authors have considered Learning Automata (LA) model, a reinforcement learning technique to understand the temporal behavior pattern of social bots at different time slots to detect malicious bots among them. The LA algorithm has been integrated with trust computational model to determine trustworthiness of a user (direct trust) and his/her neighbors (indirect trust).

Wu *et al.* [2] proposed an improved (density peak clustering based with gradient penalty) conditional generative adversarial network (improved CGAN) to overcome imbalances in the data samples from Twitter and to avoid the generation of data-augmentation noise before training the classifier to improve the detection accuracy of social bots. They used 1971 normal human accounts and 462 social bot accounts as original samples where all of the data that are original tweet content crawled from Twitter are converted into a raw data set that can be used directly through feature extraction. The model achieved an F1 score of 97.56% but the oversampling is obvious only when the humans and bots differ greatly in the dataset.

Wang *et al.* [3] proposed a bot detection framework based on a combination of a Variational Auto Encoder and an anomaly detection algorithm. Auto Encoders are used to automatically encode and decode sample features. Anomaly detection method based on k-nearest neighbors is used to reduce the number of abnormal samples participating in model training. They selected the author profiling task 2019 (CLEF2019) dataset consists of two groups of Twitter accounts, one in English and the other in Spanish and achieved 98.3% accuracy with VAE-KNN-3 model. However, the imperfections to this method are limited feature extraction of the data and lack of adaption to different language environments.

Zhang *et al.* [4] demonstrated the effectiveness and advantages of exploiting a social botnet for spam distribution and digital-influence manipulation on Twitter. They also proposed two countermeasures to defend against the two reported attacks considering the approach that a bot only retweets the spam tweets which are posted by botmaster and manipulates the influence value of each user. Evaluation of the performance of the digital-influence

measurement scheme has been carried out by using the Twitter geo-search API to collect the users in a specific metropolitan area and then crawling the latest 600 tweets of each user to extract the interactions such as retweets, replies, and mentions and achieved an accuracy of more than 70%. The limitations in this approach are network features are not taken into consideration.

Kudugunta *et al.* [5] proposed a deep neural network based on contextual long short-term memory (LSTM) architecture that exploits both content and metadata to detect bots at the tweet level: contextual features are extracted from user metadata and fed as auxiliary input to LSTM deep nets processing the tweet text. They used the dataset which was presented by Cresci and collaborators containing an entirely new breed of social bots and performed account-level and tweet-level bot detection. The former approach achieved 99.8% accuracy and the latter has achieved 96.33% accuracy with glove embedding. The drawback in this method is it is unable to capture dynamic behaviors of bots.

2.2 EXISTING SYSTEM

Malicious social bots post shortened malicious URLs in the tweet in order to redirect the requests of online social networking participants to some malicious servers. Hence, distinguishing malicious social bots from legitimate users is one of the most important tasks in the Twitter network. To detect malicious social bots, extracting URL-based features (such as URL redirection, frequency of shared URLs, and spam content in URL) consumes less amount of time in comparison with social graph- based features (which rely on the social interactions of users). In the existing system, a learning automata-based malicious social bot detection (LA-MSBD) algorithm is proposed by integrating a trust computation model with URL-based features for identifying trustworthy participants in the Twitter network. The proposed trust computation model computes direct trust which is derived from Bayes' theorem, and the indirect trust which is derived from the Dempster-Shafer theory (DST). The framework of existing system is depicted in [Fig. 1](#).

2.3 DISADVANTAGES OF EXISTING SYSTEM

Existing system is evaluated by considering two Twitter data sets, namely, Social HoneyPot1 data set and The Fake Project2 data set and has achieved precisions of 91.77% and 95.37%

The precision value obtained for The Fake Project data set is better than the Social HoneyPot data set because the Social HoneyPot data set contains noisy and untrustworthy information in its user content features than The Fake Project data set. So, the drawback of this model is that there is no noise removal technique designed in order to overcome the problem.

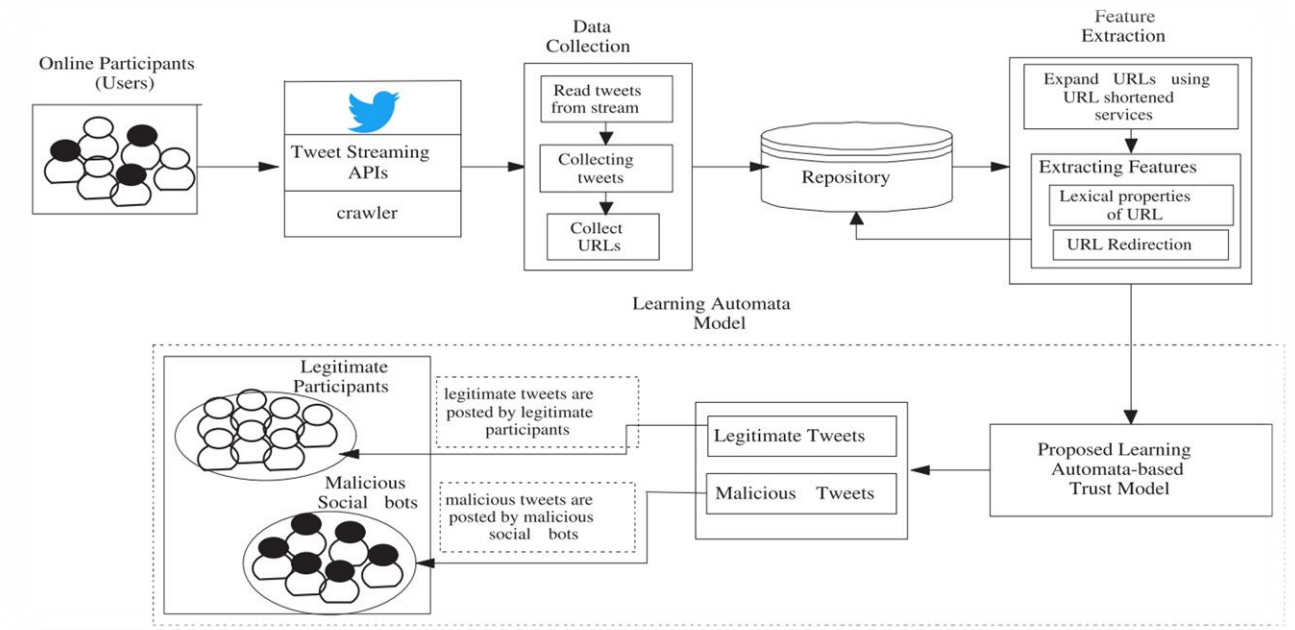


Fig. 1. Existing System

2.4 PROPOSED SYSTEM

The proposed Variational Auto Encoder Generative Adversarial Network (VAE-GAN) model as shown in [Fig. 2.](#), for malicious social bot detection consists of four steps: Data Collection, Data Balancing (GAN), Data Pre-processing (VAE), Classification.

Step 1: Data Collection

The dataset considered for our model is presented by Cresci and collaborators [\[6\]](#), which contains an entirely new breed of social bots (social spambots #1, social spambots #2 and social spambots #3, traditional spambots #1, traditional spambots #2, traditional spambots #3, traditional spambots #4) and human users (genuine accounts). All these subsets of data combined together account for 7543 records of bots and 3474 records of humans with 43 features for each. A group-wise partition is shown in [Table 1](#).

Although many established techniques use a large number of features, recent research [7,8] shows that similar high performance can be obtained by using a minimal number of features [5]. The bot detection has been done based on the account information with the following features:

- Statuses Count
- Followers Count
- Friends Count
- Favourites Count
- Listed Count

Table 1. Dataset

Dataset	No. of records
genuine accounts	3474
social spambots #1	991
social spambots #2	3457
Social spambots #3	464
traditional spambots #1	1000
traditional spambots #2	100
traditional spambots #3	403
Traditional spambots #4	1128

Step 2: Data Balancing

The minority class data is balanced to match with majority class data with the help of Generative Adversarial Networks (GAN). Different GAN models implemented are –

- FNN based GAN
- LSTM based GAN
- Bi-LSTM based GAN
- GRU based GAN

Step 3: Data Preprocessing

The features of dataset are processed to remove noise by different techniques like-

- Autoencoders
- Variational Autoencoders

Step 4: Model Building

Classification model is trained to distinguish between bots and legitimate users. Different classifiers that are implemented are –

- Random Forest with AdaBoost
- Xgboost classifier

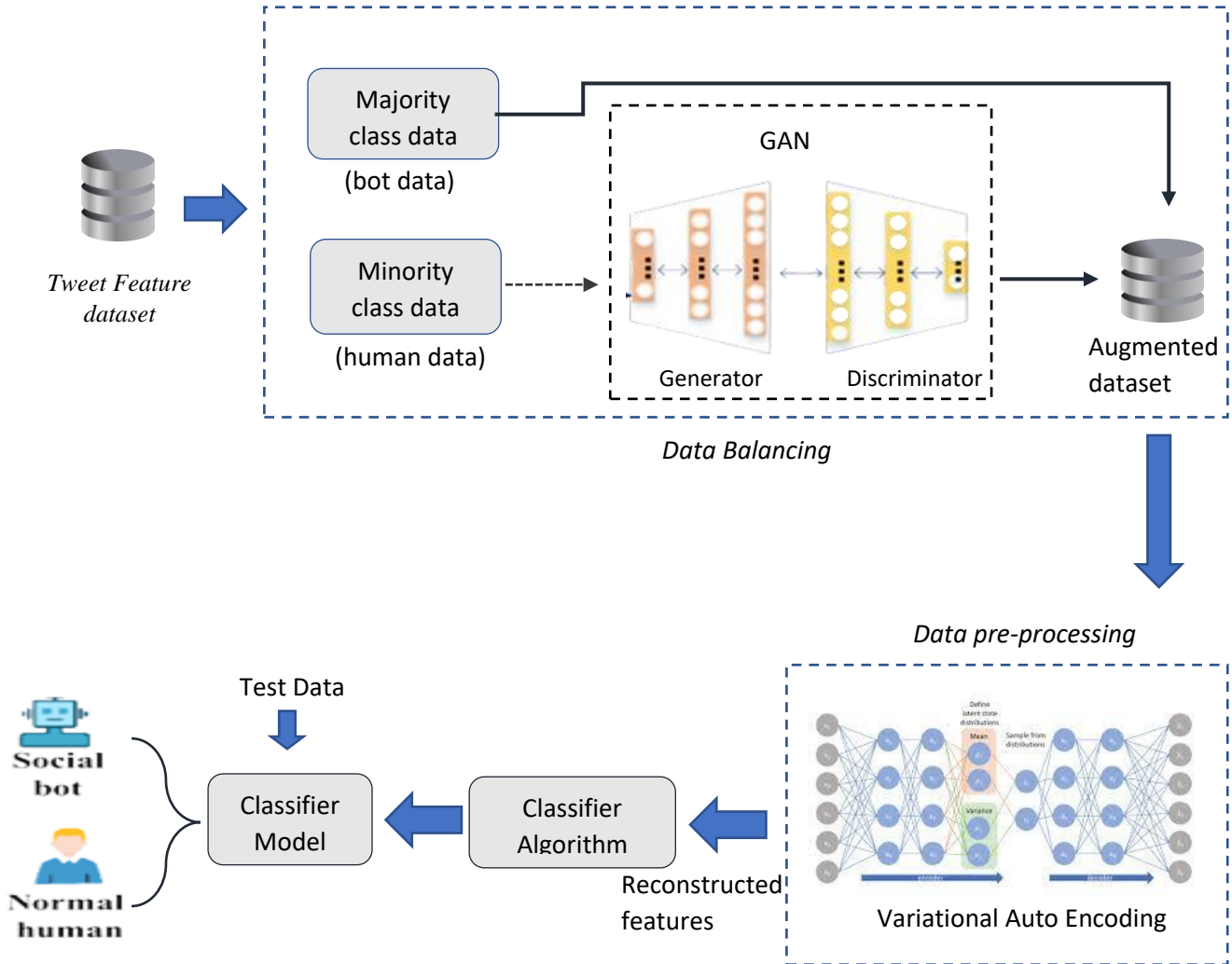


Fig. 2. Proposed System

2.5 CONCLUSION

By the survey on different existing works, it is observed that most of the conventional algorithms suffer from imbalanced dataset. Hence the proposed system consists of GAN, a data augmentation technique. Autoencoders are used as better noise removal technique.

3. ANALYSIS

3.1 INTRODUCTION

In the analysis phase, end user's requirements are analyzed and project goals get converted into the defined system functions that the organization intends to develop. The three primary activities involved in the analysis phase are as follows:

- Software Requirement Specification
- Creating content diagram
- Performing a detailed analysis by drawing a flowchart

Requirement Analysis will identify and consider the risks related to how the technology will be integrated into the standard operating procedures. Requirements Analysis will also collect the functional and system requirements of the business process, the user requirements and the operational requirements. Hence this section deals with the software and hardware requirements with respect to our project.

3.2 REQUIREMENTS SPECIFICATION

3.2.1 Hardware Requirements

Table 2. Hardware Requirements

Requirements Specification	Minimum Requirement	Host System Configuration
Processor	Intel® Core™-i3-8 th Gen	Intel® Core™-i5-10 th Gen
System Type	32-bit Operating System	64-bit Operating System
HDD	40 GB	1 TB
RAM	8 GB	8 GB

3.2.2 Software Requirements

The software requirements include the languages, libraries, packages and operating system, and different tools for developing the project.

Languages used : Python3, HTML, CSS

Software used : Jupyter Notebook, Vscode

Operating System : Windows 10

Back-end Packages used :

Table 3. Back-end packages used

S. No.	Package	Description
1.	numpy	numpy package is fast and accurate, used for scientific computing in python.
2.	pandas	Pandas is fast, scalable package used for data science operations and expensive data structures operations.
3.	matplotlib	It is a cross platform, data visualization and graphical plotting library for python and its numerical extension.
4.	seaborn	Seaborn is a matplotlib-based Python data visualization package. It has a high-level interface for creating visually appealing and instructive statistics visuals.
5.	tensorflow	It is a python-based package which provides implementing different machine learning and AI tools such as neural layers.
6.	keras	Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It consists of all the neural network layers and other packages used for deep learning computations.
7.	sklearn	Scikit-learn is an open source learn is an open-source Python package functionality supporting supervised and unsupervised learning.
8.	pickle	pickle module is used for serializing and de-serializing python object structures.
9.	statistics	It provides functions for calculating mathematical statistics.

Table 4. Front-end packages used

S. No.	Libraries used	Description
1.	django	Django is a high-level Python web framework that promotes rapid development and simple, practical design.
2.	virtualenv	virtualenv is an utility for establishing isolated Python environments, each with its own copy of Python, pip, and a location to store libraries downloaded from PyPI.
3.	jinja	Jinja is a templating engine that is quick, expressive, and extendable. The template has special placeholders that allow you to write code that looks like Python syntax.
4.	os	In Python, the OS module has functions for dealing with the operating system.

3.3 CONTENT DIAGRAM OF THE PROJECT

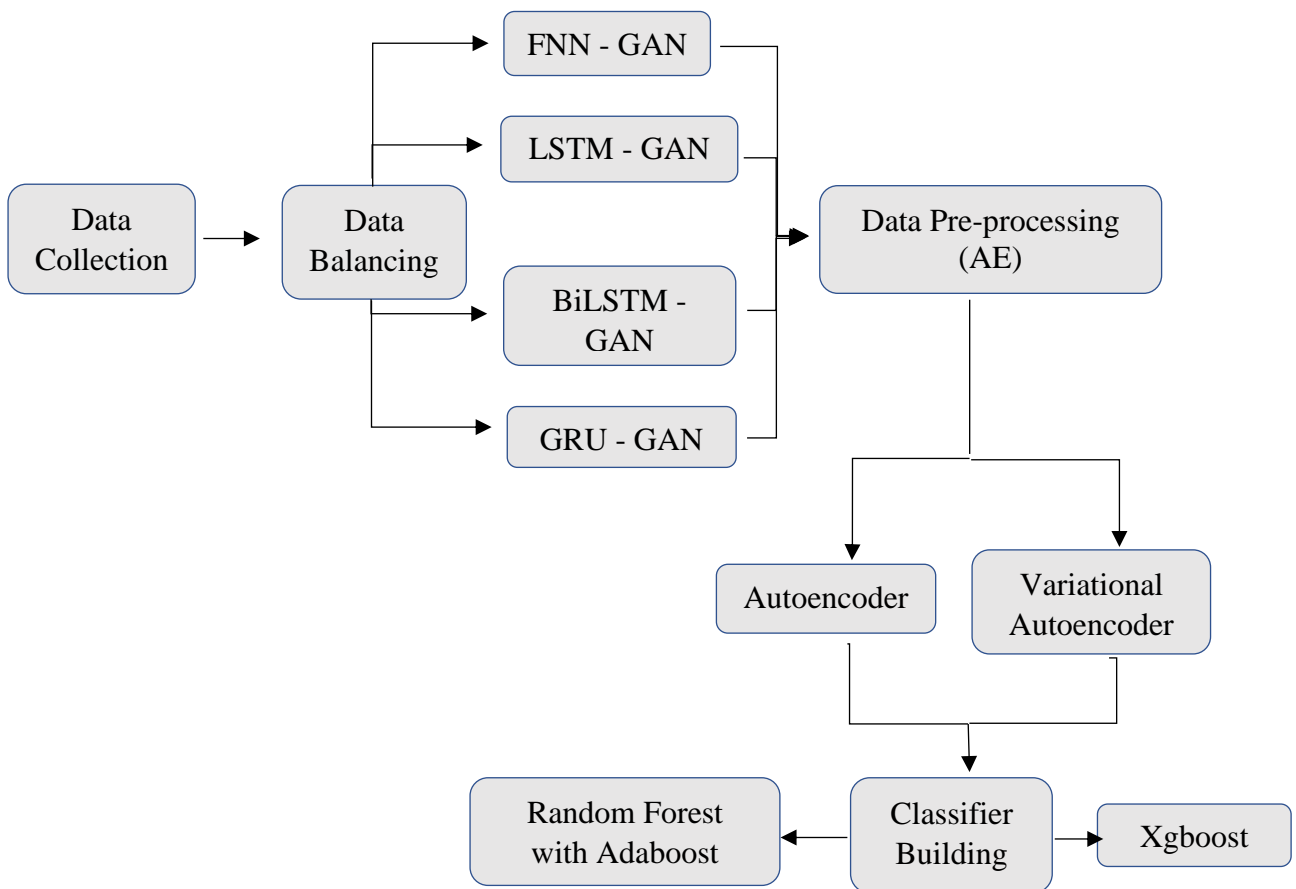


Fig. 3. Content Diagram

3.4 ALGORITHMS AND FLOWCHARTS

3.4.1 Algorithms

Generative Adversarial Networks (GAN)

In the real world the number of social bots is far less than the number of normal humans, leading to a serious sample imbalance problem in social network bot detection methods [2]. The data can be balanced either by undersampling or oversampling techniques. Undersampling is not suitable because of the loss of information in a social-bot-detection scenario which effects the model performance. Data augmentation is a technique to generate new fake samples of data from the existing data which is called oversampling and it can be achieved through GAN.

GANs discover the regularities in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset. They use two sub-models: the generator model that is trained to generate new examples, and the discriminator model that tries to classify examples as either real or fake. The generator function generates a batch of fake samples, and these are passed to the discriminator along with the real samples for classification as real or fake. The GAN model is depicted in [Fig.4](#).

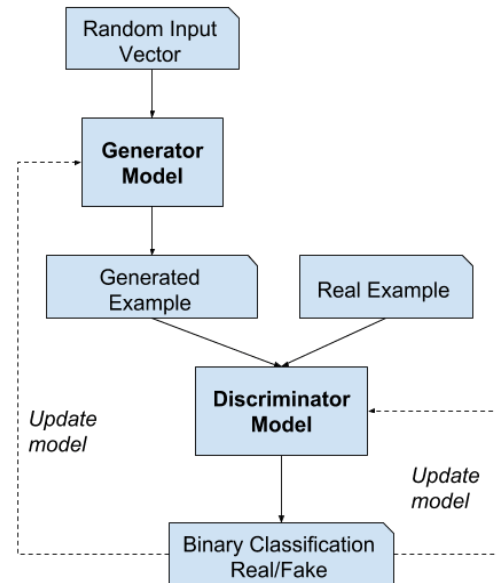


Fig. 4. GAN

The generator and discriminator functions are built with Feed forward neural network (FNN) layers. Besides neural networks, other structures can be used as discriminative models such as Long Short-Term Memory (LSTM), Bi-directional Long Short-Term Memory (Bi-LSTM) and Gated Recurrent Units (GRU).

FNN based GAN

Feed forward Neural Networks are the simple neural networks where each unit in a layer is connected to all units in previous layers (Dense Layer) as shown in [Fig. 5](#). . They do not have

back propagation. The limitation of FNN is that the learning is slow and difficult as they consider all weights in the network which may also lead to overfitting.

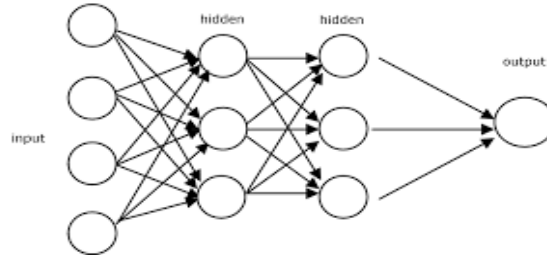


Fig. 5. FNN-GAN

LSTM based GAN

Long Short-Term Memory is a special kind of RNN, capable of learning long-term dependencies, remembering information for long periods of time. As they are recurrent networks, they backpropagate the information, overcoming the limitation in Feed Forward Networks. The architecture of the same is shown in [Fig. 6](#).

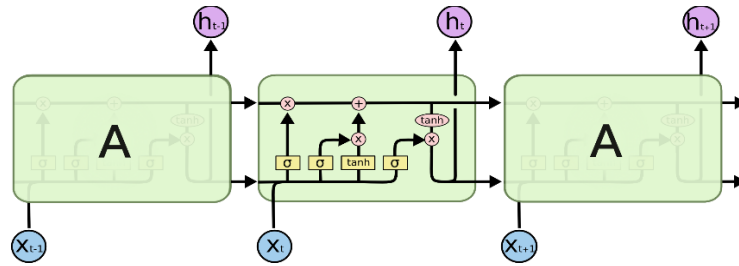


Fig. 6. LSTM-GAN

Bi-LSTM based GAN

Bi-directional Long Short Term Memory's is an extension of LSTM that trains itself both with the backward and forward information in a sequence as shown in [Fig. 7](#).

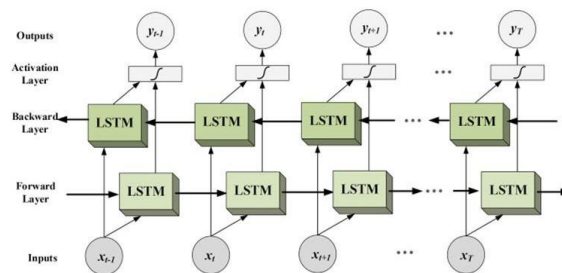


Fig. 7. Bi-LSTM-GAN

GRU based GAN

Gated Recurrent Units, shown in [Fig. 8](#), are improved version of standard recurrent neural network which are simpler, trains faster than LSTM and have lesser parameter complexity.

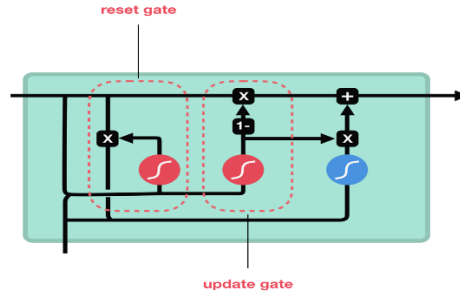


Fig. 8. GRU-GAN

Variational Autoencoders (VAE)

Variational Autoencoders are type of Autoencoders which are used for dimensionality reduction. Autoencoders compress the input into a lower-dimensional code using encoding unit and then reconstruct the output from latent space representation using decoding unit but the regularity of the latent space is not guaranteed. High degree of freedom of autoencoders with no information loss leads to severe overfitting. So Variational Autoencoders are proposed with explicit regularization to avoid overfitting. Instead of encoding an input as a single point, it is done as a distribution over the latent space. The comparison between AE and VAE is depicted in [Fig. 9](#).

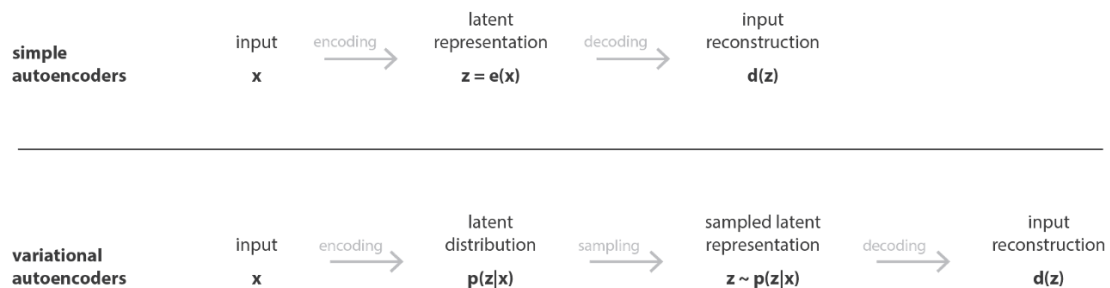


Fig. 9. AE and VAE

Random Forest with Adaboost

Random Forests are the group of decision trees built on different samples and they take majority class from all the trees as output. Boosting is a general ensemble method that creates a strong classifier from a number of weak classifiers. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added. AdaBoost is an ensemble learning model which increases the efficiency of binary classifiers by assigning more weights to wrong predictions in order to correct them in next iteration. Rather than being a model in itself, AdaBoost can be applied on top of any classifier to learn from its shortcomings and propose a more accurate model.

Xgboost

Xgboost stands for Extreme Gradient Boosting which is an improved version of Gradient Boosting. It involves computing second order gradients (parameters) to minimize loss function unlike Gradient Boosting which uses first order gradients.

3.4.2 Flowchart

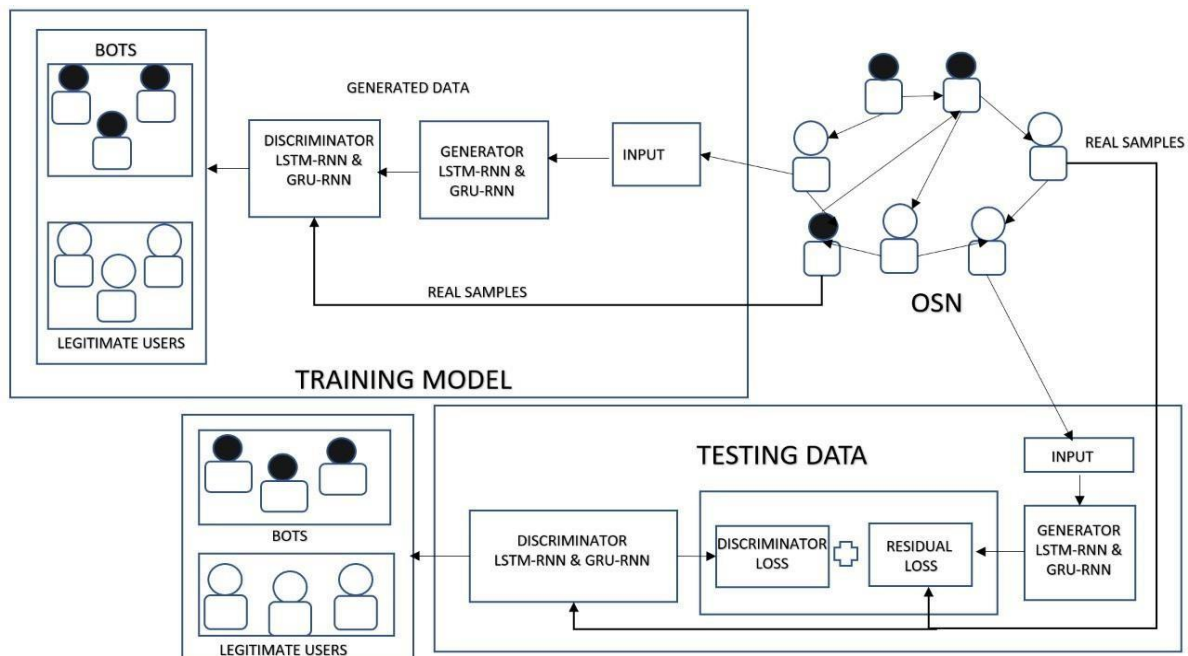


Fig. 10. Flowchart

3.5 CONCLUSION

This chapter introduces all the software requirements and hardware requirements to implement and run the system. The analysis is made on various algorithms used to implement the project and the flow of the model is described.

4. DESIGN

4.1 INTRODUCTION

It is a process of planning the new or modified system. Design is essentially a bridge between requirement specification and the final solution satisfying the requirements, Design of a system is essentially a blueprint or a solution for the system. UML diagrams are used to model the system. UML stands for Unified Modeling Language. UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

4.2 UML Diagrams

The UML consists of a number of graphical elements that combine to form diagrams. Because it's a language, the UML has rules for combining these elements. The purpose of the diagrams to present multiple views of the system, and this set of multiple views is called a Model. A UML Model of a system is something like a scale model of a building. UML model describes what a system is supposed to do. It doesn't tell how to implement the system.

- Class Diagrams
- Use Case Diagrams
- Sequence Diagrams
- Activity Diagrams

4.2.1 Use case diagram

The use case diagram summarizes the interactions among users (both internal and external) and the system. All the users are depicted as actors and the system functionalities are shown as use cases in [Fig. 11](#).

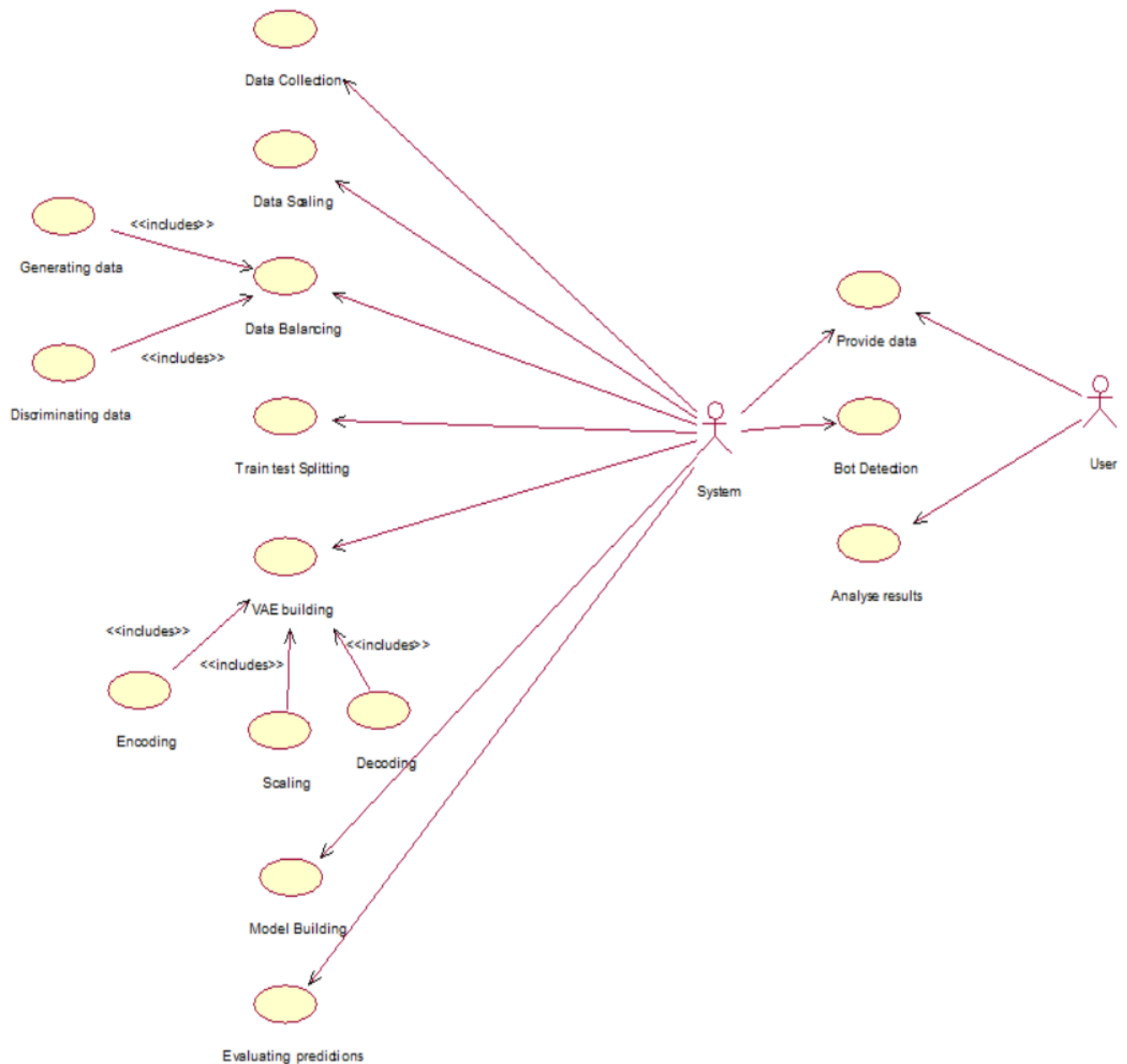


Fig. 11. Use case diagram

From the diagram, the identified actors are system and user. There are multiple actions performed by both the system and the user which are designed as use cases.

List of use cases performed by the system (internally) are –

1. Data Collection
2. Data Scaling
3. Data Balancing (includes generator and discriminator building)
4. Train Test Splitting

5. VAE building (includes encoding, scaling and decoding)
6. Model building
7. Evaluating predictions

List of use cases performed by external user (externally) are –

1. Provide new data for detection
2. Analyze the results

List of use cases performed by system (externally) are –

1. Bot Detection

4.2.2 Sequence diagram

Sequence diagrams shows the interactions among objects in a sequentially order with respect to time i.e., the order in which they interact. Sequence diagram contains actors (external objects) and elements called lifelines (internal objects). The communications among them are depicted through messages. The Sequence diagram is depicted in [Fig. 12](#).

The diagram sequentially depicts the actions in the system.

1. System collects data from the datastore.
2. Data is scaled for better results.
3. GAN model is built and data is balanced.
4. Now the data is split into train and test data.
5. Noise from the data is removed using Variational Auto Encoder (VAE).
6. Classifier model is fit and is tested on the data.
7. The model is saved for further use.
8. User provides new data through an interface.
9. System predicts the behaviour and produce results.
10. User analyzes the results.

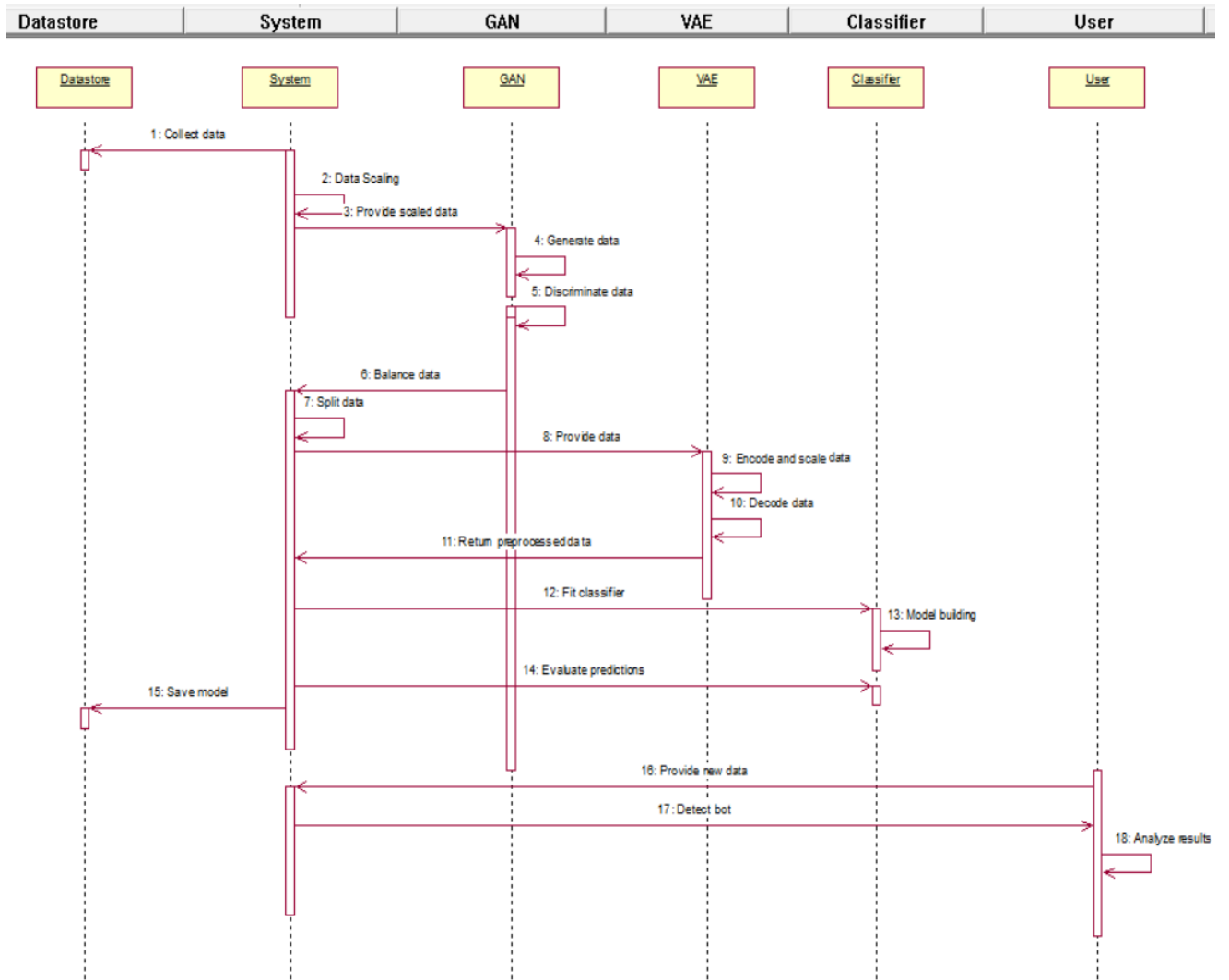


Fig. 12. Sequence diagram

4.2.3 Class diagram

Class diagram in [Fig. 13](#) describes the static view of a system by defining the attributes (properties) and operations (responsibilities) of classes and interfaces in the system. The classes identified are system, user, datastore and VAE-GAN model. The system and user are connected with 0 or more multiplicity i.e., a system can have no or any users. GAN, VAE, Classifier classes are connected to VAE-GAN model class with aggregation (whole part) relationship. The model can have one or more classifiers.

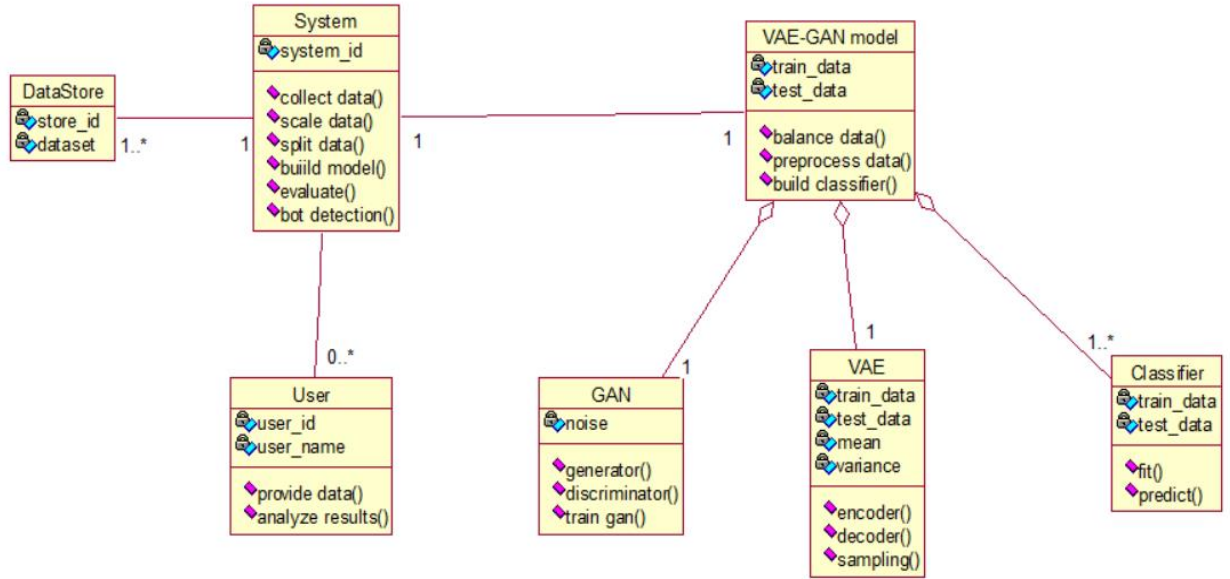


Fig. 13. Class diagram

4.3 CONCLUSION

In this section, the functionalities, classes, objects involved in the proposed VAE-GAN model are described with the help of use case diagram, sequence diagram and class diagram. The communication among various objects is described using the appropriate relationships between them.

5. IMPLEMENTATION AND RESULTS

5.1 INTRODUCTION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

5.2 EXPLANATION OF KEY FUNCTIONS

define_generator() : function defining the generator model to generate fake samples of data.

define_discriminator() : function defining the discriminator model that differentiates between fake and real samples.

define_complete_gan() : function combining both generator and discriminator model.

gan_train() : function used to train GAN model for generating new samples.

Sampling() : class that generates mean and variance used to reconstruct data in Variational Autoencoder

VAE() : class that builds Variational Autoencoder model by combining encoder and decoder.

5.3 IMPLEMENTATION

The execution flow of both the frontend and backend code of the project is depicted in the [Fig. 14.](#) The four .ipynb files depicting four different discriminator models can be executed in any order as they are independent. The connection between the frontend and backend code is established by saving the classification model using pickle.

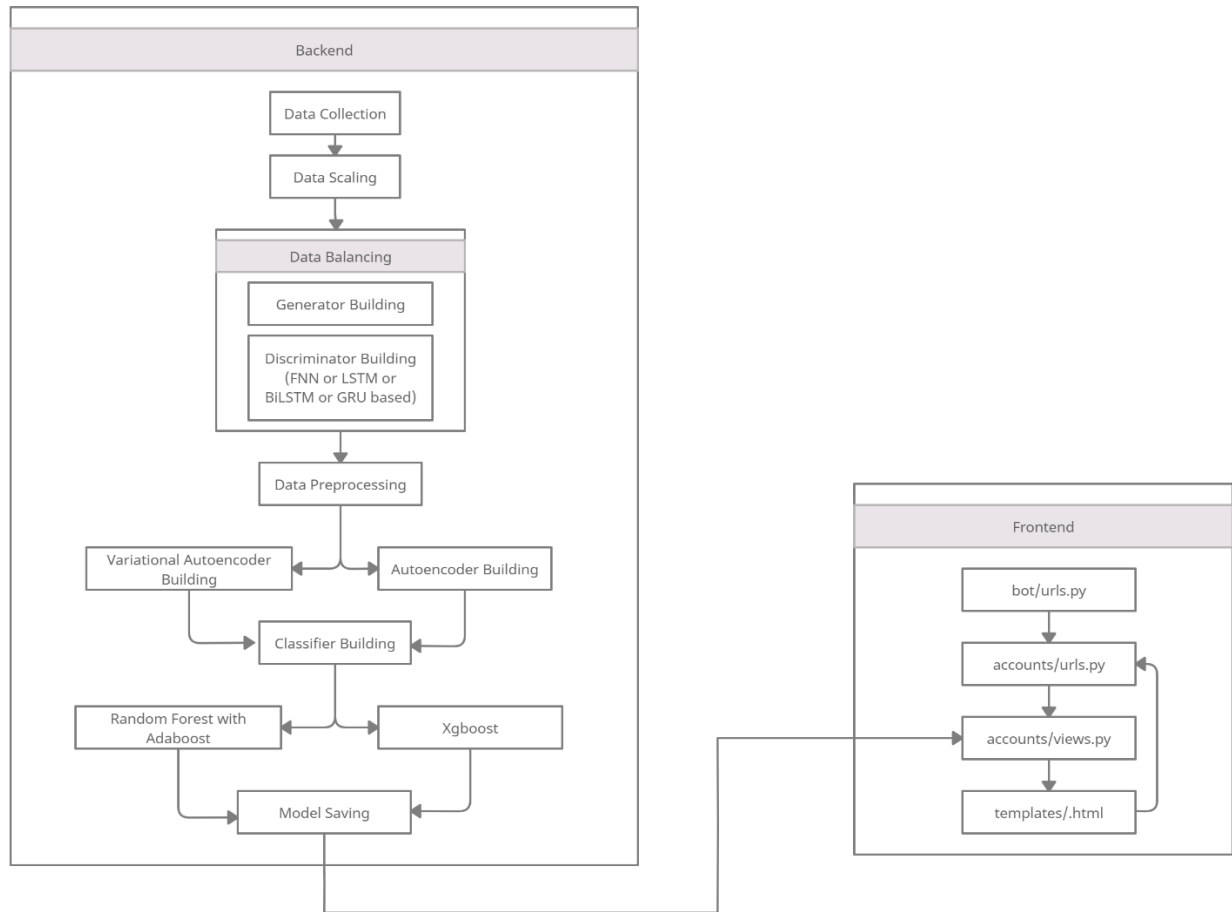


Fig. 14. Execution Flow

5.3.1 BACK-END CODE

```
data=data[['statuses_count','followers_count','friends_count','favourites_count','listed_count',
Label']]
data
```

Generator Building

```
def define_generator(numerical):
    #Inputting noise from latent space
    noise = Input(shape = (70,))
    hidden_1 = Dense(8)(noise)
    hidden_1 = LeakyReLU(0.2)(hidden_1)
    hidden_1 = BatchNormalization(momentum = 0.8)(hidden_1)
    hidden_2 = Dense(16)(hidden_1)
    hidden_2 = LeakyReLU(0.2)(hidden_2)
    hidden_2 = BatchNormalization(momentum = 0.8)(hidden_2)

    branch_12 = Dense(64)(hidden_2)
    branch_12 = LeakyReLU(0.2)(branch_12)
```

```

branch_12 = BatchNormalization(momentum = 0.8)(branch_12)
branch_12 = Dense(128)(branch_12)
branch_12 = LeakyReLU(0.2)(branch_12)
branch_12 = BatchNormalization(momentum = 0.8)(branch_12)

branch_12_output = Dense(numerical, activation = "sigmoid")(branch_12)

combined_output = concatenate([branch_12_output])
return Model(inputs = noise, outputs = combined_output)

generator = define_generator(numerical_df_rescaled.shape[1])
generator.summary()

```

Discriminator Building

CNN based

```

def define_discriminator(inputs_n):
    #Input from generator
    d_input = Input(shape = (inputs_n,))
    d = Dense(128)(d_input)
    d = LeakyReLU(0.2)(d)
    d = Dense(64)(d)
    d = LeakyReLU(0.2)(d)
    d = Dense(32)(d)
    d = LeakyReLU(0.2)(d)
    d = Dense(16)(d)
    d = LeakyReLU(0.2)(d)
    d = Dense(8)(d)
    d = LeakyReLU(0.2)(d)
    d_output = Dense(1, activation = "sigmoid")(d)

    model = Model(inputs = d_input, outputs = d_output)
    model.compile(loss = "binary_crossentropy", optimizer = Adam(lr=0.0002, beta_1=0.5),
metrics = ["accuracy"])
    return model

inputs_n = numerical_df_rescaled.shape[1]
discriminator = define_discriminator(inputs_n)
discriminator.summary()

```

LSTM based

```

def define_discriminator(inputs_n):
    spec_start = Input(shape=(1,inputs_n))
    spec_x = spec_start
    for _r in [32,32]:
        spec_x = LSTM(_r, activation='tanh', dropout=0.5, recurrent_dropout=0.5,
return_sequences=True)(spec_x)
    for _f in [32]:
        spec_x = TimeDistributed(Dense(_f))(spec_x)

```

```

spec_x = Dropout(0.5)(spec_x)
spec_x = TimeDistributed(Dense(10))(spec_x)
out = Activation('sigmoid', name='strong_out')(spec_x)
_model = Model(inputs=spec_start, outputs=out)
_model.compile(optimizer='Adam', loss='binary_crossentropy', metrics = ['accuracy'])
return _model

```

```

inputs_n = numerical_df_rescaled.shape[1]
discriminator = define_discriminator(inputs_n)
discriminator.summary()

```

Bi-LSTM based

```

def define_discriminator(inputs_n):
    spec_start = Input(shape=(1,inputs_n))
    spec_x = spec_start
    for _r in [32,32]:
        spec_x = Bidirectional(LSTM(_r, activation='tanh', dropout=0.5, recurrent_dropout=0.5,
return_sequences=True))(spec_x)
    for _f in [32]:
        spec_x = TimeDistributed(Dense(_f))(spec_x)
        spec_x = Dropout(0.5)(spec_x)
        spec_x = TimeDistributed(Dense(10))(spec_x)
        out = Activation('sigmoid', name='strong_out')(spec_x)
    _model = Model(inputs=spec_start, outputs=out)
    _model.compile(optimizer='Adam', loss='binary_crossentropy', metrics = ['accuracy'])
    return _model

```

```

inputs_n = numerical_df_rescaled.shape[1]
discriminator = define_discriminator(inputs_n)
discriminator.summary()

```

GRU-based

```

def define_discriminator(inputs_n):
    spec_start = Input((1,inputs_n))
    spec_x = spec_start
    for _r in [32,32]:
        spec_x = GRU(_r, activation='tanh', dropout=0.5, recurrent_dropout=0.5,
return_sequences=True)(spec_x)
    for _f in [32]:
        spec_x = TimeDistributed(Dense(_f))(spec_x)
        spec_x = Dropout(0.5)(spec_x)
        spec_x = TimeDistributed(Dense(10))(spec_x)
        out = Activation('sigmoid', name='strong_out')(spec_x)
    _model = Model(inputs=spec_start, outputs=out)
    _model.compile(optimizer='Adam', loss='binary_crossentropy', metrics = ['accuracy'])
    return _model

```

```

inputs_n = numerical_df_rescaled.shape[1]

```

```
discriminator = define_discriminator(inputs_n)
discriminator.summary()
```

Complete GAN

```
def define_complete_gan(generator, discriminator):
    discriminator.trainable = False
    gan_output = discriminator(generator.output)

    #Initialize gan
    model = Model(inputs = generator.input, outputs = gan_output)

    #Model Compilation
    model.compile(loss = "binary_crossentropy", optimizer = Adam(lr=0.0002, beta_1=0.5))
    return model
```

```
compleategan = define_complete_gan(generator, discriminator)
```

```
def gan_train(gan, generator, discriminator, numerical, latent_dim, n_epochs, n_batch,
n_eval):
```

```
    #Update Discriminator with half batch size
```

```
    half_batch = int(n_batch / 2)
```

```
    discriminator_loss = []
```

```
    discriminator_acc_real = []
```

```
    discriminator_acc_fake = []
```

```
    generator_loss = []
```

```
    #generate class labels for fake and real
```

```
    valid = np.ones((half_batch, 1))
```

```
    y_gan = np.ones((n_batch, 1))
```

```
    fake = np.zeros((half_batch, 1))
```

```
    #training
```

```
    for i in range(n_epochs):
```

```
        #select random batch from real categorical and numerical data
```

```
        idx = np.random.randint(0, numerical_df_rescaled.shape[0], half_batch)
```

```
        numerical_real = numerical_df_rescaled[idx]
```

```
        #concatenate categorical and numerical data for the discriminator
```

```
        real_data = np.concatenate([numerical_real], axis = 1)
```

```
        #generate fake samples from the noise
```

```
        noise = np.random.normal(0, 1, (half_batch, latent_dim))
```

```
        fake_data = generator.predict(noise)
```

```
        #train the discriminator and return losses and acc
```

```
        d_loss_real, da_real = discriminator.train_on_batch(real_data, valid)
```

```
        d_loss_fake, da_fake = discriminator.train_on_batch(fake_data, fake)
```

```
        d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
```

```
        discriminator_loss.append(d_loss)
```

```
        discriminator_acc_real.append(da_real)
```

```

discriminator_acc_fake.append(da_fake)

#generate noise for generator input and train the generator (to have the discriminator
label samples as valid)
noise = np.random.normal(0, 1, (n_batch, latent_dim))
g_loss = gan.train_on_batch(noise, y_gan)
generator_loss.append(g_loss)

#evaluate progress
if (i+1) % n_eval == 0:
    print ("Epoch: %d [Discriminator loss: %f] [Generator loss: %f] [Discriminator
Accuracy (real): %f] [Discriminator Accuracy (fake): %f]" % (i + 1, d_loss, g_loss, da_real,
da_fake))

plt.figure(figsize = (20, 10))
plt.plot(generator_loss, label = "Generator loss")
plt.plot(discriminator_loss, label = "Discriminator loss")
plt.title("Stats from training GAN",fontsize=20)
plt.xlabel('No. of epochs',fontsize=15)
plt.ylabel('Generator and Discriminator Loss',fontsize=15)
plt.grid()
plt.legend(prop={'size': 20})

plt.figure(figsize = (20, 10))
plt.plot(discriminator_acc_real, label='acc-real')
plt.plot(discriminator_acc_fake, label='acc-fake')
plt.title("Stats from training GAN",fontsize=20)
plt.xlabel('No. of epochs',fontsize=15)
plt.ylabel('Discriminator Accuracy on real and fake samples',fontsize=15)
plt.grid()
plt.legend(prop={'size': 20})

latent_dim = 70
gan_train(compleategan, generator, discriminator,numerical_df_rescaled, latent_dim,
n_epochs = 5000, n_batch = 63, n_eval = 200)

noise = np.random.normal(0, 1, (4500, 70))
generated_mixed_data = generator.predict(noise)
columns=list(numerical_df.columns)
mixed_gen_df = pd.DataFrame(data = generated_mixed_data, columns = columns)

df_generated_data = pd.DataFrame()
df_generated_data = mixed_gen_df.copy()
df_generated_data['Label']=0

```

Variational Autoencoder Building

```
class Sampling(Layer):
```

```

def call(self, inputs):
    z_mean, z_log_var = inputs
    batch = tf.shape(z_mean)[0]
    dim = tf.shape(z_mean)[1]
    epsilon = tf.keras.backend.random_normal(shape=(batch, dim))
    return z_mean + tf.exp(0.5 * z_log_var) * epsilon

latent_dim = 5
encoder_inputs = Input(shape=(5))
x=Dense(32, activation="relu")(encoder_inputs)
x=Dense(16, activation="relu")(x)
x=Dense(7, activation="relu")(x)
z_mean = Dense(latent_dim, name="z_mean")(x)
z_log_var = Dense(latent_dim, name="z_log_var")(x)
z = Sampling()([z_mean, z_log_var])
encoder = Model(encoder_inputs, [z_mean, z_log_var, z], name="encoder")
encoder.summary()

latent_inputs = keras.Input(shape=(latent_dim, ))
x=Dense(7, activation="relu")(latent_inputs)
x=Dense(16, activation="relu")(x)
x=Dense(32, activation="relu")(x)
decoder_outputs=Dense(len(x_train_scaled.columns), activation="sigmoid")(x)
decoder = Model(latent_inputs, decoder_outputs, name="decoder")
decoder.summary()

class VAE(keras.Model):
    def __init__(self, encoder, decoder, **kwargs):
        super(VAE, self).__init__(**kwargs)
        self.encoder = encoder
        self.decoder = decoder

    def train_step(self, data):
        if isinstance(data, tuple):
            data = data[0]
        with tf.GradientTape() as tape:
            z_mean, z_log_var, z = encoder(data)
            reconstruction = decoder(z)
            reconstruction_loss = tf.reduce_mean(
                keras.losses.binary_crossentropy(data, reconstruction)
            )
            reconstruction_loss *= 28 * 28
            kl_loss = 1 + z_log_var - tf.square(z_mean) - tf.exp(z_log_var)
            kl_loss = tf.reduce_mean(kl_loss)
            kl_loss *= -0.5
            total_loss = reconstruction_loss + kl_loss
        grads = tape.gradient(total_loss, self.trainable_weights)
        self.optimizer.apply_gradients(zip(grads, self.trainable_weights))
        return {

```

```

        "loss": total_loss,
        "reconstruction_loss": reconstruction_loss,
        "kl_loss": kl_loss,
    }

```

```

train_test_data = np.concatenate([x_train_scaled, x_test_scaled], axis = 0)
vae = VAE(encoder, decoder)
vae.compile(loss='mae',metrics=['mae'],optimizer='adam')
vae.fit(train_test_data, epochs = 10, batch_size = 32)

```

```

x_train = pd.DataFrame(decoder.predict(x_train_scaled))
x_test = pd.DataFrame(decoder.predict(x_test_scaled))
x_train = x_train.add_prefix('feature_')
x_test = x_test.add_prefix('feature_')

```

Random Forest with Adaboost

```

rf = RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
criterion='gini', max_depth=2, max_features='auto', max_leaf_nodes=None,
max_samples=None, min_impurity_decrease=0.0, min_samples_leaf=2,
min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=5, n_jobs=None,
oob_score=False, random_state=None, verbose=0, warm_start=False)
classifier = AdaBoostClassifier(rf)

```

```

classifier.fit(x_train,y_train)
pred = np.array(classifier.predict(x_test))

recall = rs(y_test,pred)
precision = ps(y_test,pred)
r1 = fs(y_test,pred)
ma = classifier.score(x_test,y_test)
print('*** Evaluation metrics for test dataset ***\n')
print('Recall Score: ',recall)
print('Precision Score: ',precision)
print('F1 Score: ',r1)
print('Accuracy: ',accuracy_score(y_test,pred))

```

```

classifier.fit(data.iloc[:, :-1], data['Label'])
for i in range(1):
    for j in range(1):
        fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=800)
        tree.plot_tree(classifier.estimators_[i][j],
            feature_names = X.columns,class_names = ['0','1'], filled = True)

```

Xgboost Classifier

```

xgb_cl = xgb.XGBClassifier()
xgb_cl.fit(x_train, y_train)
pred = xgb_cl.predict(x_test)
recall = rs(y_test,pred)

```

```

precision = ps(y_test,pred)
r1 = fs(y_test,pred)
ma = xgb_cl.score(x_test,y_test)
print('*** Evaluation metrics for test dataset ***\n')
print('Recall Score: ',recall)
print('Precision Score: ',precision)
print('F1 Score: ',r1)
print('Accuracy: ',ma)

false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, pred)
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.plot(false_positive_rate, true_positive_rate)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
print('roc_auc_score: ', roc_auc_score(y_test, pred))

scoring = {'accuracy': make_scorer(accuracy_score),
           'precision': make_scorer(precision_score),
           'recall': make_scorer(recall_score),
           'f1_score': make_scorer(f1_score)}

kf=KFold(n_splits=5,shuffle=True)
score=cross_validate(xgb_cl,X,Y,scoring=scoring,cv=kf)
pred=cross_val_predict(xgb_cl,X,Y,cv=kf)
print('Recall Score: ',score['test_recall'], " Average",mean(score['test_recall']))
print('Precision Score: ',score['test_precision'], " Average",mean(score['test_precision']))
print('F1 Score: ',score['test_f1_score'], " Average",mean(score['test_f1_score']))
print('Accuracy: ',score['test_accuracy'], " Average",mean(score['test_accuracy']))

```

5.3.2 FRONT-END CODE

accounts urls.py

```

from django.urls import path,include
from . import views

urlpatterns=[
    path("",views.home,name="home"),
    path('about',views.about,name="about"),
    path('contactus',views.contactus,name="contactus"),
    path('detect',views.detect,name="detect"),
    path('details',views.details,name="details"),
]

```


accounts views.py

```
from django.shortcuts import render, redirect
from django.contrib import messages
from django.conf import settings
from django.template.loader import render_to_string
import pandas as pd
import pickle
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from keras import models
from keras.models import load_model
import tensorflow as tf
import keras
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Layer, Dense
import numpy as np
from sklearn.metrics import log_loss, confusion_matrix, accuracy_score
from django.conf import settings
from django.core.mail import send_mail
import xgboost

# Create your views here.
def home(request):
    return render(request, 'home.html')

def about(request):
    return render(request, 'about.html')

def details(request):
    return render(request, 'details.html')

def contactus(request):
    if request.method == "POST":
        name = request.POST['name']
        email = request.POST['email']
        message = request.POST['message']
        subject = f'Message from {name}'
        message = f'Email: {email} {message}'
        email_from = settings.EMAIL_HOST_USER
        recipient_list = ['18jg1a0525.yasaswini@gvpcew.ac.in', ]
        send_mail( subject, message, email_from, recipient_list )
        messages.info(request, 'Message sent!!!')
        return render(request, 'contactus.html')

    return render(request, 'contactus.html')
```

```

def detect(request):
    if request.method=="POST":
        if 'clear' in request.POST:
            return render(request,'detect.html')

        data=pd.read_csv('data.csv',delimiter=',')
        sc=int(request.POST['sc'])
        fc=int(request.POST['fc'])
        foc=int(request.POST['foc'])
        fac=int(request.POST['fac'])
        lc=int(request.POST['lc'])
        gan=request.POST['gan']
        cl=request.POST['cl']

        if gan=='fnn':
            if cl=='rf':
                model=pickle.load(open('FNN_RF.sav','rb'))
            else:
                model=pickle.load(open('FNN_XG.sav','rb'))
        elif gan=='lstm':
            if cl=='rf':
                model=pickle.load(open('LSTM_RF.sav','rb'))
            else:
                model=pickle.load(open('LSTM_XG.sav','rb'))
        elif gan=='bilstm':
            if cl=='rf':
                model=pickle.load(open('BiLSTM_RF.sav','rb'))
            else:
                model=pickle.load(open('BiLSTM_XG.sav','rb'))
        else:
            if cl=='rf':
                model=pickle.load(open('GRU_RF.sav','rb'))
            else:
                model=pickle.load(open('GRU_XG.sav','rb'))

        if (sc>50000 or fc>50000 or foc>50000 or fac>50000 or lc>50000):
            messages.info(request,'Enter valid data')
            return render(request,'detect.html')

        d=[[sc,fc,foc,fac,lc]]

        data1 = data.iloc[:,1:-1].append(pd.DataFrame(d,
        columns=['statuses_count','followers_count','friends_count','favourites_count','listed_count']),
        ignore_index=True)

        ans=model.predict(data1)
        print(accuracy_score(data.iloc[:,6],ans[:-1]))
        return render(request,'detect.html',{'ans':ans[:-1]})

```

```
return render(request,'detect.html')
```

bot urls.py

```
urlpatterns = [  
    path("",include('accounts.urls')),  
    path('admin/', admin.site.urls),]
```

bot settings.py

```
"""
```

Django settings for bot project.

Generated by 'django-admin startproject' using Django 4.0.4.

For more information on this file, see
<https://docs.djangoproject.com/en/4.0/topics/settings/>

For the full list of settings and their values, see
<https://docs.djangoproject.com/en/4.0/ref/settings/>
"""

```
from pathlib import Path  
import os
```

```
# Build paths inside the project like this: BASE_DIR / 'subdir'.  
BASE_DIR = Path(__file__).resolve().parent.parent
```

```
# Quick-start development settings - unsuitable for production  
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET_KEY = 'django-insecure-  
uea9ud4wfcadw*ipxjv@w(xp4oi6tnjl%xwa**9!oge@^56h4d'
```

```
# 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',  
    'accounts',
```

```

]

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 = 'bot.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS':
[os.path.join(BASE_DIR, 'templates'), os.path.join(BASE_DIR, 'accounts/templates')],
        '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 = 'bot.wsgi.application'

# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/4.0/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/4.0/topics/i18n/

```

```
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
```

```
USE_TZ = True
```

```

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

```

```
STATIC_URL = 'static/'
```

```

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static')
]
STATIC_ROOT=os.path.join(BASE_DIR, 'assets')

```

```

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

```

```

# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

```

```
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

```
DEFAULT_FROM_EMAIL='email'
```

```

EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = 'email'
EMAIL_HOST_PASSWORD = 'Password'

```

home.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Home Page</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

  <style>

    h2
    {
      text-align: center;
      font-size: 1.5cm;
      padding: 40px;
      border: 1px;
      margin: 0%;
      background-color: rgb(192,192,192);
    }

    img{
      float:left;
    }

    h1{
      text-align:center;
      font-size:35px;
      word-spacing:5px;
      text-shadow: 4px 4px 3px darkgrey; margin-bottom: 2px;
      margin-top: 7px;
      color: darkslategrey;
    }

    .dd
    {
      padding:10px;
      background-color: #dcdcdc ;
      border-top:2px solid grey;
      border-bottom:2px solid grey;
      border-left: 2px solid grey;
      border-right: 2px solid grey;
      margin-bottom: 3px;
    }
```

```

.dd a
{
    text-decoration: none;
}
.dropbtn1, .dd a
{
    color: black;
    width:100px;
    font-size: 16px;
    border: none ;
    cursor: pointer ;
    background-color: #dcdcdc;
}

a{

    font-family: 'Times New Roman', Times, serif;
}

.logo{
    background:#DCDCDC;
    border-top:2px solid grey;
    border-bottom:2px solid grey;
}

.centered {

    position: absolute;
    top: 70%;
    left: 70%;
    transform: translate(-50%, -50%);

}

.heading {
    background: rgba(0,0,0,0.2);
    padding: 40px;
    text-align: center;
    position: absolute;
    top: 30%;
    right:50%;
    left: 10%;
}

.text{
    text-align: center;
    position: absolute;
    top: 60%;
    right:50%;
    left: 15%;
}

```

```

.current {
  font-weight: 600;
}

.current:before {
  -moz-transform: rotateZ(45deg);
  -webkit-transform: rotateZ(45deg);
  -ms-transform: rotateZ(45deg);
  transform: rotateZ(45deg);
  width: 0.75em;
  height: 0.75em;
  content: "";
  display: block;
  position: absolute;
  bottom: -0.5em;
  left: 50%;
  margin-left: -0.375em;
  background-color: #37c0fb;
  background-image: url("images/bg01.png");
}

.current a {
  color: #fff;
}

.current:before {
  opacity: 0;
}

</style>
</head>

<body>
<div class="logo">
  
  <h1>Gayathri Vidya Parishad College Of Engineering For Women</h1>
  <h3 style="text-align: center; margin-top:0px;margin-
bottom:20px;color:maroon;">(Approved by AICTE New Delhi, Affiliated to JNTUK
Kakinada)<br>
  (Accredited by National Board of Accreditation(NBA) for B.Tech CSE,ECE & IT - Valid
from 2019-20 to 2021-22)<br>
  Kommadi, Madhurawada, Visakhapatnam</h3>
</div>

<div class="dd" style="text-align:center;padding: 10px 10px;">
  <a href="#" style="padding: 10px; font-size: 30px;" class="current">Home</a>
  <a href="about" style="padding:10px; font-size:30px">About</a>

```



```

<a href="details" style="padding:10px; font-size:30px">Details</a>
<a href="detect" style="padding:10px; font-size:30px">Detect</a>
  <a href="contactus" style="padding: 10px; font-size:30px;">Contact us</a>

</div>
<div class="cont">
  <div class="image"></div>
</div>
<h1 style=" font-size:45px;" class="heading">Detection of Malicious Social Bots using
Variational AutoEncoder GAN </h1>
<div class="text"><br>
  <p style="font-size:30px;">Project Batch – A1<br>
  Academic Batch : 2018 – 2022
</p>
  <p style="font-size: 20px;">
    Project Guide -Dr. P.V.S.L. Jagadamba <br>
    Project Members<br>
    G. Bhavya Yaraswini (18JG1A0525)<br>
    G. Ragini (18JG1A0530)<br>
    K. Niharika (18JG1A0549)<br>
    I. Sahithi (18JG1A0535)<br>
  </p>
</div>
</body>
</html>

```

about.html

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Home Page</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

  <style>

    h2
    {
      text-align: center;
      font-size: 1.5cm;
      padding: 40px;
      border: 1px;
      margin: 0%;
      background-color: rgb(192,192,192);
    }

```

```
img{
  float:left;
}

h1{
  text-align:center;
  font-size:35px;
  word-spacing:5px;
  text-shadow: 4px 4px 3px darkgrey; margin-bottom: 2px;
  margin-top: 7px;
  color: darkslategrey;
}
```

```
.dd
{
  padding:10px;
  background-color: #dcdcdc ;
  border-top:2px solid grey;
  border-bottom:2px solid grey;
  border-left: 2px solid grey;
  border-right: 2px solid grey;
  margin-bottom: 3px;
}
```

```
.dd a
{
  text-decoration: none;
}
.dd a
{
  color: black;
  width:100px;
  font-size: 16px;
  border: none ;
  cursor: pointer ;
  background-color: #dcdcdc;
}
```

```
a{

  font-family: 'Times New Roman', Times, serif;
}
```

```
.logo{
  background:#DCDCDC;
  border-top:2px solid grey;
  border-bottom:2px solid grey;
```

```

}

.centered {

    position: absolute;
    top: 70%;
    left: 70%;
    transform: translate(-50%, -50%);

}

.text{
    background: rgba(0,0,0,0.2);
    padding: 40px;
    text-align: center;
    position: absolute;
    top: 45%;
    right:50%;
    left: 10%;
}

.current {
    font-weight: 600;
}

.current:before {
    -moz-transform: rotateZ(45deg);
    -webkit-transform: rotateZ(45deg);
    -ms-transform: rotateZ(45deg);
    transform: rotateZ(45deg);
    width: 0.75em;
    height: 0.75em;
    content: "";
    display: block;
    position: absolute;
    bottom: -0.5em;
    left: 50%;
    margin-left: -0.375em;
    background-color: #37c0fb;
}

.current a {
    color: #fff;
}

.current:before {
    opacity: 0;
}
body
{
    background-image: linear-gradient(180deg,#b1b1a5,#457379);

```

```

    height:2300px;
}

.content
{
    text-align: center;
    font-weight:500;
    font-size:x-large;
    width: 50%;
    margin-left: 25%;
    font-family: sans-serif;
}

.documents a{
    text-decoration: none;
    font-size: 40px;
    color: black;
    text-align: center;
    padding-left: 40%;
    font-style: italic;
}

span
{
    margin: 1em 0 0.5em 0;
    font-weight: 100;
    text-transform: uppercase;
    color: #301744;
    font-style: italic;
    font-family: 'Josefin Sans', sans-serif;
    font-size:40px;
    line-height: 54px;
    text-shadow: 2px 5px 0 rgba(0,0,0,0.2);
}

.container1,.container2
{
    display:flex;
    align-items:center;
    margin: 30px;
    border:2px solid black;
    padding:10px;
    background-color: rgb(171, 189, 190,0.2);
}

.container1 img, .container2 img{
    width:300px;
    height: 150px;
    margin-right: 20px;
}

```

```

.container1 p, .container2 p{
    font-style: italic;
    font-size:x-large;
}

.container1{
    margin-right: 49%;
    margin-left: 0%;
    border-left: 0px;
}

.container2{
    margin-left: 49%;
    margin-right: 0%;
    border-right: 0px;
}

</style>
</head>

<body>
<div class="logo">
    
    <h1>Gayathri Vidya Parishad College Of Engineering For Women</h1>
    <h3 style="text-align: center; margin-top:0px;margin-
bottom:20px;color:maroon;">(Approved by AICTE New Delhi, Affiliated to JNTUK
Kakinada)<br>
    (Accredited by National Board of Accreditation(NBA) for B.Tech CSE,ECE & IT - Valid
from 2019-20 to 2021-22)<br>
    Kommadi, Madhurawada, Visakhapatnam</h3>
</div>

<div class="dd" style="text-align:center;padding: 10px 10px;">
    <a href="/" style="padding: 10px; font-size: 30px;">Home</a>
    <a href="#" style="padding:10px; font-size:30px" class="current">About</a>
    <a href="details" style="padding:10px; font-size:30px">Details</a>
    <a href="detect" style="padding:10px; font-size:30px">Detect</a>
    <a href="contactus" style="padding: 10px; font-size:30px;">Contact us</a>

</div>
<p class="content">
    <br><span>PREDICT TO PROTECT</span><br>
    A web application to detect malicious social bots based on twitter account information,
    thus protecting users from security attacks and spam content.<br><br>
    <br><br><br><br><br><br><br><br>

```

<i>What is a bot ? A social bot represents a software program which automatically creates multiple fake accounts, frequently interacts with legitimate user accounts.</i>

<button style="background-color:#6f9186">Know more</button>

</p>

<p id="facts">

<p style="text-align: center;">Do Bots really impact?</p>

<div class="container1">

<p>1/3rd of tweets on any topic is generated by social bots.</p>

</div>

<div class="container2">

<p>According to 2020 survey, nearly 25% of Twitter users are bots.</p>

</div>

<div class="container1">

<p>In 2020, 21% of Donald Trump followers are bots.</p>

</div>

<div class="container2">

<p>In 2016 U.S. Elections, out of all generated tweets related to elections, 19% are bot generated.</p>

</div>

<div class="container1">

<p>At least 30% of the Twitter followers of both presidential candidates were bots.</p>

</div>

</p>

<p id="documents" >

<p style="text-align: center;">Related Documents</p>

<div class="documents">

Download ppt

Download documentation

Download code

</div>

</p>

</body>

</html>

details.html

<!DOCTYPE html>

```

<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Home Page</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

  <style>

    h2
    {
      text-align: center;
      font-size: 1.5cm;
      padding: 40px;
      border: 1px;
      margin: 0%;
      background-color: rgb(192,192,192);
    }

    img{
      float:left;
    }

    h1{
      text-align:center;
      font-size:35px;
      word-spacing:5px;
      text-shadow: 4px 4px 3px darkgrey; margin-bottom: 2px;
      margin-top: 7px;
      color: darkslategrey;
    }

    .dd
    {
      padding:10px;
      background-color: #dcdcdc ;
      border-top:2px solid grey;
      border-bottom:2px solid grey;
      border-left: 2px solid grey;
      border-right: 2px solid grey;
      margin-bottom: 3px;
    }

    .dd a
    {
      text-decoration: none;

```

```

}
.dd a
{
  color: black;
  width:100px;
  font-size: 16px;
  border: none ;
  cursor: pointer ;
  background-color: #dcdcdc;
}

a{

  font-family: 'Times New Roman', Times, serif;
}

.logo{
  background:#DCDCDC;
  border-top:2px solid grey;
  border-bottom:2px solid grey;
}

.centered {

  position: absolute;
  top: 70%;
  left: 70%;
  transform: translate(-50%, -50%);

}
.text{
  background: rgba(0,0,0,0.2);
  padding: 40px;
  text-align: center;
  position: absolute;
  top: 45%;
  right:50%;
  left: 10%;
}

.current {
  font-weight: 600;
}

.current:before {
  -moz-transform: rotateZ(45deg);
  -webkit-transform: rotateZ(45deg);
  -ms-transform: rotateZ(45deg);
  transform: rotateZ(45deg);
  width: 0.75em;

```



```

height: 0.75em;
content: "";
display: block;
position: absolute;
bottom: -0.5em;
left: 50%;
margin-left: -0.375em;
background-color: #37c0fb;
}

.current a {
  color: #fff;
}

.current:before {
  opacity: 0;
}
body
{
  background-image: linear-gradient(180deg,#b1b1a5,#457379);
  height:auto;
}

.content
{
  text-align: center;
  font-weight:500;
  font-size:x-large;
  width: 60%;
  margin-left: 20%;
  font-family: sans-serif;
}

.documents a{
  text-decoration: none;
  font-size: 40px;
  color: black;
  text-align: center;
  padding-left: 40%;
  font-style: italic;
}

span
{
  margin: 1em 0 0.5em 0;
  font-weight: 100;
  text-transform: uppercase;
  color: #301744;
  font-style: italic;
  font-family: 'Josefin Sans', sans-serif;
}

```

```

font-size:40px;
line-height: 54px;
text-shadow: 2px 5px 0 rgba(0,0,0,0.2);
}

.container1,.container2
{
display:flex;
align-items:center;
margin: 30px;
border:2px solid black;
padding:10px;
background-color: rgb(171, 189, 190,0.2);
}

.container1 img, .container2 img{
width:300px;
height: 150px;
margin-right: 20px;
}

.container1 p, .container2 p{
font-style: italic;
font-size:x-large;
}

.container1{
margin-right: 49%;
margin-left: 0%;
border-left: 0px;
}

.container2{
margin-left: 49%;
margin-right: 0%;
border-right: 0px;
}

.cont
{
font-size:x-large;
width: 75%;
text-align: justify;
margin-left:12%;
}

table,td{
text-align: center;
font-family: 'Times New Roman', Times, serif;
font-weight:700;

```

}

</style>
</head>

<body>
 <div class="logo">

 <h1>Gayathri Vidya Parishad College Of Engineering For Women</h1>
 <h3 style="text-align: center; margin-top:0px;margin-bottom:20px;color:maroon;">(Approved by AICTE New Delhi, Affiliated to JNTUK Kakinada)

 (Accredited by National Board of Accreditation(NBA) for B.Tech CSE,ECE & IT - Valid from 2019-20 to 2021-22)

 Kommadi, Madhurawada, Visakhapatnam</h3>
 </div>

<div class="dd" style="text-align:center;padding: 10px 10px;">
 Home
 About
 Details
 Detect
 Contact us
</div>

<p class="content">

 <button style="background-color:#6f9186">Introduction</button>
 <button style="background-color:#6f9186">Proposed System</button>
 <button style="background-color:#6f9186">Architectural Diagram</button>
 <button style="background-color:#6f9186">Methodology</button>
 <button style="background-color:#6f9186">Performance</button>
</p>
<p id="intro">

<p style="text-align: center;">Introduction</p>
 <p class="cont">Online social networks (OSNs) have influenced users to share information related to social activities like news, links, opinion and promote products and services.

It also provides an additional space for an attacker to steal a user's personal information and to perform malicious activities such as generating fake identities and performing phishing attacks in OSNs. Malicious social bots represents software programs that automatically creates multiple fake accounts, frequently interacts


```

</p>
</p>

<p id="per">
  <br><p style="text-align: center;"><span>Performance</span></p><br>
  <p class="cont" >
    <center style="font-size:x-large;">Various models performance in terms of
accuracy.</center>
    <table border="2px solid black" style="font-size:xx-large;margin-left: 13%;"
width="70%" cellpadding:10 bordercolor="black">
      <tr style="text-align: center;">
        <td></td>
        <td colspan="3">Randomforest with Adaboost</td>
        <td colspan="3">Xgboost</td>
      </tr>

      <tr>
        <td></td>
        <td>Without AE</td>
        <td>AE</td>
        <td>VAE</td>
        <td>Without AE</td>
        <td>AE</td>
        <td>VAE</td>
      </tr>

      <tr>
        <td>Without GAN</td>
        <td>89.7</td>
        <td>84.8</td>
        <td>86.02</td>
        <td>88.8</td>
        <td>88.2</td>
        <td>89.9</td>
      </tr>

      <tr>
        <td>FNN-GAN</td>
        <td>92.6</td>
        <td>94.1</td>
        <td>94.8</td>
        <td>91.4</td>
        <td>92.7</td>
        <td>92.9</td>
      </tr>

      <tr>
        <td>LSTM-GAN</td>
        <td>92.3</td>
        <td>94.9</td>

```

	96.3
	91.5
	92.5
	93.3
BiLSTM-GAN	92.4
	94.5
	97.4
	91.9
	92.8
	94.1
GRU-GAN	93.8
	95.9
	97.3
	92.0
	93.6
	93.9

Performance of Xgboost with 5 fold cross validation.

Model	Recall	Precision	F1 score	Accuracy
FNN-GAN	89.8	92.4	91.1	90.3
LSTM-GAN				

```
 <td>89.5</td>  <td>93.5</td>  <td>91.4</td>  <td>90.8</td> </tr>  <tr>  <td>BiLSTM-GAN</td>  <td>89.2</td>  <td>93.7</td>  <td>91.4</td>  <td>92.8</td> </tr>  <tr>  <td>GRU-GAN</td>  <td>89.0</td>  <td>93.8</td>  <td>91.3</td>  <td>92.8</td> </tr> </table> </p> | | | | | | | | | | | | | |
```

```

</body>
</html>

```

detect.html

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Home Page</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

  <style>

  h2
  {
    text-align: center;
    font-size: 1.5cm;
    padding: 40px;
    border: 1px;
    margin: 0%;
    background-color: rgb(192,192,192);
  }

```

```
img{
  float:left;
}

h1{
  text-align:center;
  font-size:35px;
  word-spacing:5px;
  text-shadow: 4px 4px 3px darkgrey; margin-bottom: 2px;
  margin-top: 7px;
  color: darkslategrey;
}
```

```
.dd
{
  padding:10px;
  background-color: #dcdcdc ;
  border-top:2px solid grey;
  border-bottom:2px solid grey;
  border-left: 2px solid grey;
  border-right: 2px solid grey;
  margin-bottom: 3px;
}
```

```
.dd a
{
  text-decoration: none;
}
.dd a
{
  color: black;
  width:100px;
  font-size: 16px;
  border: none ;
  cursor: pointer ;
  background-color: #dcdcdc;
}
```

```
a{

  font-family: 'Times New Roman', Times, serif;
}
```

```
.logo{
  background:#DCDCDC;
  border-top:2px solid grey;
  border-bottom:2px solid grey;
```



```

}

.centered {

    position: absolute;
    top: 70%;
    left: 70%;
    transform: translate(-50%, -50%);

}

.text{
    background: rgba(0,0,0,0.2);
    padding: 40px;
    text-align: center;
    position: absolute;
    top: 45%;
    right: 50%;
    left: 10%;
}

.current {
    font-weight: 600;
}

.current:before {
    -moz-transform: rotateZ(45deg);
    -webkit-transform: rotateZ(45deg);
    -ms-transform: rotateZ(45deg);
    transform: rotateZ(45deg);
    width: 0.75em;
    height: 0.75em;
    content: "";
    display: block;
    position: absolute;
    bottom: -0.5em;
    left: 50%;
    margin-left: -0.375em;
    background-color: #37c0fb;
}

.current a {
    color: #fff;
}

.current:before {
    opacity: 0;
}
body
{
    background-image: linear-gradient(180deg,#b1b1a5,#457379);

```

```

height:800px;
}

form {
  background: rgba(0,0,0,0.2);
  padding: 30px;
  text-align: center;
  position: absolute;
  top: 30%;
  left: 5%;
}

#contact {

  text-align: center; padding: 2px;
  font-size: 20px;
}

#form p input{ padding-bottom: 3px;
background-color: green; border-radius: 8px; border: none;
font-size: 15px; padding: 6px; color: white;
}

.output
{
  position:absolute;
  margin-left: 30%;
  margin-top: 2%;
  font-size:xx-large;
  font-weight: bold;
  text-align: center;
}

.accuracy
{
  position:absolute;
  margin-top: 23%;
  margin-left:7%;
  font-size:x-large;
  font-weight:500;
  text-align: center;
}

.message{
  color:darkred;
  margin-bottom:5mm;
  font-size: large;
}

```

```

</style>
</head>

<body>
  <div class="logo">
    
    <h1>Gayathri Vidya Parishad College Of Engineering For Women</h1>
    <h3 style="text-align: center; margin-top:0px;margin-
bottom:20px;color:maroon;">(Approved by AICTE New Delhi, Affiliated to JNTUK
Kakinada)<br>
    (Accredited by National Board of Accreditation(NBA) for B.Tech CSE,ECE & IT - Valid
from 2019-20 to 2021-22)<br>
    Kommadi, Madhurawada, Visakhapatnam</h3>
  </div>

  <div class="dd" style="text-align:center;padding: 10px 10px;">
    <a href="/" style="padding: 10px; font-size: 30px;">Home</a>
    <a href="/about" style="padding:10px; font-size:30px" >About</a>
    <a href="details" style="padding:10px; font-size:30px">Details</a>
    <a href="#" style="padding:10px; font-size:30px" class="current">Detect</a>
    <a href="contactus" style="padding: 10px; font-size:30px;">Contact us</a>
  </div>

  <div class="cont">
    <div class="image"></div>

    <div class="output">
      {% if ans == 1 %}
      IT'S A BOT<br>
      
      {% elif ans == 0 %}
      IT'S A HUMAN<br>
      
      {% endif %}
    </div>

    <form id="form" name="contact-form" method="post" action="" >
      {% csrf_token %}
      <div class="message">
        {% for m in messages %}
        {{m}}
        {% endfor %}
      </div>
      <p style="color: maroon;">Enter only positive numerical data</p>
      <p style="color: maroon;">Maximum Range: 50000</p><br>
      <label for="gan">Choose GAN model:</label><br>

```



```

margin: 0%;
background-color: rgb(192,192,192);
}

img{
float:left;
}

h1{
text-align:center;
font-size:35px;
word-spacing:5px;
text-shadow: 4px 4px 3px darkgrey; margin-bottom: 2px;
margin-top: 7px;
color: darkslategrey;
}

.dd
{
padding:10px;
background-color: #dcdcdc ;
border-top:2px solid grey;
border-bottom:2px solid grey;
border-left: 2px solid grey;
border-right: 2px solid grey;
margin-bottom: 3px;
}

.dd a
{
text-decoration: none;
}
.dd a
{
color: black;
width:100px;
font-size: 16px;
border: none ;
cursor: pointer ;
background-color: #dcdcdc;
}

a{

font-family: 'Times New Roman', Times, serif;
}

```

```

.logo{
  background:#DCDCDC;
  border-top:2px solid grey;
  border-bottom:2px solid grey;
}

.centered {

  position: absolute;
  top: 70%;
  left: 70%;
  transform: translate(-50%, -50%);

}

.text{
  background: rgba(0,0,0,0.2);
  padding: 40px;
  text-align: center;
  position: absolute;
  top: 45%;
  right:50%;
  left: 10%;
}

.current {
  font-weight: 600;
}

.current:before {
  -moz-transform: rotateZ(45deg);
  -webkit-transform: rotateZ(45deg);
  -ms-transform: rotateZ(45deg);
  transform: rotateZ(45deg);
  width: 0.75em;
  height: 0.75em;
  content: "";
  display: block;
  position: absolute;
  bottom: -0.5em;
  left: 50%;
  margin-left: -0.375em;
  background-color: #37c0fb;
}

.current a {
  color: #fff;
}

.current:before {
  opacity: 0;
}

```

```

}
body
{
background-image: linear-gradient(180deg,#b1b1a5,#457379);
height:800px;
}
#contact {

text-align: center; padding: 2px;
font-size: 20px;
}

.section-header{ text-align: center; font-size: 25px;
margin-bottom: 10px; font-weight: bold;
text-transform: uppercase;

}

#contact .contact-info { margin-bottom: 10px; text-align: center;
}

#contact .contact-info i { font-size: 48px; display: inline-block; margin-bottom: 10px;

color: #18d26e;

}

.contact-address{ padding-left: 35px; float: left;
}

#contact .contact-info address, #contact .contact-info p { margin-bottom: 0;
color: #000;

}

#contact .contact-info h3 { font-size: 18px;
margin-bottom: 15px; font-weight: bold;
text-transform: uppercase; color: #999;
}

.contact-phone{ float: right;
padding-right: 15%;

}

.contact-info a {

color: #000;

text-decoration: none;

```

```

}

#contact .contact-info a:hover { color: #18d26e;
}

#contact .contact-address, #contact .contact-phone, #contact .contact-email { margin-
bottom: 20px;
}

.contact-email{ padding-right: 30%;
}

#form p input{ padding-bottom: 3px;
background-color: green; border-radius: 8px; border: none;
font-size: 15px; padding: 6px; color: white;
}

.message{
color:darkred;
margin-bottom:5mm;
}

</style>
</head>

<body>
<div class="logo">

<h1>Gayathri Vidya Parishad College Of Engineering For Women</h1>
<h3 style="text-align: center; margin-top:0px;margin-
bottom:20px;color:maroon;">(Approved by AICTE New Delhi, Affiliated to JNTUK
Kakinada)<br>
(Accredited by National Board of Accreditation(NBA) for B.Tech CSE,ECE & IT - Valid
from 2019-20 to 2021-22)<br>
Kommadi, Madhurawada, Visakhapatnam</h3>
</div>

<div class="dd" style="text-align:center;padding: 10px 10px;">
<a href="/" style="padding: 10px; font-size: 30px;">Home</a>
<a href="/about" style="padding:10px; font-size:30px" >About</a>
<a href="details" style="padding:10px; font-size:30px">Details</a>
<a href="detect" style="padding:10px; font-size:30px">Detect</a>
<a href="#" style="padding: 10px; font-size:30px;" class="current">Contact us</a>
</div>
<!-- Contact Us -->

<section id="contact" class="section-bg wow fadeInUp">

```



```

<div class="container">

<div class="row contact-info">

<div class="col-md-4">

<div class="contact-address">

<i class="ion-ios-location-outline"></i>

<h3 style="color: black;">Address</h3>

<address>Madhurawada, Visakhapatnam-530048, AP, India.</address>

</div>

</div>

<div class="col-md-4">

<div class="contact-phone">

<i class="ion-ios-telephone-outline"></i>

<h3 style="color: black;">Phone Number</h3>

<p><a href="tel:+919494130993">+91 9494130993</a></p>

</div>

</div>

<div class="col-md-4">

<div class="contact-email">

<i class="ion-ios-email-outline"></i>

<h3 style="color: black;">Email</h3>

<p><a href="mailto:18jg1a0525.bhavya@gvpcew.ac.in">18jg1a0525.bhavya@gvpcew.ac.in</a></p>

</div>

</div>

</div>

</div>

<div class="message">

```

```

    {% for m in messages %}
    {{m}}
    {% endfor %}
</div>

<form id="form" name="contact-form" method="post" action="" >
    {% csrf_token %}
    <input type="text" name="name" required="required" id="contact-name"
placeholder="Your Name"><br><br>
    <input type="email" name="email" required="required" id="contact-email"
placeholder="Your Email"><br><br>
    <textarea placeholder="Message" name="message" required="required" id="contact-
message" cols="70" rows="6"></textarea><br><br>
    <p><input type="submit" value="Send message" name="contact-submit"><p>

</form>
</div>
</div>
</section>
</body>
</html>

```

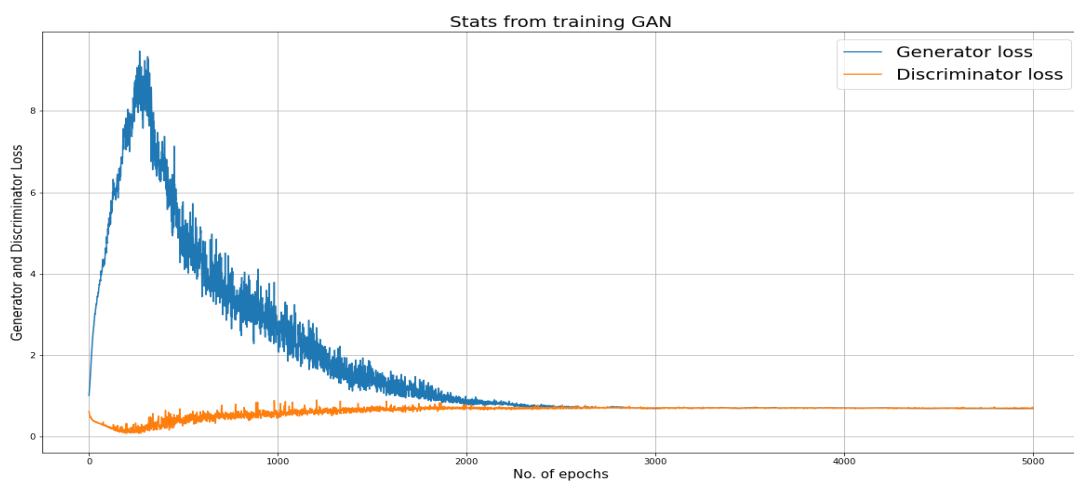
5.4 RESULTS

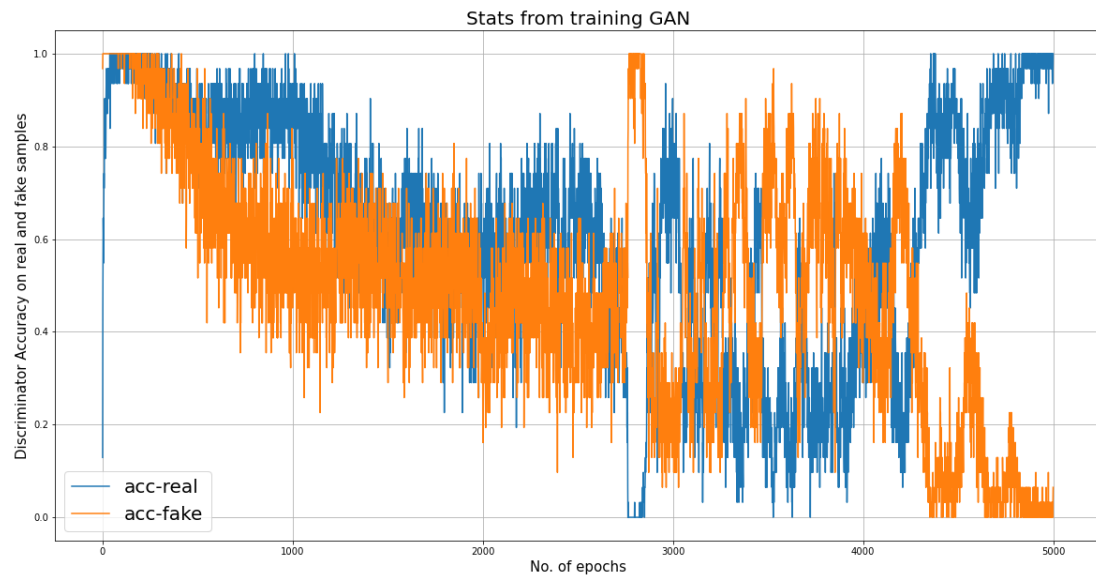
5.4.1 RESULT ANALYSIS

The generator and discriminator models are trained with 5000 epochs. The performance of both the models (generator and discriminator) are listed below in [Table 5](#). Both the generator and discriminator models train themselves to produce fake samples which are hard to be distinguished from real. The better GAN model is created when the loss values of both the models are nearly same i.e., the generator is generating better fake data and discriminator is differentiating them correctly. As we can see, in the 5000 epoch the generator and discriminator loss are nearly same. Now the accuracy of the discriminator on real and fake samples is 1 when the model started training. Accuracy 1 on fake samples says that the discriminator is easily identifying the fake samples from real. When the model is completely trained, it can be observed that the accuracy on fake samples is very low (0.06) showing the fact that the model is generating very good fake samples. The visualization is shown in [Screen 1](#). and [Screen 2](#).

Table 5. GAN Analysis

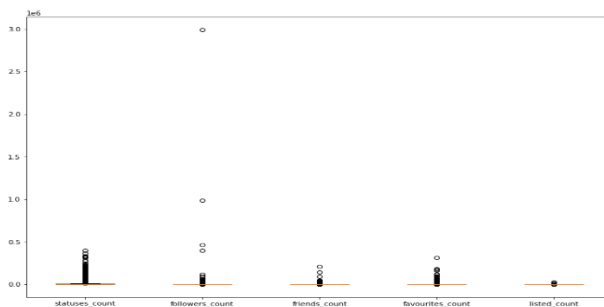
Epoch	Loss		Discriminator Accuracy	
	Generator	Discriminator	Real Samples	Fake Samples
200	0.07	7.15	1.00	1.00
400	0.33	6.74	0.83	0.83
600	0.52	3.43	0.83	0.61
800	0.49	3.59	0.90	0.64
....				
4200	0.69	0.69	0.29	0.87
4400	0.69	0.68	0.93	0.06
4600	0.68	0.69	0.74	0.16
4800	0.69	0.68	0.90	0.16
5000	0.69	0.68	1.00	0.06

**Screen 1.** Generator and Discriminator Loss



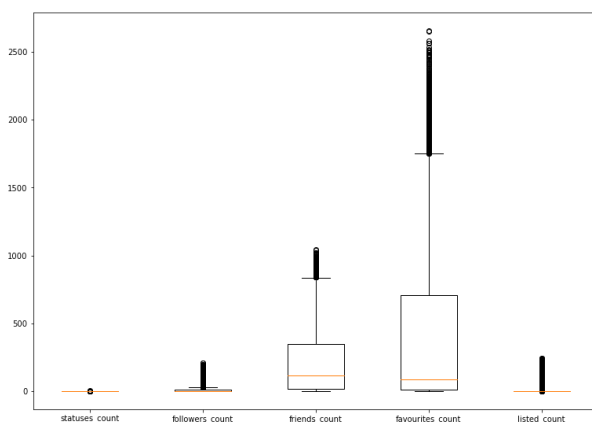
Screen 2. Discriminator Accuracy for real and fake samples

The no. of outliers in the dataset before and after applying preprocessing is shown in [Screen 3.](#) [Screen 4.](#) and [Screen 5.](#)



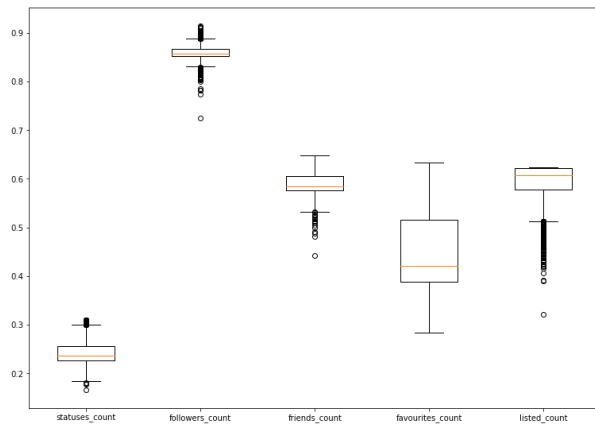
statuses_count 1845
followers_count 1037
friends_count 1449
favourites_count 2346
listed_count 1508

Screen 3. Noise before AE



statuses_count 225
followers_count 489
friends_count 158
favourites_count 621
listed_count 2255

Screen 4. Noise after AE



statuses_count 325
 followers_count 258
 friends_count 17
 favourites_count 0
 listed_count 180

Screen 5. Noise after VAE

The accuracy of Xgboost and AdaBoost model with different implementations of GAN and Auto-encoders is shown in [Table 6](#). It is clearly evident that the model is giving better performance with balanced dataset and Variational Auto-Encoders.

Table 6. Model Comparison in terms of Accuracy

	AdaBoost			Xgboost model		
	Without AE	AE	VAE	Without AE	AE	VAE
Without GAN	89.7	84.8	86.02	88.8	88.2	89.9
FNN-GAN	92.6	94.1	94.8	91.4	92.7	92.9
LSTM-GAN	92.3	94.9	96.3	91.5	92.5	93.3
BiLSTM-GAN	92.4	94.5	97.4	91.9	92.8	94.1
GRU-GAN	93.8	95.9	97.3	92.0	93.6	93.9

Cross validation is a technique of evaluating a model by using different parts of the data for training and testing in each fold so that it reduces overfitting. The data is partitioned into fixed folds and model is trained and evaluated. The final performance of the model is the average

of overall estimate. The performance of Xgboost with 5fold cross validation is depicted in [Table 7](#).

Table 7. Performance of Xgboost with 5 fold cross validation

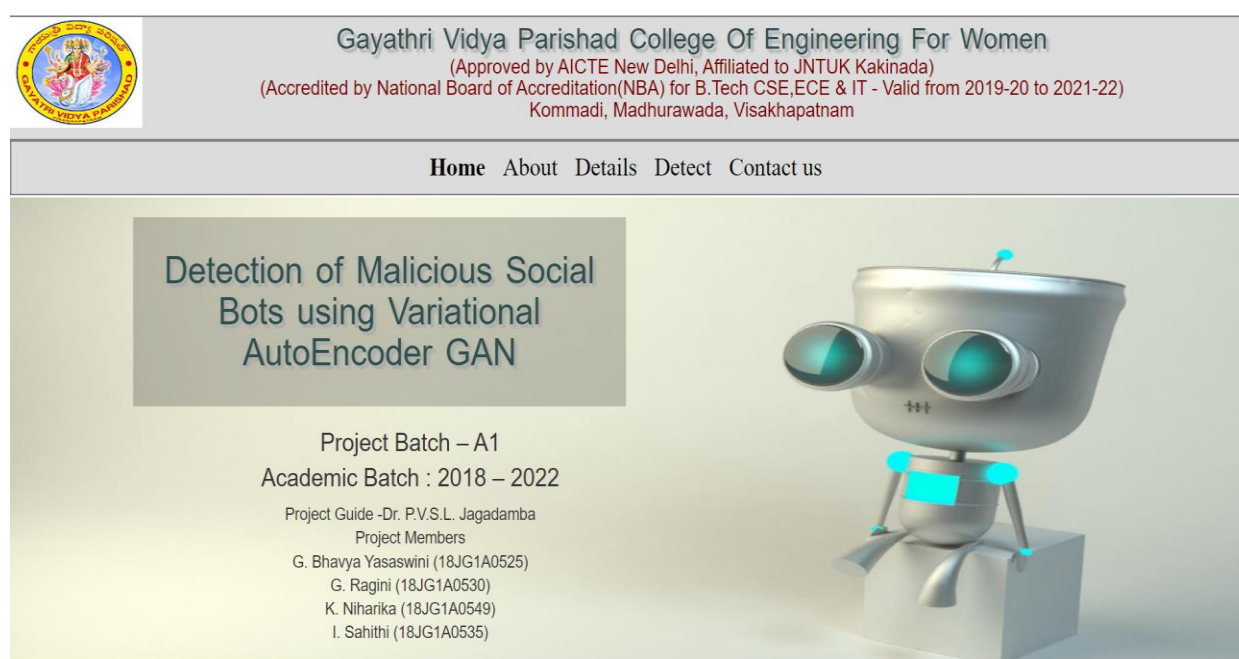
Model	Recall	Precision	F1 Score	Accuracy
FNN-GAN	89.8	92.4	91.1	90.3
LSTM-GAN	89.5	93.5	91.4	90.8
Bi LSTM-GAN	89.2	93.7	91.4	92.8
GRU-GAN	89.0	93.8	91.3	92.8

5.4.2 OUTPUT SCREENS

Home page

It contains About, Details, Detect and Contact us navigation buttons.

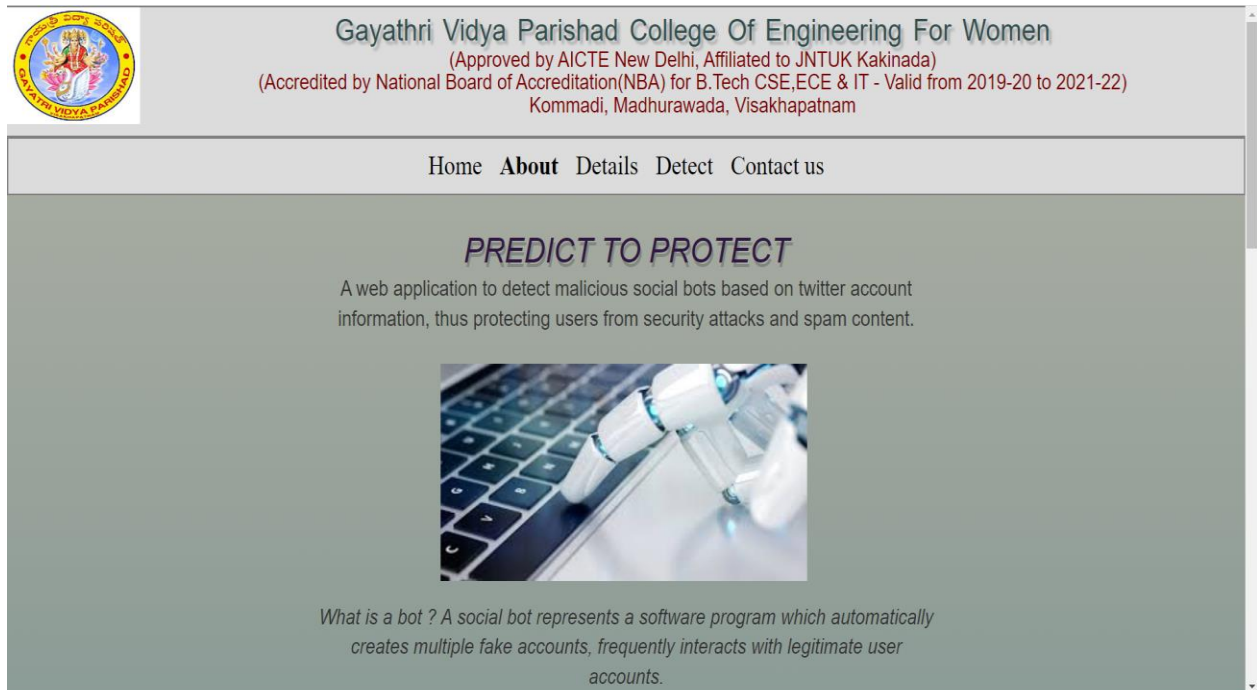
- About: The Page gives a brief idea about the web application.
- Details: This page contains all the necessary information about the implementation
- Detect: This page is used to predict social bots
- Contact us: It contains information required to contact and direct message sending is available.



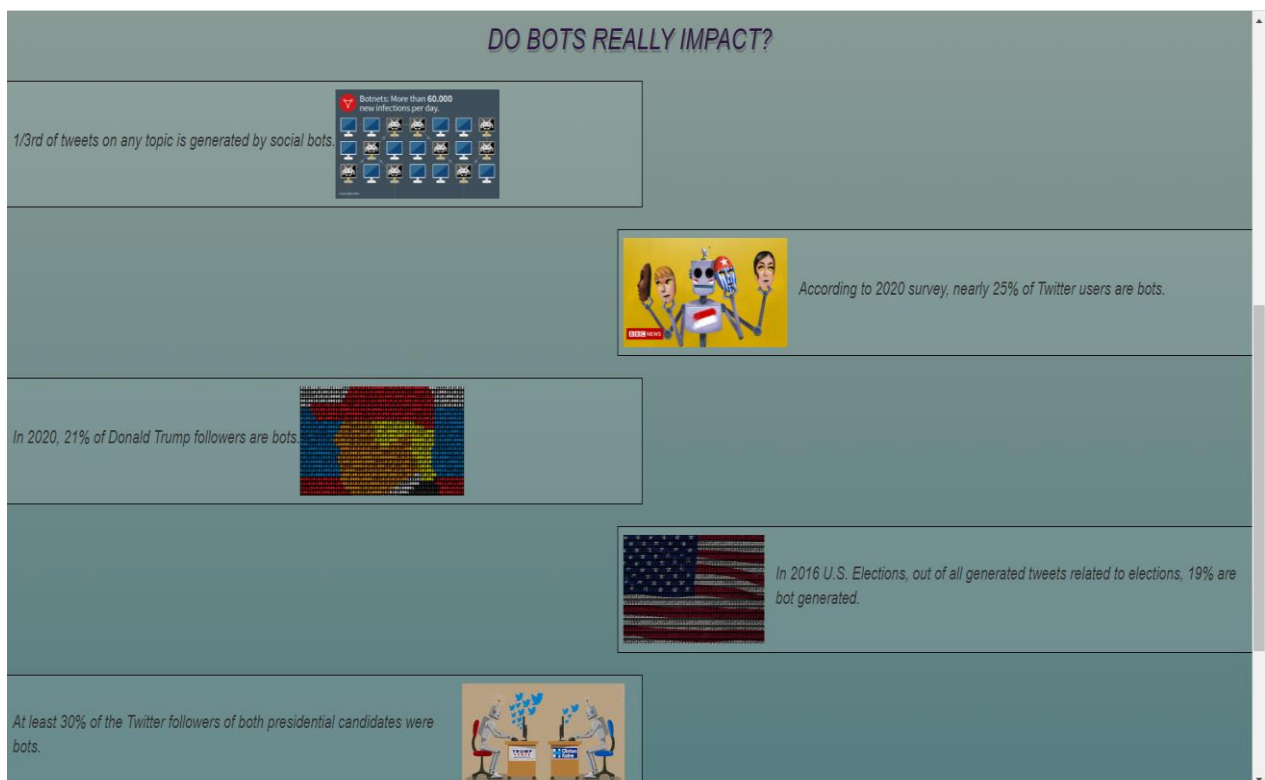
Screen 6. Home page

About page

It gives idea about what the app is and some facts about the social bots.




Screen 7. About page



Screen 8. About page (facts)

Details page

It contains navigation buttons for Introduction, Proposed System, Architectural Diagram, Methodology and Performance of the model.



Gayathri Vidya Parishad College Of Engineering For Women

(Approved by AICTE New Delhi, Affiliated to JNTUK Kakinada)
(Accredited by National Board of Accreditation(NBA) for B.Tech CSE,ECE & IT - Valid from 2019-20 to 2021-22)
Kommadi, Madhurawada, Visakhapatnam

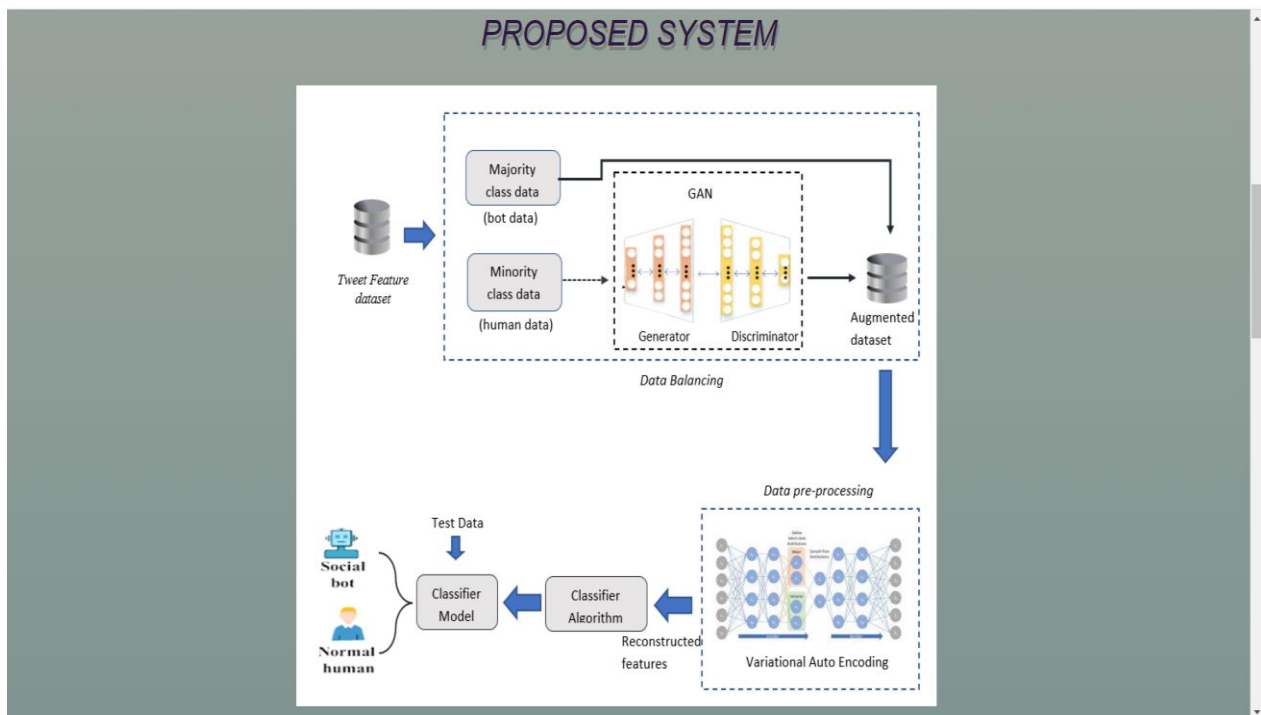
Home About **Details** Detect Contact us

Introduction Proposed System Architectural Diagram Methodology Performance

INTRODUCTION

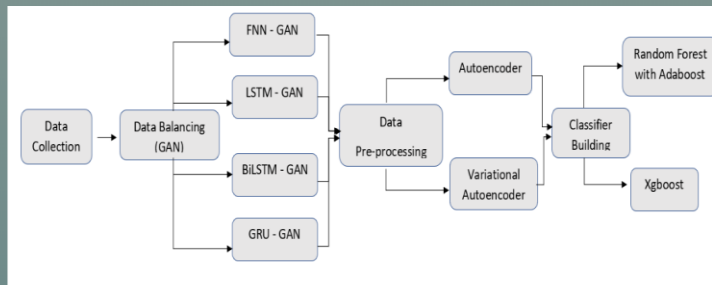
Online social networks (OSNs) have influenced users to share information related to social activities like news, links, opinion and promote products and services. It also provides an additional space for an attacker to steal a user's personal information and to perform malicious activities such as generating fake identities and performing phishing attacks in OSNs. Malicious social bots represents software programs that automatically creates multiple fake accounts, frequently interacts with legitimate user accounts. In many OSNs, the bots are much less in number when compared to legitimate users. Most of the conventional supervised learning models often suffer from a highly imbalanced dataset with many legitimate users belonging to one class and only a few malicious social bots belonging to another class. To solve the imbalanced distribution of malicious social bots, a Variational Auto-encoder Generative Adversarial Network (VAE-GAN) model is proposed which balances the data from Twitter Network and further improves the accuracy of malicious social bot detection.

Screen 9. Details page (Introduction)



Screen 10. Details page (Proposed System)

ARCHITECTURAL DIAGRAM



METHODOLOGY

The proposed Variational Auto Encoder Generative Adversarial Network (VAE-GAN) model, for malicious social bot detection consists of four modules: Data Collection, Data Balancing (GAN), Data Pre-processing (VAE), Classification.

The major contributions are as follows.

1. Balance the minority distribution of data in Twitter Network using Generative Adversarial Network (GAN).
Different GAN models implemented are : FNN-GAN, LSTM-GAN, BiLSTM-GAN, GRU-GAN
2. Reconstruct the features by removing noise from the data using Variational Auto-Encoders (VAE).
Different classifier models implemented are : Random Forest with Adaboost, Xgboost classifier
3. Design different Classifiers to analyse the behaviour of users in different environments.
4. Design of a VAE-GAN algorithm by integrating GAN model with Variational AE on balanced data.
5. Evaluate the performance of the proposed VAE-GAN model in terms of precision, recall, F-measure, and accuracy in the Twitter network.

Screen 11. Details page (Architecture and Methodology)

PERFORMANCE

Various models performance in terms of accuracy.

	Randomforest with Adaboost			Xgboost		
	Without AE	AE	VAE	Without AE	AE	VAE
Without GAN	89.7	84.8	86.02	88.8	88.2	89.9
FNN-GAN	92.6	94.1	94.8	91.4	92.7	92.9
LSTM-GAN	92.3	94.9	96.3	91.5	92.5	93.3
BiLSTM-GAN	92.4	94.5	97.4	91.9	92.8	94.1
GRU-GAN	93.8	95.9	97.3	92.0	93.6	93.9

Performance of Xgboost with 5 fold cross validation.

Model	Recall	Precision	F1 score	Accuracy
FNN-GAN	89.8	92.4	91.1	90.3
LSTM-GAN	89.5	93.5	91.4	90.8
BiLSTM-GAN	89.2	93.7	91.4	92.8
GRU-GAN	89.0	93.8	91.3	92.8

Screen 12. Details page (Performance)

Detect page

This page contains a form to take user features as input and drop down buttons to select GAN and classifier models. An alert message is produced when negative or out of range data is given.

After clicking predict, the output from the selected model is shown on the page.

Gayathri Vidya Parishad College Of Engineering For Women
(Approved by AICTE New Delhi, Affiliated to JNTUK Kakinada)
(Accredited by National Board of Accreditation(NBA) for B.Tech CSE,ECE & IT - Valid from 2019-20 to 2021-22)
Kommadi, Madhurawada, Visakhapatnam

Home About Details **Detect** Contact us

Enter only positive numerical data
Maximum Range: 50000

Choose GAN model:
GRU

Choose Classifier model:
Random Forest with Adaboost

858
22
40
1
0

Predict Reset

Screen 13. Detect page

Gayathri Vidya Parishad College Of Engineering For Women
(Approved by AICTE New Delhi, Affiliated to JNTUK Kakinada)
(Accredited by National Board of Accreditation(NBA) for B.Tech CSE,ECE & IT - Valid from 2019-20 to 2021-22)
Kommadi, Madhurawada, Visakhapatnam

Home About Details **Detect** Contact us

IT'S A BOT

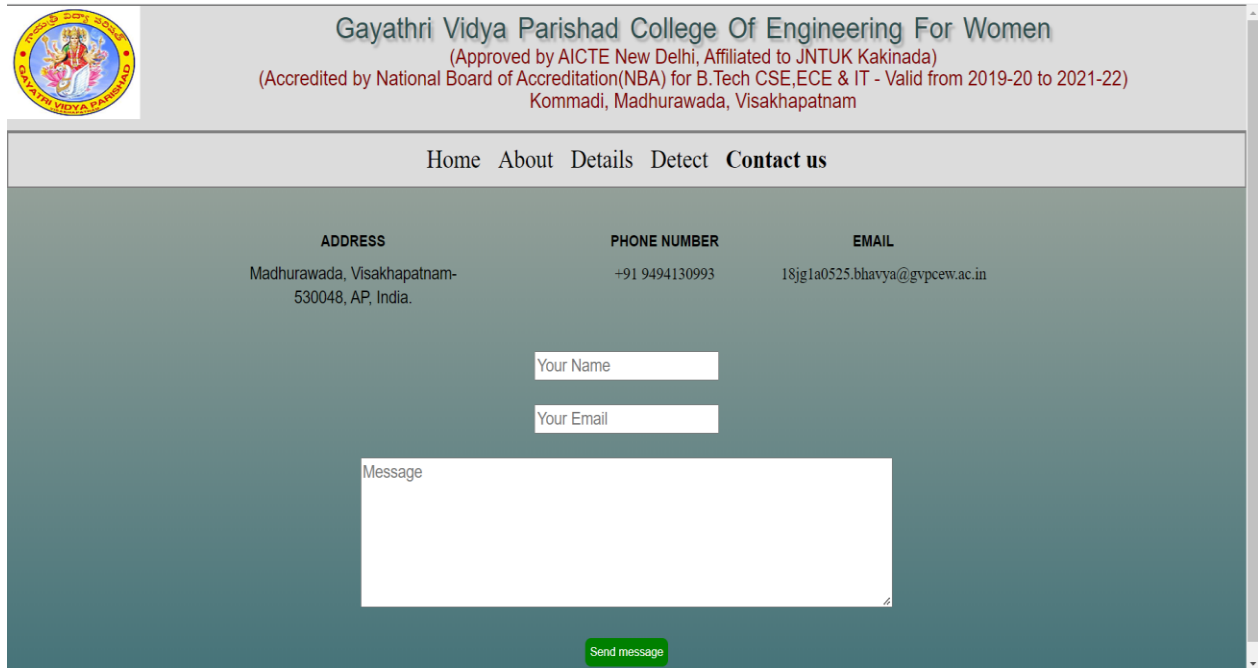
Statuses Count
Friends Count
Followers Count
Favorites Count
Listed Count

Predict Reset

Screen 14. Detect page (after predict)

Contact us page

This page is used to send message to the host user email.



Gayathri Vidya Parishad College Of Engineering For Women
(Approved by AICTE New Delhi, Affiliated to JNTUK Kakinada)
(Accredited by National Board of Accreditation(NBA) for B.Tech CSE,ECE & IT - Valid from 2019-20 to 2021-22)
Kommadi, Madhurawada, Visakhapatnam

Home About Details Detect **Contact us**

ADDRESS	PHONE NUMBER	EMAIL
Madhurawada, Visakhapatnam- 530048, AP, India.	+91 9494130993	18jg1a0525.bhavya@gvpcew.ac.in

Your Name

Your Email

Message

Screen 15. Contact us page

5.5 CONCLUSION

In this chapter the implementation and results are compared. Out of all GAN models, Bi-LSTM and GRU based GAN gave better accuracy with Variational Autoencoders. This part also includes the coding and front-end designing part.

6. TESTING AND VALIDATION

6.1 INTRODUCTION

Testing is a process of executing a program with the intent of finding an error. Testing presents an interesting anomaly for software engineering. The goal of the software testing is to convince system developers and customers that the software is good enough for operational use. Testing is a process intended to build confidence in the software. Testing is a set of activities that can be planned in advance and conducted systematically. Software testing is often referred to as verification & validation.

6.1.1 Scope

A primary purpose for testing is to detect software failures so that defects may be uncovered and corrected. This is a non-trivial Pursuit. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of the that code in various environments and conditions as well as examining the aspects of code: does it do what it is supported to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team.

6.1.2 Defects and failures

Not all software defects are caused by coding errors. One common source of expensive defects is caused by requirements gaps, e.g., unrecognized requirements that result in errors of omission by the program designer. A common source of requirements gaps is non-functional requirements such as testability, scalability, maintainability, usability, performance, and security. Software faults occur through the following process. A programmer makes an error (mistake), which results in a defect (fault, bug) in the software source code. If this defect is indeed, in certain situations the system will produce wrong results, causing a failure. Not all defects will necessarily result in failures. For example, defects in dead code will never result in failures. A defect can turn into a failure when the environment is changed. Examples of these changes in environment include the software being run on a new hardware platform,

alterations of source data or interacting with different software. A single defect may result in wide range failure symptoms.

6.2 TYPES OF TESTING

Black Box Testing

It's also called behavioral testing. It focuses on the functional requirements of the software. It is a complementary approach that is likely to uncover different classes of errors than white box errors. A black box testing enables a software engineer to derive a set of input conditions that will fully exercise all functional requirements for a program. Functional testing typically involves four steps:

- The identification of functions that the software is expected to perform.
- The creation of input data based on the function's specifications.
- The determination of output based on the function's specifications.

Unit Testing

In this testing we test each module individually and integrate with the overall system. It focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during the programming stage itself. In this each module is found to work satisfactorily as regard to the expected output from the module.

White Box Testing

It is also called glass-box testing. It is a test case design method that uses the control structure of the procedural design to derive test cases. Using white box testing methods, the software engineer can derive test cases that guarantee that all independent parts within a module have been exercised at least once and exercise all logical decisions on their true and false sides.

System Testing

Testing of the debugging programs is one of the most critical aspects of the computer programming triggers, without programs that work, the system would never produce the output

for which it was designed. Testing is best performed when use development is asked to assist in identifying all errors and bugs. The sample data are used for testing.

Verification & Validation

In software testing and software engineering, verification and validation is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control. It is the responsibility of software testers as part of the SDLC. Validation checks that the product design satisfies or fits the Intended use i.e., the software meets the user requirements. This is done through dynamic testing and other forms of review. Boehm succinctly expressed the difference between. Verification: Are we building the product, right? Validation: Are we building the right product?

6.3 DESIGN OF TEST CASES AND SCENARIOS

Table 8. Testcases and Scenarios

S.no	Test Case	Expected Result	Observed result	Positive/Negative	Pass/ Fail
1.	Giving bot features	It's a bot	It's a bot	Positive	Pass
2.	Giving human features as input	It's a human	It's a human	Positive	Pass
3.	Giving negative features as input	Enter valid data	Enter valid data	Negative	Pass
4.	Giving input out of range	Enter valid data	Enter valid data	Negative	Pass

6.4 CONCLUSION

This chapter briefly explains about the testing and validation of our project and the accuracy obtained in our project is 95.4 percentage.

7. CONCLUSION

Our project can help us to detect malicious social bots. We developed a web application which takes features as input from the user and predict whether it is a bot or a human. Various combinations of GAN and classification models have been implemented to best predict the bot behavior and the results are analyzed. An analysis was made to identify the limitations and the future scope of the work.

Future Scope: The prediction was done based on static account information. Our future enhancements are to make bot prediction on temporal data with tweet level classification.

8. REFERENCES

- [1] R. R. Rout, G. Lingam and D. V. L. N. Somayajulu, "Detection of Malicious Social Bots Using Learning Automata With URL Features in Twitter Network," in *IEEE Transactions on Computational Social Systems*, vol. 7, no. 4, pp. 1004-1018, Aug. 2020, doi: 10.1109/TCSS.2020.2992223.
- [2] B. Wu, L. Liu, Y. Yang, K. Zheng and X. Wang, "Using Improved Conditional Generative Adversarial Networks to Detect Social Bots on Twitter," in *IEEE Access*, vol. 8, pp. 36664-36680, 2020, doi: 10.1109/ACCESS.2020.2975630.
- [3] Wang, Xiujuan & Zheng, Qianqian & Zheng, Kangfeng & Sui, Yi & Cao, Siwei & Shi, Yutong. (2021). Detecting Social Media Bots with Variational AutoEncoder and k-Nearest Neighbor. *Applied Sciences*. 11. 5482. 10.3390/app11125482.
- [4] Jinxue Zhang, Rui Zhang, Yanchao Zhang, and Guanhua Yan, The rise of social botnets: Attacks and countermeasures, *IEEE Transactions on Dependable and Secure Computing* (2016).
- [5] Kudugunta S., Ferrara E. Deep neural networks for bot detection *Inform.Sci.*, 467 (2018), pp. 312-322
- [6] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi. A Criticism to Society (as seen by Twitter analytics). In *DASec*. IEEE, 2014.
- [7] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini. The rise of social bots. *Commun. ACM*, 59(7):96–104, 2016.
- [8] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. N. Choudhary. Towards online spam filtering in social networks. In *NDSS*. Internet Society, 2012.
- [9] Ma, Jing & Gao, Wei & Wong, Kam-Fai. (2019). Detect Rumors on Twitter by Promoting Information Campaigns with Generative Adversarial Learning. 10.1145/3308558.3313741.
- [10] Zhi Yang, Jilong Xue, Xiaoyong Yang, Xiao Wang, and Yafei Dai, Votetrust: Leveraging friend invitation graph to defend against social network sybils, *IEEE Transactions on Dependable and Secure Computing* 13 (2016), no. 4, 488–501.
- [11] Jing Wang and Ioannis Ch Paschalidis, Botnet detection based on anomaly and community detection, *IEEE Transactions on Control of Network Systems* (2016).
- [12] Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia, Detecting automation of twitter accounts: Are you a human, bot, or cyborg?, *IEEE Transactions on Dependable and Secure Computing* 9 (2012), no. 6, 811–824.
- [13] Sudipta Chowdhury, Mojtaba Khanzadeh, Ravi Akula, Fangyan Zhang, Song Zhang, Hugh Medal, Mohammad Marufuzzaman, and Linkan Bian, Botnet detection using graph-based feature clustering, *Journal of Big Data* 4 (2017), no. 1, 14.

- [14] Peining Shi, Zhiyong Zhang, and Kim-Kwang Raymond Choo, Detecting malicious social bots based on clickstream sequences, *IEEE Access* 7 (2019), 28855–28862.
- [15] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi. A Criticism to Society (as seen by Twitter analytics). In *DASec*. IEEE, 2014.
- [16] E. Alothali, N. Zaki, E. A. Mohamed and H. Alashwal, "Detecting Social Bots on Twitter: A Literature Review," *2018 International Conference on Innovations in Information Technology (IIT)*, 2018, pp. 175-180, doi: 10.1109/INNOVATIONS.2018.8605995.
- [17] Orabi, M., Mouheb, D., Al Aghbari, Z. and Kamel, I., 2020. Detection of bots in social media: a systematic review. *Information Processing & Management*, 57(4), p.102250.
- [18] Cresci, S., 2020. A decade of social bot detection. *Communications of the ACM*, 63(10), pp.72-83.
- [19] Wei, F. and Nguyen, U.T., 2019, December. Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)* (pp. 101-109). IEEE.
- [20] Heidari, M., Jones, J.H. and Uzuner, O., 2020, November. Deep contextualized word embedding for text-based online user profiling to detect social bots on twitter. In *2020 International Conference on Data Mining Workshops (ICDMW)* (pp. 480-487). IEEE.
- [21] Gilmary, R., Venkatesan, A. and Vaiyapuri, G., 2021. Discovering social bots on Twitter: a thematic review. *International Journal of Internet Technology and Secured Transactions*, 11(4), pp.369-395.
- [22] Lingam, G., Rout, R.R., Somayajulu, D.V. and Das, S.K., 2020, October. Social botnet community detection: a novel approach based on behavioral similarity in twitter network using deep learning. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security* (pp. 708-718).