Chapter 1

1 Introduction

In recent decades, social media has significantly influenced interpersonal relationships, transforming the internet into a virtual platform for online development, trade, and exchanging knowledge by individuals and their organizations. The various social communication systems have value chains aimed at certain user groups. For example, users may reunite with old acquaintances by browsing their Facebook profiles, and social media such as Twitter provides relevant updates and news of the following profiles. On the other hand, there are social network sites with different purposes, like LinkedIn, which is intended to serve as a support system for professional groups. Therefore, users are encouraged to fill their profiles with a significant number of personal data and to explore other users who share the same interests. According to usage rates, Facebook is the most popular social media platform, with 800 million monthly visits.

Estimates provided by Cloudmark suggest that between 20 and 40 percent of accounts on both Facebook and Twitter might be fake profiles. It is becoming more challenging to tell a real user from a fake due to the high levels of user engagement that occur every day and the millions of transactions that take place each day. The anticipated outcomes from the efforts to elicit user participation in identifying fake accounts have not been attained. In addition, when it comes to networks that have strict user privacy policies, there is a tiny amount of available public data. Thus, differentiating between fake and valid profile pages has become quite tricky systematically before trusting a possible association. In this piece of work, a way for distinguishing authentic accounts from false accounts in a logical manner is proposed. The approach is based on the limited publicly accessible account data on websites with strict privacy regulations, such as LinkedIn.

Social media’s growth can potentially raise people’s social evaluation and popularity. In particular, social network users may gain popularity by amassing many likes, follows, and remarks. On the other hand, establishing fake profiles is much too simple, and such accounts can be purchased online at little cost. For instance, purchasing followers and comments on social media platforms such as Facebook and Twitter may be done more easily on the internet. Analysis of activity changes is one of the most common techniques open social networking methods use to spot strange accounts. The activities that people engage in throughout time tend to shift and evolve. Therefore, the server can identify a potential scam account by monitoring for sudden changes in access patterns to the content and activity it requires. In case of unsuccessful identification, the deviant might fill the systems with fake information. Fig. 1 demonstrates a common schema of fake account detection on social networks.

1.1 Problem Statement

The proliferation of social media platforms has led to a surge in the creation of fake profiles, which are often used for malicious purposes such as spreading misinformation, scamming, or engaging in cyberbullying. Detecting these fake profiles manually is a daunting task due to the sheer volume of users and the sophistication of fraudulent activities.

The aim of this project is to develop a machine learning model capable of accurately identifying fake profiles on social media platforms. This model will analyze various features and patterns associated with both genuine and fake profiles to distinguish between them effectively.

Another type of fake account is a Cyborg, which a human uses to communicate with real users. It lowers the legitimacy of the user and employs the hijacked account to disseminate false information, create disinformation, and polarize public opinion. On the other hand, several communities suggest doing a variety of dataset analyses in conjunction with machine learning techniques to solve the issue. For example, one learning approach allows the model to calculate user categorization by “training” on the attributes data throughout some time. Several other articles examine the identification of false nodes using statistical approaches, distributed spatial using a density-based strategy, support vector machines (SVM), and hybrid models to identify social network fake profiles. Social network accounts include various personally identifiable information, such as the username, the user’s complete name, phone number, etc. The critical material may be compromised or manipulated by a skilled attacker if personal protection is used, which is a disadvantage of the method. Cyber fraudsters can use social and industrial design tactics and create dummy accounts to steal information and modify data.

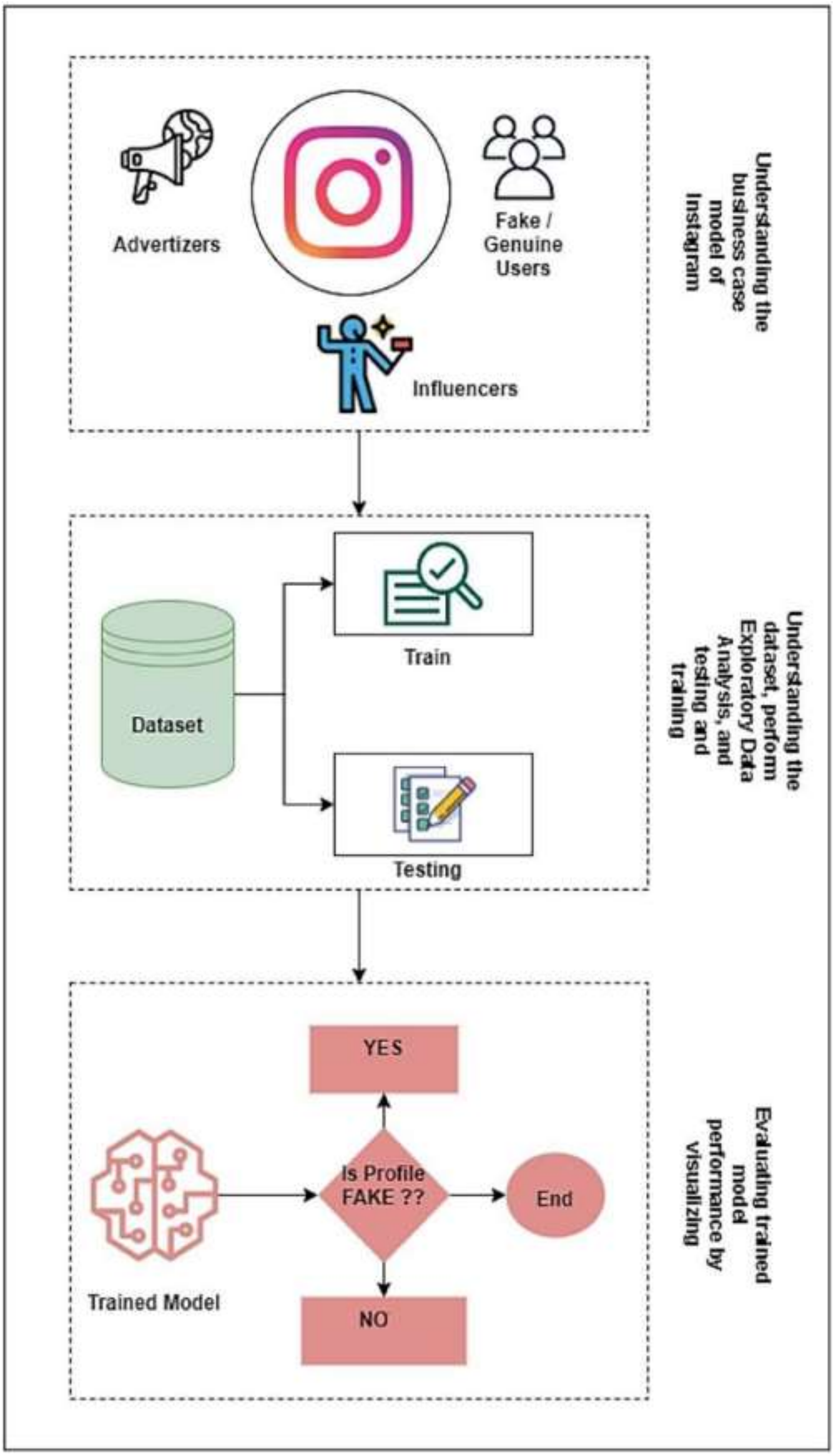

Advertizers
Fake / Genuine Users
Influencers
Understanding the business case model of Instagram
Dataset
Train
Testing
Understanding the dataset, perform Exploratory Data Analysis, and testing and training
Trained Model
Is Profile FAKE ??
YES
NO
End
Evaluating trained model performance by visualizing

Figure 1.1: Dataset and feature collection procedure

According to the reviewed publications, the issues surrounding the protection and credibility of social networks have become more significant. Moreover, it is necessary to have a trustworthy security model, particularly in light of threats' rising complexity and diversity.

While there are several methods for detecting fake profiles on social networks, they are not foolproof and come with their disadvantages. Here are some of the disadvantages of these methods:

Inaccuracy: One of the main challenges of detecting fake profiles is the methods' accuracy. Algorithms and models may not always accurately identify fake profiles and may also flag legitimate profiles as fake. This can lead to false positives and false negatives, which can be frustrating for users.

Limited data access: Detecting fake profiles often requires access to private data, such as Internet Protocol (IP) addresses, device fingerprints, and browsing histories. This raises privacy concerns, as users may not be comfortable sharing this information with social networks or third-party apps.

The difficulty of distinguishing between real and fake content: It can be challenging to distinguish between real and fake content on social networks, especially regarding user-generated content. Fake profiles may generate content that appears legitimately, making it difficult to detect them.

Limited resources: Detecting fake profiles can be resource-intensive, requiring advanced algorithms and machine-learning models to process vast amounts of data. Small social networks or those with limited resources may be unable to afford or implement these tools, leaving them vulnerable to fake profiles.

Ethical considerations: Detecting fake profiles raises ethical considerations, particularly regarding user privacy and social networks' responsibility to protect users. Social networks may need to balance the need for fraud prevention with the rights and privacy of their users.

Overall, while there are various methods for detecting fake profiles on social networks, they have their disadvantages. It is a continual challenge to stay ahead of the evolving tactics of fake profile creators.

To the best of our knowledge and based on our poll results, this is the only way to deal with fake social media profiles that involve training certain features using a deep learning approach. A neural network is a type of machine-learning model that is inspired by the structure and function of the human brain. It consists of layers of interconnected nodes, or neurons, that process information and makes predictions. A deep neural network (DNN) is a neural network with multiple hidden layers. The term “deep” refers to the number of layers in the network. In contrast, a general neural network may have just one or two hidden layers.

Adding multiple hidden layers in a deep neural network allows it to learn more complex features and relationships in the data, leading to better performance in tasks such as image and speech recognition, natural language processing, and many others. However, deep neural networks can be more challenging to train and suffer from overfitting or vanishing gradients.

To overcome these challenges, researchers have developed various techniques such as regularization, normalization, dropout, and gradient clipping to improve the training of deep neural networks. Overall, deep neural networks have revolutionized the field of machine learning and are widely used in various applications, from computer vision to natural language processing to autonomous vehicles.

The following is a summary of the most important contributions that our study has made, notably in addressing the fake profile classification task:

- A deep learning approach provided a unique way of identifying fraudulent accounts inside social networks.
- Sixteen profile-based features to train models for fake account detection problems were determined.
- We put the proposed deep model through its paces by conducting an exhaustive examination to acquire cutting-edge findings, particularly regarding detecting fraudulent profiles.

The remainder of the proposed paper is as follows: Next section reviews the literature by exploring the related works. Section 3 demonstrates the materials and methods applied in this research by demonstrating the proposed framework, method, and dataset. Section 4 illustrates the obtained results and compares them with the state-of-the-art results. Section 5 discusses the obtained results by referencing the advantages, limitations of current work and future perspectives. In the end, the paper was concluded by demonstrating the obtained results.

Chapter 2

2 Literature Review

Nowadays, social networking is expanding at an astonishingly rapid rate, which is significant for marketing initiatives and for celebrities attempting to advertise themselves by expanding their network of followers and admirers. Nevertheless, dummy accounts that seem to have been established on behalf of companies or individuals have the potential to tarnish their reputations and lead to a decline in the number of likes and follows they get. Fake updates and misunderstandings also plague them with other individuals. Fake accounts of any sort can lead to negative consequences that cancel out the benefits of social networks for companies in terms of promotion and marketing. They also prepare the ground for cyberbullying to occur. Users have varying worries about protecting their personal information in an online setting. Freelon et al. outlined the dangers that lurk in social networking sites when members are often oblivious to them.

Loss of privacy and identity, as well as viruses, fake accounts, and harassment, are some of the issues that might arise due to cyber fraudsters' activity on social media. In terms of popularity, social networks now have billions of people signed up for their services. Facebook has established itself as the most well-known social network, with over a billion active users. There are fundamentally four different types of threats on social media: traditional threats, contemporary threats, combination threats, and threats explicitly aimed at children. Several potential responses to these dangers may be grouped into one of three categories: operator responses, business responses, or scholarly responses. The processes included within each of these classes have the potential to assist in overcoming the challenges posed by social networks. Social engineering is the most common source of social network privacy and security risks. The primary methods of social engineering include socio-technical, technical, physical, and social. These methods are often carried out with the assistance of either software or actual users. Email, instant messaging, the telephone, messengers, Voice over Internet Protocol (VoIP), open social networks, the cloud, websites, and even physical routes may all be used as vectors for social engineering. In addition, there are modern forms of assault, such as social phishing, context-aware spam, false profiles, spear phishing, and phony identities stored in the cloud. Investigating the factors contributing to social network vulnerabilities revealed that fake accounts are the most significant factor. Detecting fake profiles is essential before the profiles in question are enrolled as social networking site members. Methods of detection similar to these will be covered later in this article. To acquire meaningful insights about the vast amount of accessible data, many companies have begun to explore the unstructured data available on social media.

Previous methods operate on the assumption that machine learning methods are complicated to implement because scammers produce patterns that computers cannot learn. However, recent research has shown that adversarial learning may be effectively implemented using many traditional machine learning methods, including ensembles of classifiers, Random Forests, SVM, adaptive boosting (AdaBoost), and Naive Bayes. Furthermore, the grouping and categorization of profiles according to their qualities are accomplished via several machine learning methods. This review on effective machine learning introduces several machine learning techniques. Furthermore, it examines the capacity of such algorithms to handle large amounts of data concerning the accuracy of their predictions. The computational needs of a model, the amount of memory required the least, and the ratio of the cost of computing to the accuracy of predictions are used to determine a model’s performance. In addition, clustering methods were applied to analyze social network graphs to identify fake profiles.

It is common practice to utilize fake social media accounts to gain users’ confidence and then distribute malware or a link. In addition, these scammers are involved in various other forms of criminal operations. So far, a considerable amount of research has been cantered on identifying false accounts on social media platforms to find solutions to these issues. This taxonomy has generally been used since it was described in. The methods for detecting fake accounts on social networking sites can be broken down into two categories: those that focus on analyzing individual accounts and those that concentrate on capturing coordinated activities that span many profiles. For instance, one research report identified and classified ghost accounts in popular social networking games. The research investigates Facebook games, known as the online game “Fighters club,” which offers rewards and a competitive edge to users who request their friends to participate. The researchers claim that the game stimulates its layers to develop phony profiles by initiating such an offer. As a result, users would improve their incentive value by putting such fake profiles into the game. The authors begin by gathering thirteen characteristics for each user and then go on to the classification process using support vector machines. According to the findings of this research, the approaches above do not provide any clear discriminators that might differentiate genuine users from phony ones. This most recent study employed graph-based characteristics, such as local clustering coefficient, betweenness centrality, and bi-directional connections ratio, as well as neighbor-based and timing-based features, to design several classifiers. The obtained accuracy resulted in 86% true and 5% false positives. However, subsequent efforts have made use of a number of the more common machine learning techniques. Several different machine learning techniques are used to classify profiles according to their attributes of those profiles. The review of research on effective machine learning presents some algorithms and examines their processing abilities concerning prediction accuracy. Several machine learning techniques are applied to detect false accounts that may exist inside social media platforms. Numerous approaches to machine learning have been used in various studies.

Using supervised learning methods, the technique that Singh and Sharma suggested to identify spammer patterns was discussed. Honey profiles were dispersed around Twitter and Facebook with the intention of coaxing spammers into exposing their identities by forming links with the honey profiles. All these accounts that were found to have established connections with the honey profiles were carefully evaluated and classified as either spammers or genuine users. On Facebook, a classifier was built using 1,000 tagged profiles; out of them, 173 of 1000 profiles were recognized as belonging to spam bots and were removed from the classifier. The authors used an unlabeled dataset, including 790,951 profiles, to assess the built classifier. Only 130 of these profiles were identified as belonging to spammers.

Cresci employed machine learning and honeypots to identify spammers on MySpace and Twitter social networks. They conducted their research in the same year. Rampersad et al. identified real-time disasters like earthquakes and typhoons by using data analysis of tweets sent by Twitter users. Behavior and content analysis were used by Umer et al. to identify spammers on Twitter. They amassed a dataset including 54 million individual profiles. To develop their predictor, they employed a training set that had 8,207 manually labeled individuals, of whom 355 turned out to be spammers and 7,852 were not considered as spammers. Due to a disproportionately large number of spammers compared to genuine users, 710 legitimate users were chosen randomly and included in the training set along with the spammers. In addition, a regular SVM classifier was applied for the phase that dealt with classification.

Consequently, the research demonstrated a true-positive percentage of 70.1% when detecting spammers, while their false-positive rate was 3.6%. Phantom profiles, also known as profiles constructed to get a strategic edge in social games, were the primary focus of Mourão et al. They successfully built a phantom profiles identification classifier by employing data from profiles and gaming activity on Facebook. The obtained results have shown 86.4% true-positive and 13.4% false-positive rates.

While research in Deep Neural Networks for detecting fake profiles in social networks has shown promising results, some limitations and challenges still need to be addressed. Some of these limitations include the following:

Limited Training Data: One of the main challenges in training a Deep Neural Network for detecting fake profiles is limited labeled training data availability. The lack of diverse and large-scale datasets can result in overfitting and poor model generalization to unseen data.

Adversarial Attacks: Adversarial attacks can trick the Deep Neural Network into misclassifying fake profiles as genuine ones. Attackers can use various techniques, such as data poisoning or model inversion, to manipulate the model’s outputs and bypass the detection mechanism.

Ethical Concerns: There are ethical concerns surrounding using Deep Neural Networks to detect fake profiles. For example, the algorithm’s accuracy may vary based on cultural differences and demographics, leading to prediction bias. Furthermore, using such algorithms can violate user privacy and raise concerns about using personal data.

Limited Interpretability: Deep Neural Networks are often considered black boxes, making it difficult to interpret their decision-making processes. This lack of interpretability can make understanding how the model identifies fake profiles challenging and limit its effectiveness.

Computational Complexity: Deep Neural Networks can be computationally expensive, requiring significant computational resources to train and operate. This complexity can limit the model’s scalability and practicality for real-time applications.

While Deep Neural Networks have shown promising results in detecting fake profiles in social networks, researchers must address these limitations to develop more accurate, ethical, and scalable detection mechanisms.

Chapter 3

3.1 Materials and Methods

As is the case with most social networking sites, the public Facebook social network developer application programming interface (API) only presents users' public information. Therefore, it is impossible to acquire access to the different activities of specific customers, and this occurs most of the time when a client has already changed the mode of their account to private. This annoyance is considered an obstruction to the system of records series since it causes a lot of trouble. Therefore, to address the issues and crawl the customers' data, a specialized crawler for data extraction and a function series device, both described in the following stages, were developed. Fig. 2 demonstrates the dataset creation and feature collection process.

3.2 Modules

1. **Service Provider**

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

2. **View and Authorize**

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

3. **Remote User**

   In this module, the rear en numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like REGISTER AND LOGIN,PREDICT PROFILE IDENTIFICATION STATUS,VIEW YOUR PROFILE

4. **Decision tree classifiers:** Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C1, C2, …, Ck is as follows**:**

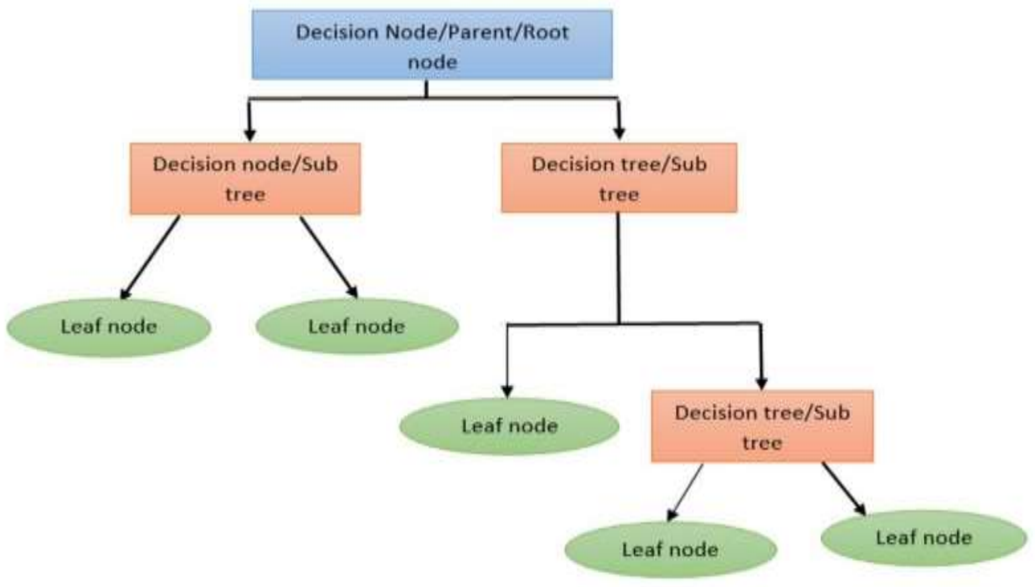
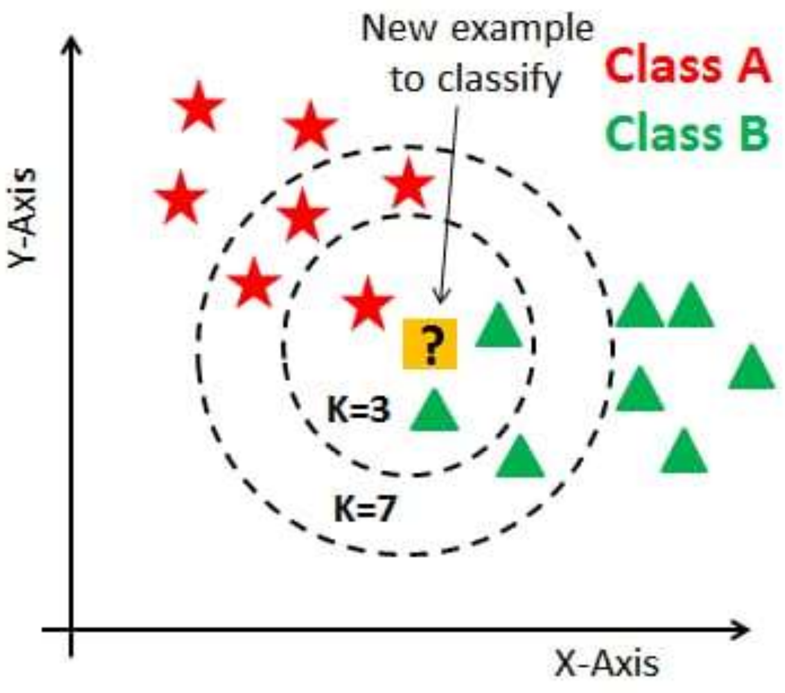


Figure 3.2 : Decision tree classifier

K-Nearest Neighbours(KNN):

- Classifies based on a similarity measure
- Non-parametric
- Lazy learning
- Does not "learn" until the test example is given
- When ever we have an we data to classify, we find its K-nearest neighbours from the training data

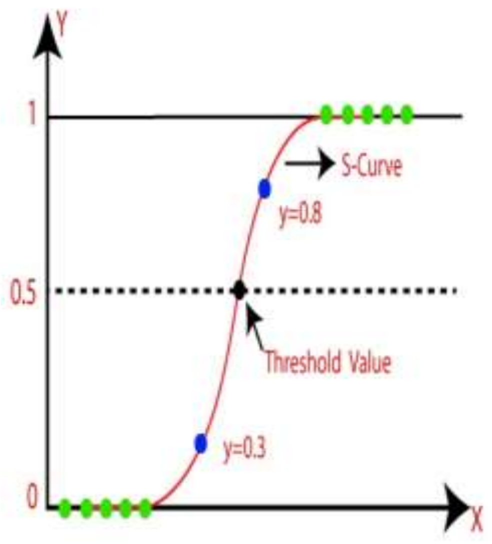
$$\text{Euclidean distance} = \sqrt{\sum_{i=1}^{n} \left(x_i - y_i\right)^2}$$



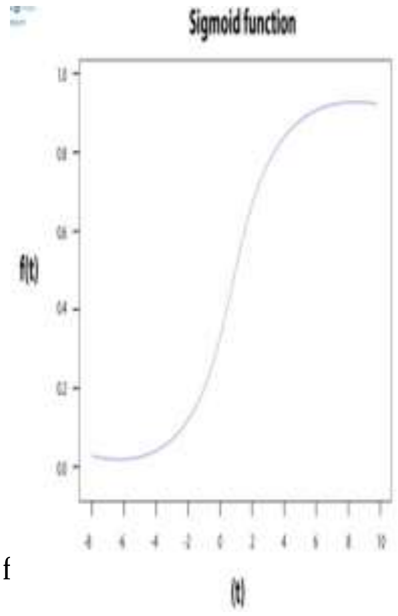
3.4 Logistic Regression:

Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name logistic regression is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name multinomial logistic regression is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar. Logistic regression competes with discriminate analysis as a method for analyzing categorical-response variables. Many statisticians feel that logistic regression is more versatile and better suited for modeling most situations than is discriminate analysis. This is because logistic regression does not assume that the independent variables are normally distributed, as discriminate analysis does.

This program computes binary logistic regression and multinomial logistic regression on both numeric and categorical independent variables. It reports on the regression equation as well as the goodness of fit, odds ratios, confidence limits, likelihood, and deviance.



f



3.5 Naïve Bayes:

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature .

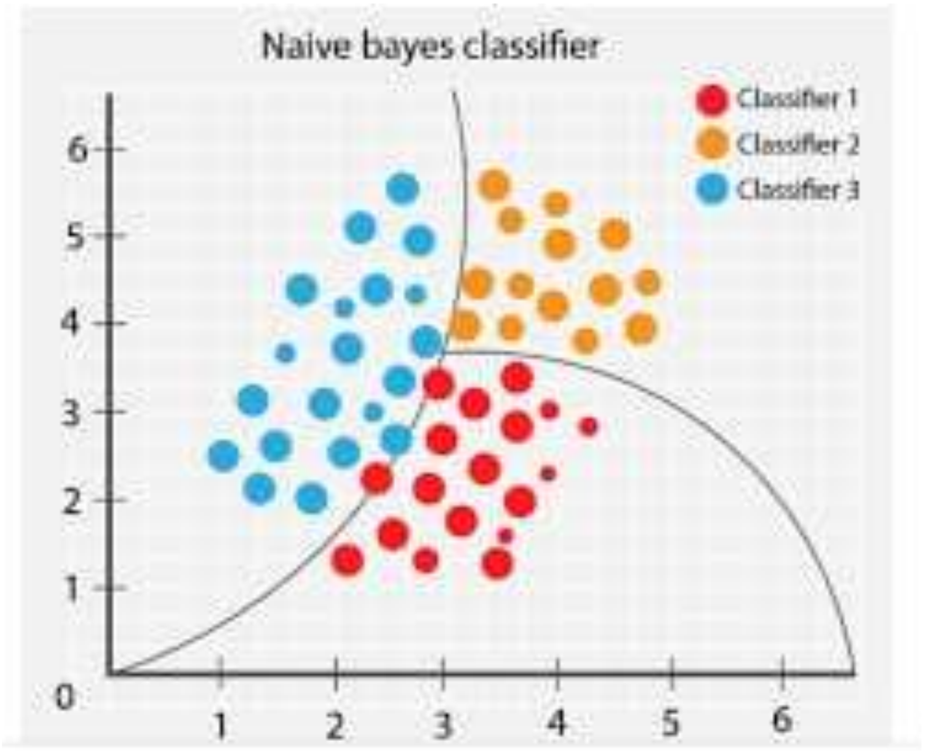
Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant

analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).

On the other hand, the final users do not obtain a model easy to interpret and deploy, they does not understand the interest of such a technique.

Thus, we introduce in an presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminate analysis and the linear SVM

$$P(Y|X) = \frac{P(X\ and\ Y)}{P(X)}$$



3.6 Data Collection

The 6868 regular customers, including celebrities, corporations, and daily legitimate

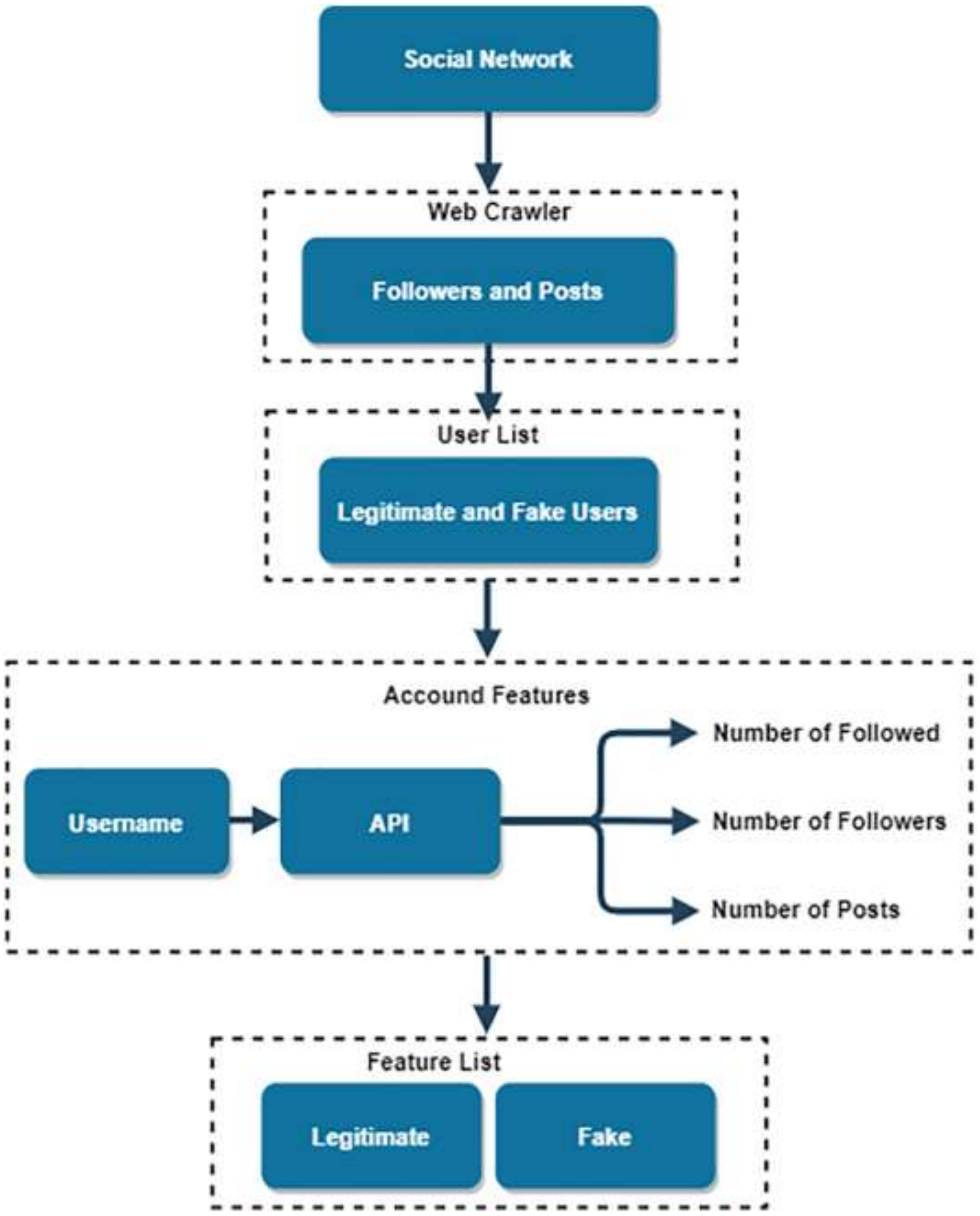
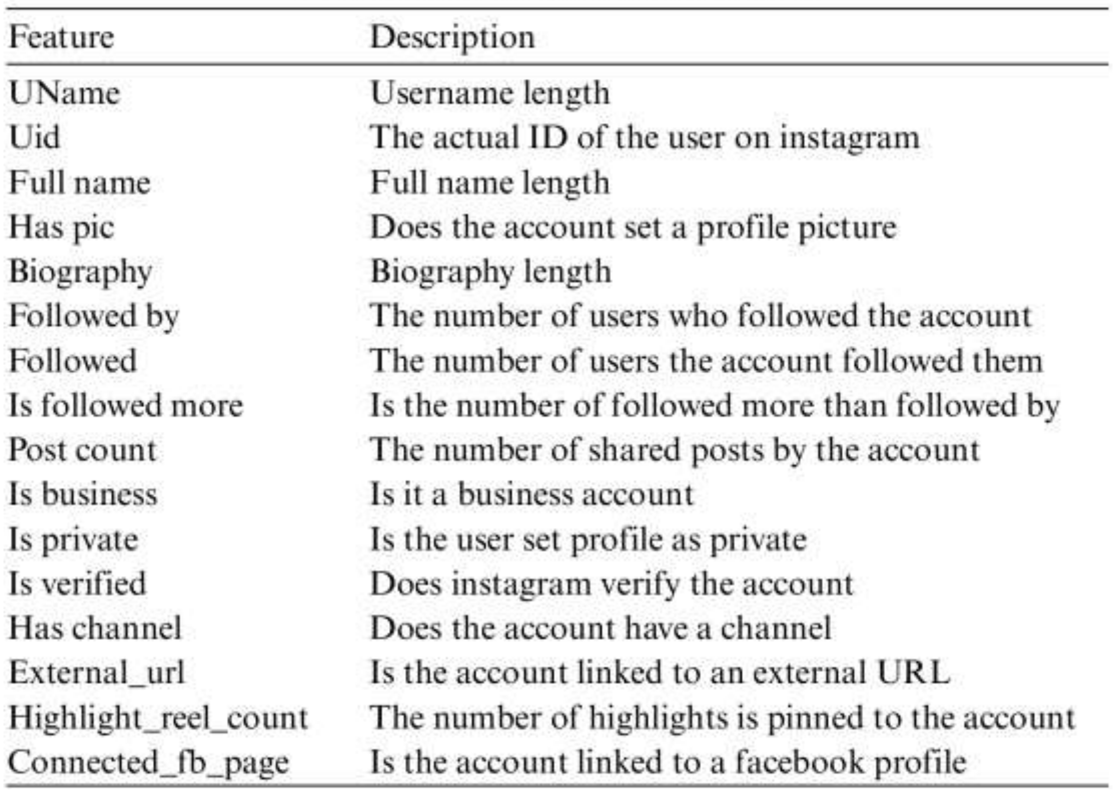


Figure 3.6 : Data Collection

consumers, and the 3132 anomaly customers who were personally checked and chosen have been collected inside the dataset. Additionally, the dataset contains 3132 customers that were deemed to be anomalous. We have developed more sophisticated types of record crawlers, one for reaching typical clients and another for finding unusual ones. The daily user crawler used the find feature on Facebook to locate ordinary users to be included in the list of everyday users in the dataset. The Explore section of Facebook displays recently published photos and videos that capture the attention of other users, indicating that the content shared on Facebook is, for the most part, genuine and authentic. In addition, to find and harvest fake customers on Facebook, an advanced crawler was initially utilized to acquire fake customer identifications (IDs) through the follower listings of customers who considered a wide variety of fake customers of their follower listings. Secondly, another system that allows the manual test of all false archived customers included in the dataset was developed. Thus, it will allow us to be sure about the customers' identities and improve the dataset's quality.

The Facebook application programming interface (API) crawled a few public records for each user. An overview of the dataset and a description of the crawled capabilities can be found indexed in Tables 1 and 2, respectively. The compiled capabilities are cataloged in Table 2.

Table 2: The list of collected features



Feature	Description
UName	Username length
Uid	The actual ID of the user on instagram
Full name	Full name length
Has pic	Does the account set a profile picture
Biography	Biography length
Followed by	The number of users who followed the account
Followed	The number of users the account followed them
Is followed more	Is the number of followed more than followed by
Post count	The number of shared posts by the account
Is business	Is it a business account
Is private	Is the user set profile as private
Is verified	Does instagram verify the account
Has channel	Does the account have a channel
External_url	Is the account linked to an external URL
Highlight_reel_count	The number of highlights is pinned to the account
Connected_fb_page	Is the account linked to a facebook profile

3.7 Random Forest:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho[1] using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.).The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman[13] in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (fi - yi)^2$$

Where N is the number of data points,
fi is the value returned by the model and
yi is the actual value for data point *i*.

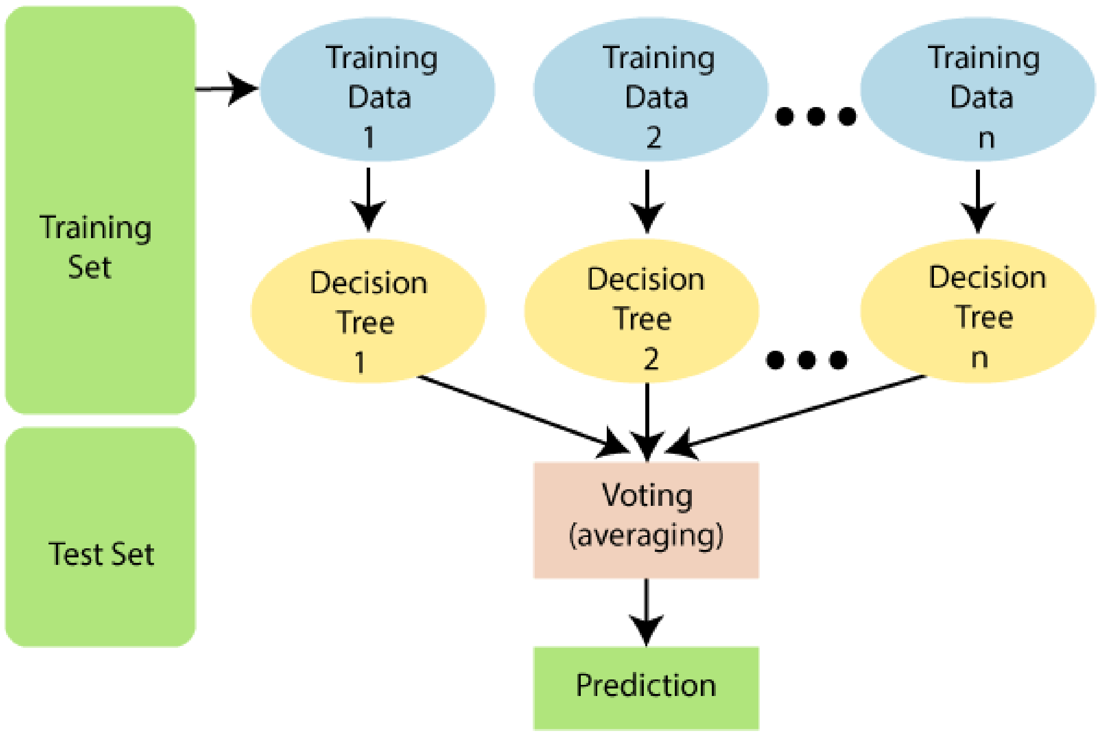


Figure 3.7 : Random Forest

3.8 Feature Preparation

After a significant amount of manual tagging, 1002 real accounts, and 201 fake accounts were acquired for the dataset. These accounts included debts from a variety of countries and professions throughout the world. The criteria that are taken into consideration during this data collection process include follower and following counts, media counts, media posting dates or frequencies, comments on posts made by social network users, the number of accompanying and following accounts, the lifespan of the profile picture, and the username of the profile.

A specific example: noticing different fake accounts from the dataset is possible. As can be seen, it has a large number of followers, 3949, and a small number of coffee followers, 15. Additionally, it does not have a profile picture or any published material.

The selected essential functions may be indexed in the dataset in the following manner:

- The whole range of activities by the account.
- Remember that the account is vital to the followers.
- After taking into consideration the history.
- The number of digits that may be found in the account username.

Regardless of whether or not the account is private, none of the functions are connected to the user’s media; hence, the set of restrictions does not violate the user’s account privacy. In this day and age of fake debts, some debts are manufactured by adding various numbers to the same name, which is why account usernames must have a diverse range of digits. It’s possible to make out the several ways the digits are distributed.

As can be seen, more than half of the fake debts have more than one number, while most real accounts have zero or one digit, accounting for around 89% of the total.

3.9 The Proposed Method

In this part of the research, a deep convolutional neural network (CNN) model was proposed for recognizing fake accounts on social networks. Fig. 3 demonstrates the proposed deep CNN model. The primary function of the suggested CNN consists of a backward pass and a forward pass that will operate concurrently throughout each iteration. The forward pass is computed by the proposed CNN using the following equation.

ynj=∑iknij·xni$yjn=\sum ikijn\cdot xin$(1)

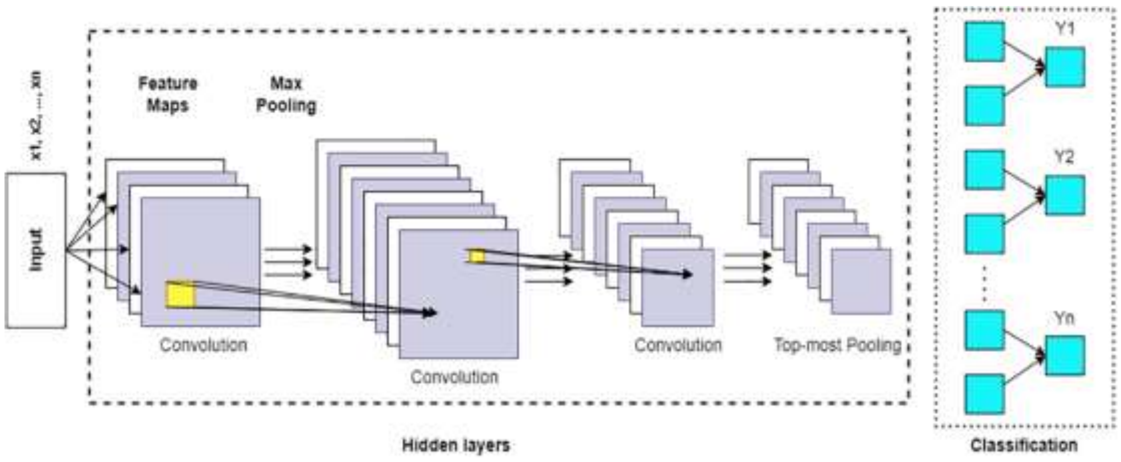


Figure 3: Architecture of the proposed deep CNN

In formula (1) shown above, the first network is responsible for computing the feature maps, which are then provided as the input layer to the proposed CNN.

Depending on the signification of xnixin is the i-th input feature map of the sample n and knij$kijn$ is the ij input filter of the sample n. Then, j−th becomes the output feature map of sample n networks. Using the formulae, the proposed convolutional neural network does the calculation for a backward pass as in the following equation:

$$\partial l\partial xni = \sum j(\partial l\partial ynj)\cdot(knij)\partial l\partial xin = \sum j(\partial l\partial yjn)\cdot(kijn)(2)$$

$$\partial l\partial knij = (\partial l\partial knj)\cdot xni\partial l\partial kijn = (\partial l\partial kjn)\cdot xin(3)$$

The CNN goes over each neuron to calculate the loss function, working its way up through the layers. The forward pass of the proposed neural network refers to the forward propagation of errors. In contrast, the forward pass of the proposed network refers to the forward propagation of errors using gradient descent to estimate the gradient of the loss function concerning the network parameters. The backward pass of CNN describes how mistakes are propagated in the other direction (weight and bias). To update the learning parameters, gradient descent is necessary to attain the minor loss function possible [30]. Besides, the Cross-Entropy (CE) loss function was applied during the study, a primary loss function for classification issues [31]. The calculation of the following accomplishes this:

$$H(y) = -\sum iy'ilog(yi)H(y) = -\sum iyi'\log(yi)(4)$$

Regarding the preceding operation, y_i' is the target label, and y_i is the output of the classification model. Finally, the CE function is applied to get an output with a probability distribution, which is the most popular loss function for use with SoftMax.

Classification

A deep learning model in binary classification has been applied to distinguish between dummy and valid profiles. Converting the array into binary tensors is a crucial step before supplying it with input, the matrix, into the network to facilitate adaptation. The suggested CNN begins by collecting input data in the first layer and crossing it onto hidden layers as its first operational phase. It employs social network characteristics as its input matrix at the beginning of the process.

The network performs a convolutional operation and a pooling operation, as well as a calculation using fully connected layers to create the output. The purpose of the convolutional layer is to extract numeric characteristics from the input data by iteratively sliding a smaller matrix filter over it. On the other hand, the convolutional method produces vast arrays. To simplify the array, we have optimized the pooling procedure using an innovative pooling function. Pooling allows us to restrict the number of generated feature maps while retaining the component that delivers the most information. This is accomplished by the downsampling approach, which helps us prevent overfitting in a CNN.

We trained the model in the hidden layer by supplying multiple layers to test how well the proposed deep convolutional neural network performs with the activation layer. In this study, to minimize the number of parameters and boost the network's capacity to generalize its results, the automated weight-sharing characteristics of the network were modified. By reducing the number of degrees of freedom that make up the complexity of the network, overfitting may be avoided by sharing the weight of neurons.

To downsample the input feature map, reducing its size while retaining the essential information, MaxPooling was applied. This operation effectively reduces the spatial resolution of the feature map but preserves the essential features by selecting the maximum value. Max pooling has several benefits, including reducing the computational complexity of the network, making it more robust to small translations in the input, and helping to prevent overfitting by enforcing a form of regularization.

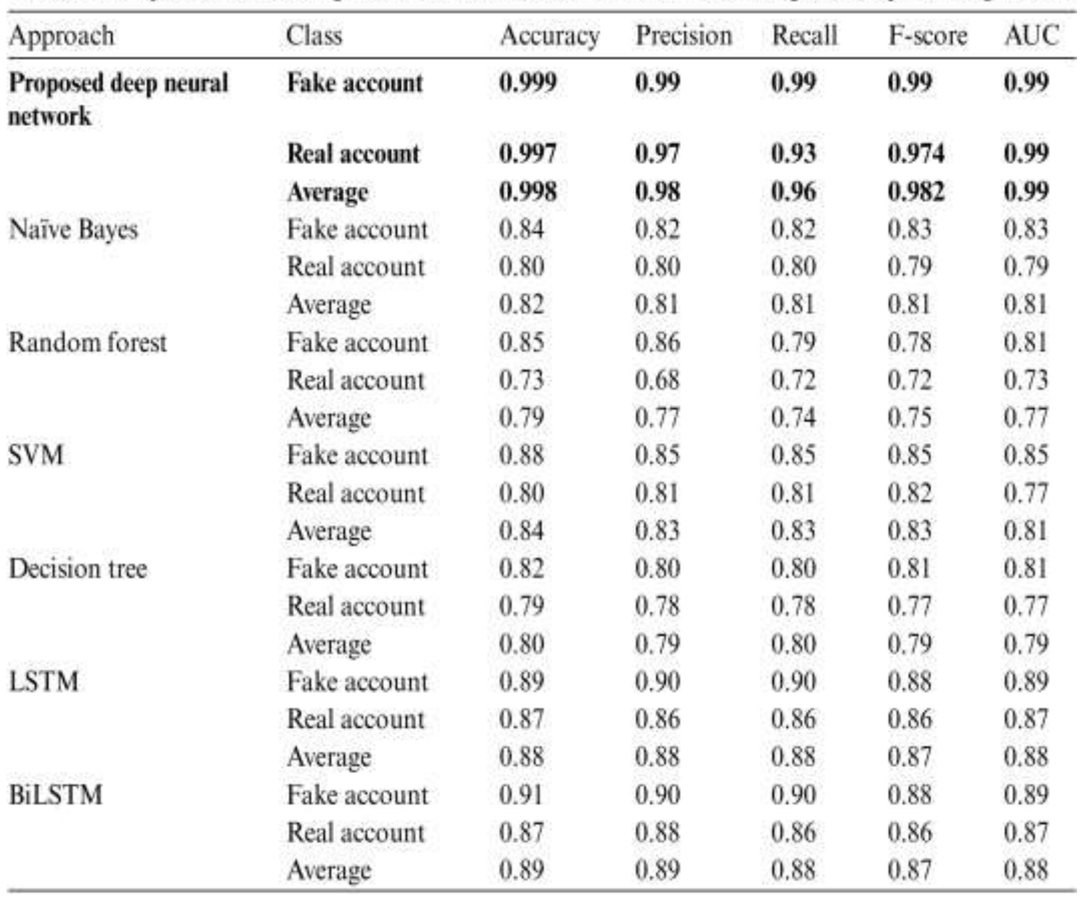
Chapter 4

Results

4.1 Comparison of Existing Methods

The experiment allowed us to achieve a balance between the amount of time it takes and how accurately it performs. We put gradient descent through its paces and experiment with various optimization algorithms’ hyperparameters. One of the most vital factors for improving the network’s training quality is choosing a hyperparameter value. As a result, we achieved the best possible performance of the network by optimizing the hyper-parameters. During training and testing, the epoch to equal 15 and the batch size to equal 50 were tuned. To train a model effectively and have enough data to evaluate its performance of the proposed model, the dataset was divided between training and testing set as 80% to 20%. The performance of the DeepProfile CNN concerning computing validation accuracy and loss is detailed in Table 3.

Table 3: Comparison of existing methods with traditional machine learning and deep learning models

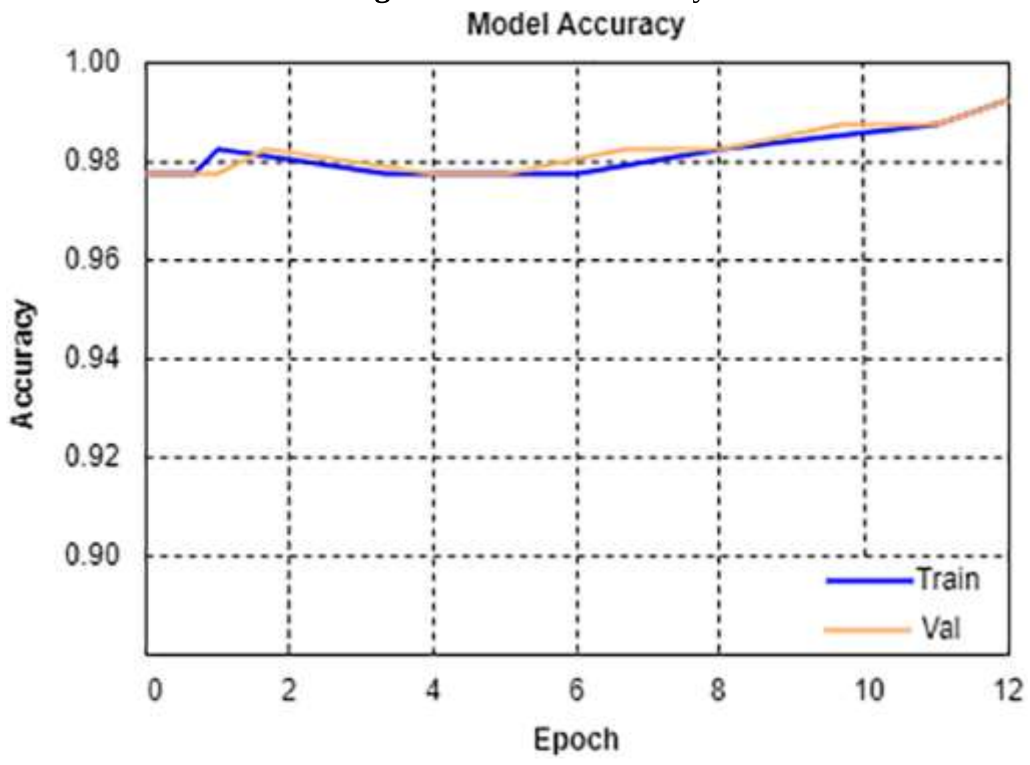


Approach	Class	Accuracy	Precision	Recall	F-score	AUC
Proposed deep neural network	**Fake account**	**0.999**	**0.99**	**0.99**	**0.99**	**0.99**
	Real account	**0.997**	**0.97**	**0.93**	**0.974**	**0.99**
	Average	**0.998**	**0.98**	**0.96**	**0.982**	**0.99**
Naïve Bayes	Fake account	0.84	0.82	0.82	0.83	0.83
	Real account	0.80	0.80	0.80	0.79	0.79
	Average	0.82	0.81	0.81	0.81	0.81
Random forest	Fake account	0.85	0.86	0.79	0.78	0.81
	Real account	0.73	0.68	0.72	0.72	0.73
	Average	0.79	0.77	0.74	0.75	0.77
SVM	Fake account	0.88	0.85	0.85	0.85	0.85
	Real account	0.80	0.81	0.81	0.82	0.77
	Average	0.84	0.83	0.83	0.83	0.81
Decision tree	Fake account	0.82	0.80	0.80	0.81	0.81
	Real account	0.79	0.78	0.78	0.77	0.77
	Average	0.80	0.79	0.80	0.79	0.79
LSTM	Fake account	0.89	0.90	0.90	0.88	0.89
	Real account	0.87	0.86	0.86	0.86	0.87
	Average	0.88	0.88	0.88	0.87	0.88
BiLSTM	Fake account	0.91	0.90	0.90	0.88	0.89
	Real account	0.87	0.88	0.86	0.86	0.87
	Average	0.89	0.89	0.88	0.87	0.88

These findings were gleaned by applying the machine learning models to the selected dataset. Graphs, comparison charts, and ROC curves are the presentation formats for the findings produced during the study. In addition, the accuracy or loss patterns under each model were considered. Further discussion is on the accuracy or loss computed during the training and validation. X Since Google Colab allows for the use of free GPUs, the platform for the training of our models was chosen. The NVIDIA Tesla K80 GPU from Google Colab has 12 GB of memory and can operate nonstop for up to 12 h. As a main language to write down all of the models, Python3 was used. The following findings were obtained after all the models were trained and validated. The model accuracy, model loss *vs.* the epoch graphs, and the receiver operating characteristic (ROC) curve for the random forest, eXtreme Gradient Boosting (XG boost), and other approaches are presented for the proposed neural network. Model accuracy comparisons are also performed.

Figs. 4 and 5, which represent the trained neural network, provide the accuracy and loss graphs for the model, respectively. The accuracy and loss graphs above let the algorithm run for 15 epochs. Beginning at a value of 0.97 and continuing down the route, the accuracy gradually improves until it achieves its highest possible value, which is 0.98. Similarly, the loss graph for testing data starts at 1, whereas the loss graph for validation data starts at 4. Both graphs ultimately converge on a minimum point that is smaller than 0.5. The binary cross-entropy function is used to compute the loss amount. At first, each feature receives a completely random weight, and in the end, the machine gives a weight that is unique to each feature.

Figure 4.1: Model accuracy



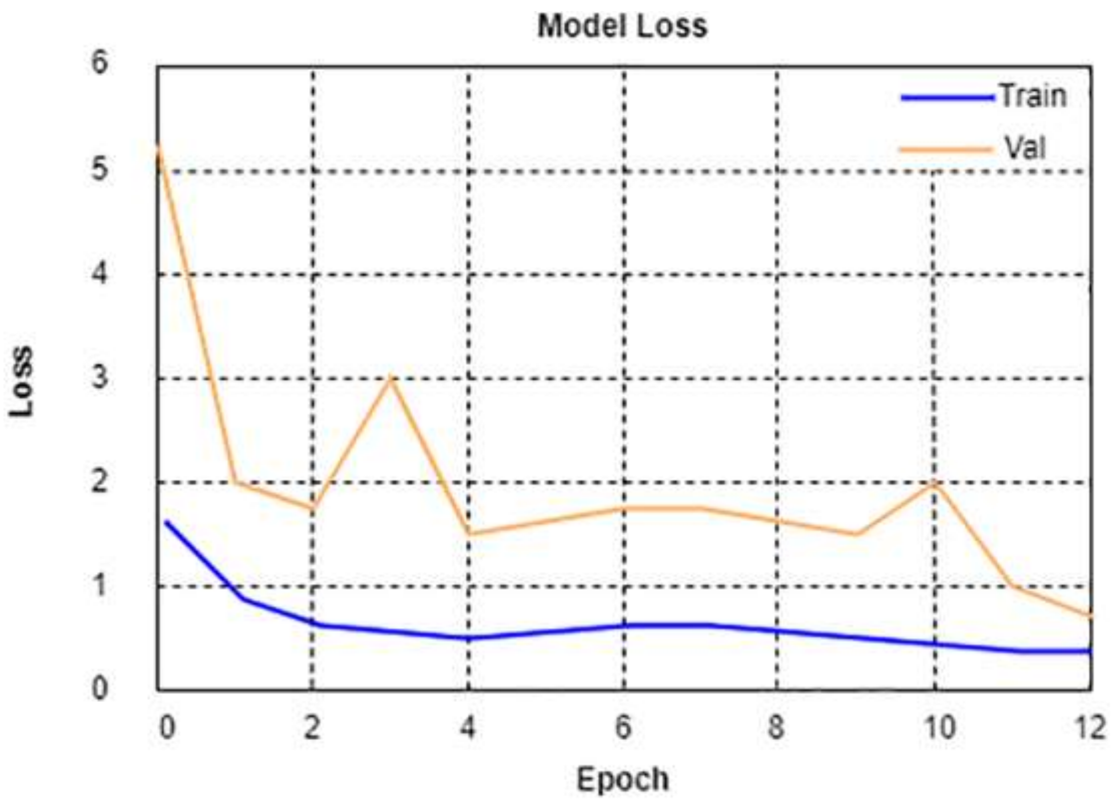


Figure 4.1: Model loss

Fig. 5 demonstrates the training and validation loss of the proposed deep convolution neural networks for 12 training and testing epochs. As the figure illustrates, train loss sharply decreased from the first training epoch, and validation loss remains with less error rate. The results show that the proposed model has achieved high accuracy with less error rate.

Comparison to Other Methods: In the chart that compares the various models' accuracy, we can see the performance of different machine learning algorithms. The histogram for comparing the levels of accuracy, as well as the Area under the ROC Curve (AUC-ROC) curves, have been provided in Fig. 6. As it can be seen from the figure, the proposed deep neural network demonstrates higher performance in terms of all the evaluation parameters including accuracy, precision, recall, F-score, and AUC-ROC.

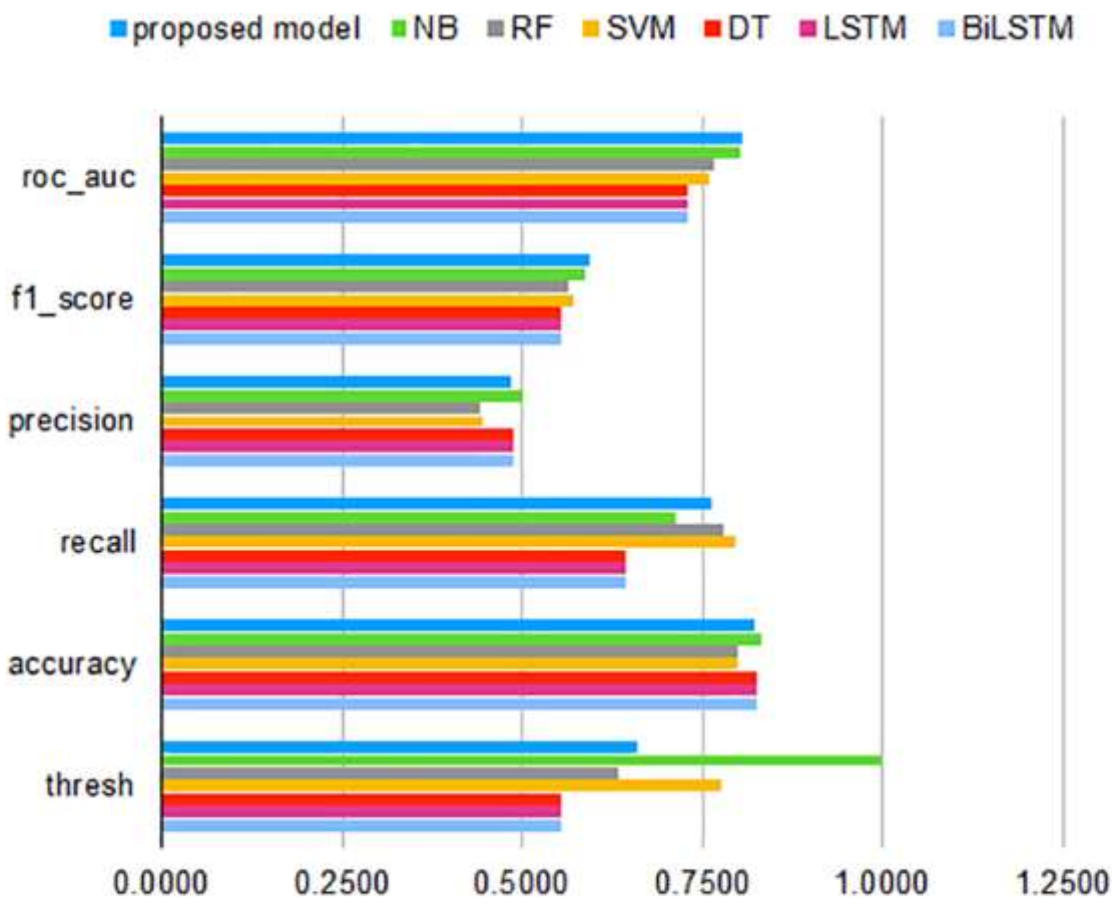


Figure 4.1 : Comparison of the proposed network and machine learning methods

Fig. 7 demonstrates the AUC-ROC curve of the proposed deep convolutional neural network. The horizontal axis represents false positive rates, while the vertical axis represents true positive rates. As the figure illustrates, the proposed network shows a high rate of AUC-ROC that is very useful in practical use.

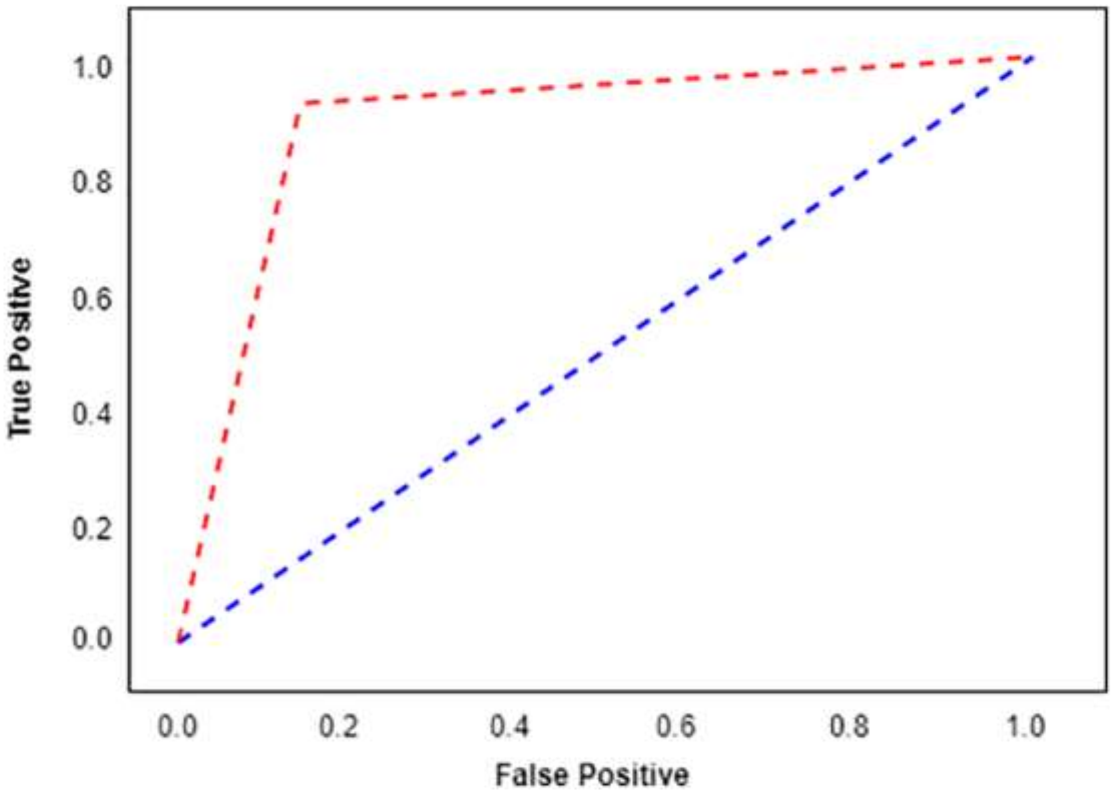
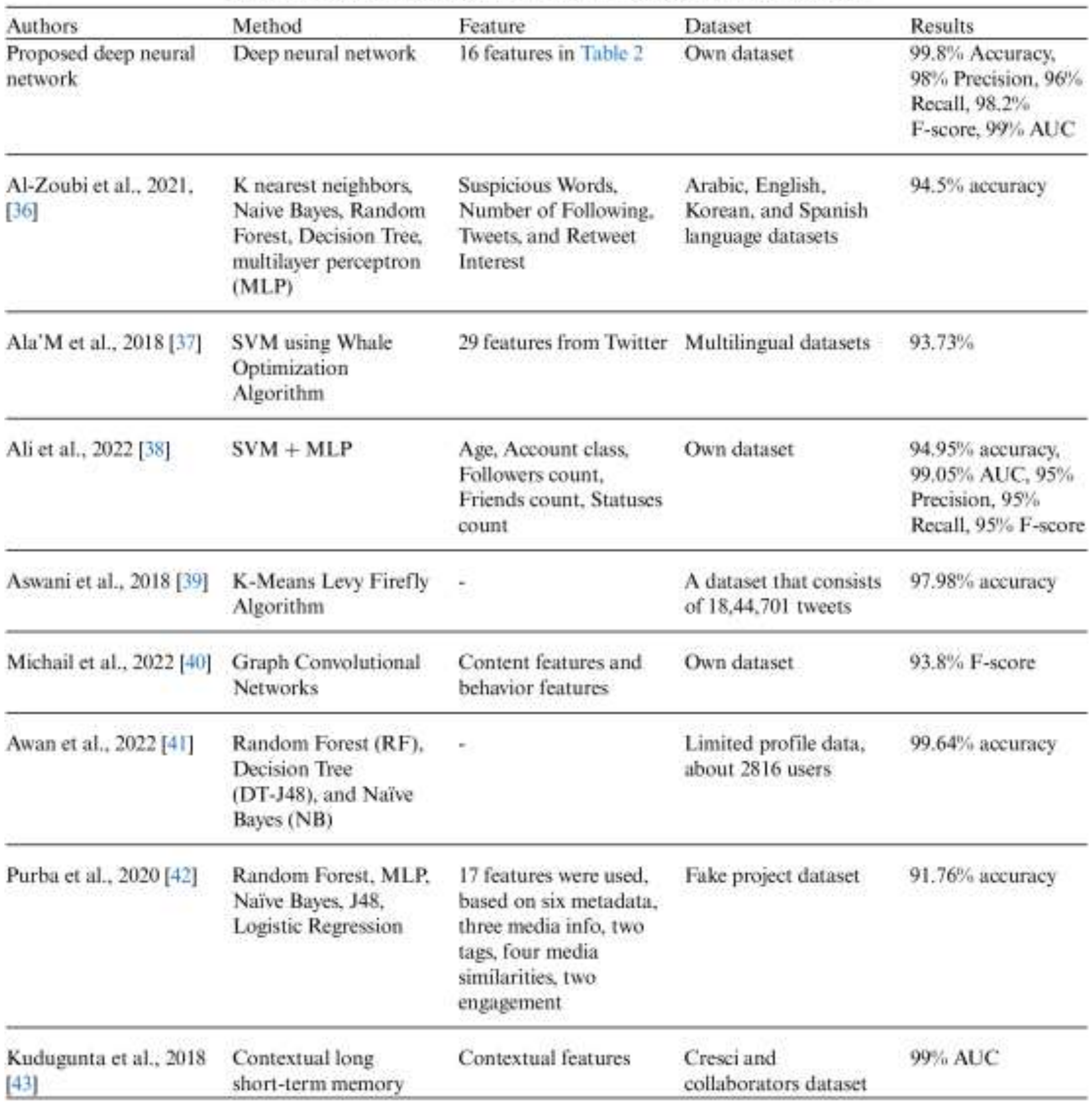


Figure 4.1: AUC-ROC curve in detecting fake profiles

Table 3 compares the proposed deep neural network with the other machine learning and deep learning models on Kaggle's Social Network Fake Account Dataset. The results show that the proposed deep neural networks cope better than traditional machine learning and deep learning models.

To summarize the suggested learning method in detecting fake profiles, the provided approach may be a future choice for analyzing accounts in a huge dataset. The results show that the proposed method can serve as a protection strategy in social networks. Furthermore, the expenditures of research and development for harmful account analysis may be reduced under this approach, which is another advantage of using this method. We have concluded that the proposed model can achieve an efficient result and one that the neural network can do with more precision. In a real-world scenario, the convolutional neural network is a potentially viable approach for addressing difficulties related to harmful behavior on social networks. Table 4 compares state-of-the-art research results with the proposed convolutional neural network regarding different evaluation parameters.

Table 4: Comparison of methods for fake profiles detection



Authors	Method	Feature	Dataset	Results
Proposed deep neural network	Deep neural network	16 features in Table 2	Own dataset	99.8% Accuracy, 98% Precision, 96% Recall, 98.2% F-score, 99% AUC
Al-Zoubi et al., 2021, [36]	K nearest neighbors, Naive Bayes, Random Forest, Decision Tree, multilayer perceptron (MLP)	Suspicious Words, Number of Following, Tweets, and Retweet Interest	Arabic, English, Korean, and Spanish language datasets	94.5% accuracy
Ala'M et al., 2018 [37]	SVM using Whale Optimization Algorithm	29 features from Twitter	Multilingual datasets	93.73%
Ali et al., 2022 [38]	SVM + MLP	Age, Account class, Followers count, Friends count, Statuses count	Own dataset	94.95% accuracy, 99.05% AUC, 95% Precision, 95% Recall, 95% F-score
Aswani et al., 2018 [39]	K-Means Levy Firefly Algorithm	-	A dataset that consists of 18,44,701 tweets	97.98% accuracy
Michail et al., 2022 [40]	Graph Convolutional Networks	Content features and behavior features	Own dataset	93.8% F-score
Awan et al., 2022 [41]	Random Forest (RF), Decision Tree (DT-J48), and Naïve Bayes (NB)	-	Limited profile data, about 2816 users	99.64% accuracy
Purba et al., 2020 [42]	Random Forest, MLP, Naïve Bayes, J48, Logistic Regression	17 features were used, based on six metadata, three media info, two tags, four media similarities, two engagement	Fake project dataset	91.76% accuracy
Kudugunta et al., 2018 [43]	Contextual long short-term memory	Contextual features	Cresci and collaborators dataset	99% AUC

4.2 Unit Testing:

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

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to betested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the in crèmental integrati on testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software systemor – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Preparation of TestData

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmer so analysts often ask users to key in a set of data from their normal

activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the lived at a entered are in fact

typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

4.3 USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

MAINTAINENCE

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirement sin future. The coding and designing is simple and easy to understand which will make maintenance easier.

TESTING STRATEGY:

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

SYSTEM TESTING:

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

UNIT TESTING:

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail pathsare tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company.For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key consider at ions involved in the feasibility analysis are

ECONOMICAL FEASIBILITY

TECHNICAL FEASIBILITY

SOCIAL FEASIBILITY

4.4 SYSTEM DESIGN AND DEVELOPMENT

INPUT DESIGN

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts,

This system has input screens in almost all the modules. Error messages are developed to alert the user whenever he commits some mistakes and guides him in the right way so that invalid entries are not made. Let us see deeply about this under module design.

Input design is the process of converting the user created input into a computer- based format. The goal of the input design is to make the data entry logical and free from errors. The error is in the input are controlled by the input design. The application has been developed in user-friendly manner. The forms have been designed in such a way during the processing the cursor is placed in the position where must be entered. The user is also provided with in an option to select an appropriate input from various alternatives related to the field in certain cases.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

OUTPUT DESIGN

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, an ew project may be assigned to the client.

User authentication procedures are maintained at the initial stages itself. A new user may be created by the administrator himself or a user can himself register as a new user but the task of assigning projects and validating a new user rests with the administrator only.

The application starts running when it is executed for the first time. The server has to be started and then the internet explorer in used as the browser. The project will run on the local area network so the server machine will serve as the administrator while the other connected systems can act as the clients. The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

4.5 RESULTS & DISCUSSIONS

The application of machine learning (ML) and natural language processing (NLP) techniques represents a significant stride forward in the ongoing battle against fake profiles within social networks. By leveraging sophisticated algorithms to scrutinize vast amounts of user-generated content, these technologies empower platforms to identify suspicious accounts with greater accuracy and efficiency than traditional methods. ML models can discern subtle linguistic cues, such as grammar usage, sentiment patterns, and vocabulary choices, which often distinguish authentic user interactions from those generated by bots or malicious actors. Additionally, NLP algorithms enable the extraction of meaningful insights from unstructured textual data, allowing for deeper analysis of user behaviors, network structures, and content propagation dynamics.

Table 4.5: Results and Discussion

MODEL	Accuracy
KNN	95.489
Naive Bayes	95.791
SVM	96.376

One of the key advantages of employing ML and NLP in fake profile identification is their adaptability to evolving tactics employed by perpetrators. As perpetrators continually refine their strategies to evade detection, ML models can be trained on updated datasets to recognize new patterns and anomalies indicative of fraudulent activity. Moreover, the integration of ML-based anomaly detection algorithms can enhance the scalability and real-time responsiveness of fake profile detection systems, enabling platforms to promptly flag and mitigate emerging threats.

Nevertheless, the deployment of ML and NLP technologies in fake profile detection also raises important ethical considerations and challenges. Privacy concerns regarding the collection and analysis of user data must be carefully balanced with the imperative to safeguard users from online deception. Transparent communication and robust data protection measures are essential to build and maintain user trust in the integrity of detection mechanisms. Additionally, biases inherent in training data and algorithmic decision-making processes must be actively addressed to prevent the amplification of existing social inequalities or the unjust targeting of specific user groups.

Looking ahead, the continued advancement of ML and NLP techniques holds immense potential for further enhancing the effectiveness and sophistication of fake profile detection systems. Collaborative efforts between researchers, platform operators, and regulatory bodies are essential to foster innovation while upholding ethical standards and safeguarding user rights. Furthermore, interdisciplinary approaches that integrate insights from psychology, sociology, and criminology can enrich our understanding of online deception dynamics and inform the development of more robust detection methodologies.

In summary, the fusion of ML and NLP technologies represents a transformative force in the ongoing fight against fake profiles in social networks. By harnessing the power of data-driven insights and computational intelligence, platforms can better defend against malicious actors and cultivate a safer, more trustworthy online environment for users worldwide. However, achieving this vision requires a concerted commitment to ethical practices, ongoing research, and collaboration across diverse stakeholders to ensure the responsible and equitable deployment of these cutting-edge technologies.

To evaluate a model some other metrics also need to be considered. Recall, Precision and F1-score were considered to evaluate.

Recall: Recall measures the ability of a model to correctly identify all relevant instances, which are the positive instances. It's also known as sensitivity or the true positive rate.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Precision: Precision measures the ability of a model to correctly identify only the relevant instances among those it classifies as positive. It quantifies how many of the positively classified instances were actually positive.

$$Precision = \frac{TruePositive}{True\ Positive + False\ Positive}$$

F1-score: The F1 Score is the harmonic mean of precision and recall. It provides a balance between these two metrics, allowing you to assess the model's overall performance.

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$

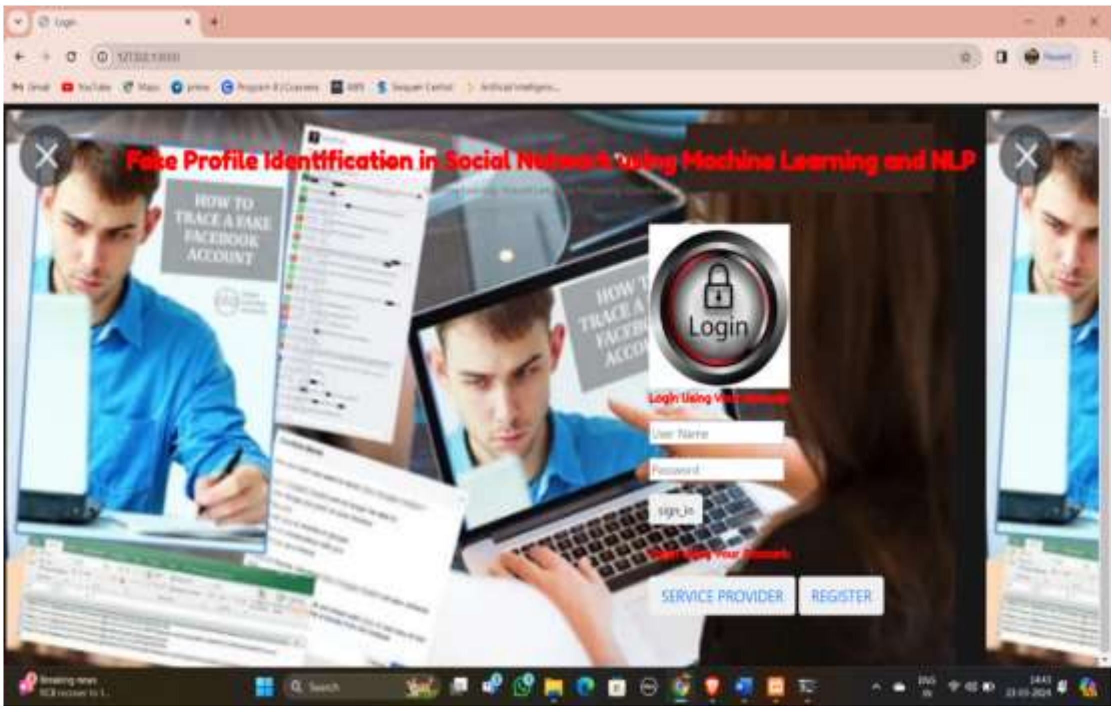
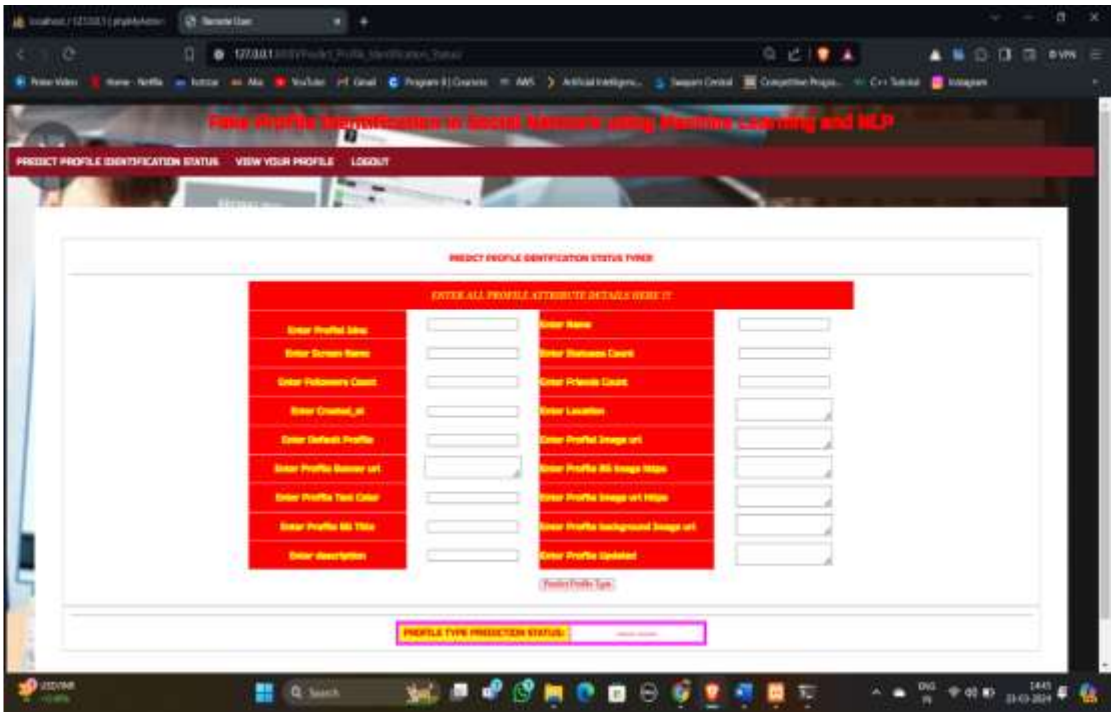

Fake Profile Identification in Social Network using Machine Learning and NLP
HOW TO TRACE A FAKE FACEBOOK ACCOUNT
Login
User Name
Password
sign_in
SERVICE PROVIDER
REGISTER

Fig 4.5: Login Page



Fake Profile Identification in Social Network using Machine Learning and NLP

PREDICT PROFILE IDENTIFICATION STATUS　VIEW YOUR PROFILE　LOGOUT

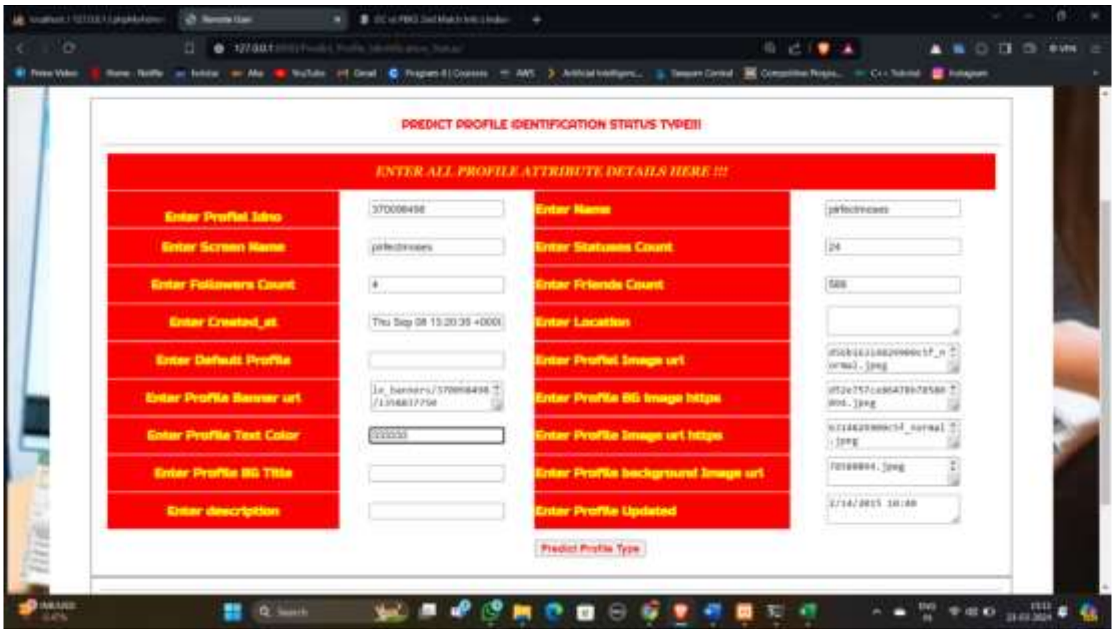
PREDICT PROFILE IDENTIFICATION STATUS TYPE!!!

ENTER ALL PROFILE ATTRIBUTE DETAILS HERE !!!

Enter Profiel Idno		Enter Name	
Enter Screen Name		Enter Statuses Count	
Enter Followers Count		Enter Friends Count	
Enter Created_at		Enter Location	
Enter Default Profile		Enter Profiel Image url	
Enter Profile Banner url		Enter Profile BG Image https	
Enter Profile Text Color		Enter Profile Image url https	
Enter Profile BG Title		Enter Profile background Image url	
Enter description		Enter Profile Updated	

Predict Profile Type

PROFILE TYPE PREDICTION STATUS:



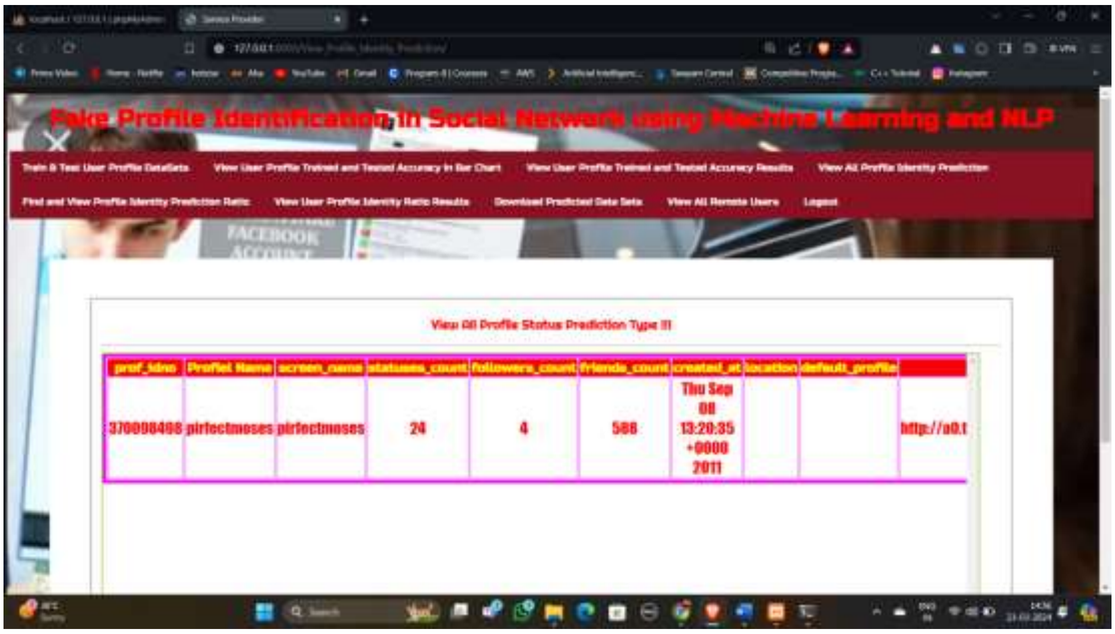
PREDICT PROFILE IDENTIFICATION STATUS TYPE!!!

ENTER ALL PROFILE ATTRIBUTE DETAILS HERE !!!

Enter Profiel Idno	370008498	Enter Name	perfectroses
Enter Screen Name	perfectroses	Enter Statuses Count	24
Enter Followers Count	4	Enter Friends Count	588
Enter Created_at	Thu Sep 08 15:20:35 +0000	Enter Location	
Enter Default Profile		Enter Profiel Image url	[illegible]_normal.jpeg
Enter Profile Banner url	le_banners/370008498/1356837750	Enter Profile BG Image https	[illegible].jpeg
Enter Profile Text Color	333333	Enter Profile Image url https	[illegible]_normal.jpeg
Enter Profile BG Title		Enter Profile background Image url	[illegible].jpeg
Enter description		Enter Profile Updated	2/14/2015 10:40

Predict Profile Type

Fig 4.5: Data Enter page


Fake Profile Identification in Social Network using Machine Learning and NLP
Train & Test User Profile DataSets
View User Profile Trained and Tested Accuracy in Bar Chart
View User Profile Trained and Tested Accuracy Results
View All Profile Identity Prediction
Find and View Profile Identity Prediction Ratio
View User Profile Identity Ratio Results
Download Predicted Data Sets
View All Remote Users
Logout

View All Profile Status Prediction Type !!!

prof_idno	Profiel Name	screen_name	statuses_count	followers_count	friends_count	created_at	location	default_profile	
370098468	pirfectmoses	pirfectmoses	24	4	588	Thu Sep 08 13:20:35 +0000 2011			http://a0.t

Fig 4.5: Profile Status Page

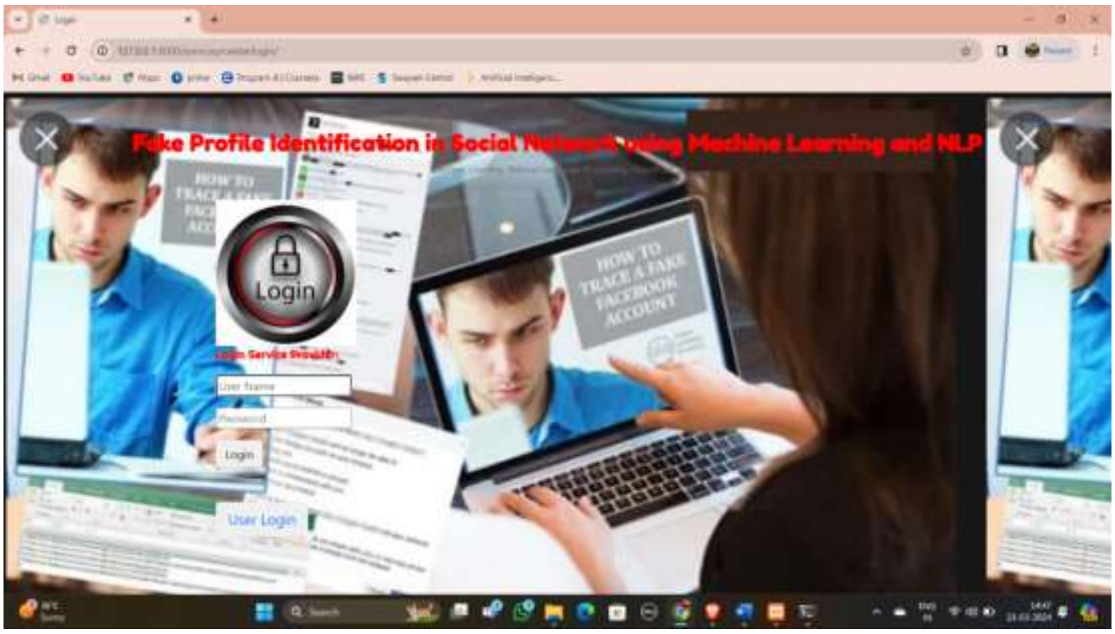

Fake Profile Identification in Social Network using Machine Learning and NLP
Login
Login Service Provider
User Name
Password
Login
User Login

Fig 4.5: Service Provider Login Page

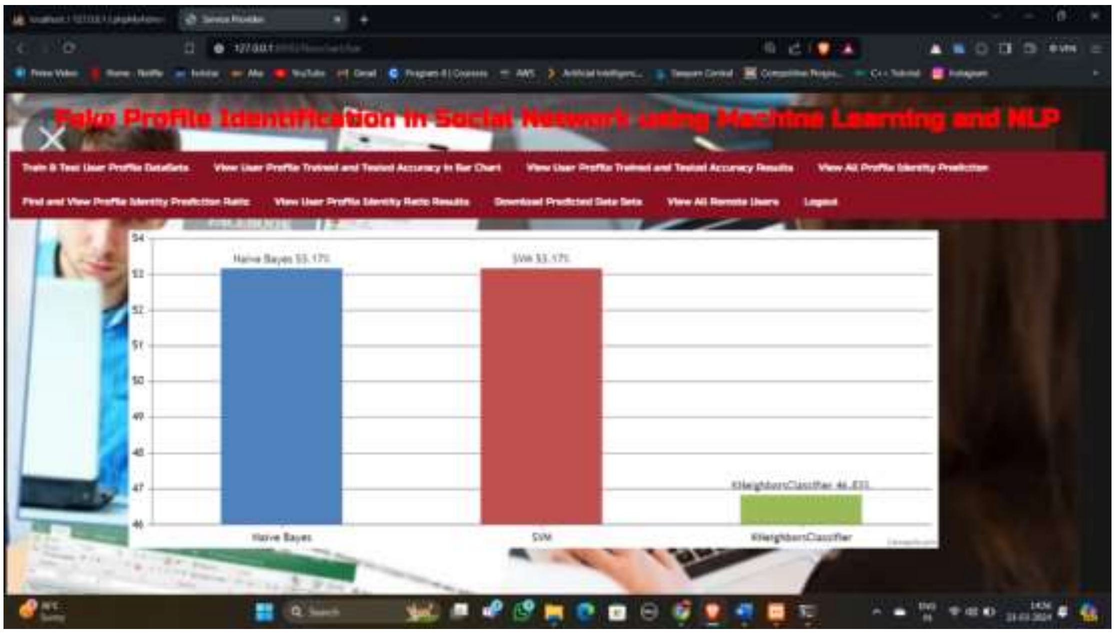

Fake Profile Identification in Social Network using Machine Learning and NLP
Train & Test User Profile DataSets
View User Profile Trained and Tested Accuracy in Bar Chart
View User Profile Trained and Tested Accuracy Results
View All Profile Identity Prediction
Find and View Profile Identity Prediction Ratio
View User Profile Identity Ratio Results
Download Predicted Data Sets
View All Remote Users
Logout
54
53
52
51
50
49
48
47
46
Naive Bayes 53.17%
SVM 53.17%
KNeighborsClassifier 46.83%
Naive Bayes
SVM
KNeighborsClassifier

Fig 4.5: Bar Chart

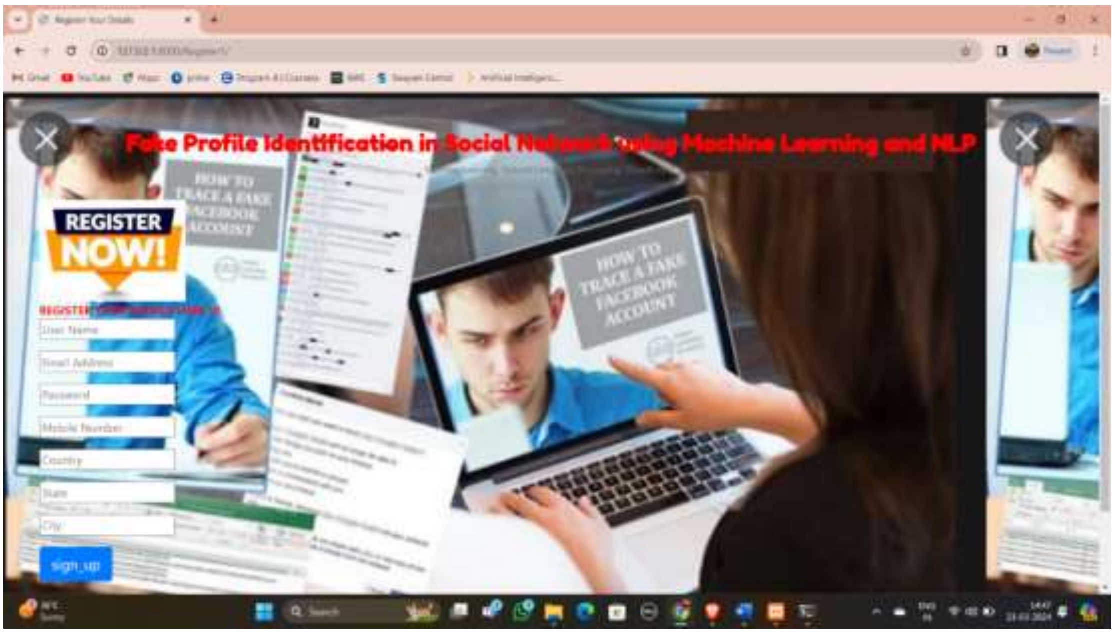


Fig 4.5: Registration Page

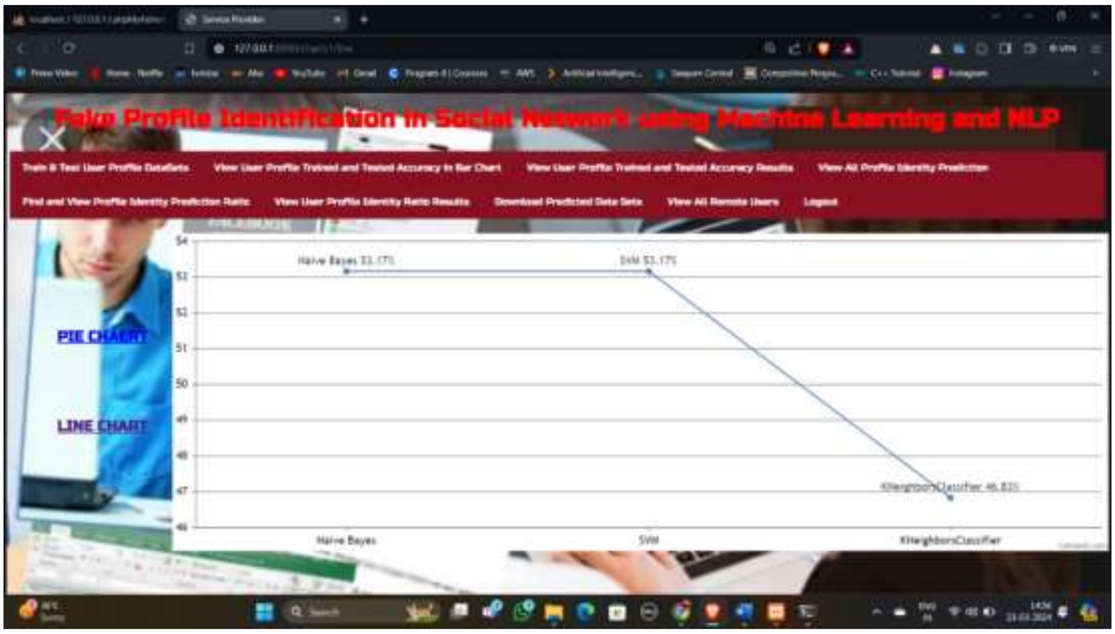

Fake Profile Identification in Social Network using Machine Learning and NLP
Train & Test User Profile DataSets
View User Profile Trained and Tested Accuracy in Bar Chart
View User Profile Trained and Tested Accuracy Results
View All Profile Identity Prediction
Find and View Profile Identity Prediction Ratio
View User Profile Identity Ratio Results
Download Predicted Data Sets
View All Remote Users
Logout
PIE CHART
LINE CHART
Naive Bayes 53.175
SVM 53.175
KNeighborsClassifier 46.835
54
53
52
51
50
49
48
47
46
Naive Bayes
SVM
KNeighborsClassifier

Fig 4.5: Graph

CODE:

```
import tkinter as tk

from tkinter import Tk

from tkinter import filedialog

import pandas as pd

import numpy as np

from sklearn.naive_bayes import MultinomialNB

from sklearn.svm import LinearSVC

import matplotlib.figure

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score,
classification_report, confusion_matrix

from sklearn.neighbors import KNeighborsClassifier
```

```
import matplotlib.pyplot as plt

from tkinter import ttk

from tkinter.filedialog import askopenfilename

from collections import OrderedDict


filename=None

ErrorrateMeans=list()

AccuracyMeans=list()


def browse() :

        Tk().withdraw()

        global filename

        filename = askopenfilename()

        print (filename)




def Create_Input_Window():

        input_window = tk.Toplevel(root)

        input_window.geometry("400x500")

        input_window.resizable(0, 0)


        def retrieve_input():

                if e1 :

                 inputframe =pd.DataFrame(

                 OrderedDict(
                {
                'UserID':[e1.get()],
                'No Of Abuse Report':[e2.get()],
                'Rejected Friend Requests':[e3.get()],
                'No Of Freind Requests Thar Are Not Accepted':[e4.get()],
                'No Of Friends':[e5.get()],
                'No Of Followers':[e6.get()],
                'No Of Likes To Unknown Account':[e7.get()],
                'No Of Comments Per Day':[e8.get()],
                }))
                inputframe = inputframe[['UserID', 'No Of Abuse Report','No Of
Freind Requests Thar Are Not Accepted','No Of Friends','No Of Followers','No Of
Likes To Unknown Account','No Of Comments Per Day']]
                print(inputframe.loc[0])
                Naive_Bayes_Manual_Input(inputframe)



        def show_predicted_label(label):
                if label == 1:
                    account_type_label=tk.Label(input_window, text="Account
Type : Fake").grid(row=24)
                else :
                    account_type_label=tk.Label(input_window, text="Account
Type : Not Fake").grid(row=24)
```

```
def Linear_Svc_Manual_Input ():
          if  filename :
                    print("here")
                    if e1 :
                     inputframe =pd.DataFrame(
                     OrderedDict(
                    {
                    'UserID':[e1.get()],
                    'No Of Abuse Report':[e2.get()],
                    'Rejected Friend Requests':[e3.get()],
                    'No Of Freind Requests Thar Are Not Accepted':[e4.get()],
                    'No Of Friends':[e5.get()],
                    'No Of Followers':[e6.get()],
                    'No Of Likes To Unknown Account':[e7.get()],
                    'No Of Comments Per Day':[e8.get()],
                    }))
                    inputframe = inputframe[['UserID', 'No Of Abuse Report','No
Of Freind Requests Thar Are Not Accepted','No Of Friends','No Of Followers','No Of
Likes To Unknown Account','No Of Comments Per Day']]
                    print(inputframe.loc[0])

                    df=pd.read_csv(filename)
                    msk = np.random.rand(len(df)) < 0.7
                    train = df[msk]

                    test = inputframe

                    testing_data=test.values[:, 0:7]

                    features = train.values[:, 0:7]

                    labels   = train.values[:, 8].astype('int')

                    model2 = LinearSVC()

                    model2.fit(features,labels)

                    predictions_model2 = model2.predict(testing_data)


                    print('2.Linear SVC :\n')

                    print('\n Predicted Class :',predictions_model2[0])

                    show_predicted_label(predictions_model2[0])



def Naive_Bayes_Manual_Input ():

            if  filename :

                    print("here")


                    if e1 :

                     inputframe =pd.DataFrame(

                     OrderedDict(

                    {

                    'UserID':[e1.get()],

                    'No Of Abuse Report':[e2.get()],

                    'Rejected Friend Requests':[e3.get()],
                    'No Of Freind Requests Thar Are Not Accepted':[e4.get()],
                    'No Of Friends':[e5.get()],
                    'No Of Followers':[e6.get()],
                    'No Of Likes To Unknown Account':[e7.get()],
                    'No Of Comments Per Day':[e8.get()],
                    }))
                    inputframe = inputframe[['UserID', 'No Of Abuse Report','No Of Freind Requests Thar Are Not Accepted','No Of Friends','No Of Followers','No Of Likes To Unknown Account','No Of Comments Per Day']]
                    print(inputframe.loc[0])



                    df=pd.read_csv(filename)
                    msk = np.random.rand(len(df)) < 0.7
                    train = df[msk]
                    test = inputframe
                    testing_data=test.values[:, 0:7]
                    features = train.values[:, 0:7]
                    labels   = train.values[:, 8].astype('int')
                    model1 = MultinomialNB()
                    model1.fit(features,labels)
                    predictions_model1 = model1.predict(testing_data)

                    print('\n1.Multinomial Naive Bayes :\n')
                    print('\n Predicted Class :',predictions_model1[0])
```

```
                        show_predicted_label(predictions_model1[0])


        def Knn__Manual_Input():

                if  filename :

                        print("here")


                        if e1 :

                         inputframe =pd.DataFrame(

                         OrderedDict(

                        {

                        'UserID':[e1.get()],

                        'No Of Abuse Report':[e2.get()],

                        'Rejected Friend Requests':[e3.get()],

                        'No Of Freind Requests Thar Are Not Accepted':[e4.get()],

                        'No Of Friends':[e5.get()],

                        'No Of Followers':[e6.get()],

                        'No Of Likes To Unknown Account':[e7.get()],

                        'No Of Comments Per Day':[e8.get()],

                        }))

                        inputframe = inputframe[['UserID', 'No Of Abuse Report','No
Of Freind Requests Thar Are Not Accepted','No Of Friends','No Of Followers','No Of
Likes To Unknown Account','No Of Comments Per Day']]

                        print(inputframe.loc[0])
```

```
                    df=pd.read_csv(filename)

                    msk = np.random.rand(len(df)) < 0.7

                    train = df[msk]

                    test = inputframe

                    testing_data=test.values[:, 0:7]

                    features = train.values[:, 0:7]

                    labels   = train.values[:, 8].astype('int')

                    model3 = KNeighborsClassifier(n_neighbors=3)

                    model3.fit(features,labels)

                    predictions_model3 = model3.predict(testing_data)

                    print('3.K-Nearest Neighbors  :\n')

                    print('\n Predicted Class :',predictions_model3[0])

                    show_predicted_label(predictions_model3[0])
```

```
        tk.Label(input_window, text="Enter UserID").grid(row=0)

        tk.Label(input_window, text="Enter No Of Abuse Report").grid(row=3)

        tk.Label(input_window, text="Enter No Of Rejected Friend Requests").grid(row=5)

        tk.Label(input_window, text="Enter No Of Freind Requests Thar Are Not Accepted").grid(row=7)

        tk.Label(input_window, text="Enter No Of Friends").grid(row=9)

        tk.Label(input_window, text="Enter No Of Followers").grid(row=11)

        tk.Label(input_window, text="Enter No Of Likes To Unknown Account").grid(row=13)
```

```
    tk.Label(input_window, text="Enter No Of Comments Per Day").grid(row=15)



    e1 = tk.Entry(input_window)
    e2 = tk.Entry(input_window)
    e3 = tk.Entry(input_window)
    e4 = tk.Entry(input_window)
    e4 = tk.Entry(input_window)
    e5 = tk.Entry(input_window)
    e6 = tk.Entry(input_window)
    e7 = tk.Entry(input_window)
    e8 = tk.Entry(input_window)

    e1.grid(row=2, column=0)
    e2.grid(row=4, column=0)
    e3.grid(row=6, column=0)
    e4.grid(row=8, column=0)
    e5.grid(row=10, column=0)
    e6.grid(row=12, column=0)
    e7.grid(row=14, column=0)
    e8.grid(row=16, column=0)

    tk.Label(input_window,
             text="Predict",
             fg = "dark violet",

             bg = "yellow2",

             width=35,

             height=1,

             font = "Helvetica 15 bold italic").grid(row=32)


    Naive_Bayes_button = tk.Button(input_window,

                                text="Naive Bayes",

                                fg="black",

                                bg="light steel blue",

                                width=10,

                                height=2,

                                command=Naive_Bayes_Manual_Input,

                                font = "Helvetica 10 bold italic",

                                )

    Naive_Bayes_button.place(relx=0.025, rely=0.8)



    Linear_Svc_Button = tk.Button(input_window,

                                text="Linear SVC",

                                fg="black",

                                bg="misty rose",

                                width=15,

                                height=2,

                                font = "Helvetica 10 bold italic",
                                          command=Linear_Svc_Manual_Input,
                                          )
     Linear_Svc_Button.place(relx=0.31, rely=0.8)



     KNN_button = tk.Button(input_window,
                                          text="KNN",
                                          fg="black",
                                          bg="salmon1",
                                          width=15,
                                          height=2,
                                          font = "Helvetica 10 bold italic",
                                          command=Knn__Manual_Input,
                                          )
     KNN_button.place(relx=0.7, rely=0.8)




def Naive_Bayes():
     if filename:
          global AccuracyMeans,ErrorrateMeans
          df=pd.read_csv(filename)
          msk = np.random.rand(len(df)) < 0.7
          train = df[msk]
          test = df[~msk]
```

```
            testing_data=test.values[:, 0:7]

            testing_data_labels=test.values[:, 8]

            features = train.values[:, 0:7]

            labels   = train.values[:, 8].astype('int')

            model1 = MultinomialNB()

            model1.fit(features,labels)

            predictions_model1 = model1.predict(testing_data)


            accuracy=accuracy_score(testing_data_labels,
predictions_model1)*100

            AccuracyMeans.append(accuracy)

            error_rate=100-accuracy

            ErrorrateMeans.append(error_rate)

            precision=precision_score(testing_data_labels,
predictions_model1)*100

            recall=recall_score(testing_data_labels, predictions_model1)*100


            print('\n1.Multinomial Naive Bayes :\n')

            print('Confusion Matrix :')

            print(confusion_matrix(testing_data_labels, predictions_model1))

            print('Accuracy Is : '+str(accuracy )+' %')

            print('Error Rate Is : '+str(error_rate)+' %')

            print('Precision Is : '+str(precision)+' %')

            print('Recall Is : '+str(recall)+' %\n\n')
```

```
                labels = ['Error Rate', 'Accuracy ']

                sizes = [error_rate,accuracy]

                explode = (0, 0.1)

                fig1, ax1 = plt.subplots()

                ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',

                                shadow=True, startangle=90)

                plt.title('Naive Bayes Algorithm')

                ax1.axis('equal')

                plt.tight_layout()

                plt.show()


def Linear_Svc():

        if filename:

                global AccuracyMeans,ErrorrateMeans

                df=pd.read_csv(filename)

                msk = np.random.rand(len(df)) < 0.7

                train = df[msk]

                test = df[~msk]

                testing_data=test.values[:, 0:7]

                testing_data_labels=test.values[:, 8]

                features = train.values[:, 0:7]

                labels   = train.values[:, 8].astype('int')

                model2 = LinearSVC()

                model2.fit(features,labels)

                predictions_model2 = model2.predict(testing_data)
```

```
            accuracy=accuracy_score(testing_data_labels,
predictions_model2)*100
            AccuracyMeans.append(accuracy)
            error_rate=100-accuracy
            ErrorrateMeans.append(error_rate)
            precision=precision_score(testing_data_labels,
predictions_model2)*100
            recall=recall_score(testing_data_labels, predictions_model2)*100

            print('2.Linear SVC :\n')
            print('Confusion Matrix :')
            print(confusion_matrix(testing_data_labels, predictions_model2))
            print('Accuracy Is : '+str(accuracy )+' %')
            print('Error Rate Is : '+str(error_rate)+' %')
            print('Precision Is : '+str(precision)+' %')
            print('Recall Is : '+str(recall)+' %\n\n')

            labels = ['Error Rate', 'Accuracy ']
            sizes = [error_rate,accuracy]
            explode = (0, 0.1)
            fig1, ax1 = plt.subplots()
            ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
                        shadow=True, startangle=90)
```

```
        plt.title('Linear SVC Algorithm')

        ax1.axis('equal')

        plt.tight_layout()

        plt.show()




def Knn():

    if filename:

        global AccuracyMeans,ErrorrateMeans

        df=pd.read_csv(filename)

        msk = np.random.rand(len(df)) < 0.7

        train = df[msk]

        test = df[~msk]

        testing_data=test.values[:, 0:7]

        testing_data_labels=test.values[:, 8]

        features = train.values[:, 0:7]

        labels   = train.values[:, 8].astype('int')


        model3 = KNeighborsClassifier(n_neighbors=3)

        model3.fit(features,labels)

        predictions_model3 = model3.predict(testing_data)


        accuracy=accuracy_score(testing_data_labels,
predictions_model3)*100

        AccuracyMeans.append(accuracy)
        error_rate=100-accuracy
        ErrorrateMeans.append(error_rate)
        precision=precision_score(testing_data_labels,
predictions_model3)*100
        recall=recall_score(testing_data_labels, predictions_model3)*100

        print('3.K-Nearest Neighbors  :\n')
        print('Confusion Matrix :')
        print(confusion_matrix(testing_data_labels, predictions_model3))
        print('Accuracy Is : '+str(accuracy )+' %')
        print('Error Rate Is : '+str(error_rate)+' %')
        print('Precision Is : '+str(precision)+' %')
        print('Recall Is : '+str(recall)+' %\n\n')

        labels = ['Error Rate', 'Accuracy ']
        sizes = [error_rate,accuracy]

        explode = (0, 0.1)
        fig1, ax1 = plt.subplots()
        ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
                shadow=True, startangle=90)

        plt.title('KNN Algorithm')
        ax1.axis('equal')
        plt.tight_layout()
```

```
            plt.show()




def compare():

        N = 3

        ind = np.arange(N)

        width = 0.35

        p1 = plt.bar(ind, AccuracyMeans, width )

        p2 = plt.bar(ind, ErrorrateMeans, width,bottom=AccuracyMeans )

        plt.ylabel('Scores')

        plt.title('Performance By Classifiers')

        plt.xticks(ind, ('Naive Bayes', 'Linear SVC', 'KNN',))

        plt.yticks(np.arange(0, 110, 10))

        plt.legend((p1[0], p2[0]), ('Accuracy', 'Error Rate'))

        plt.show()



root = tk.Tk()

root.title("X Fake Account Detector")

root.grid_columnconfigure(0, weight=1)

root.geometry("600x500")

root.resizable(0, 0)
```

```
tk.Label(root,
                text="Fake Account Detector",
                fg = "dark violet",
                bg = "light blue",
                width=400,
                height=2,
                font = "Helvetica 35 bold italic").pack()


browsebutton = tk.Button(root,
            text="Browse File",
            fg="blue",
            bg="thistle",
            width=45,
            height=2,
            font = "Helvetica 15 bold italic",
            command=browse,
            )
browsebutton.pack(padx=0,pady=10)


tk.Label(root,
                text="Classifiers ",
                fg = "dark violet",
                bg = "light green",
                width=40,
                height=1,
                font = "Helvetica 35 bold italic").pack(padx=0,pady=0)


Manual_Input_Button = tk.Button(root,
            text="Give Manual Input",
            fg="black",
            bg="LightGoldenRod1",
            width=35,
            height=2,
            font = "Helvetica 10 bold italic",
            command=Create_Input_Window,
            )
Manual_Input_Button.place(relx=0.25, rely=0.63)


Naive_Bayes_button = tk.Button(root,
            text="Naive Bayes",
            fg="black",
            bg="light steel blue",
            width=25,
            height=2,
            command=Naive_Bayes,
            font = "Helvetica 10 bold italic",
            )
Naive_Bayes_button.pack( side=tk.LEFT)


Linear_Svc_Button = tk.Button(root,
            text="Linear SVC",
            fg="black",
            bg="misty rose",
            width=25,
            height=2,
            font = "Helvetica 10 bold italic",
            command=Linear_Svc,
            )
Linear_Svc_Button.pack(side=tk.LEFT)

KNN_button = tk.Button(root,
            text="KNN",
            fg="black",
            bg="salmon1",
            width=25,
            height=2,
            font = "Helvetica 10 bold italic",
            command=Knn,
            )
KNN_button.pack( side=tk.LEFT)

All_statistics_Button = tk.Button(root,
            text="Compare",
            fg="black",
            bg="aquamarine2",
            width=35,
            height=2,
            font = "Helvetica 10 bold italic",
            command=compare,
            )
All_statistics_Button.place(relx=0.25, rely=0.9)
root.mainloop()
```

Chapter 5

5 Conclusion

To sum it up, our research proposes an approach for categorizing profiles on social networking sites as either real or actual accounts. The convolutional neural network-based deep model served as the foundation for the algorithm designed to extract patterns of descriptive writing from the contents of posts. Experiments have been carried out using classifiers using our proposed dataset collected from the internet. It was shown that the suggested technique generates relatively high detection performance, coming in at 99.8% overall. When the findings are considered, the proposed text-based model shows promising accuracy in categorizing the user type based on the writing style they use. In our opinion, the proposed methodology has the potential to be of great assistance in the fight against fraud on social networking sites. Further investigation into diverse deep-learning techniques may provide additional intriguing findings.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. M. Mansoor, Z. Rehman, M. Shaheen, A. Khan and M. Habib, "Deep learning based semantic similarity detection using text data," *Information Technology and Control*, vol. *49*, no. *4*, pp. 495–510, 2020.
2. M. Shaheen and M. Shahbaz, "An algorithm of association rule mining for microbial energy prospection," *Scientific Reports*, vol. *7*, no. *1*, pp. 1–12, 2017.
3. R. Kaliyar, A. Goswami and P. Narang, "FakeBERT: Fake news detection in social media with a BERT-based deep learning approach," *Multimedia Tools and Applications*, vol. *80*, no. *8*, pp. 11765–11788, 2021.
4. F. Pierri and S. Ceri, "False news on social media: A data-driven survey," *ACM Sigmod Record*, vol. *48*, no. *2*, pp. 18–27, 2019.
5. D. Sultan, A. Toktarova, A. Zhumadillayeva, S. Aldeshov, S. Mussiraliyeva et al., "Cyberbullying-related hate speech detection using shallow-to-deep learning," *Computers, Materials & Continua*, vol. *74*, no. *1*, pp. 2115–2131, 2023.
6. B. Ghanem, P. Rosso and F. Rangel, "An emotional analysis of false information in social media and news articles," *ACM Transactions on Internet Technology*, vol. *20*, no. *2*, pp. 1–18, 2020.
7. M. Ahvanooey, M. Zhu, W. Mazurczyk, K. Choo, M. Conti et al., "Misinformation detection on social media: Challenges and the road ahead," *IT Professional*, vol. *24*, no. *1*, pp. 34–40, 2022.
8. G. Pennycook and D. Rand, "The psychology of fake news," *Trends in Cognitive Sciences*, vol. *25*, no. *5*, pp. 388–402, 2021.
9. D. Sultan, B. Omarov, Z. Kozhamkulova, G. Kazbekova, L. Alimzhanova et al., "A review of machine learning techniques in cyberbullying detection," *Computers, Materials & Continua*, vol. *74*, no. *3*, pp. 5625–5640, 2023.
10. D. Barud and A. Salau, "Detection of fake news and hate speech for Ethiopian languages: A systematic review of the approaches," *Journal of Big Data*, vol. *9*, no. *1*, pp. 1–17, 2022.
11. D. Freelon, M. Bossetta, C. Wells, J. Lukito, Y. Xia et al., "Black trolls matter: Racial and ideological asymmetries in social media disinformation," *Social Science Computer Review*, vol. *40*, no. *3*, pp. 560–578, 2022.
12. K. Hartley and M. Vu, "Fighting fake news in the COVID-19 era: Policy insights from an equilibrium model," *Policy Sciences*, vol. *53*, no. *4*, pp. 735–758, 2020.

13. B. Omarov, N. Saparkhojayev, S. Shekerbekova, O. Akhmetova, M. Sakypbekova et al., "Artificial intelligence in medicine: Real time electronic stethoscope for heart diseases detection," *Computers, Materials & Continua*, vol. *70*, no. *2*, pp. 2815–2833, 2022.

14. N. Kanagavalli and S. BaghavathiPriya, "Social networks fake account and fake news identification with reliable deep learning," *Intelligent Automation & Soft Computing*, vol. *33*, no. *1*, pp. 191–205, 2022.

15. D. Koggalahewa, Y. Xu and E. Foo, "An unsupervised method for social network spammer detection based on user information interests," *Journal of Big Data*, vol. *9*, no. *1*, pp. 1–35, 2022.

16. B. Omarov, S. Narynov, Z. Zhumanov, A. Gumar and M. Khassanova, "A Skeleton-based approach for campus violence detection," *Computers, Materials & Continua*, vol. *72*, no. *1*, pp. 315–331, 2022.

17. A. Bruns, "After the 'APIcalypse': Social media platforms and their fight against critical scholarly research," *Information, Communication & Society*, vol. *22*, no. *11*, pp. 1544–1566, 2019.

18. S. Tsao, H. Chen, T. Tisseverasinghe, Y. Yang, L. Li et al., "What social media told us in the time of COVID-19: A scoping review," *The Lancet Digital Health*, vol. *3*, no. *3*, pp. e175–e194, 2021.

19. N. Brashier and D. Schacter, "Aging in an era of fake news," *Current Directions in Psychological Science*, vol. *29*, no. *3*, pp. 316–323, 2020.

20. B. Singh and D. Sharma, "Predicting image credibility in fake news over social media using multi-modal approach," *Neural Computing and Applications*, vol. *34*, no. *24*, pp. 21503–21517, 2022.

21. S. Cresci, "A decade of social bot detection," *Communications of the ACM*, vol. *63*, no. *10*, pp. 72–83, 2020.

22. G. Rampersad and T. Althiyabi, "Fake news: Acceptance by demographics and culture on social media," *Journal of Information Technology & Politics*, vol. *17*, no. *1*, pp. 1–11, 2020.

23. M. Umer, Z. Imtiaz, S. Ullah, A. Mehmood, G. Choi et al., "Fake news stance detection using deep learning architecture (CNN-LSTM)," *IEEE Access*, vol. *8*, no. *1*, pp. 156695–156706, 2020.

24. R. Mourão and C. Robertson, "Fake news as discursive integration: An analysis of sites that publish false, misleading, hyperpartisan and sensational information," *Journalism Studies*, vol. *20*, no. *14*, pp. 2077–2095, 2019.

25. M. Awan, M. Khan, Z. Ansari, A. Yasin and H. Shehzad, “Fake profile recognition using big data analytics in social media platforms,” *International Journal of Computer Applications in Technology*, vol. *68*, no. *3*, pp. 215–222, 2022.

26. U. Ahmed, J. Lin and G. Srivastava, “Social media multiaspect detection by using unsupervised deep active attention,” *IEEE Transactions on Computational Social Systems*, vol. *6*, no. *1*, pp. 1–9, 2022.

27. R. Dougnon, P. Fournier-Viger, J. Lin and R. Nkambou, “Inferring social network user profiles using a partial social graph,” *Journal of Intelligent Information Systems*, vol. *47*, no. *1*, pp. 313–344, 2016.

28. Z. Guo, K. Yu, Y. Li, G. Srivastava and J. Lin, “Deep learning-embedded social internet of things for ambiguity-aware social recommendations,” *IEEE Transactions on Network Science and Engineering*, vol. *9*, no. *3*, pp. 1067–1081, 2021.

29. P. Wanda, “RunMax: Fake profile classification using novel nonlinear activation in CNN,” *Social Network Analysis and Mining*, vol. *12*, no. *1*, pp. 1–11, 2022.

30. M. Irfan and M. Munsif, “Deepdive: A learning-based approach for virtual camera in immersive contents,” *Virtual Reality & Intelligent Hardware*, vol. *4*, no. *3*, pp. 247–262, 2022.

31. L. Allein, M. Moens and D. Perrotta, “Preventing profiling for ethical fake news detection,” *Information Processing & Management*, vol. *60*, no. *2*, pp. 1–22, 2023.

32. M. Munsif, H. Afridi, M. Ullah, S. Khan, F. Cheikh et al., “A lightweight convolution neural network for automatic disasters recognition,” in *2022 10th European Workshop on Visual Information Processing*, IEEE, Lisbon, Portugal, pp. 1–6, 2022.

33. D. Choi, S. Chun, H. Oh, J. Han and T. Kwon, “Rumor propagation is amplified by echo chambers in social media,” *Scientific Reports*, vol. *10*, no. *1*, pp. 1–10, 2020.

34. M. Vicario, W. Quattrociocchi, A. Scala and F. Zollo, “Polarization and fake news: Early warning of potential misinformation targets,” *ACM Transactions on the Web*, vol. *13*, no. *2*, pp. 1–22, 2019.

35. A. Shrestha and F. Spezzano, “Characterizing and predicting fake news spreaders in social networks,” *International Journal of Data Science and Analytics*, vol. *13*, no. *4*, pp. 385–398, 2022.

36. A. Al-Zoubi, J. Alqatawna, H. Faris and M. Hassonah, “Spam profiles detection on social networks using computational intelligence methods: The effect of the lingual context,” *Journal of Information Science*, vol. *47*, no. *1*, pp. 58–81, 2021.

37. A. Ala’M, H. Faris, J. Alqatawna and M. Hassonah, “Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts,” *Knowledge-Based Systems*, vol. *153*, no. *1*, pp. 91–104, 2018.

38. A. Ali and A. Abdullah, “Fake accounts detection on social media using stack ensemble system,” *International Journal of Electrical & Computer Engineering*, vol. *12*, no. *3*, pp. 3013–3022, 2022.

39. R. Aswani, A. Kar and P. Ilavarasan, “Detection of spammers in twitter marketing: A hybrid approach using social media analytics and bio inspired computing,” *Information Systems Frontiers*, vol. *20*, no. *3*, pp. 515–530, 2018.

40. D. Michail, N. Kanakaris and I. Varlamis, “Detection of fake news campaigns using graph convolutional networks,” *International Journal of Information Management Data Insights*, vol. *2*, no. *2*, pp. 1–10, 2022.

41. S. Mussiraliyeva, B. Omarov, P. Yoo and M. Bolatbek, “Applying machine learning techniques for religious extremism detection on online user contents,” *Computers, Materials & Continua*, vol. *70*, no. *1*, pp. 915–934, 2022.

42. K. Purba, D. Asirvatham and R. Murugesan, “Influence maximization diffusion models based on engagement and activeness on instagram,” *Journal of King Saud University-Computer and Information Sciences*, vol. *34*, no. *6*, pp. 2831–2839, 2020.

43. S. Kudugunta and E. Ferrara, “Deep neural networks for bot detection,” *Information Sciences*, vol. *467*, no. *1*, pp. 312–322, 2018.